

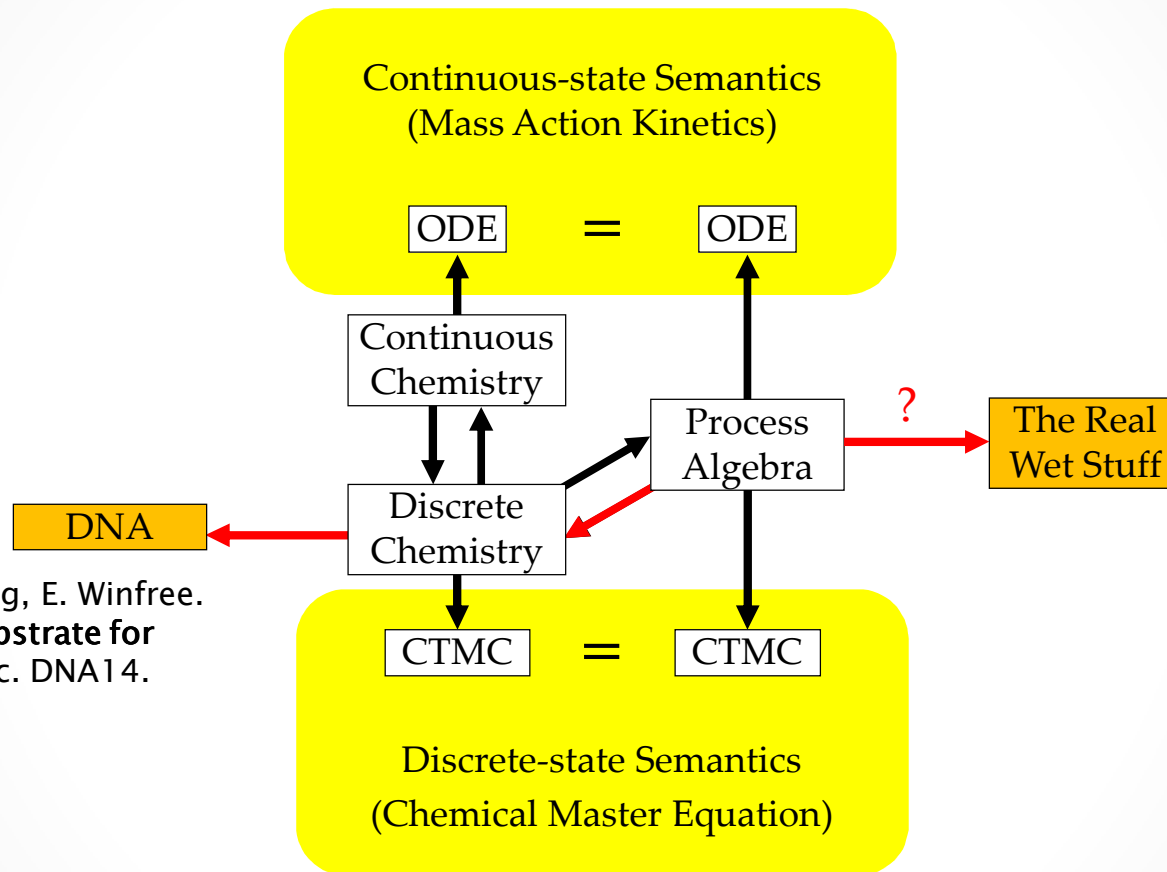


Algebras and Languages for Molecular Programming

Luca Cardelli
Microsoft Research

CS2Bio Amsterdam, 2010-05-06
<http://lucacardelli.name>

Molecules as Automata



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.

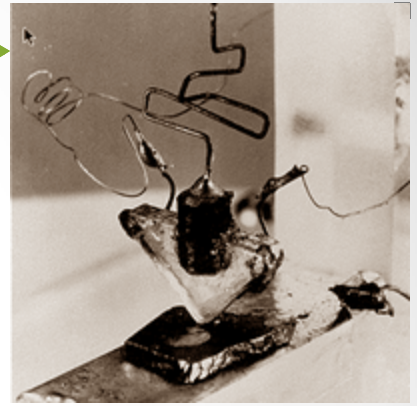
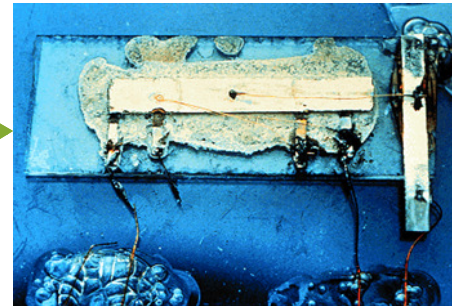
Sep. 1958. Jack Kilby builds the first integrated circuit.

50 years later

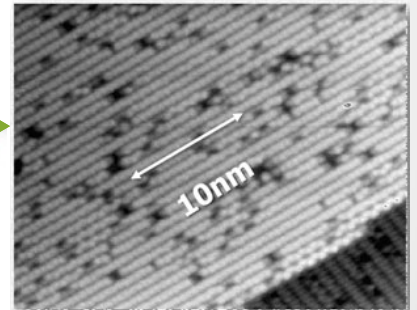
Jan. 2010. Intel and Micron announce 25nm NAND flash.

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

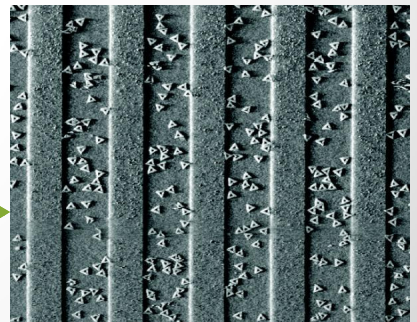
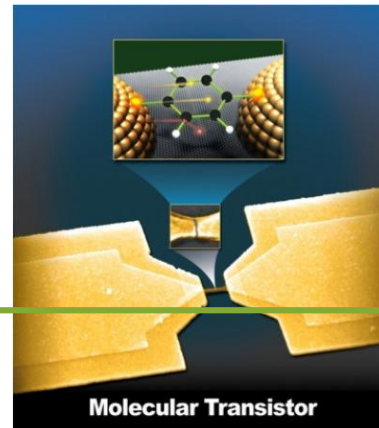
<10 iterations of Moore's Law left!
The race is on for *molecular scale integrated circuits*.



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



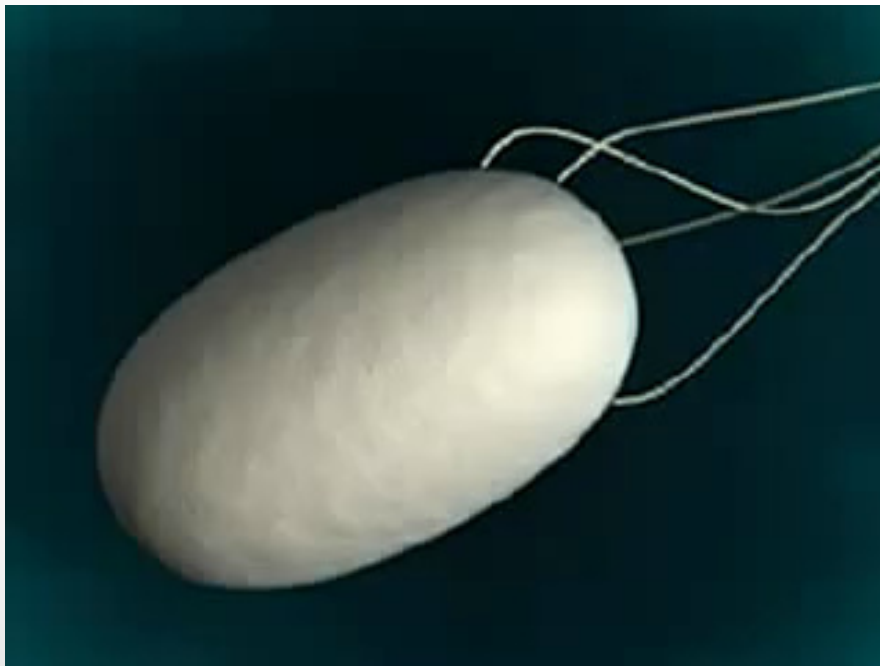
Observation of molecular orbital gating. *Nature*, 2009; 462 (7276): 1039



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

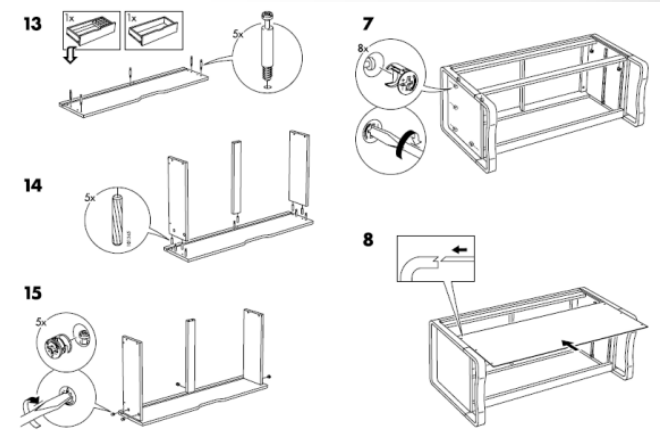
Building The *Smallest* Things

- 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*.

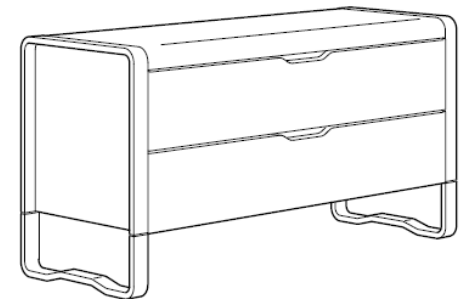


Molecular IKEA

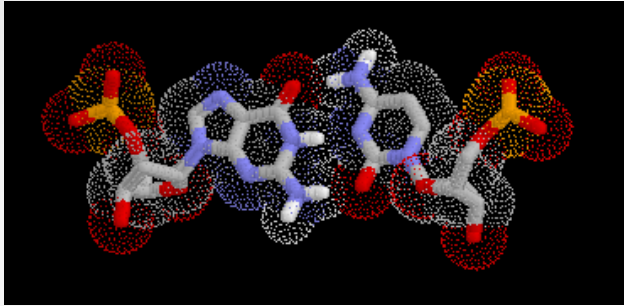
- Nature can self-assemble.
Can we?
- *“Dear IKEA, please send me a chest of drawers that assembles itself.”*
- We need a magical material where the pieces are pre-programmed to fit into to each other.
- At the molecular scale many such materials exist; let’s pick one...



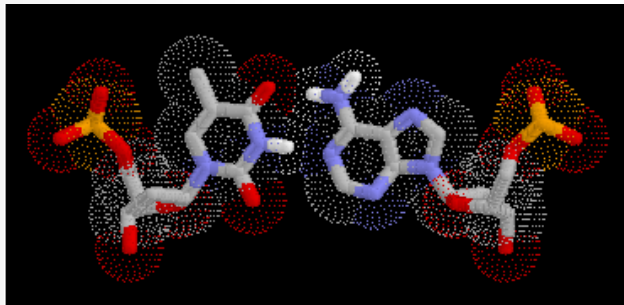
Add water



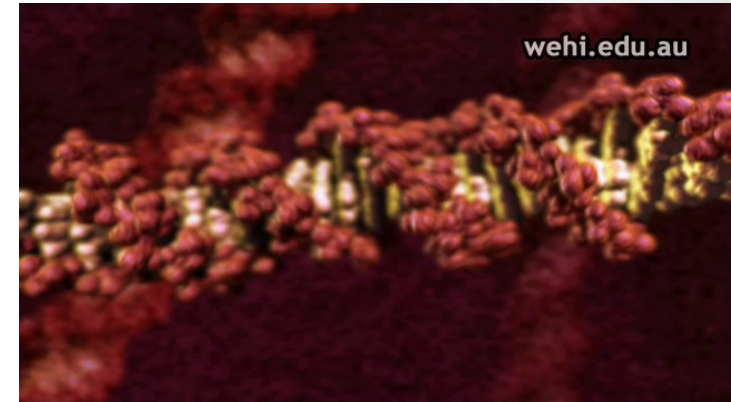
DNA



GC Base Pair
Guanine-Cytosine

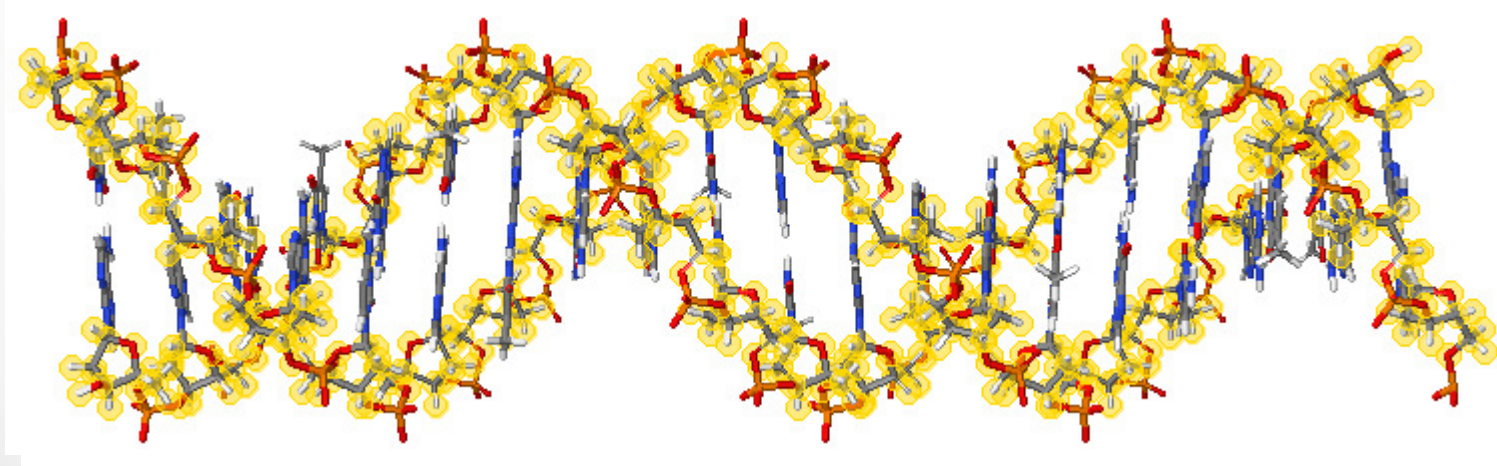


TA Base Pair
Thymine-Adenine



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

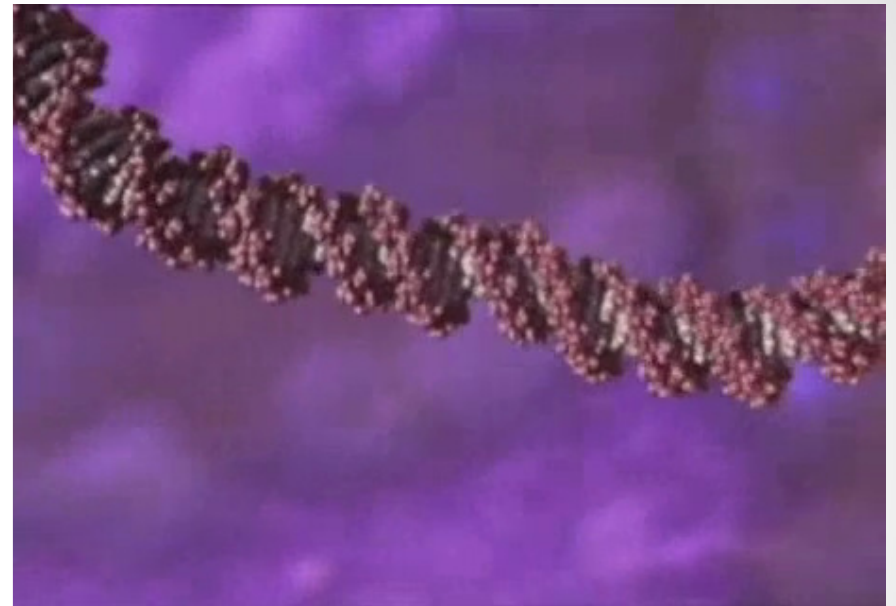
(<http://www.biosciences.bham.ac.uk/labs/minchin/tutorials/dna.html>)



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 6 μ m 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



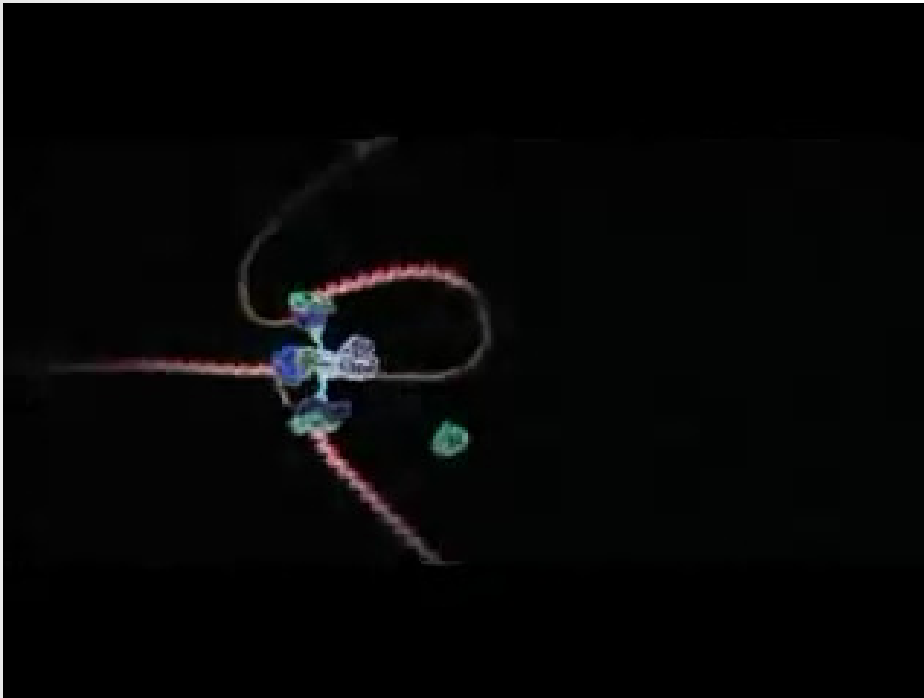
DNA wrapping into chromosomes



Andromeda Galaxy
2.5 million light years

Zippering 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*

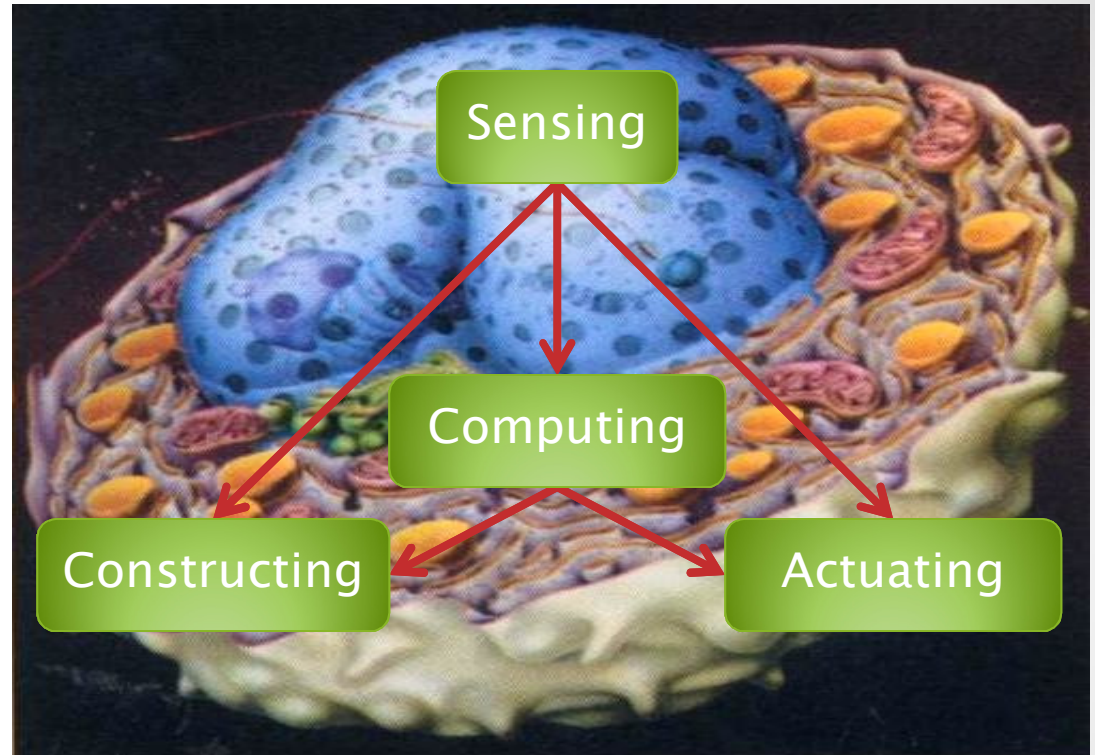
RNA polymerase II: 15–30 base/second

Drew Berry

<http://www.wehi.edu.au/wehi-tv>

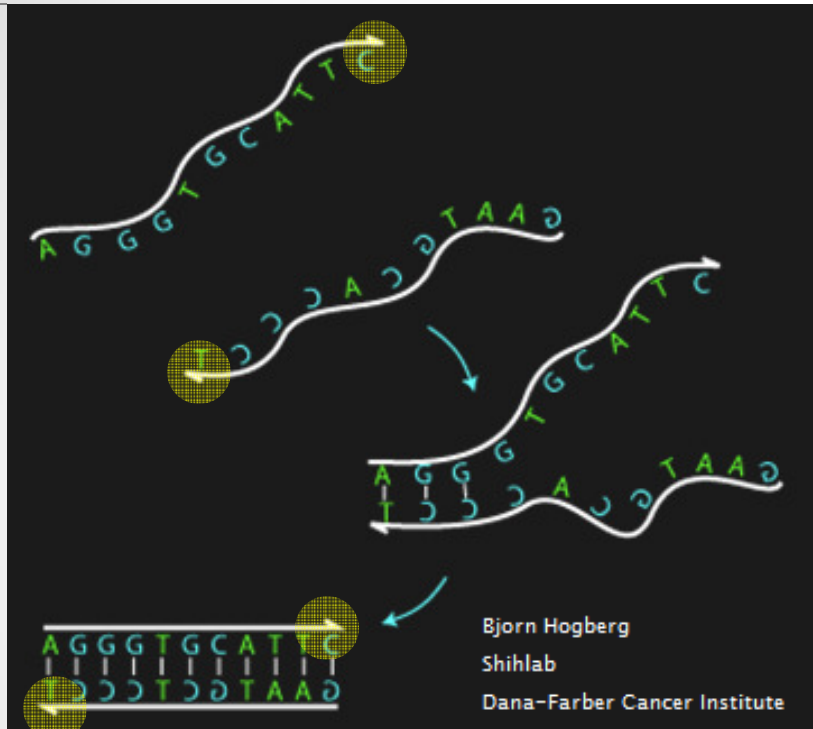
Nanoscale Engineering

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

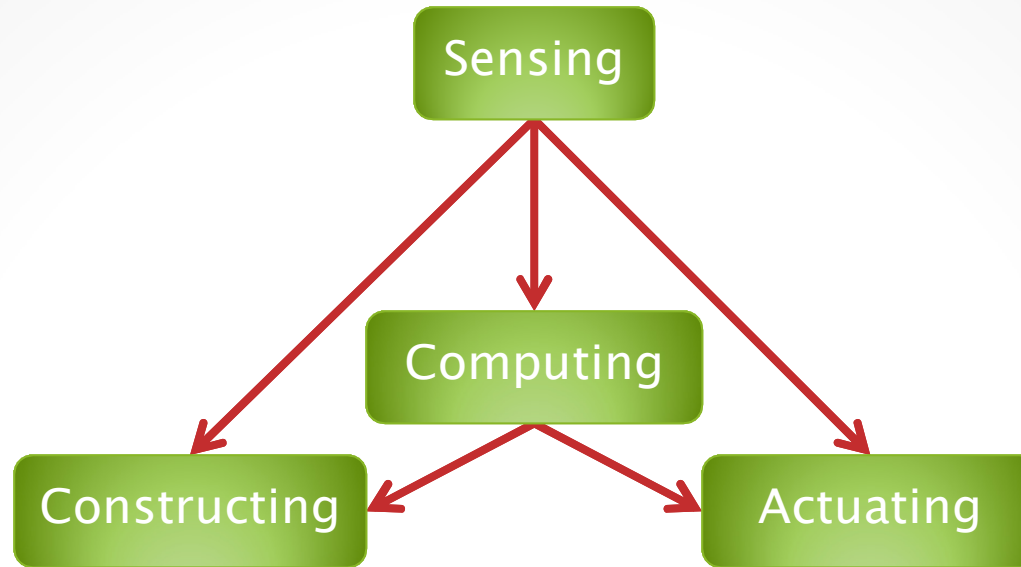


Nucleic Acids can do all this.
And interface to **biology**.

