

Algebras and Languages for Molecular Programming

Luca Cardelli

Microsoft Research

Exeter, 2010-03-10

<http://lucacardelli.name>

Smaller and Smaller

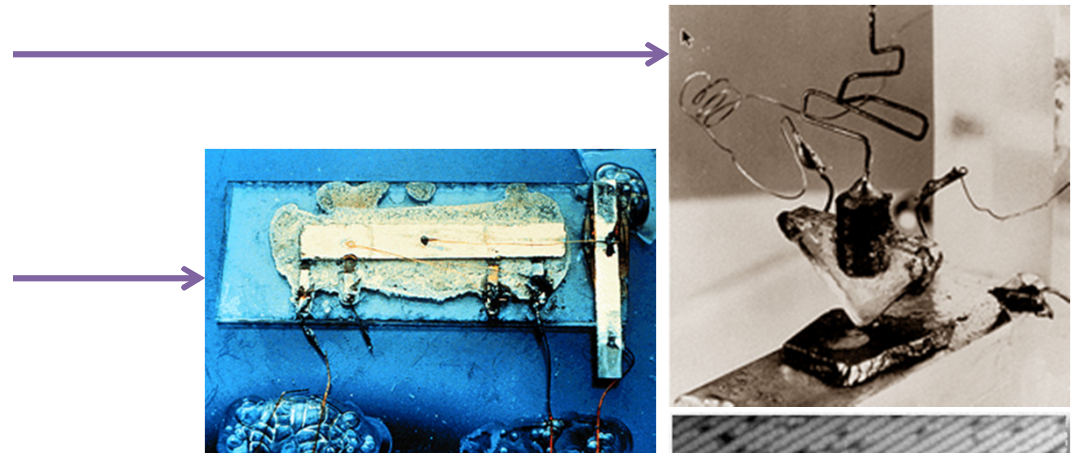
Dec. 23, 1947. John Bardeen and Walter Brattain show the first working transistor.

September 1958. Jack Kilby builds the first integrated circuit.

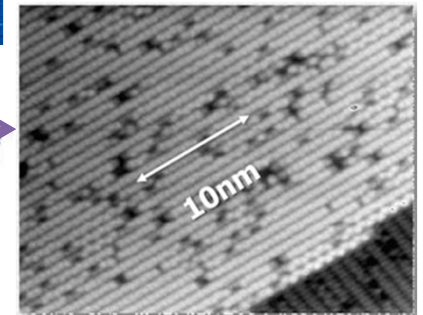
Jan 30, 2010. Intel and Micron announce 25nm NAND flash.

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

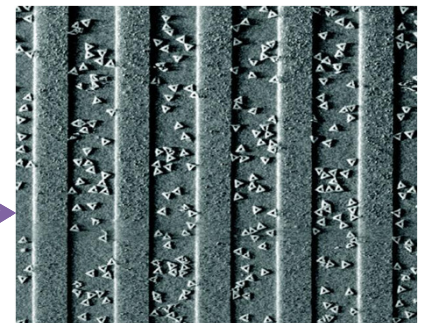
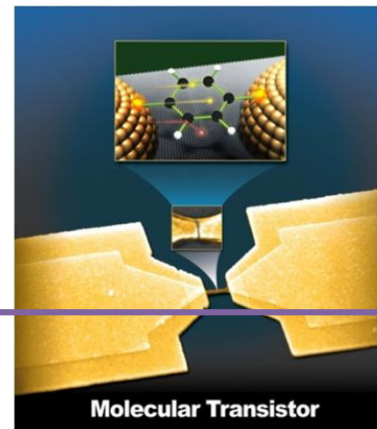
The race is on for *molecular scale integrated circuits*.



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



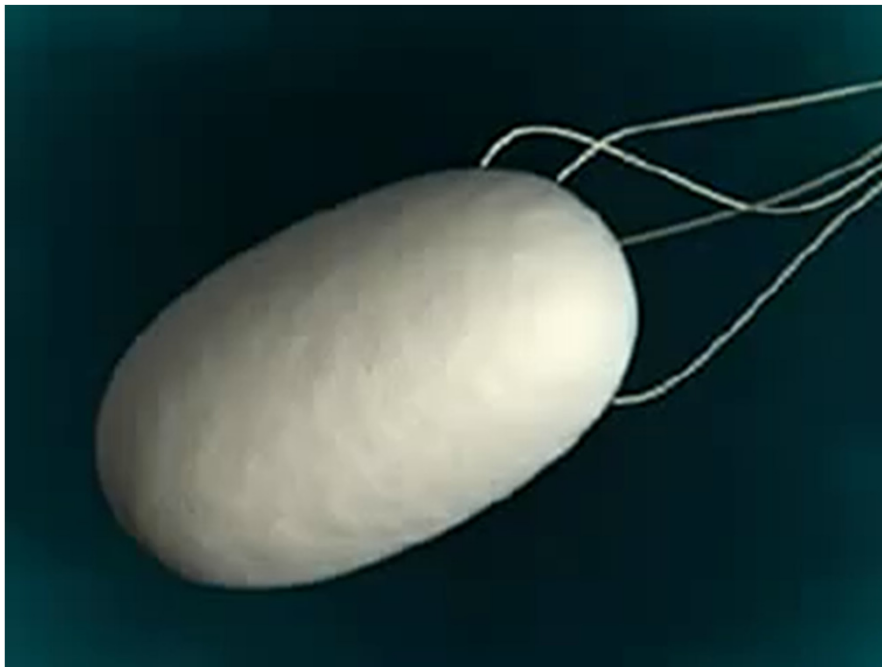
Observation of molecular orbital gating. *Nature*, 2009; 462 (7276): 1039



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