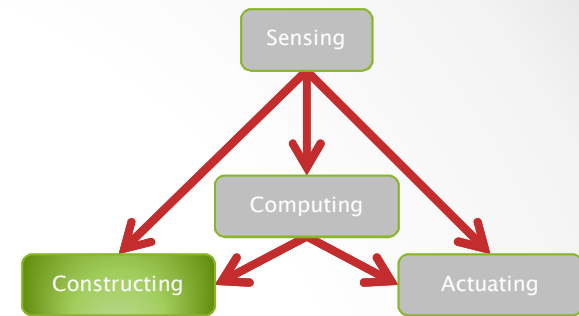
Hybridization



- 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.



Hybridization Tricks



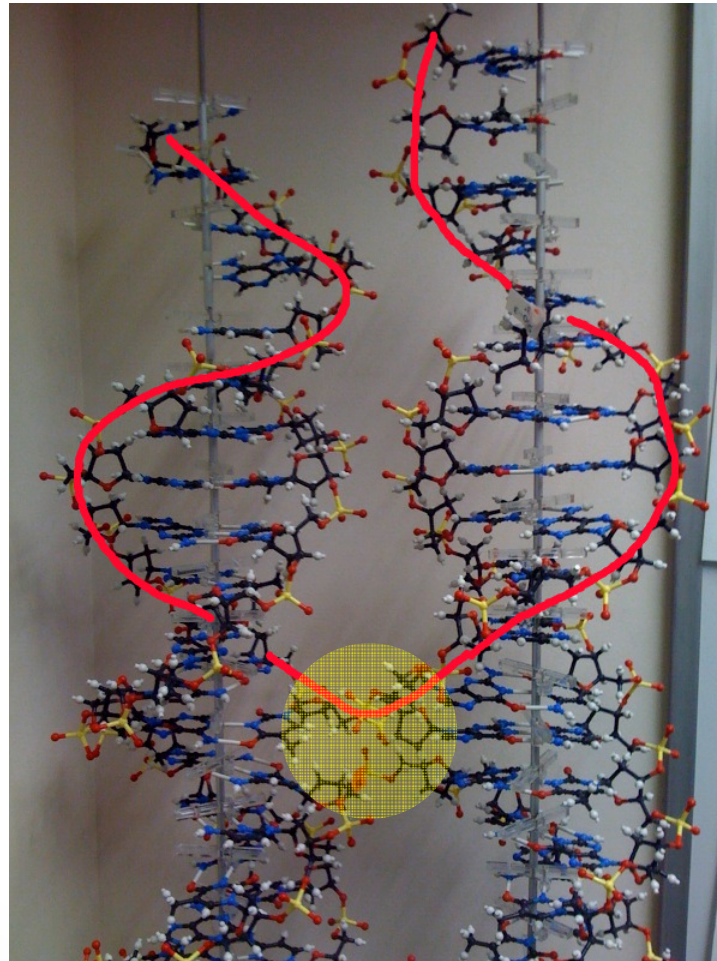
Constructing

...

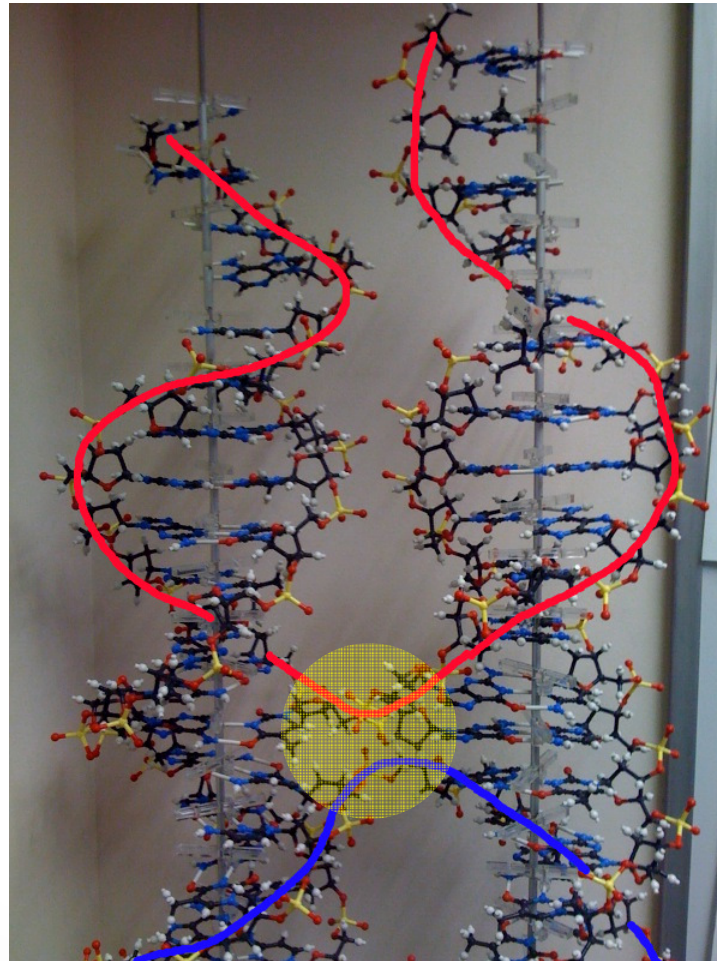
Crosslinking



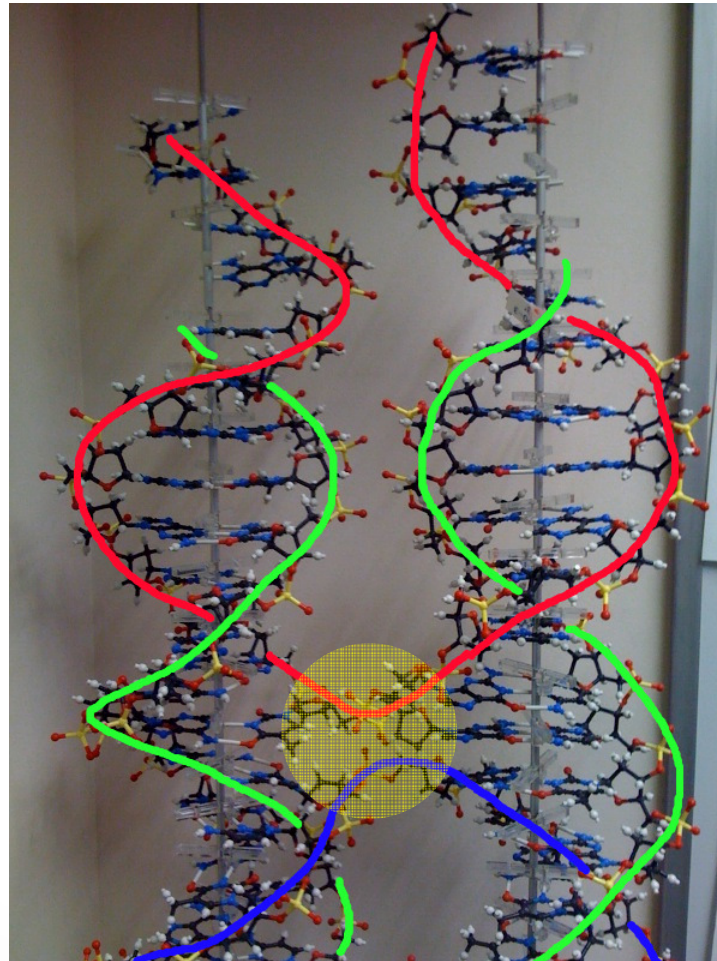
Crosslinking



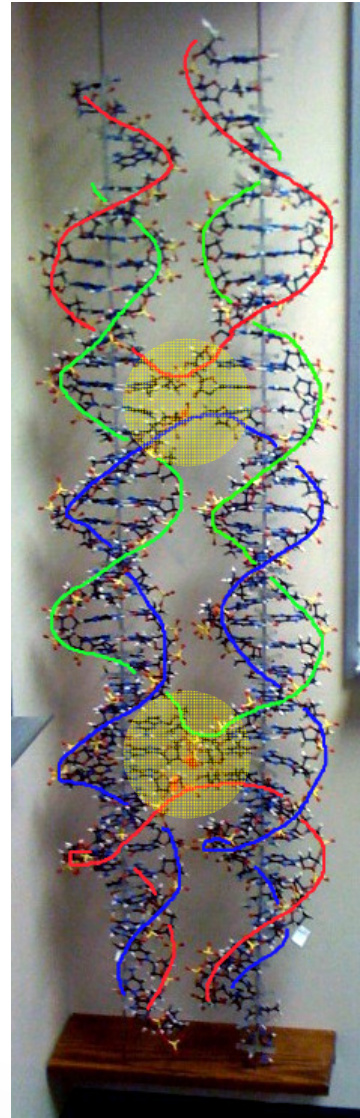
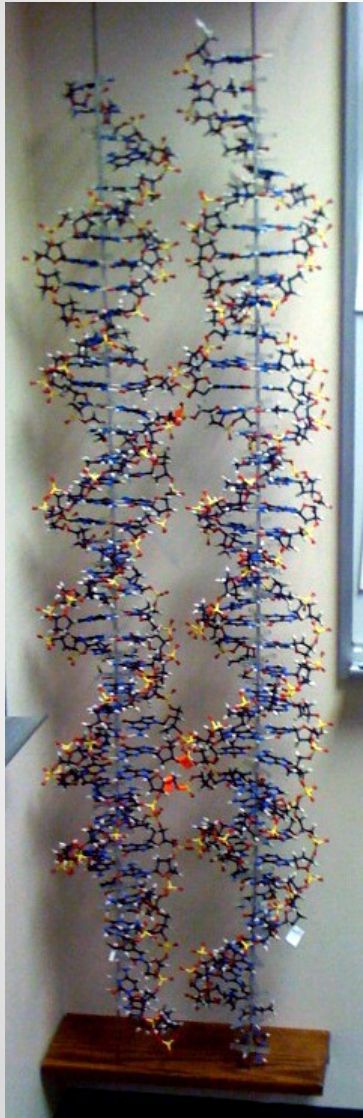
Crosslinking



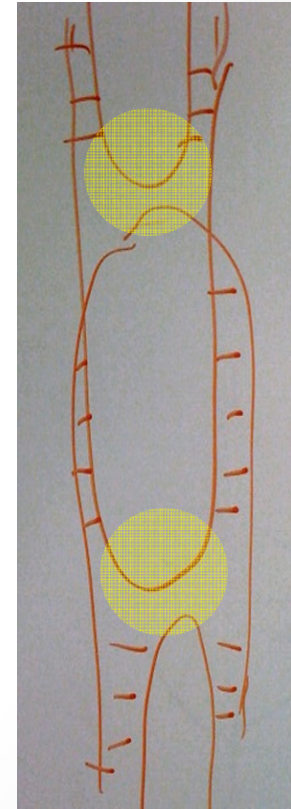
Crosslinking



Crosslinking

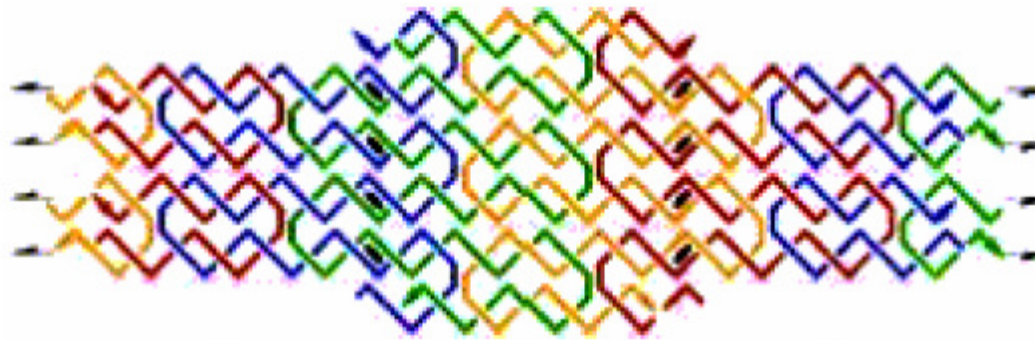
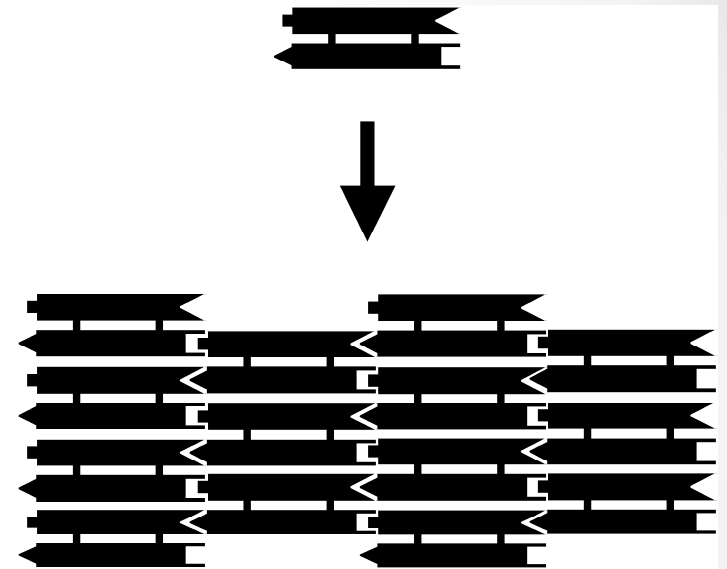
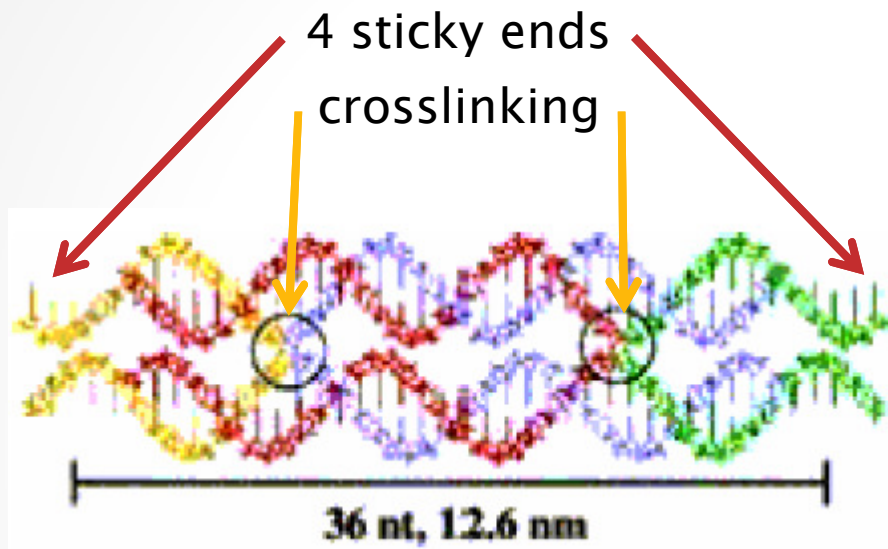


In nature, crosslinking is deadly (blocks DNA replication).



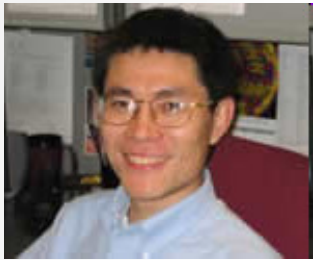
In engineering, crosslinking is the key to using DNA as a construction material.

DNA Tiling

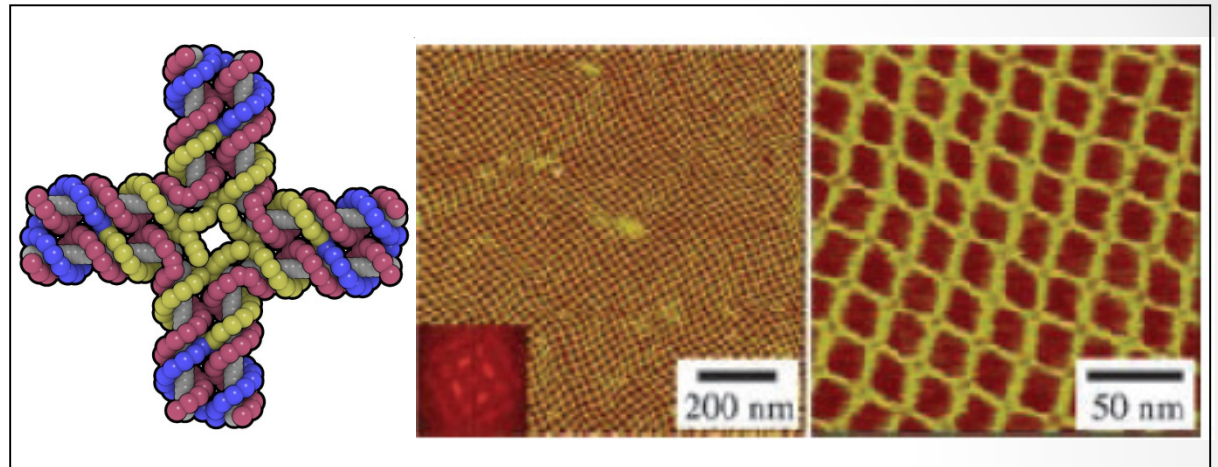


Construction and manipulation of DNA tiles in free space

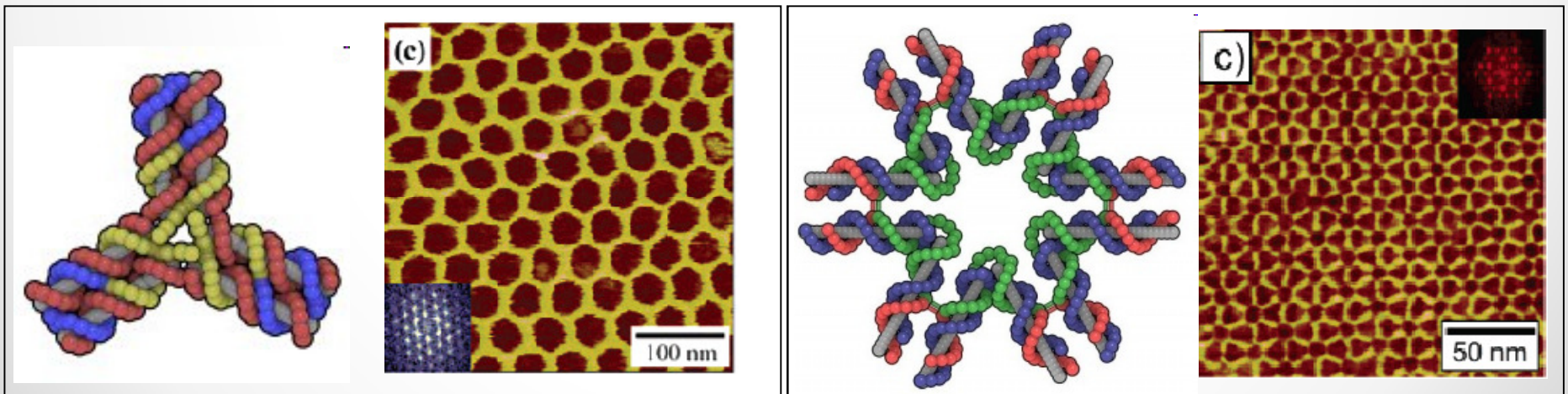
2D DNA Lattices



Chengde Mao
Purdue University, USA



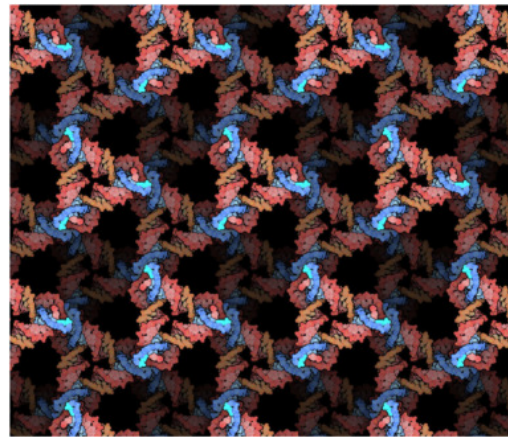
N-point Stars



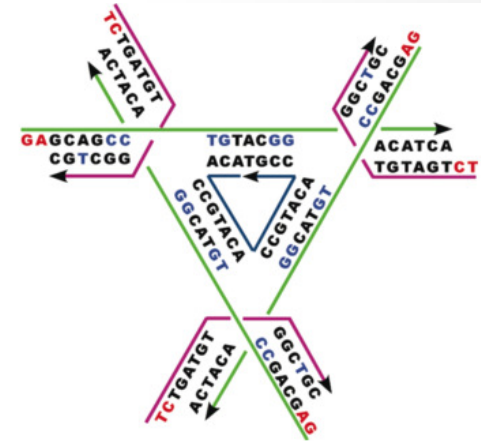
3D DNA Structures



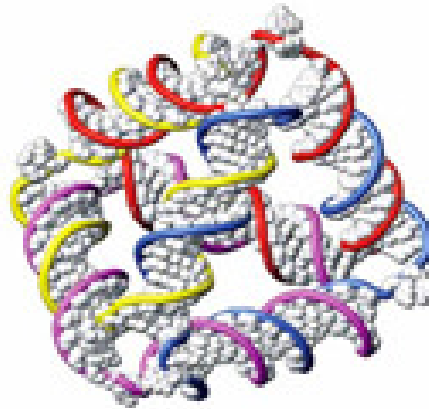
Ned Seeman
NYU



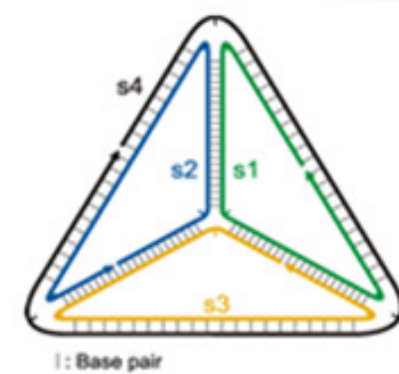
3D Crystal



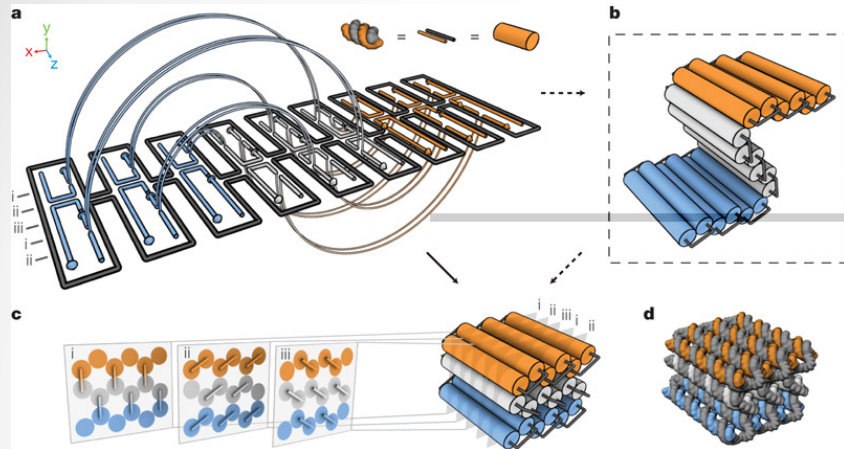
Andrew Tuberfield
Oxford



Tetrahedron

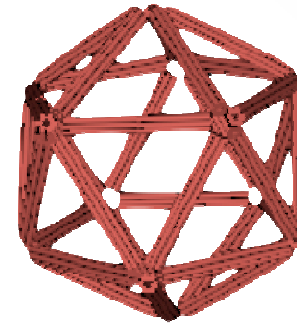


CADnano

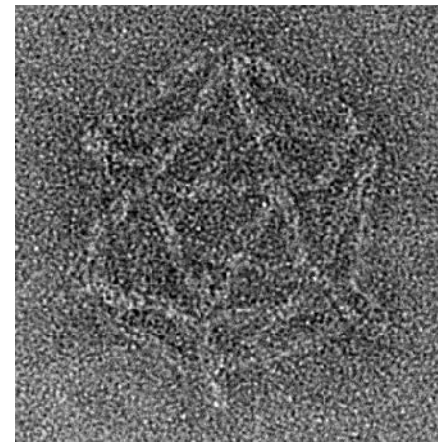


Folding DNA into Twisted and Curved Nanoscale Shapes

Hendrik Dietz, Shawn M. Douglas, & William M. Shih
[Science, 325:725–730, 7 August 2009.](#)



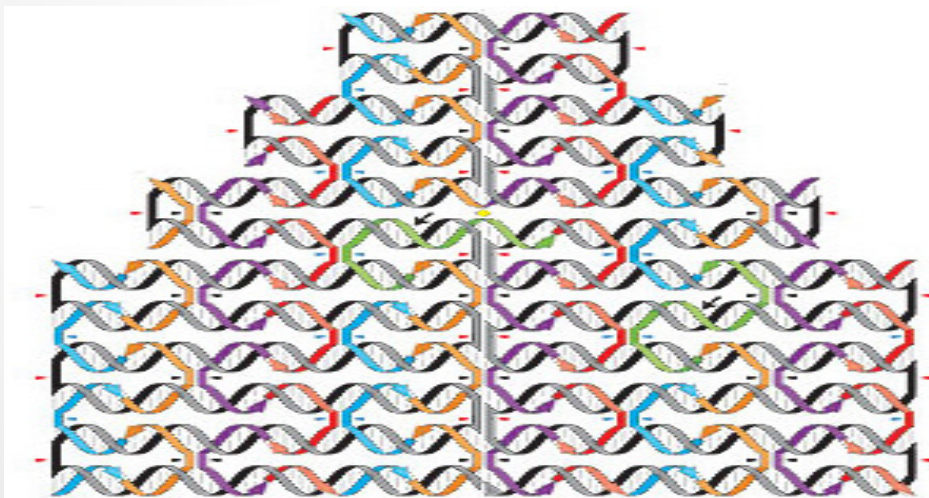
William Shih
Harvard



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)

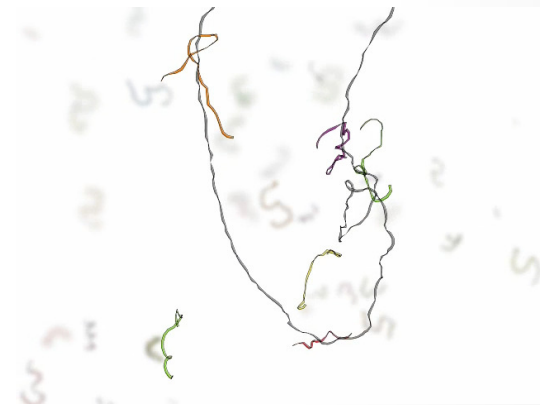
DNA Origami

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

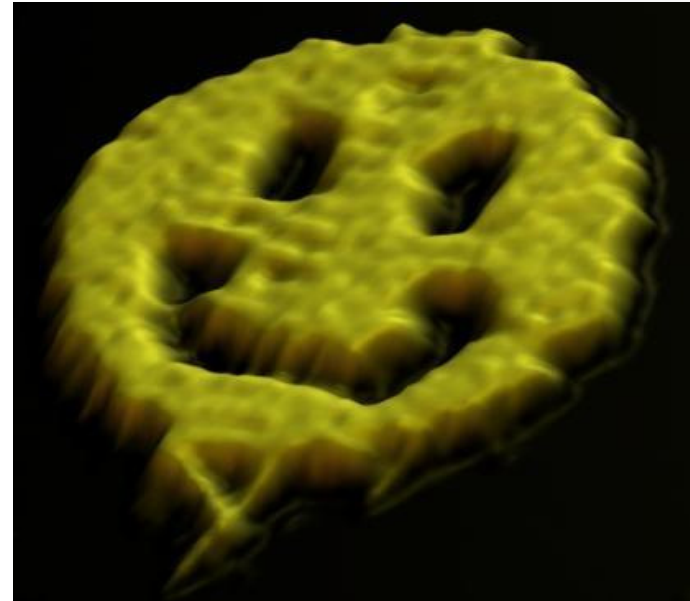
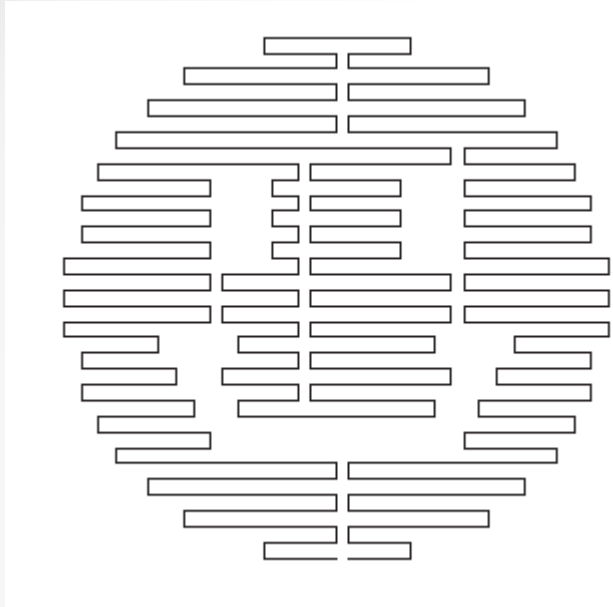


PWK Rothemund, *Nature* 440, 297 (2006)

Black: long viral strand
Color: short staple strands



DNA Origami



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

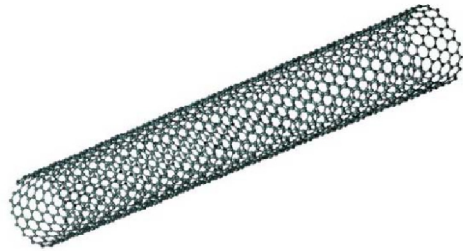


Paul W K Rothemund
California Institute of Technology

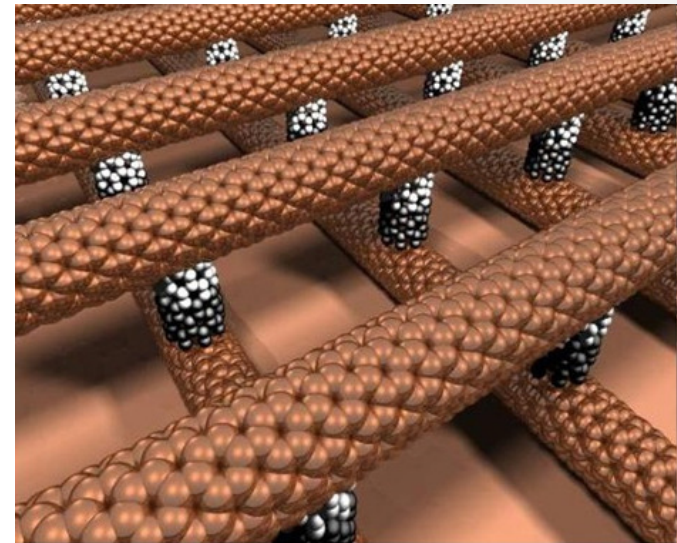
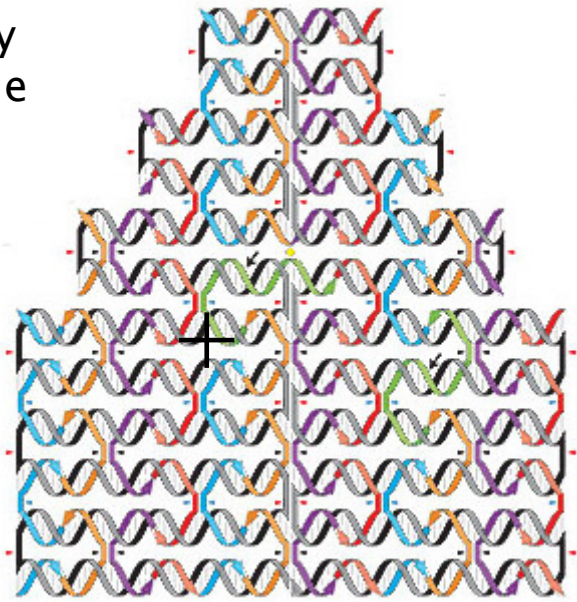
This means we can already self-assemble meso-scale structures.

DNA Circuit Boards

DNA-wrapped
nanotubes



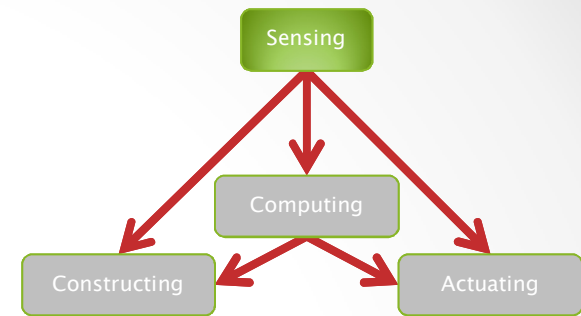
6 nm grid of
individually
addressable
pixels



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

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



Sensing

...

Aptamers

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

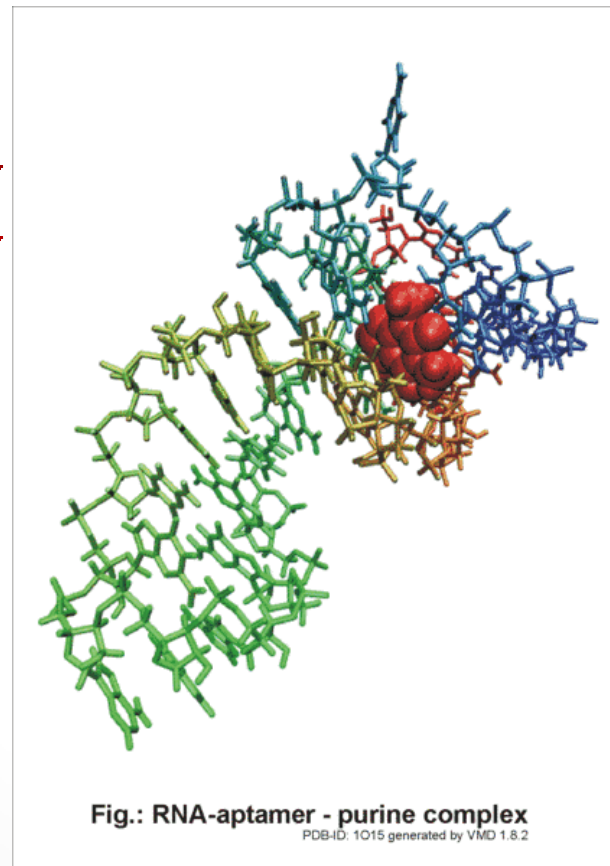
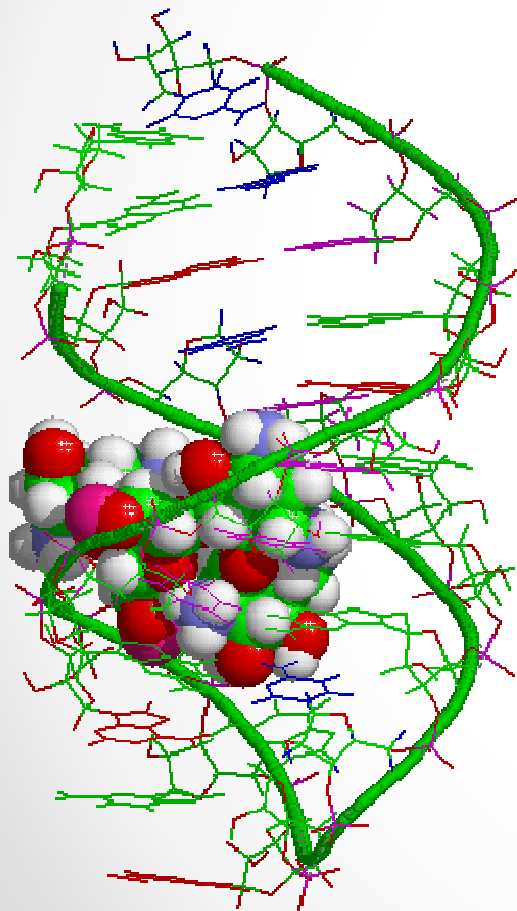
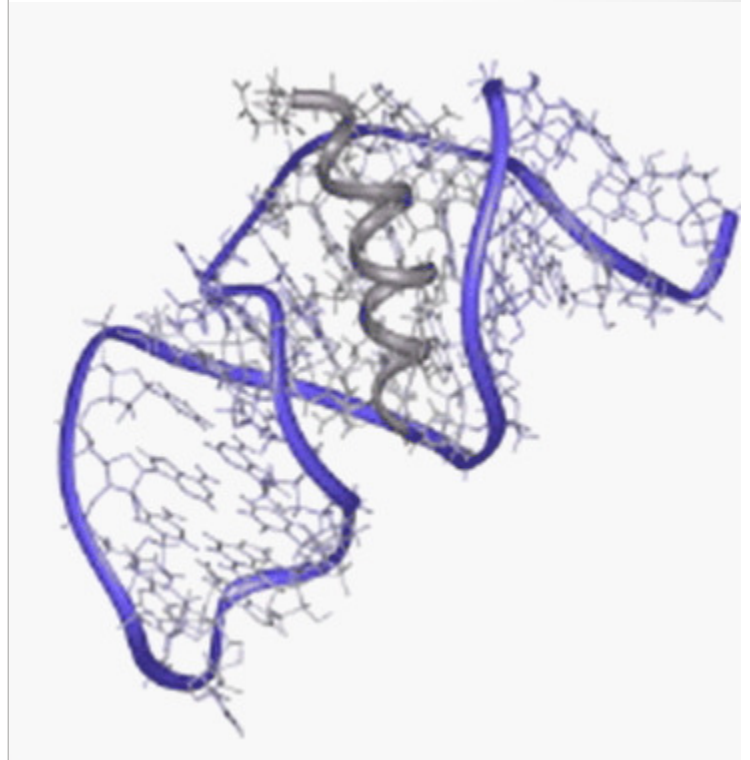


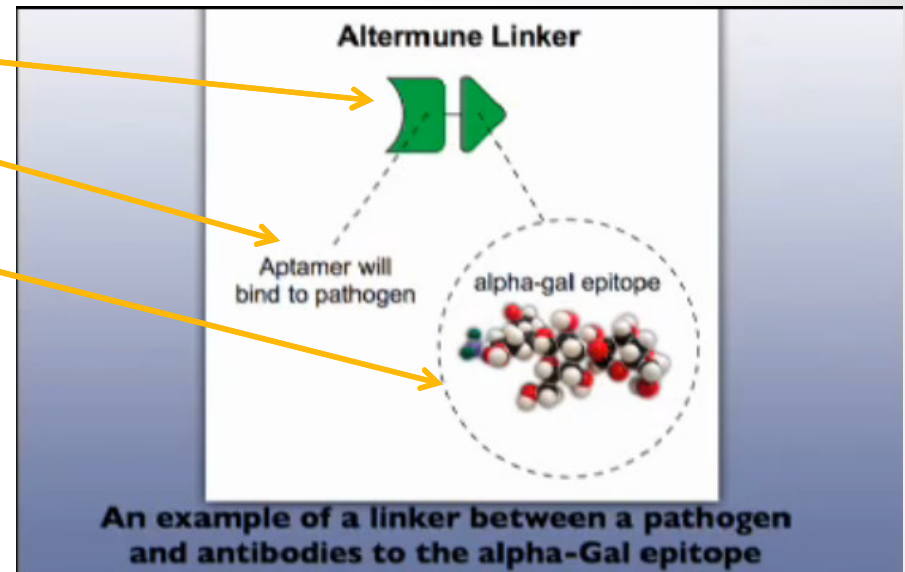
Fig.: RNA-aptamer - purine complex

PDB-ID: 1O15 generated by VMD 1.8.2



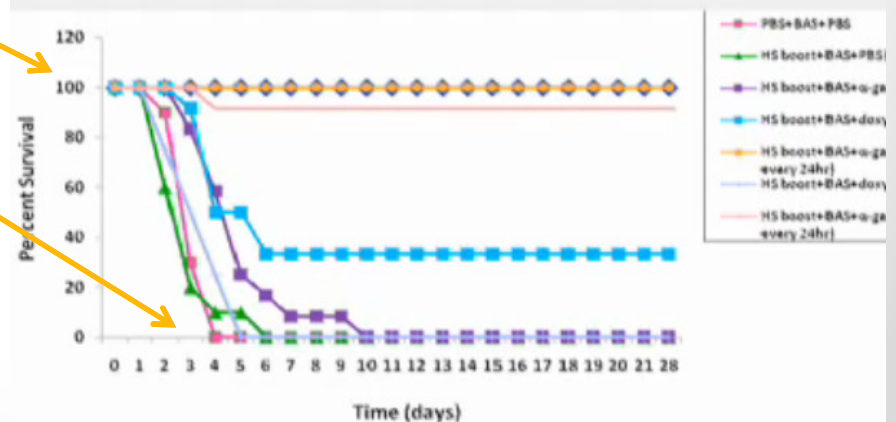
Pathogen Spotlights

- 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

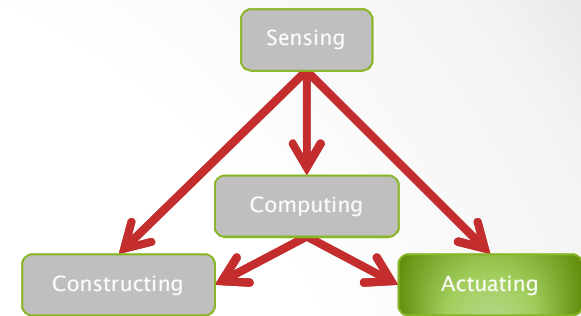


- Result: instant immunity
 - Mice poisoned with Anthrax plus aptamer (100% survival)
 - Mice poisoned with Anthrax (not so good)

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



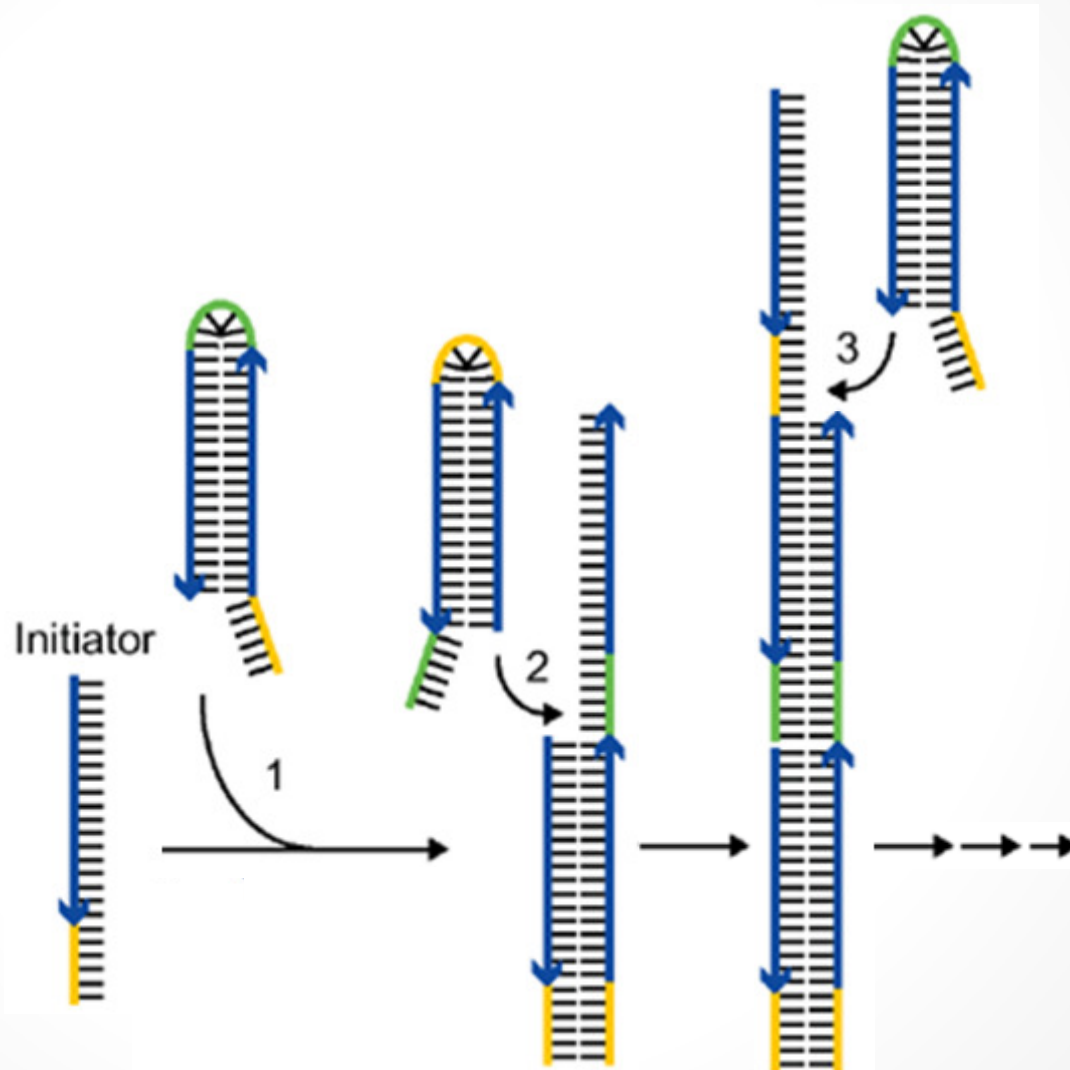
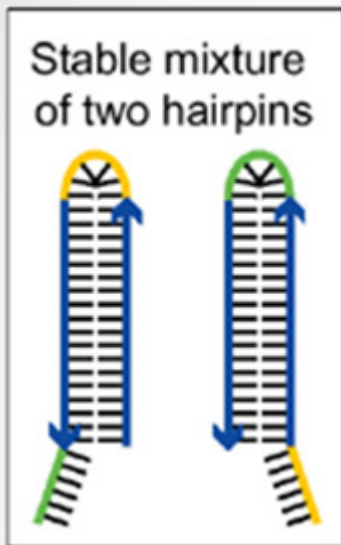
Kary Mullis (incidentally, also Nobel prize for inventing the Polymerase Chain Reaction)



Actuating

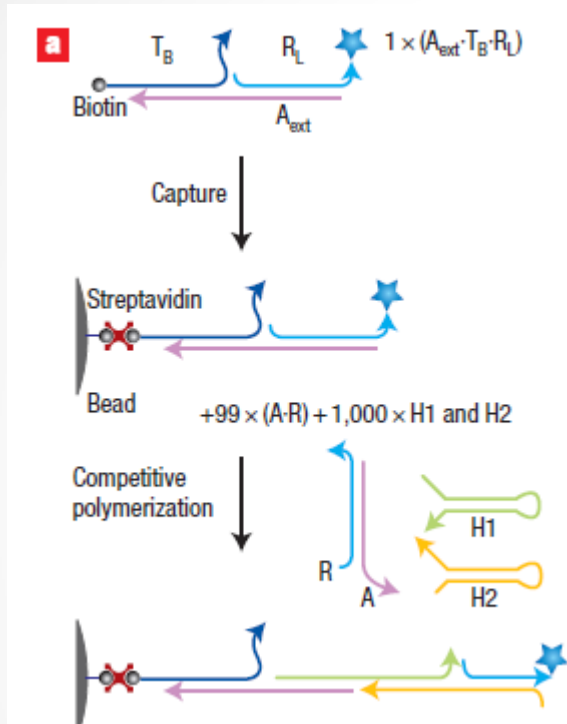
...

Hybridization Chain Reaction



chain reaction by hybridization

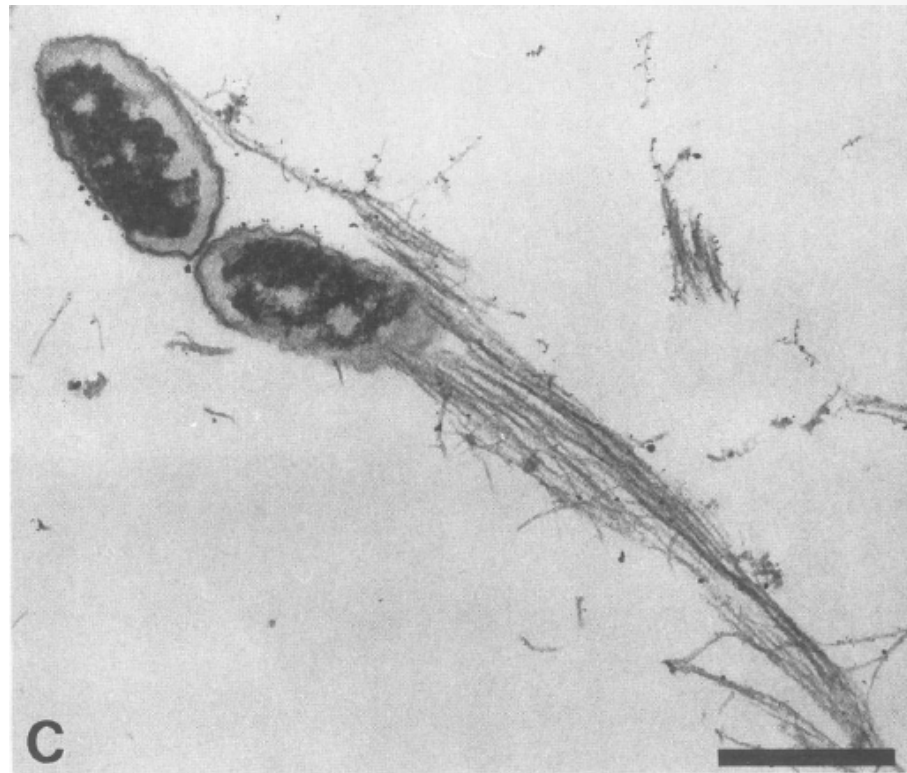
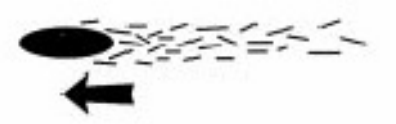
Polymerization Motor



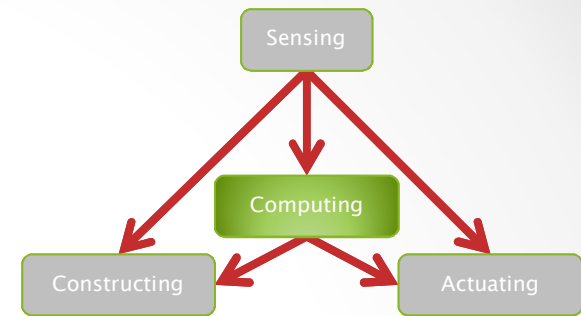
An autonomous polymerization motor powered by DNA hybridization

SUVIR VENKATARAMAN¹, ROBERT M. DIRKS¹, PAUL W. K. ROTHMUND^{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*



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.

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

Rules of the Game

- Short complementary segments hybridize reversibly

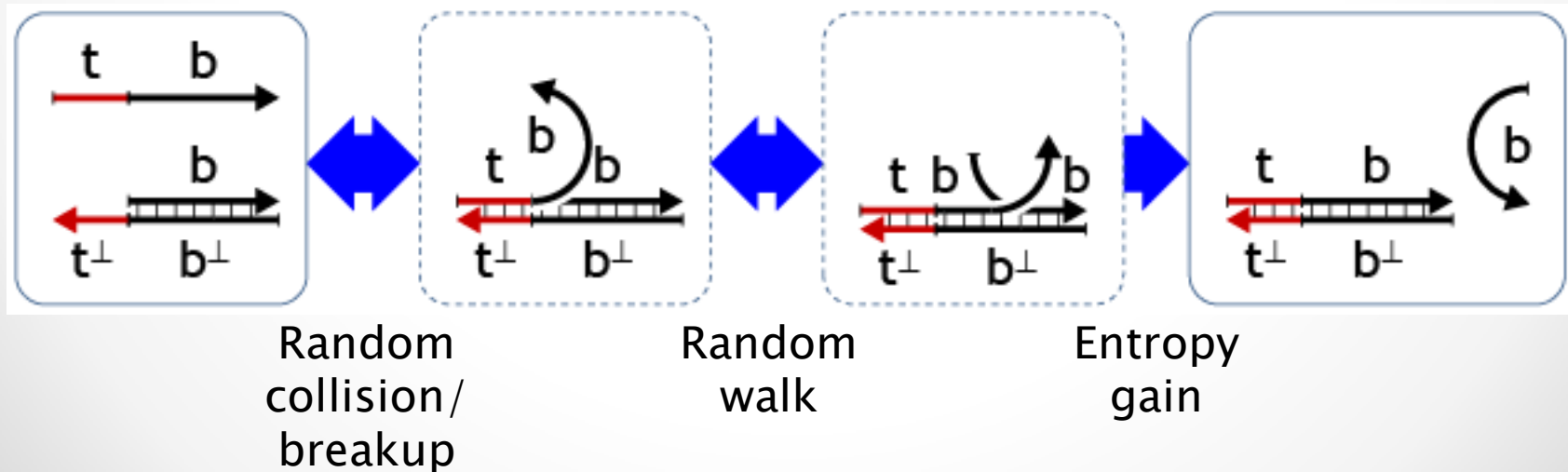
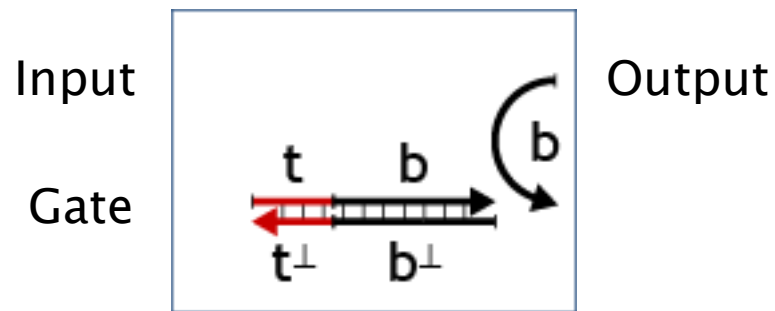


- Long complementary segments hybridize irreversibly



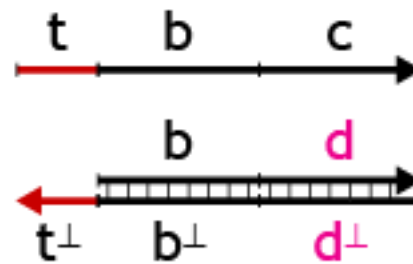
DNA Strand Displacement

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

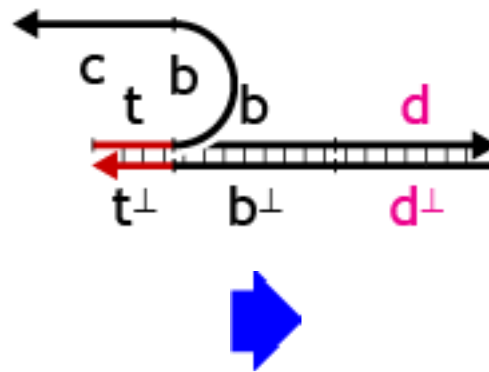


Failed Strand Displacement

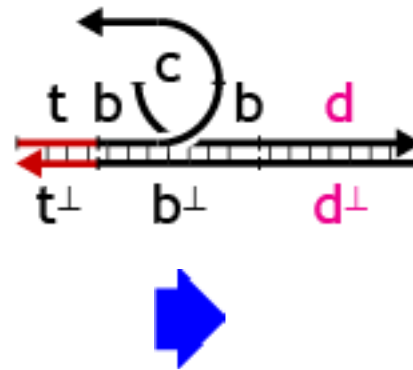
- What if the input does not match the gate?



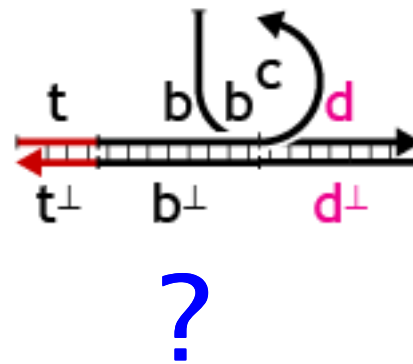
Failed Strand Displacement



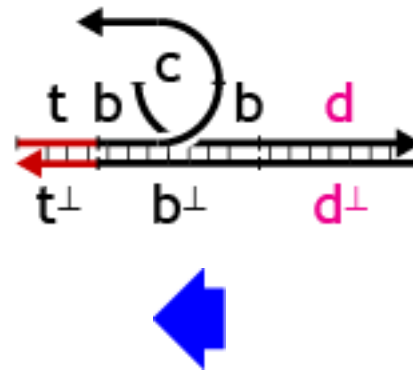
Failed Strand Displacement



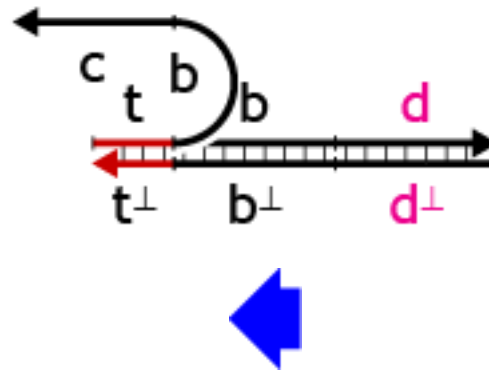
Failed Strand Displacement



Failed Strand Displacement

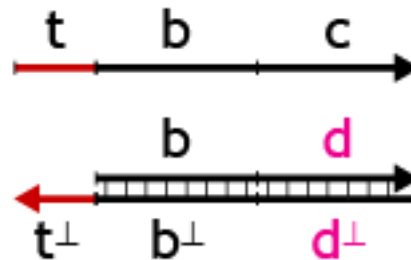


Failed Strand Displacement

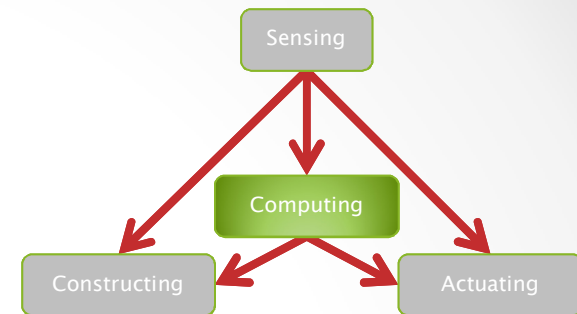


Failed Strand Displacement

- 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



Computing

...

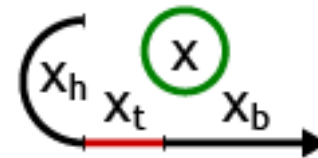
Implementing “Arbitrary”
Computing Functions

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

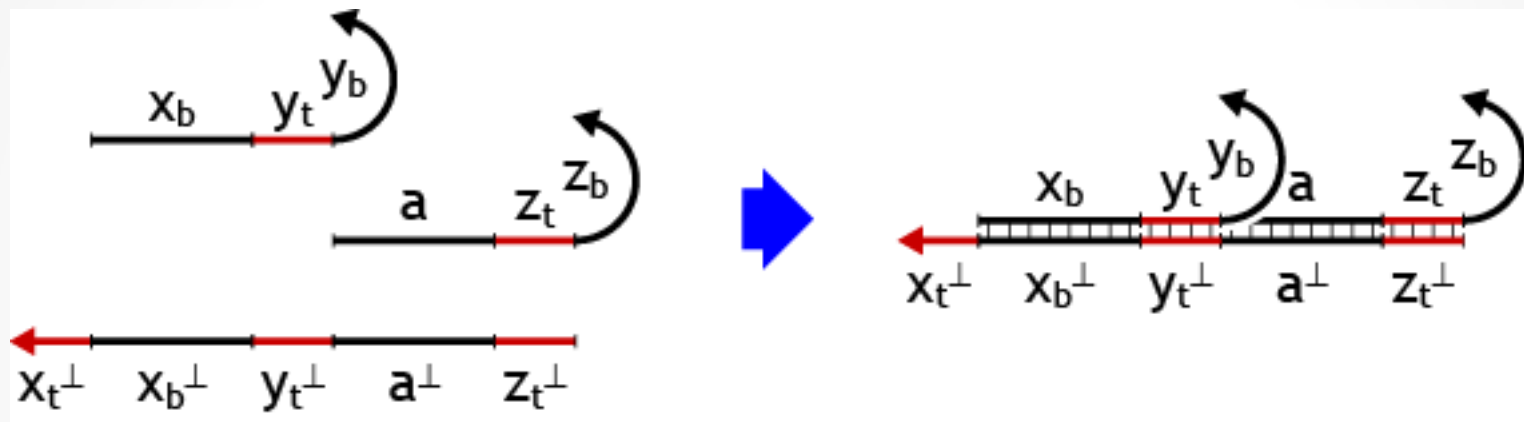
- x_h : hystory (ignore)
- x_t : toehold (binding)
- x_b : body (recognition)



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

Gates

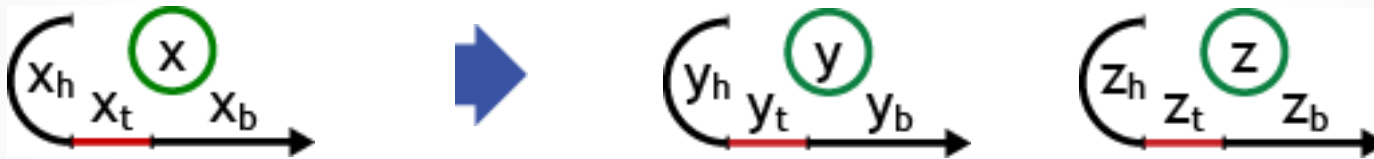
- Double-stranded structures with free toeholds



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

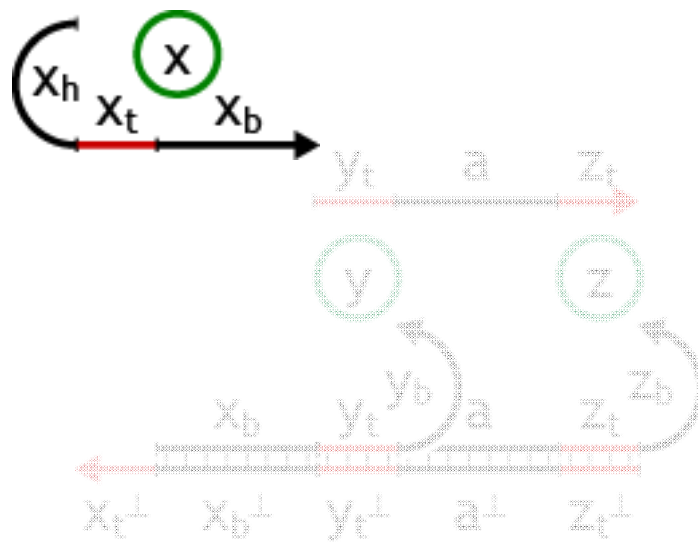
Fork Gate

- $x \rightarrow y + z$



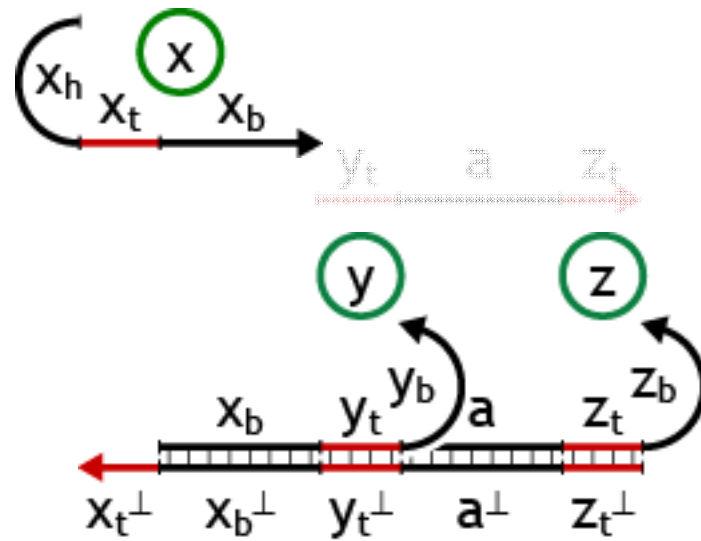
- $x \rightarrow y + 0$ transform x to y (transducer)
- $x \rightarrow x + y$ linear production of y (catalyst)
- $x \rightarrow x + x$ exponential production of x (amplifier)

Fork Gate

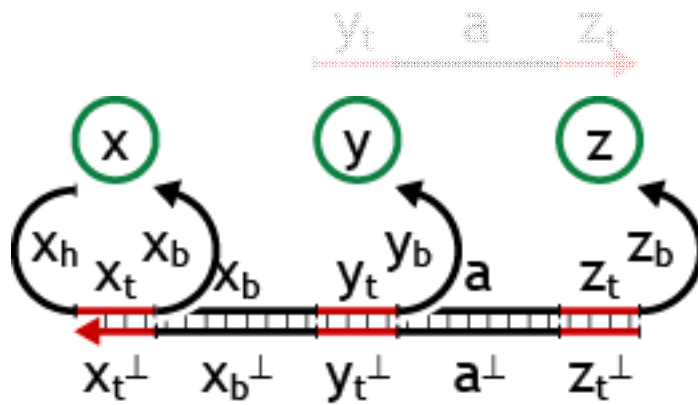


This is the
Fork Gate
structure

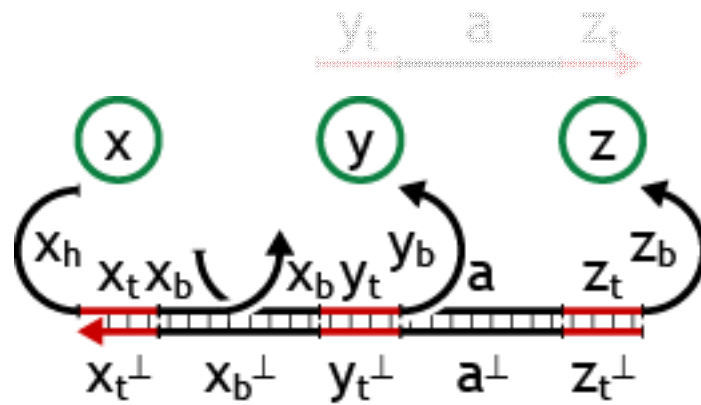
Fork Gate



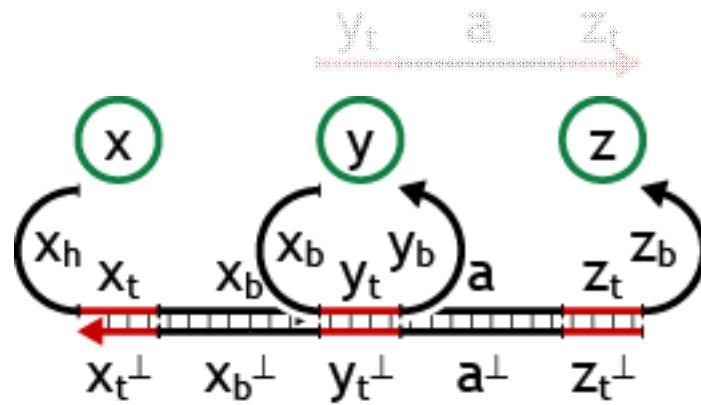
Fork Gate



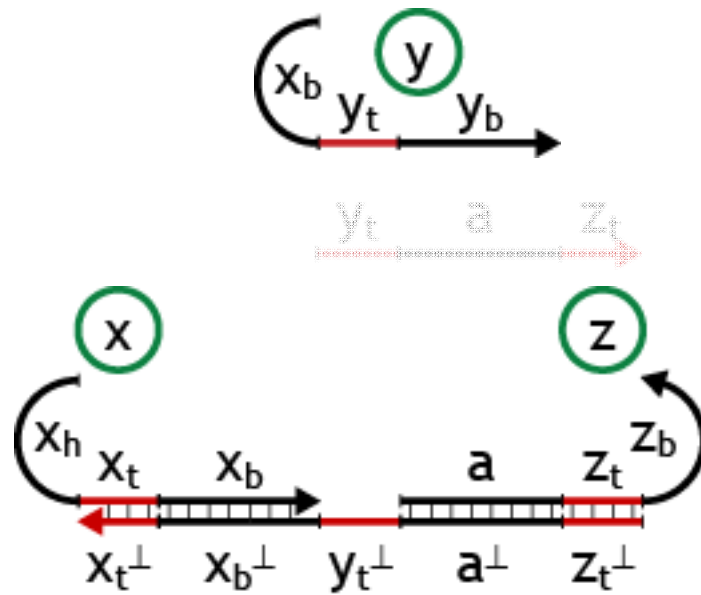
Fork Gate



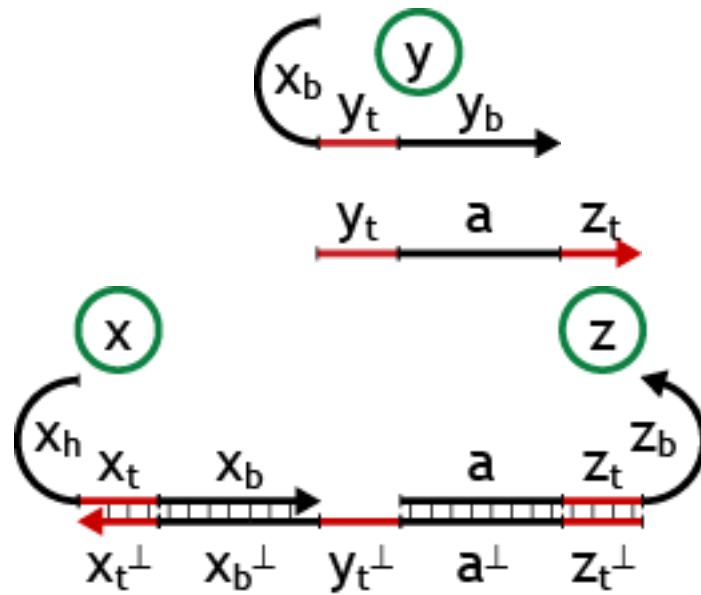
Fork Gate



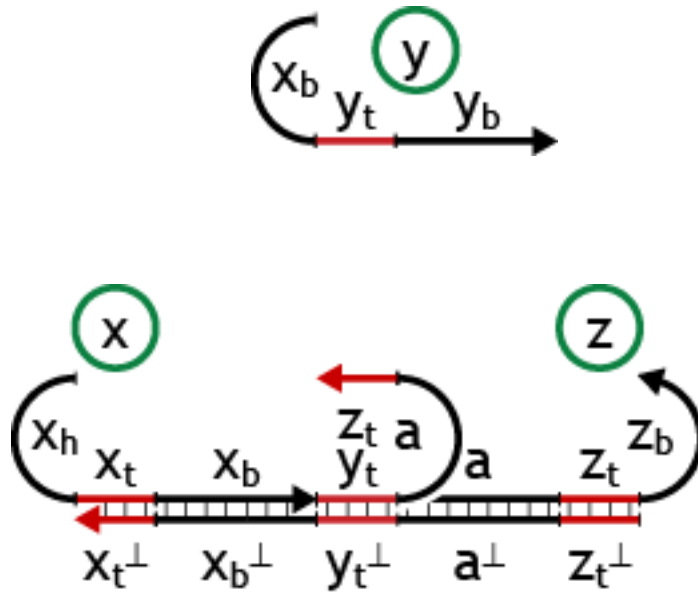
Fork Gate



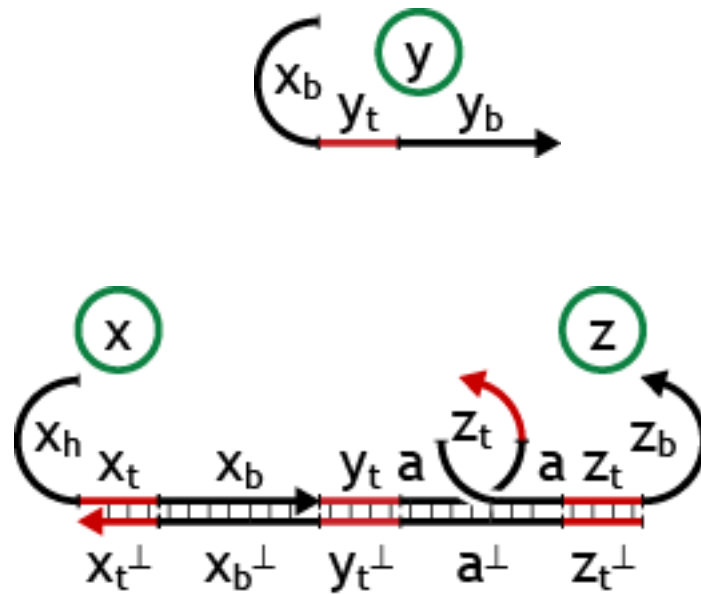
Fork Gate



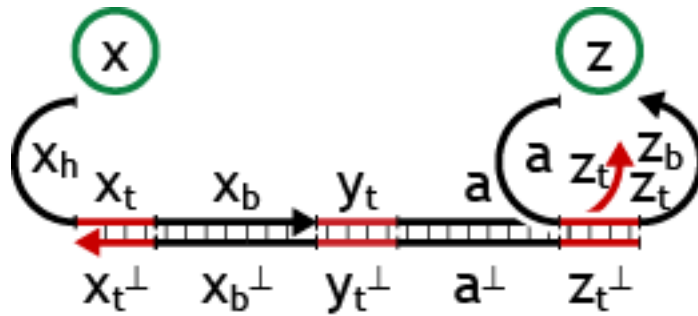
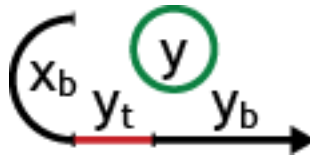
Fork Gate



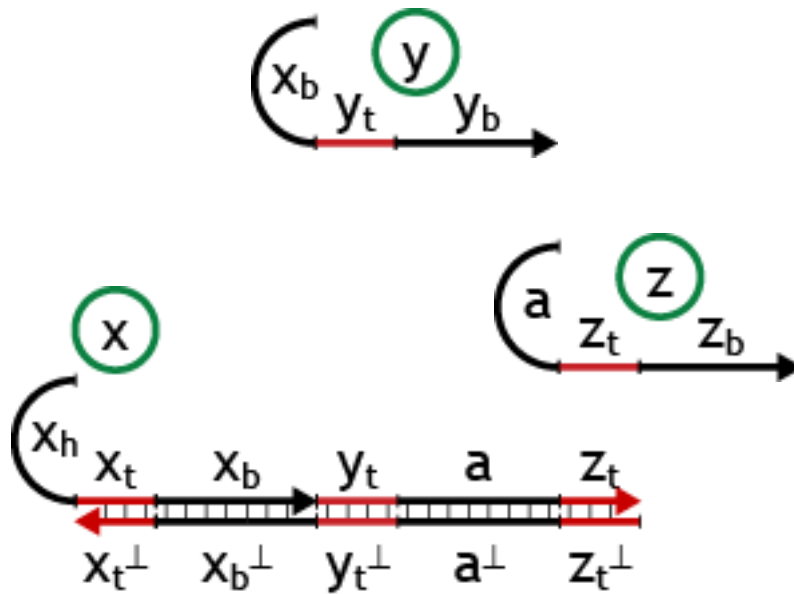
Fork Gate



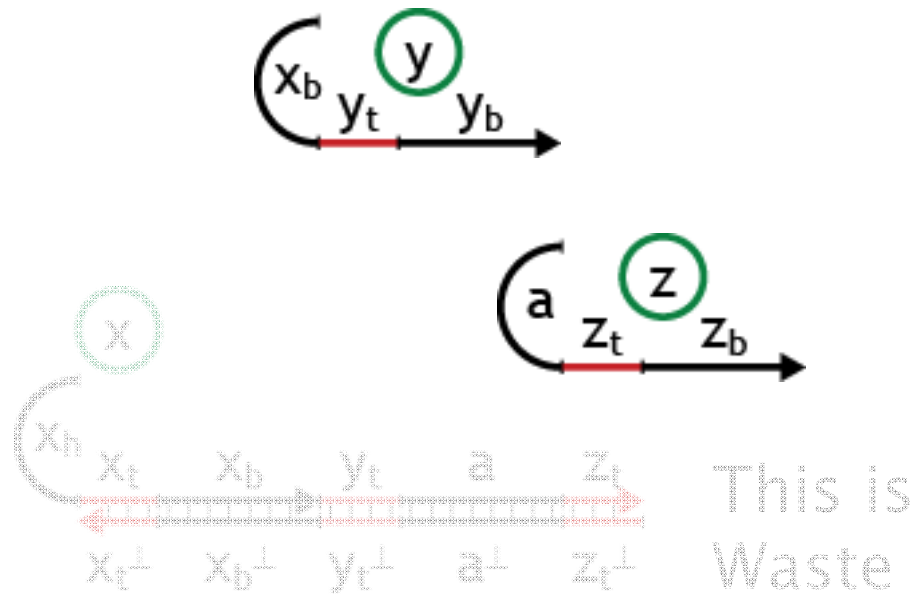
Fork Gate



Fork Gate

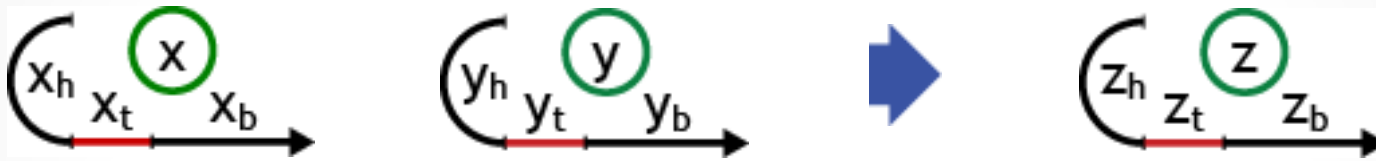


Fork Gate

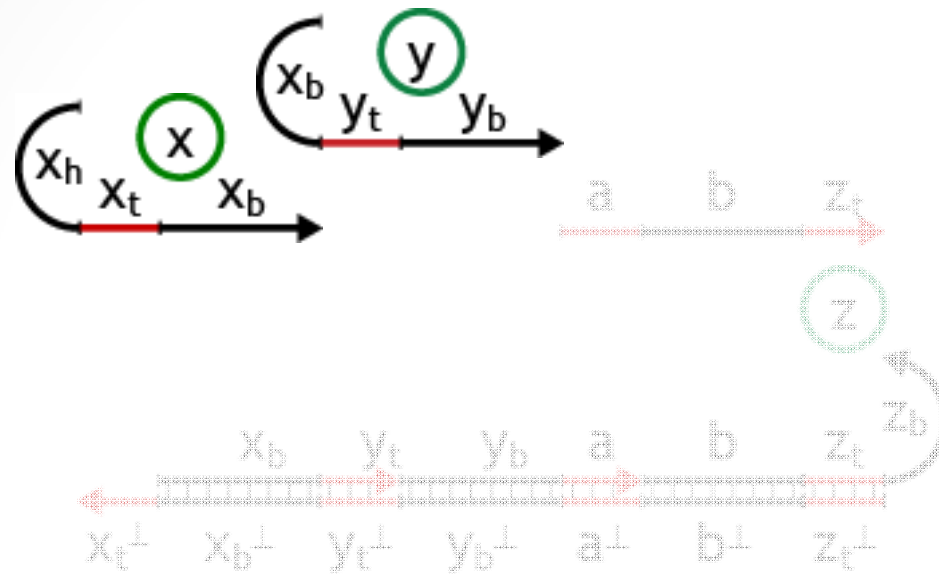


Join Gate

- $x + y \rightarrow z$

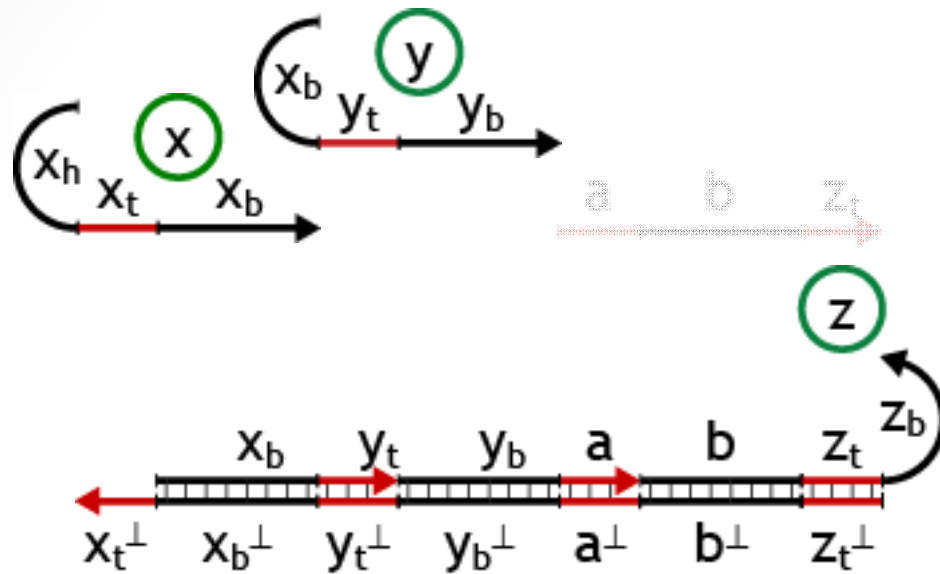


Join Gate

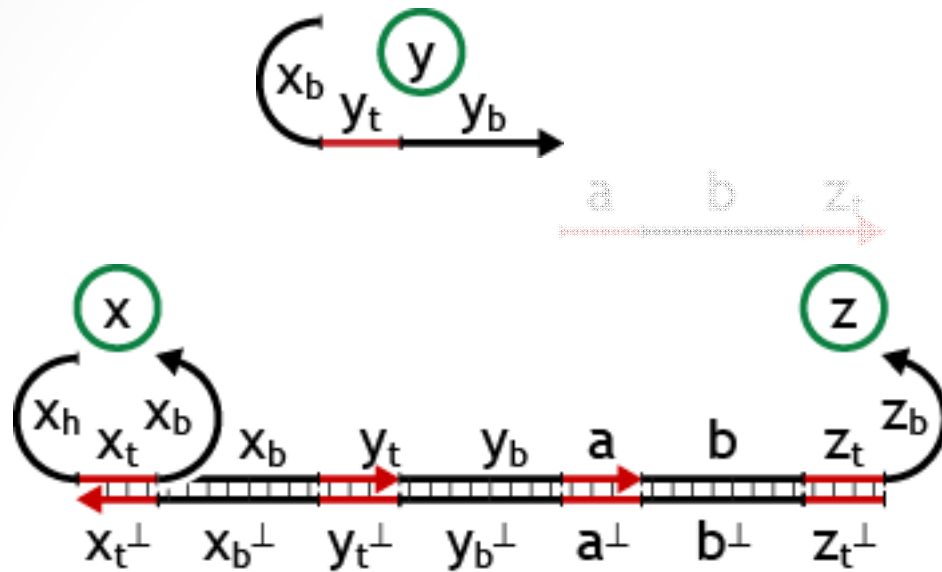


This is the
Join Gate
structure

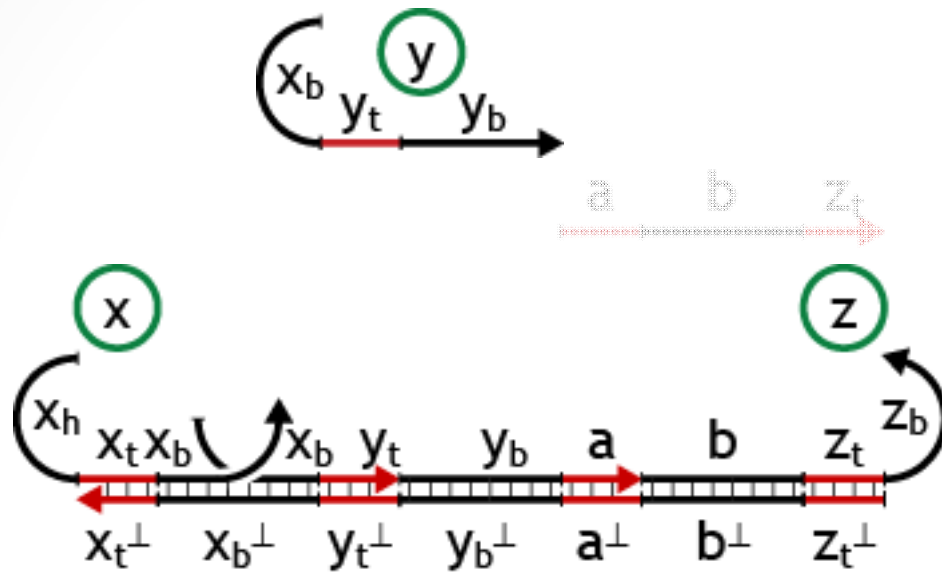
Join Gate



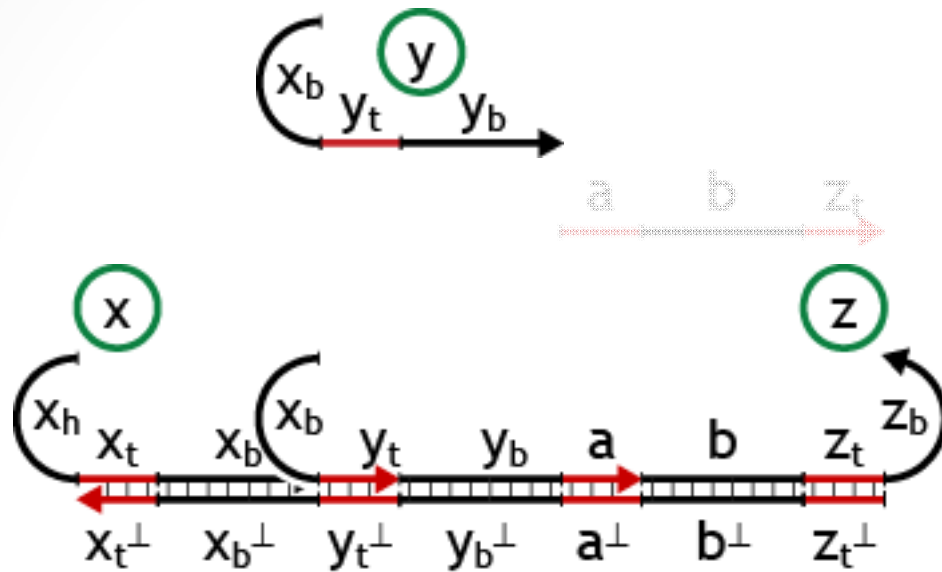
Join Gate



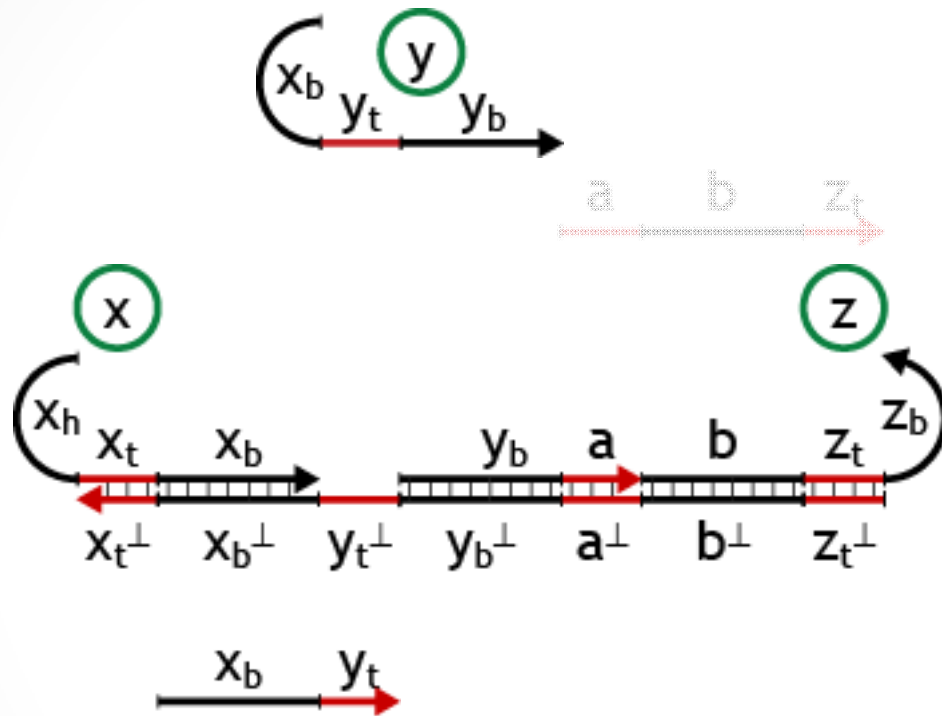
Join Gate



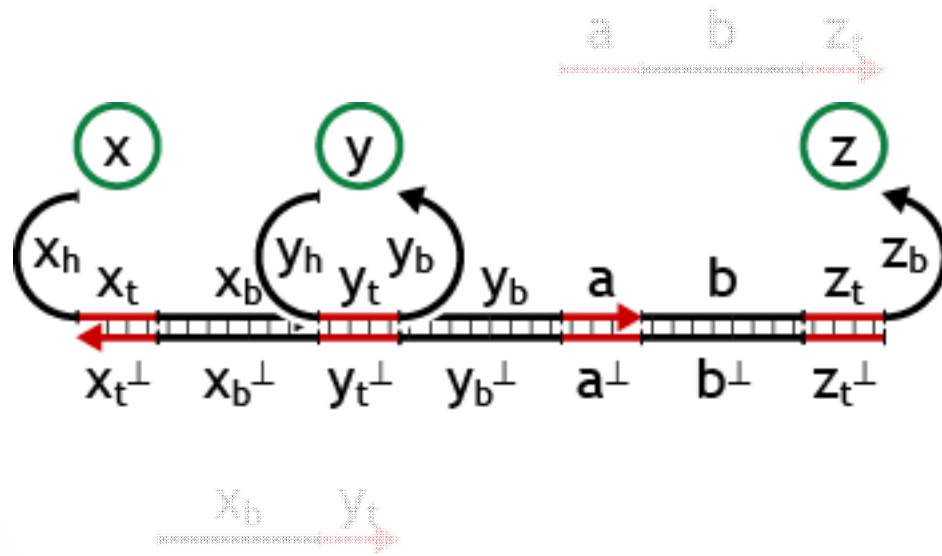
Join Gate



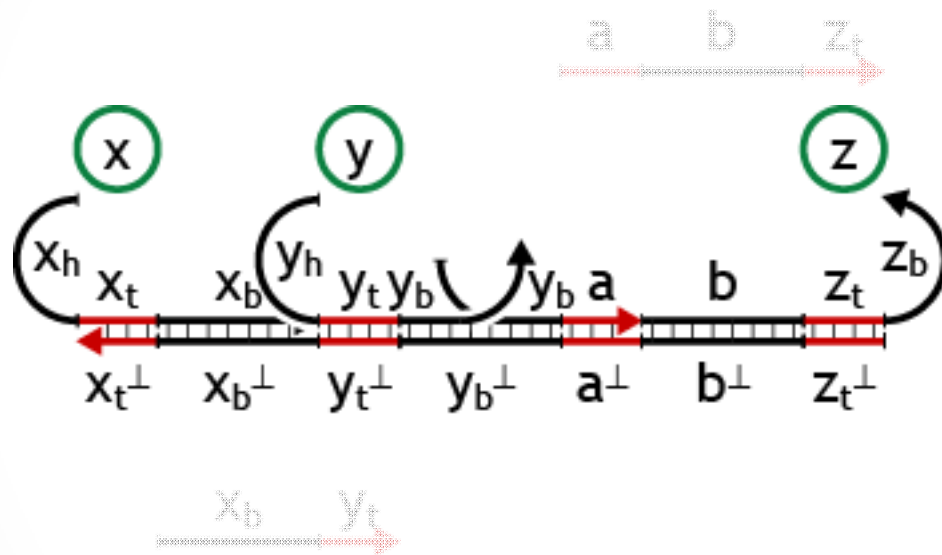
Join Gate



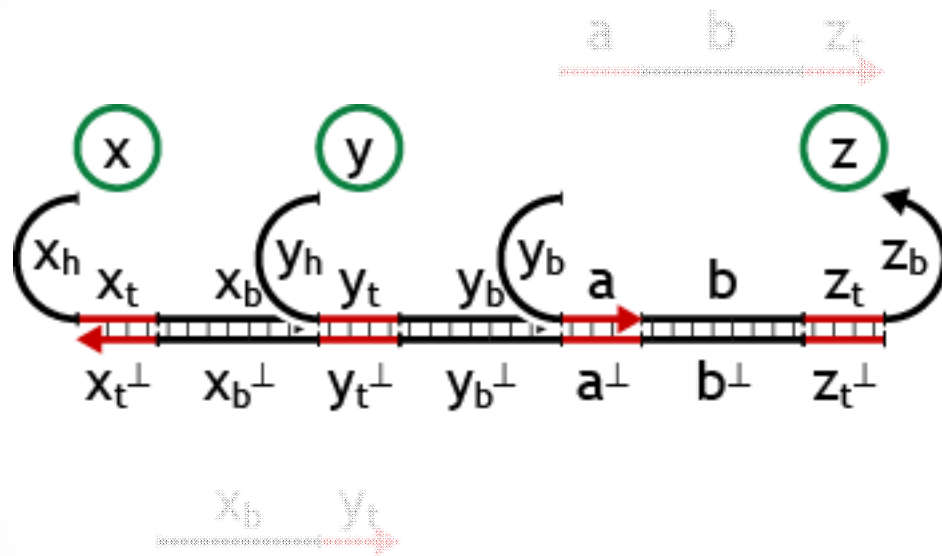
Join Gate



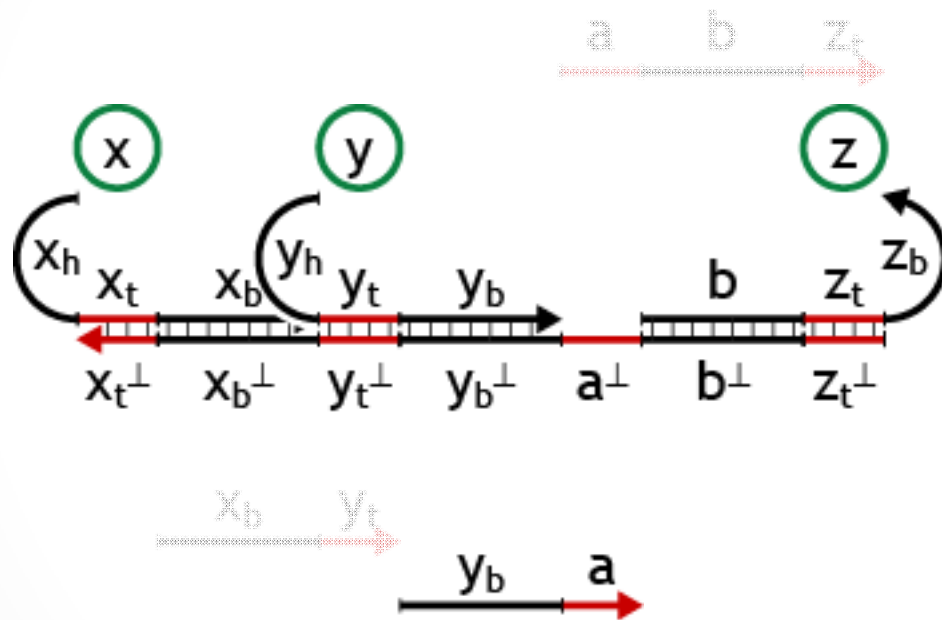
Join Gate



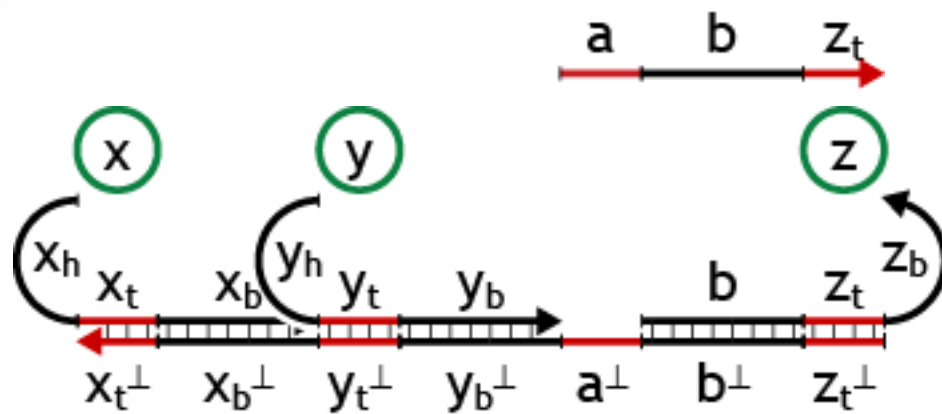
Join Gate



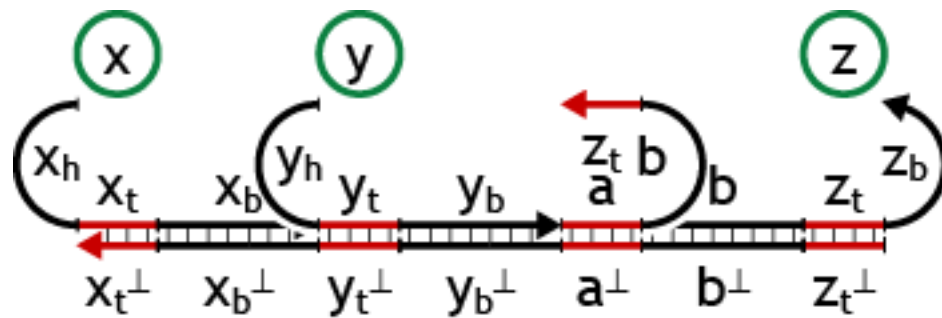
Join Gate



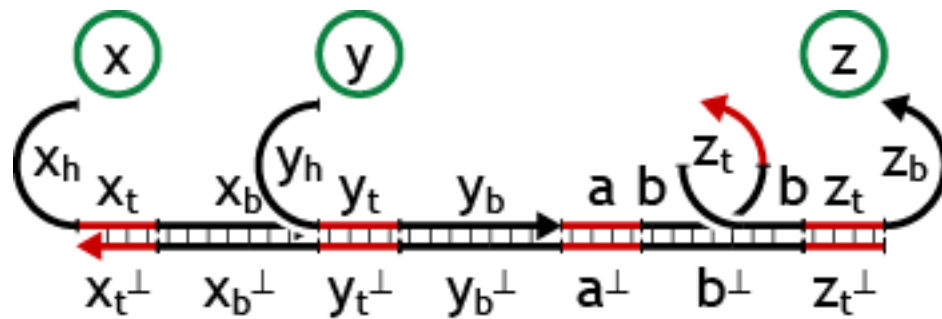
Join Gate



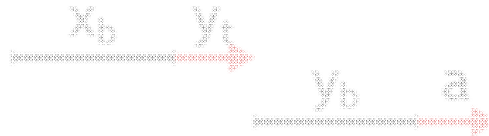
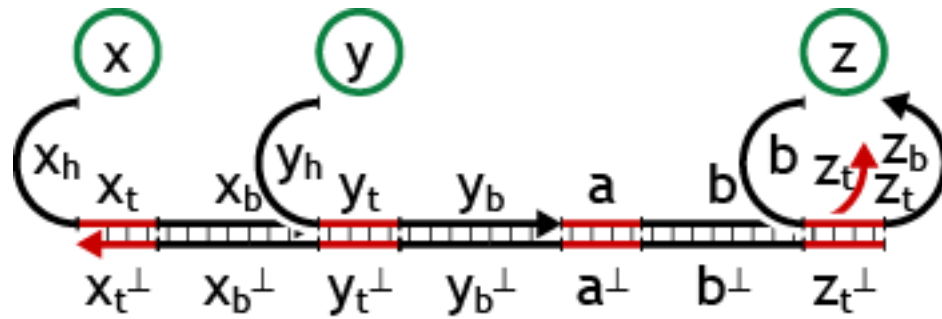
Join Gate



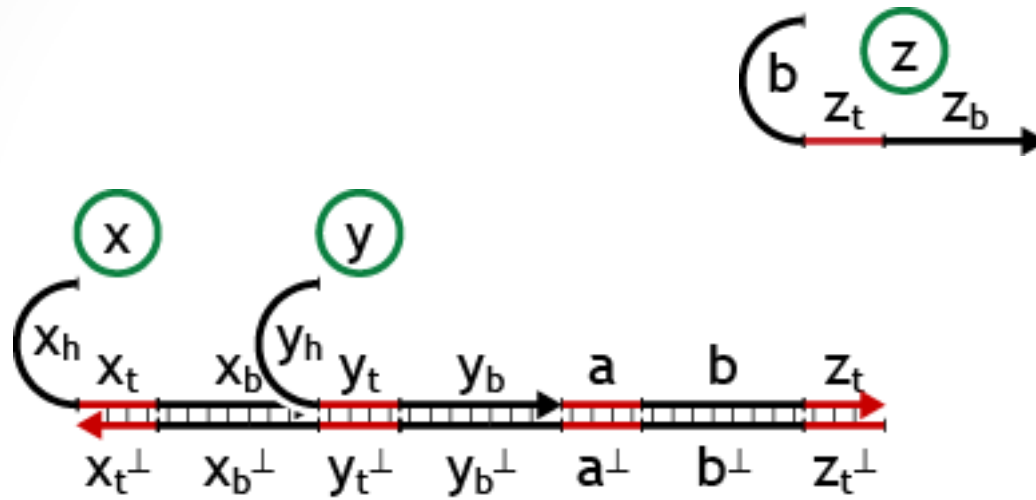
Join Gate



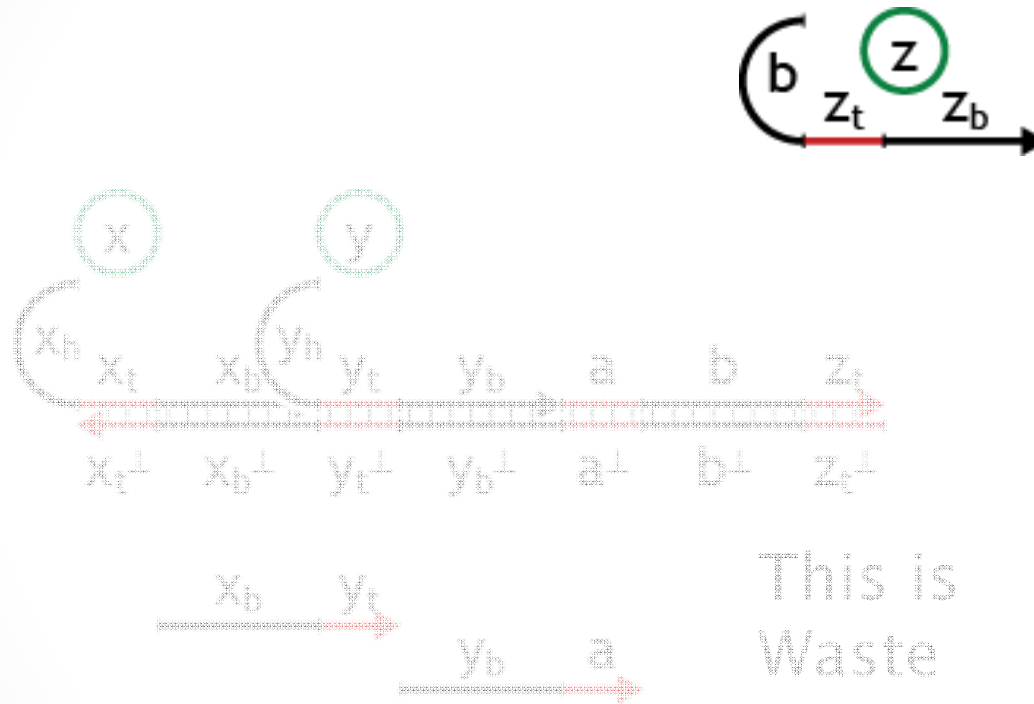
Join Gate



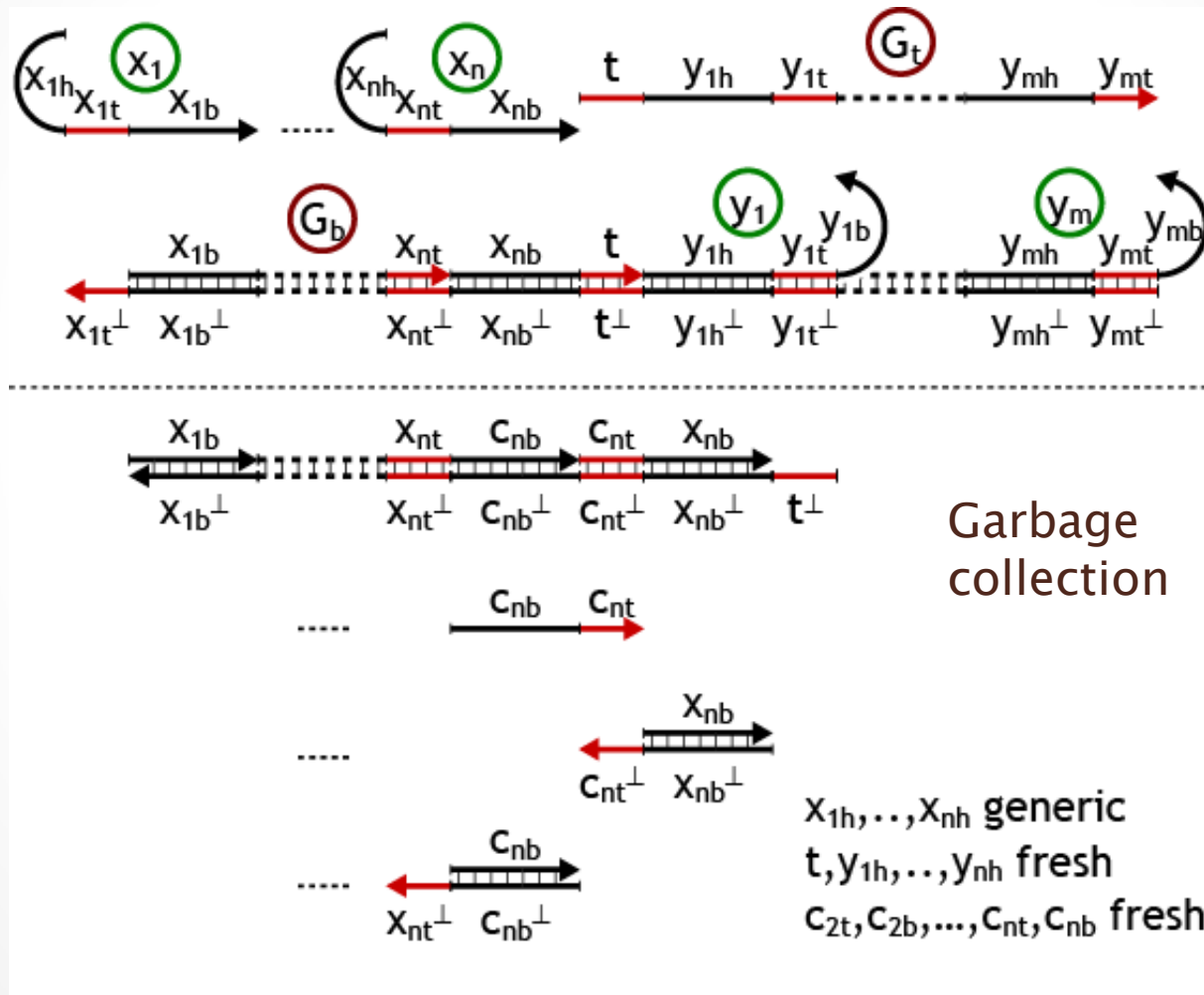
Join Gate



Join Gate



General n-Join/m-Fork Gate



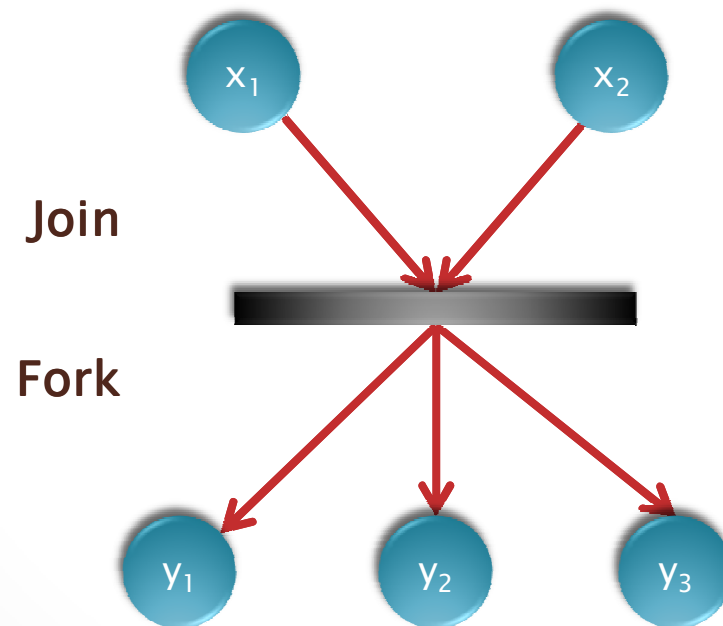
Gate Design Verification

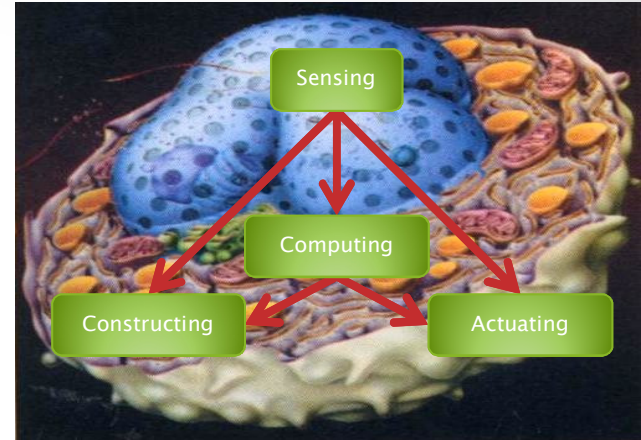
- Active garbage
 - The active join residuals slow down the performance of following joins.
 - → Add a garbage collector to remove the active residuals.
- Interference between gates
 - The join garbage collector interferes with the fork gate.
 - → Modify the fork gate to remove the interference.
- What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

Strand Algebra

$$x_1 \mid \dots \mid x_n \mid [x_1, \dots, x_n] \cdot [y_1, \dots, y_m] \rightarrow y_1 \mid \dots \mid y_m$$

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





Curing

...

A Doctor in Each Cell

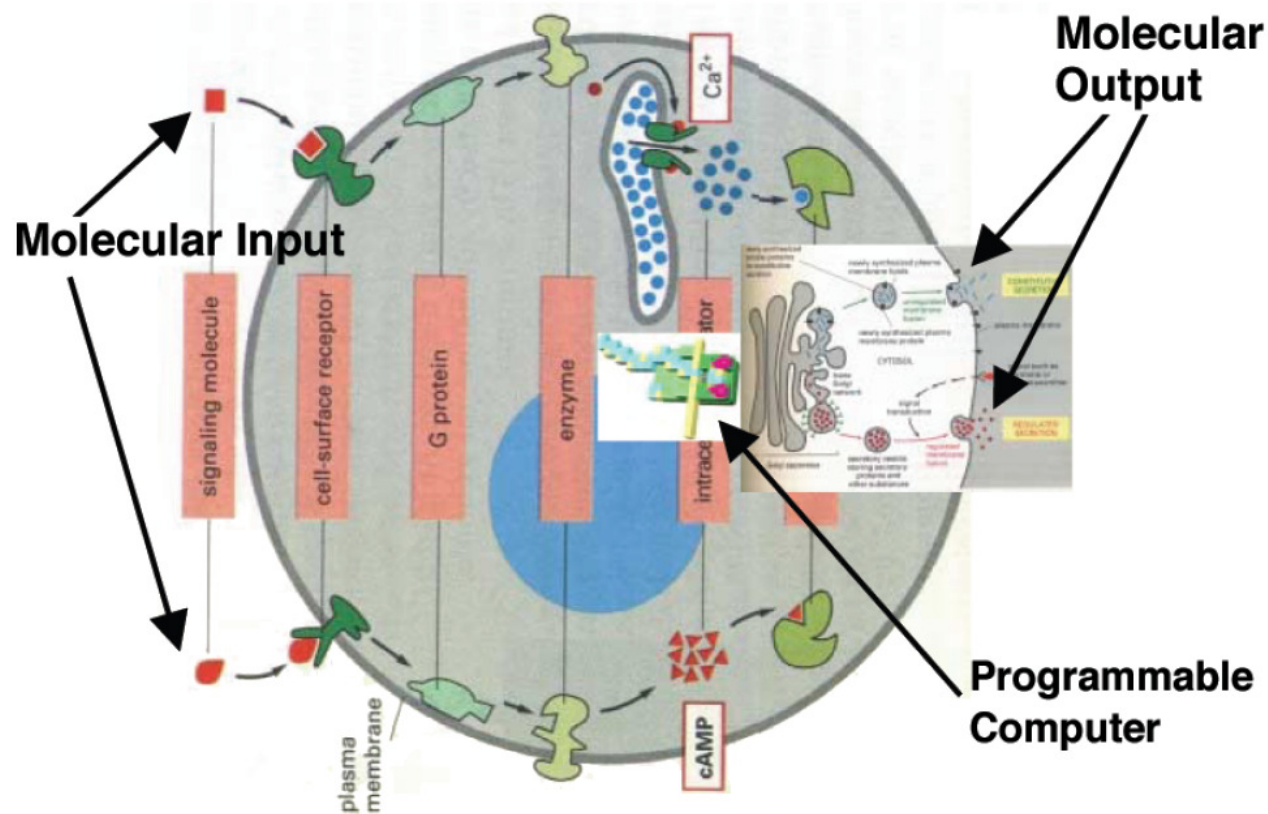


Fig. 1 Medicine in 2050: "Doctor in a Cell"

Ehud Shapiro

Rivka Adar
Kobi Benenson
Gregory Linshitz
Aviv Regev
William Silverman

**Molecules and
computation**

Tools

...



Sequence Design

NUPACK BETA
nucleic acid package

Analysis Design Downloads

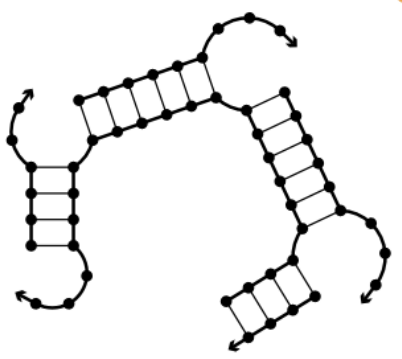
Input References Demos Help

Nucleic acid type: RNA DNA

Number of designs: 1

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

Preview:



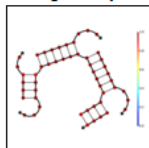
Input

NUPACK BETA
nucleic acid package

Analysis Design Downloads

Input Results References Demos Help

Designability summary



Sequence designs

Average percentage of correct nucleotides	Average number of incorrect nucleotides	GC content	Sequence
99.1%	0.475	74.5%	GCCUC+GCAAGCACC+GCC AGCUUG+GCUC+GAGCGCUG GCGCUUGC GCCGUG

Analyze

Output

Copyright © 2007-2009 Caltech. All rights reserved. | [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/>

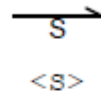
Syntax

A programming language for composable DNA circuits

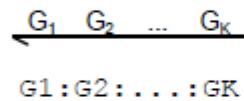
Andrew Phillips* and Luca Cardelli

A. Syntax of DNA molecules D

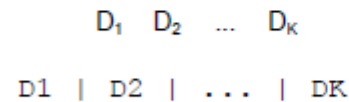
Upper strand with sequence complementary to S



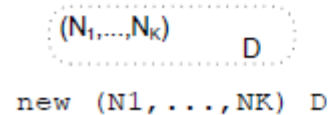
Molecule with segments G_1, \dots, G_k



Parallel molecules D_1, \dots, D_k



Molecules D with private domains N_1, \dots, N_k

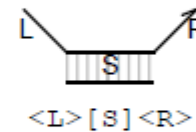


B. Syntax of DNA segments G

Lower strand with toehold N^c

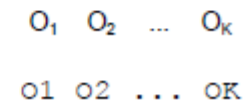


Double strand with sequence S and overhangs L, R

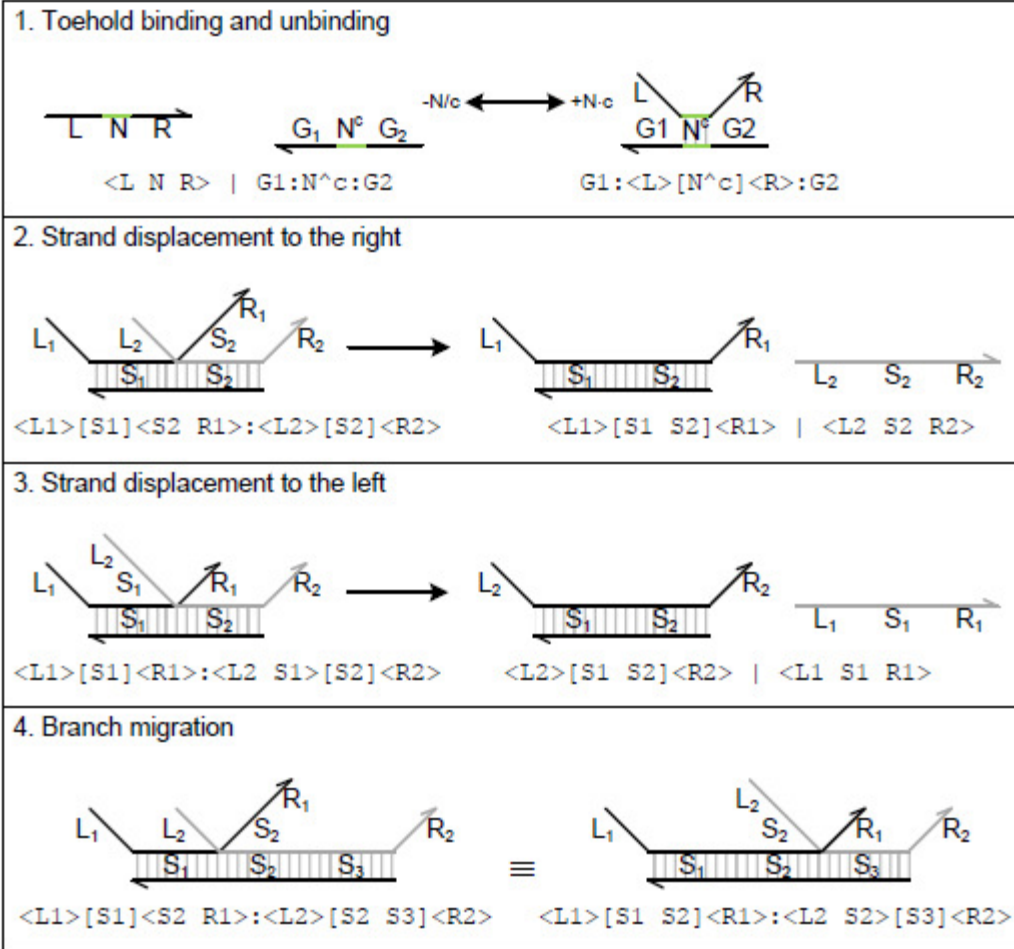


C. Syntax of DNA sequences S, L, R

Sequence of domains O_1, \dots, O_k



Dynamics



Initial Species

The screenshot displays the Visual DSD software interface. The title bar reads "Visual DSD - localhost". The main menu includes "Examples:", "Solve", "Simulate", and "Pause". Below the menu, there are settings for "Rules: Default", "Simulation: Stochastic", "View options:", and checkboxes for "Unproductive:", "Leaks:", and "Domains:". A version number "v0.12-20100224-1521" and an "Update" button are also present.

The interface is divided into two main panels. The left panel contains a diagram of a DNA strand with segments labeled 2, 3, 4, and 5. Segment 3 is highlighted in red, and segment 5 is highlighted in green. The right panel is currently empty, with a zoom level of 109% and buttons for "Fit" and "Save...".

The "Initial" tab is highlighted with a red circle, indicating the current view.

Reaction Graph

Visual DSD - localhost

Examples: Rules: Default Simulation: Stochastic View options: Unproductive: Leaks: Domains: v0.12-20100224-1521

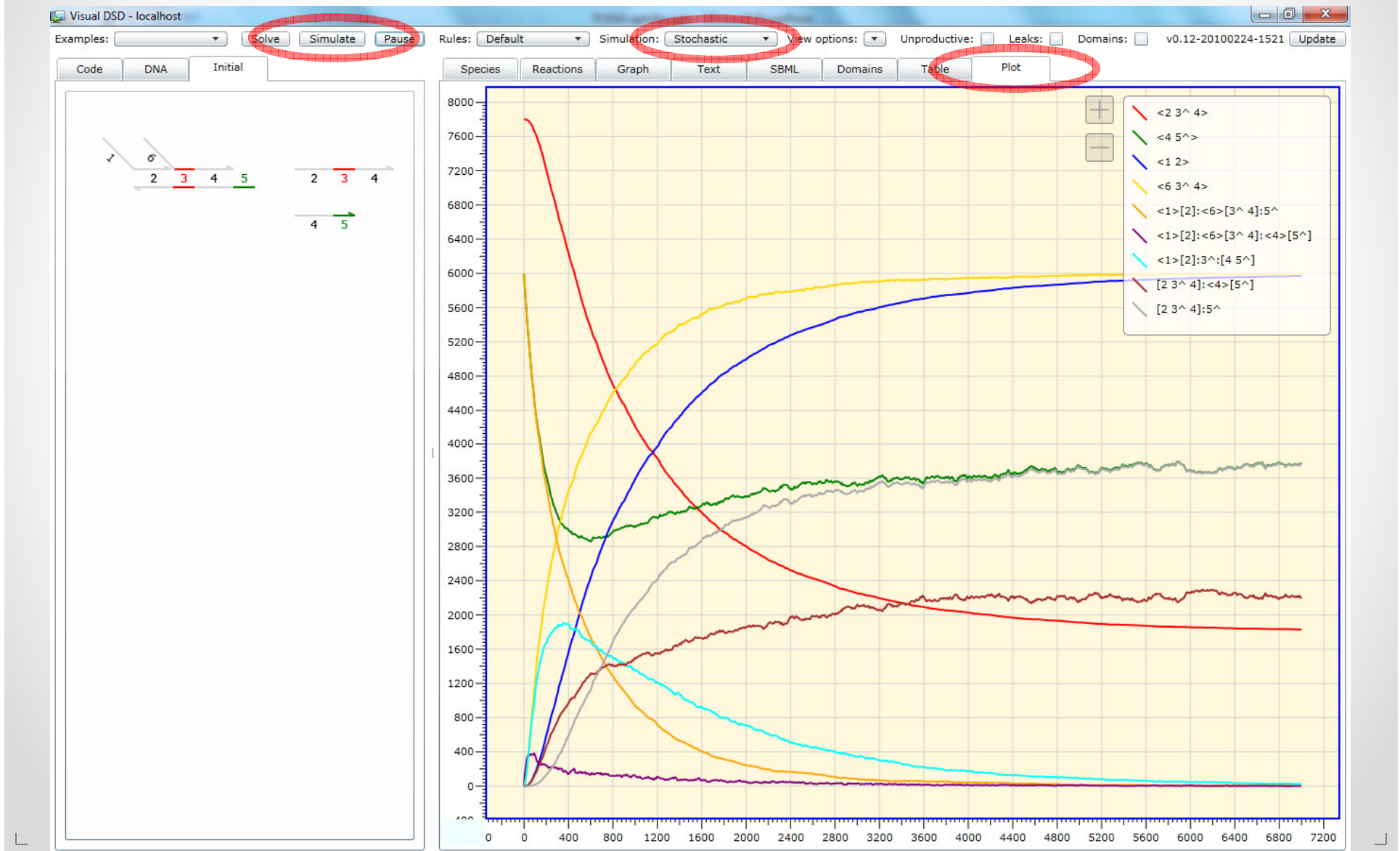
Code DNA Initial

Species Reactions **Graph** Text SBML Domains Table Plot

Zoom 109% Fit Save...

The screenshot shows the Visual DSD software interface. The top menu bar contains 'Solve' and 'Simulate' buttons, which are circled in red. Below the menu are tabs for 'Species', 'Reactions', 'Graph', and 'Text', with 'Graph' selected. The main window displays a reaction graph with nodes containing chemical structures and arrows. The graph shows a sequence of reactions starting from a root node at the top, branching out to various intermediate states, and finally leading to a terminal state at the bottom right. The nodes are connected by arrows, indicating the flow of the reaction sequence. The chemical structures in the nodes consist of horizontal lines with numbers (1-6) and colored arrows (red, green, black) indicating the state of the system. The zoom slider is set to 109%.

Simulation



Abstract Reactions

Visual DSD - localhost

Examples: Solve Simulate Pause Rules: **Infinite** Simulation: Stochastic View options: Unproductive: Leaks: Domains: v0.12-20100224-1521 Update

Code DNA Initial

Species Reactions Graph Text SBML Domains Table Plot

Zoom 150% Fit Save...

The diagram illustrates a reaction network starting from a root node (a DNA strand with segments 2, 3, 4, 5). The network branches into several nodes, some of which are circled in black:

- Root node: DNA strand with segments 2, 3, 4, 5.
- Node 1: DNA strand with segments 6, 3, 4.
- Node 2: DNA strand with segments 2, 3, 4.
- Node 3: DNA strand with segments 2, 3, 4, 5.
- Node 4: DNA strand with segments 4, 5.
- Node 5: DNA strand with segments 1, 2.

The nodes are connected by arrows, indicating the direction of the reactions. The root node is connected to Node 1, Node 2, and Node 3. Node 2 is connected to Node 4 and Node 5. Node 3 is connected to Node 5.

Detailed Reactions

Visual DSD - localhost

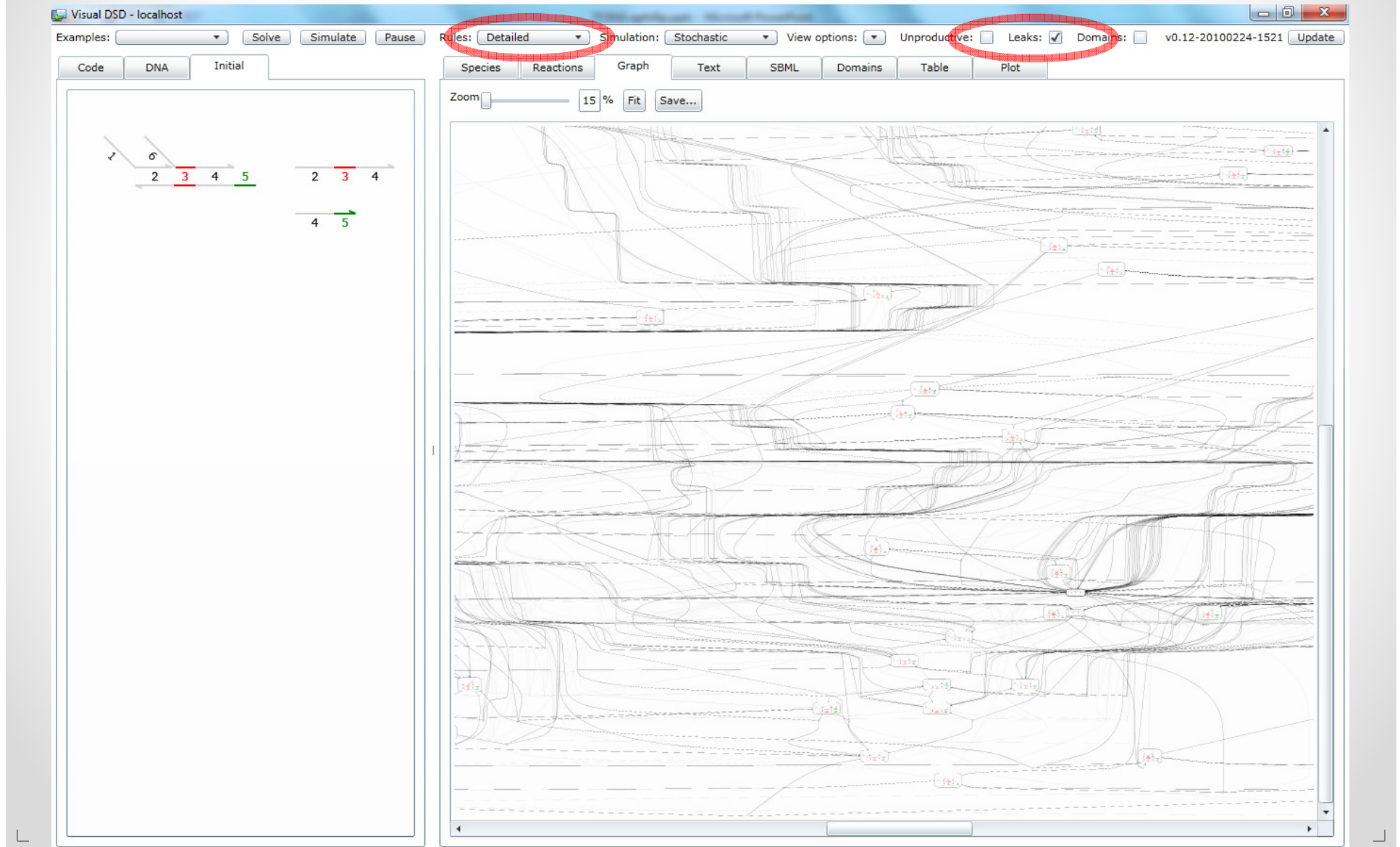
Examples: Solve Simulate Pause Rules: **Detailed** Simulation: Stochastic View options: Unproductive: Leaks: Domains: v0.12-20100224-1521 Update

Code DNA Initial Species Reactions Graph Text SBML Domains Table Plot

Zoom 64% Fit Save...

The diagram illustrates a reaction network starting from a DNA strand with segments 1, 2, 3, 4, 5. The network branches out into various states, including configurations like '2 3 4', '4 5', and '1 2'. The 'Reactions' tab is selected, and the 'Graph' view shows a hierarchical tree of reaction states.

Detailed Leak Reactions!



Just-in-Time Simulation

Visual DSD - localhost

Examples: Solve Simulate Pause Rules: **Detailed** Simulation: **JIT** View options: Unproductive: Leaks: Domains: v0.12-20100224-1521 Update

Code DNA Initial

Species Reactions Graph Text SBML Domains Table Plot

Zoom 38% Fit Save...

The screenshot shows the Visual DSD software interface. At the top, the title bar reads "Visual DSD - localhost". Below it, a toolbar contains several buttons: "Examples:", "Solve", "Simulate", "Pause", "Rules:", "Simulation:", "View options:", "Unproductive:", "Leaks:", "Domains:", and "Update". The "Rules:" dropdown is set to "Detailed" and the "Simulation:" dropdown is set to "JIT", both of which are circled in red. To the right of these dropdowns, there are checkboxes for "Unproductive:" (unchecked), "Leaks:" (checked), and "Domains:" (unchecked). Below the toolbar, there are tabs for "Code", "DNA", and "Initial". The main window is divided into two panes. The left pane shows DNA sequence diagrams with colored segments (red, green, black) and arrows indicating interactions. The right pane shows a complex reaction network graph with nodes and edges, and a zoom level of 38%. The graph contains many nodes, each representing a DNA sequence with colored segments, and edges representing interactions between these sequences. The nodes are arranged in a hierarchical structure, with some nodes at the top and others at the bottom. The edges connect these nodes, forming a dense network. The zoom level is 38%, and there are "Fit" and "Save..." buttons above the graph.

DNA Sequences

The screenshot displays the Visual DSD software interface. At the top, the title bar reads "Visual DSD - localhost". Below it, a control bar includes buttons for "Solve", "Simulate", and "Pause", along with dropdown menus for "Rules: Default" and "Simulation: Stochastic". There are also checkboxes for "Unproductive", "Leaks", and "Domains", and a version number "v0.12-20100302-1033" with an "Update" button.

The main interface is divided into several sections:

- Code**: A tabbed interface with "DNA" selected. It contains a "Check sequences" checkbox (checked) and a "Reset" button.
- TOEHOLD SEQUENCES**: A list of 16 DNA sequences:

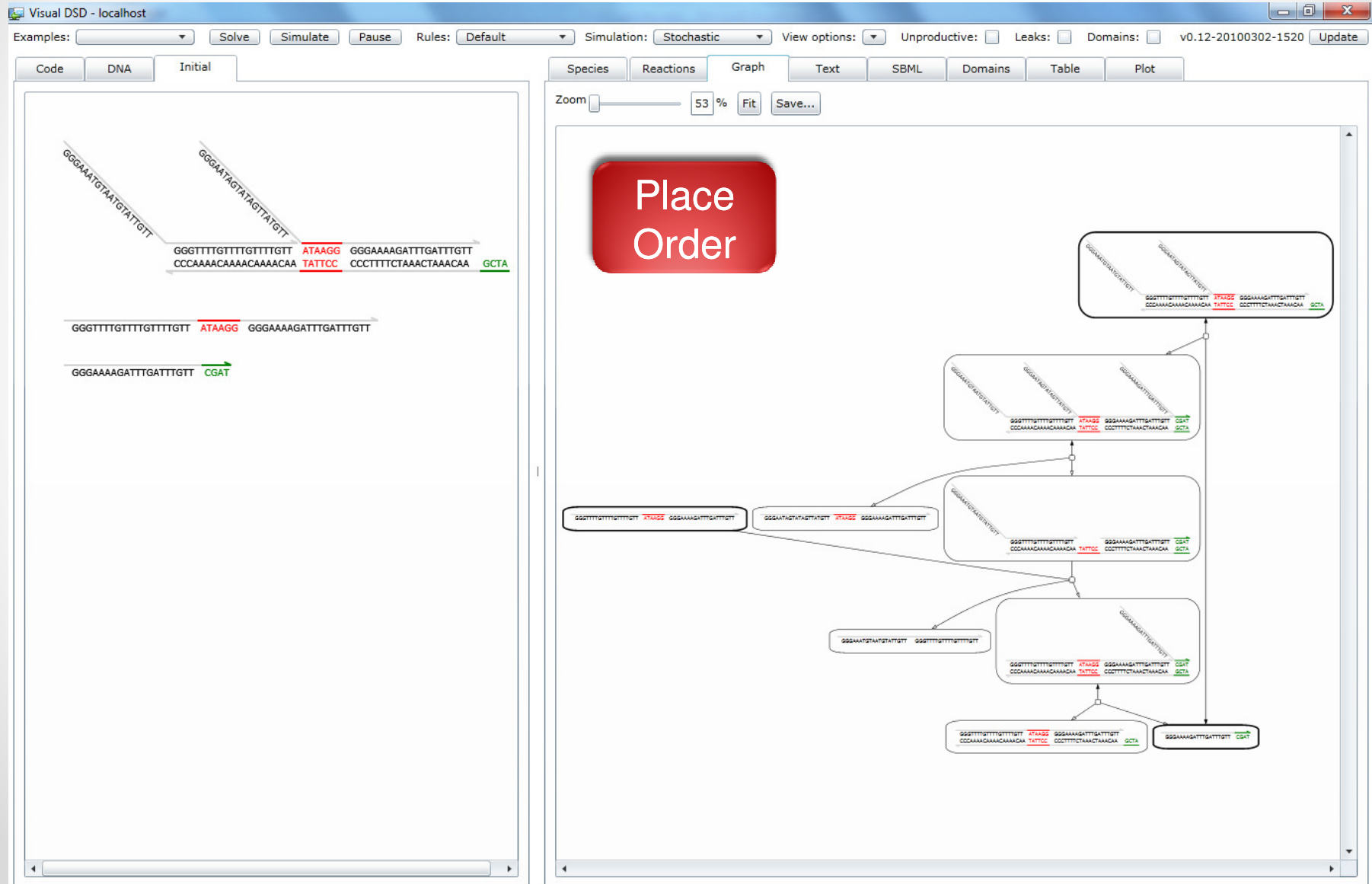
```
TATTCC
GCTA
GTCA
TACCAA
CATCG
ACTACAC
CTCAG
CTCAATC
CCTACG
TCTCCA
CCCT
GACA
ACCT
TAGCCA
CACACA
AGAC
```
- SPECIFICITY SEQUENCES**: A scrollable list of 24 DNA sequences:

```
CCCAAAACAAAACAAAACAA
CCCTTTTCTAACTAAACAA
CCCTTTACATTACATAACAA
CCCTTATCATATCAATACAA
CCCTTAACTTAACAAATCTA
CCCTATTCAATTCAAATCAA
CCCTATACTATACAATACTA
CCCTAATCTAATCATAACTA
CCCTAAACTTATCTAAACAT
CCCATTTCAAATCAAACCTT
CCCATTAATAATCAATTCAA
CCCATATCTATACATTACAA
CCATAACTATTCTAAACTA
CCCAATTCTTAACATATCAA
CCCAATACTATTATAACAT
CCCAAATCTTAACATATACTA
CCTATACCTTAACCTTAACAA
CCATATCCATAACTTTTACAA
CCATAACCTATACTTATCAA
CCATTTCCCTTTCTTAACCTA
CCATTACCATATCTTATCAT
CCAAAACCATAAACATAACTT
```
- Species**: A tabbed interface with "SEML" selected. It contains a "Zoom" slider and a list of sequences:

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

The "SEML" tab is highlighted with a red dashed circle. The "Table" and "Plot" tabs are also visible.

Final DNA Circuit



Next-Day Oligos!

XX-IDT
INTEGRATED DNA
TECHNOLOGIES

Chat is now closed.
Please click to email
a representative.

[Logout]
Spain

Items: €1,00

Home Products Order Support Services SciTools Search Go

SameDay® Oligo Service

Only € 1,44 EUR / Base!

The Current Time is 22:40 (GMT)

Specifications:

- Order online 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](#)

© Copyright 2010 Integrated DNA Technologies, Inc

Place Order NOW!

XX IDT
INTEGRATED DNA TECHNOLOGIES

Chat is now closed.
Please click to email a representative.

[LogIn]
Spain

0 Items € 0,00

Home Products Order Support Services SciTools Search Go

Order Oligos

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

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

25 nmole DNA oligo Purification: Standard Desalting

Sequence Name # Bases: 21

5'-ACT GCA CCA TAA GCA ACT TTT 3'

Notes: Enter your notes here. Please do not enter modifications

ADD TO ORDER
ADD TO WISH LIST

Help 5' mods Internal Mods 3' mods Services Mixed Bases

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

Customized Labels (more detail)
Stock IDT Label FREE

DNA by Mail



It runs!

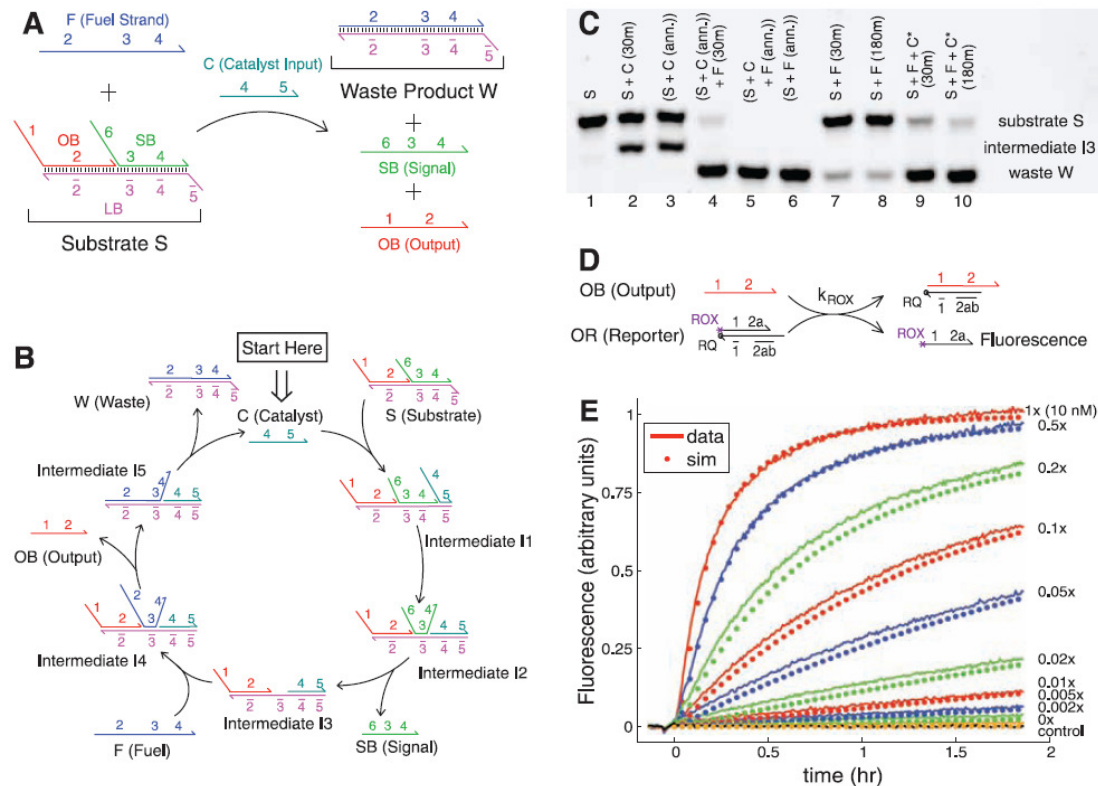
- (Nothing to do with us.)

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

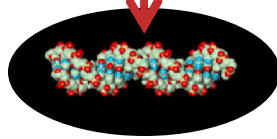
Monolithic
Compilers



Language
Design #1

Boolean
Networks

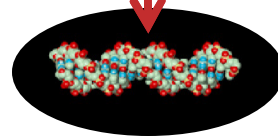
Language
Implementation #1



Language
Design #2

Petri
Nets

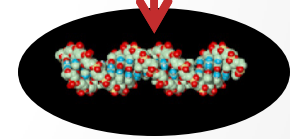
Language
Implementation #2



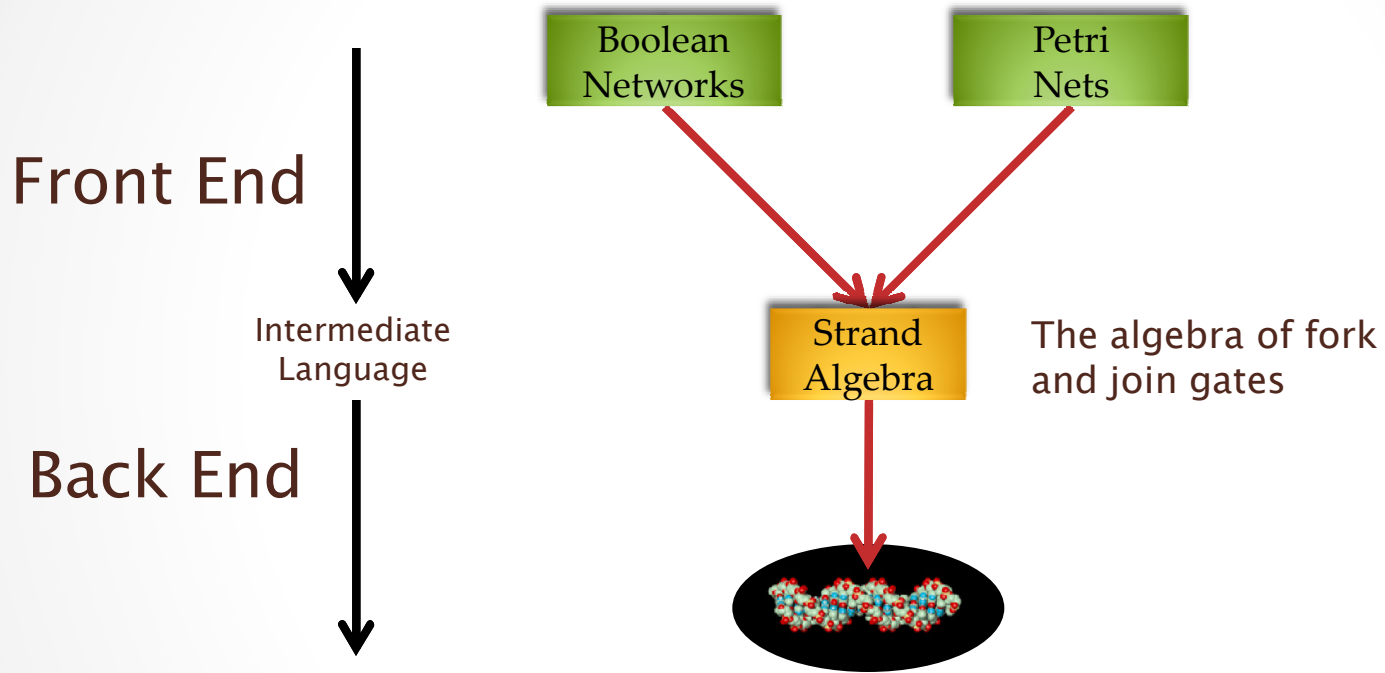
Language
Design #3

...

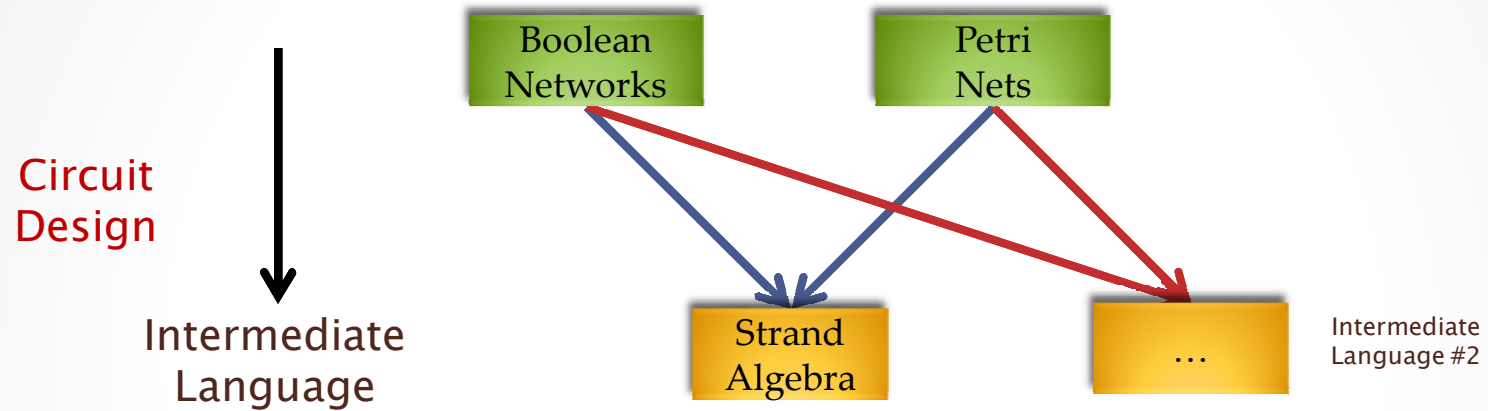
Language
Implementation #3



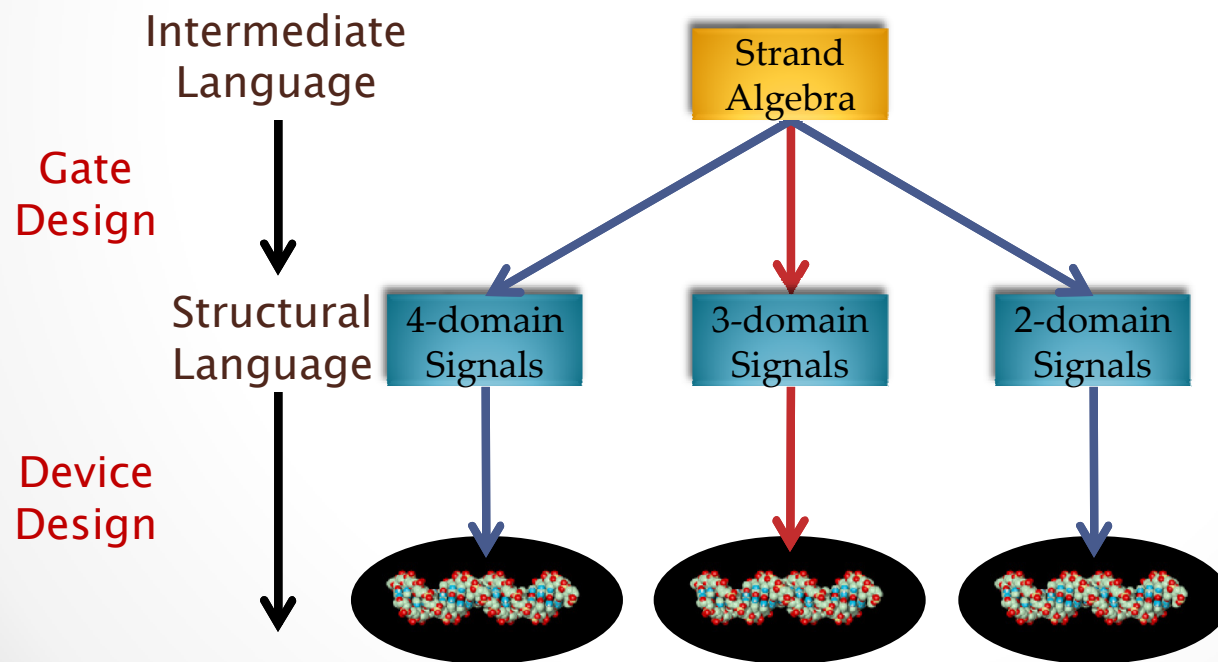
Intermediate Languages



Front Ends



Back Ends



Strand Algebra

...

Strand Algebra

$P ::= x \mid [x_1, \dots, x_n] \cdot [y_1, \dots, y_m] \mid 0 \mid P \mid P \mid P^* \quad n \geq 1, m \geq 0$

$n \times m$ gates

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

Reaction Rule

$x_1 \mid \dots \mid x_n \mid [x_1, \dots, x_n] \cdot [y_1, \dots, y_m] \rightarrow y_1 \mid \dots \mid y_m$

Auxiliary rules (axioms of diluted well-mixed solutions)

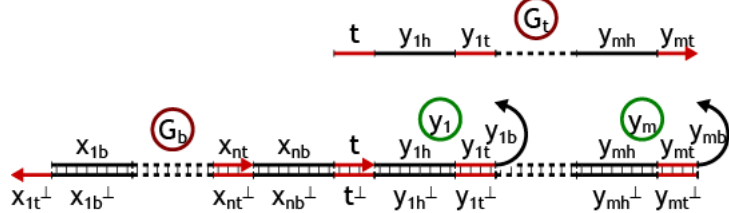
$P \rightarrow P' \Rightarrow P \mid P'' \rightarrow P' \mid P''$ Dilution
 $P \equiv P_1, P_1 \rightarrow P_2, P_2 \equiv P' \Rightarrow P \rightarrow P'$ Well Mixing

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

Compiling Strand Algebra to DNA

$$P ::= x \mid [x_1, \dots, x_n] \cdot [y_1, \dots, y_m] \mid 0 \mid P \mid P' \mid P^* \quad n \geq 1, m \geq 0$$

- $\text{compile}(x) =$ 

- $\text{compile}([x_1, \dots, x_n] \cdot [y_1, \dots, y_m]) =$ 

- $\text{compile}(0) =$ empty solution

- $\text{compile}(P \mid P') = \text{mix}(\text{compile}(P), \text{compile}(P'))$

- $\text{compile}(P^*) = \text{population}(\text{compile}(P))$

More in the DNA15 Paper

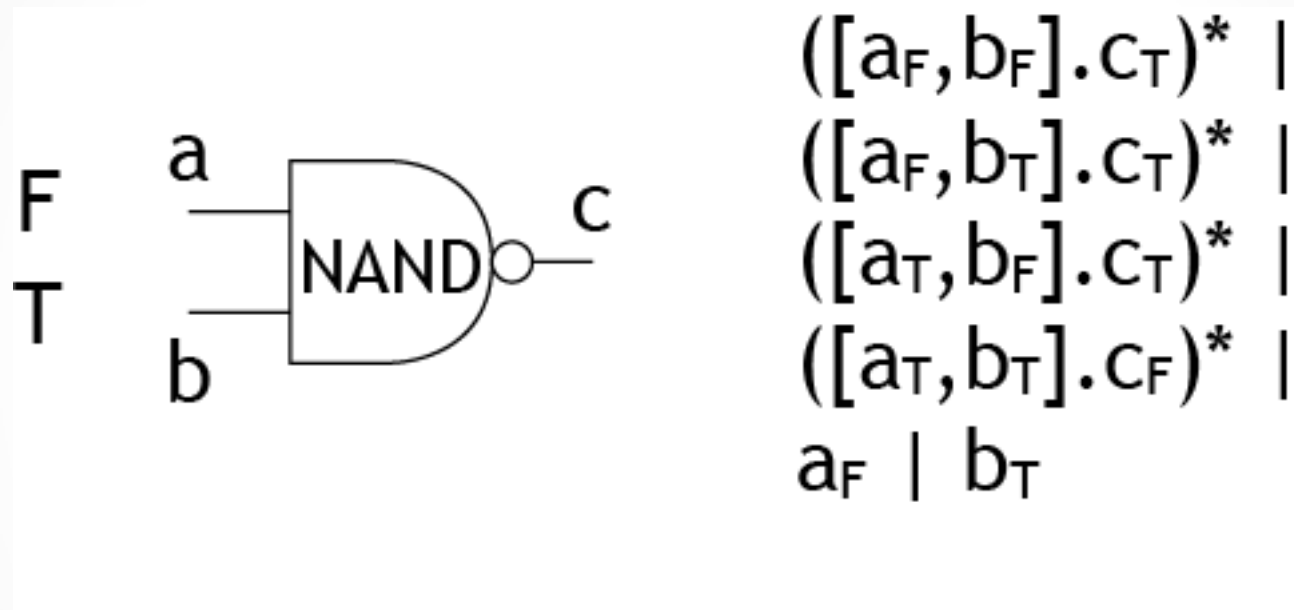
- Stochastic strand algebra
 - Matches the stochastic semantics of interacting automata
 - Uses a technique for implementing constant buffered populations, to replace P^* with finite populations
- Nested strand algebra
 - An higher-level language (with nested expressions)
 - A compilation algorithm into the basic strand algebra

Compiling Abstract Machines

...

Boolean Networks

Boolean Networks to Strand Algebra

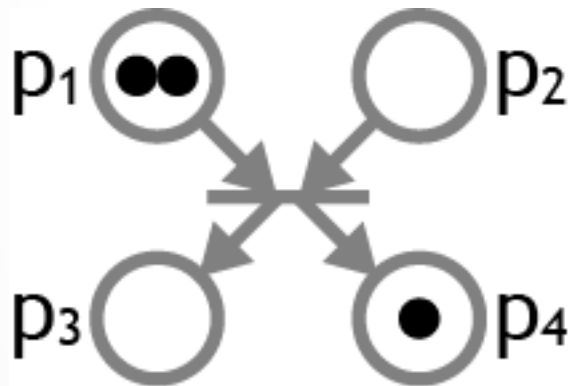


- This encoding is *compositional*, and can encode *any* Boolean network:
- multi-stage networks can be assembled (**combinatorial logic**)
 - network loops are allowed (**sequential logic**)

Petri Nets

Petri Nets to Strand Algebra

Transitions as Gates
Place markings as Signals



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

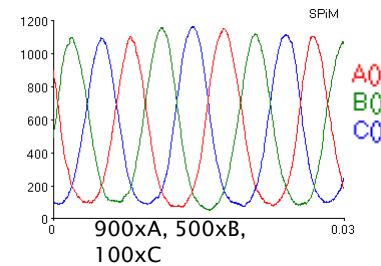
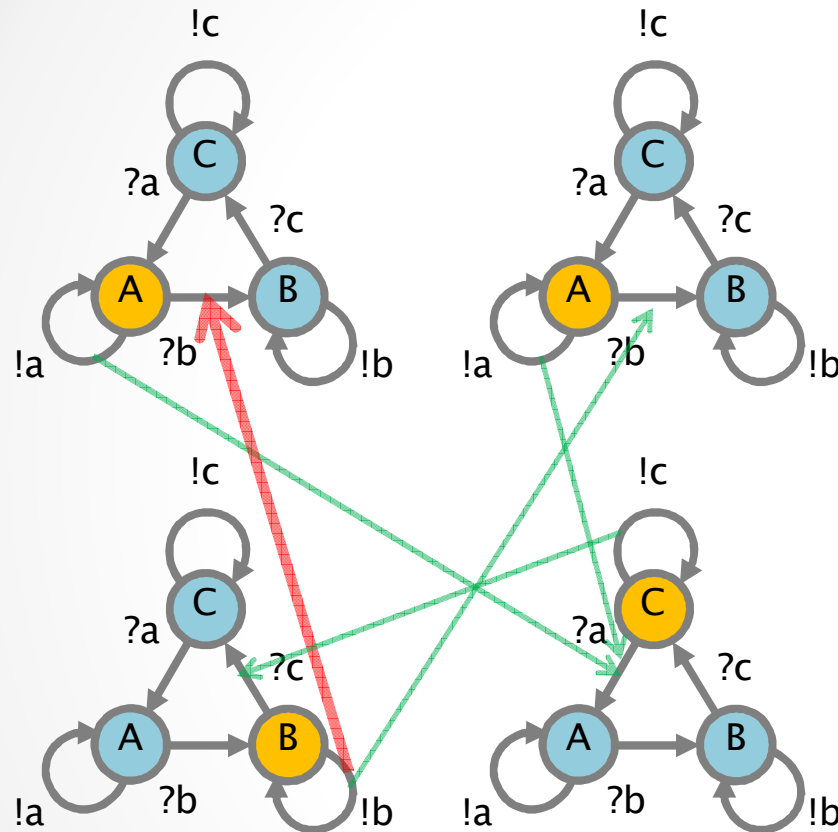
Chemical Reaction Networks

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

Species become signals
Reactions become gates



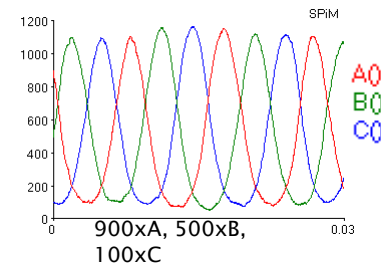
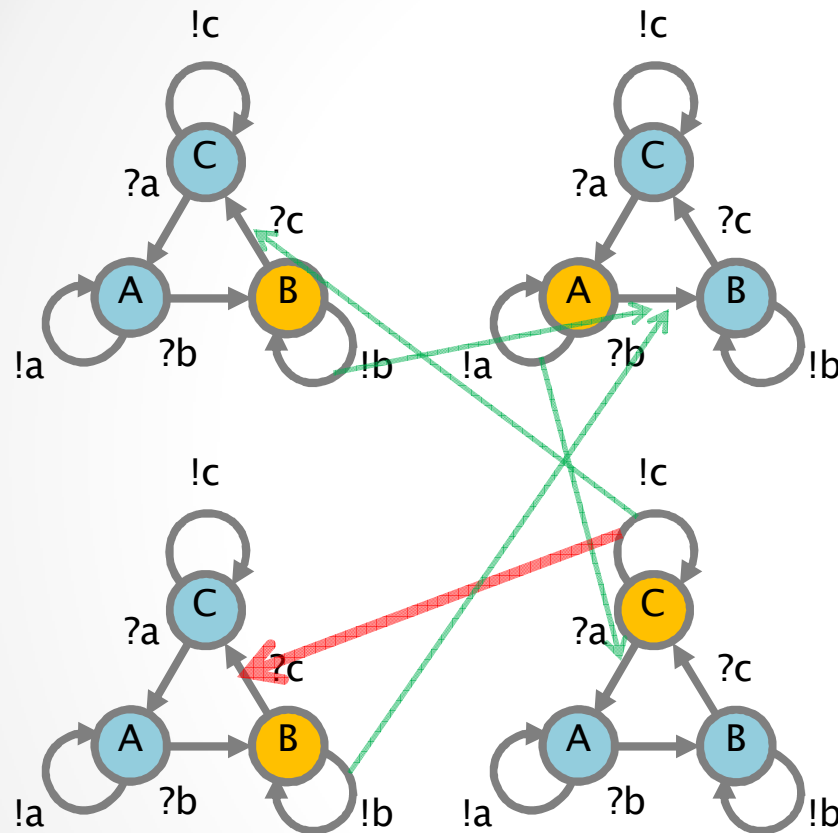
Interacting Automata



$([A, B]. [B, B])^* \mid$
 $([B, C]. [C, C])^* \mid$
 $([C, A]. [A, A])^* \mid$
 $A \mid A \mid B \mid C$

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

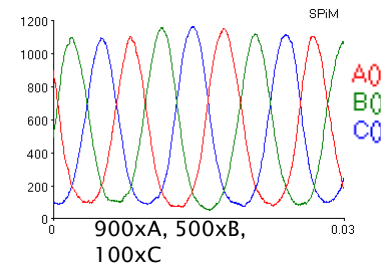
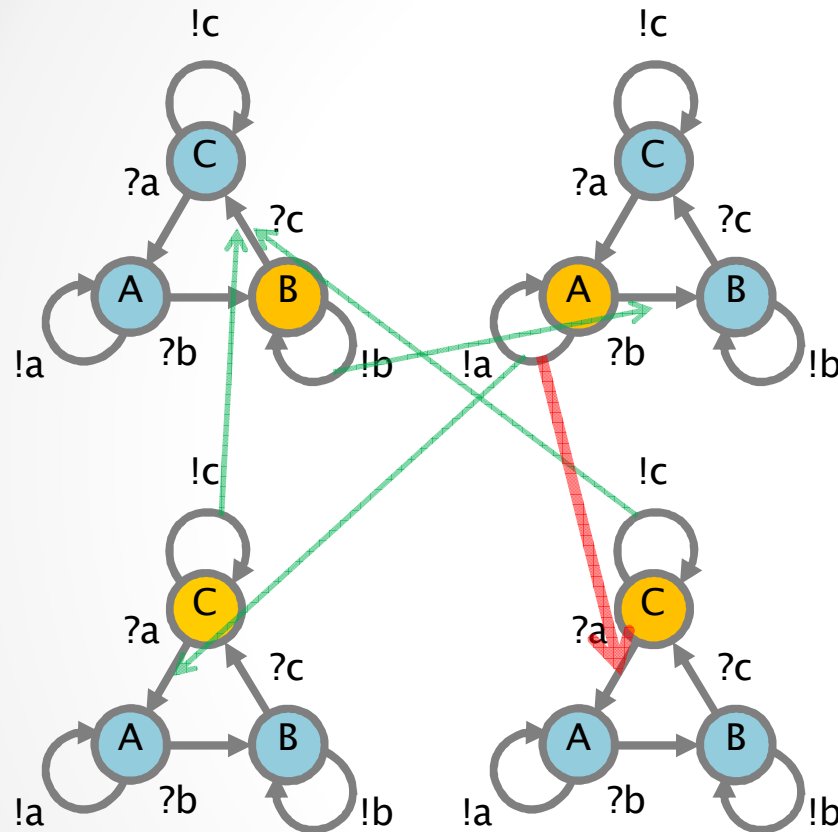
Interacting Automata



$([A, B]. [B, B])^* \mid$
 $([B, C]. [C, C])^* \mid$
 $([C, A]. [A, A])^* \mid$
 $A \mid B \mid B \mid C$

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

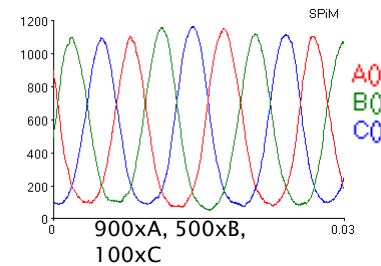
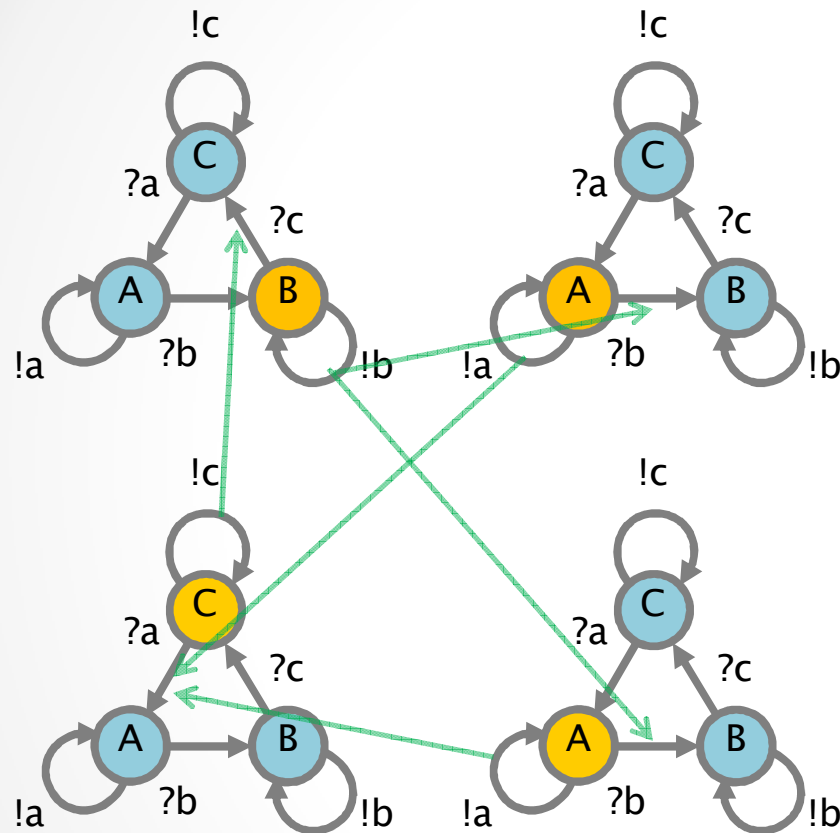
Interacting Automata



$([A, B]. [B, B])^* \mid$
 $([B, C]. [C, C])^* \mid$
 $([C, A]. [A, A])^* \mid$
 $A \mid B \mid C \mid C$

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

Interacting Automata



$([A, B]. [B, B])^* \mid$
 $([B, C]. [C, C])^* \mid$
 $([C, A]. [A, A])^* \mid$
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

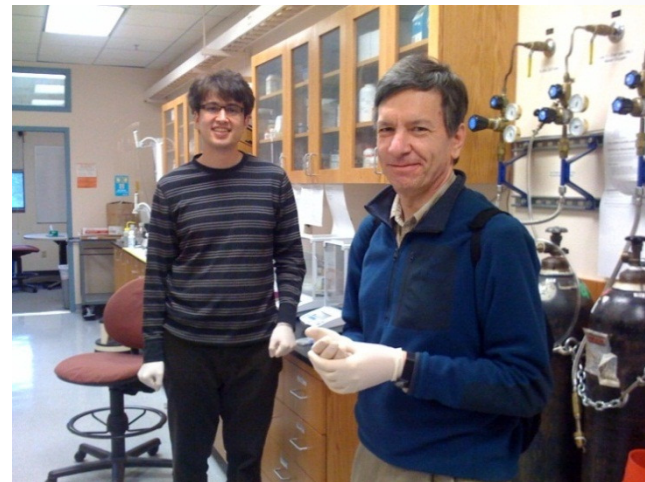
Conclusions

- **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



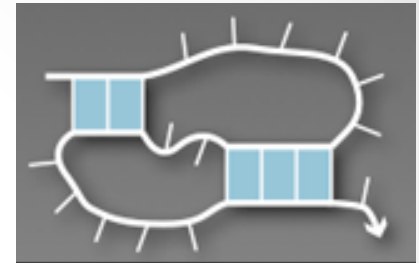
- Illustrations
 - John Reif, Duke
 - Ned Seeman, NYU
 - Erik Winfree, Caltech
 - Bernard Yurke, Boise State
 - Wikipedia
 - YouTube
- David Soloveichik



The Molecular Programming Project

- Caltech & U.Washington

- National Science Foundation's Expeditions in Computing
- Shuki Brooks, Erik Klavins, Richard Murray, Niles Pierce, Paul Rothmund, 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.

Q?

<http://lucacardelli.name>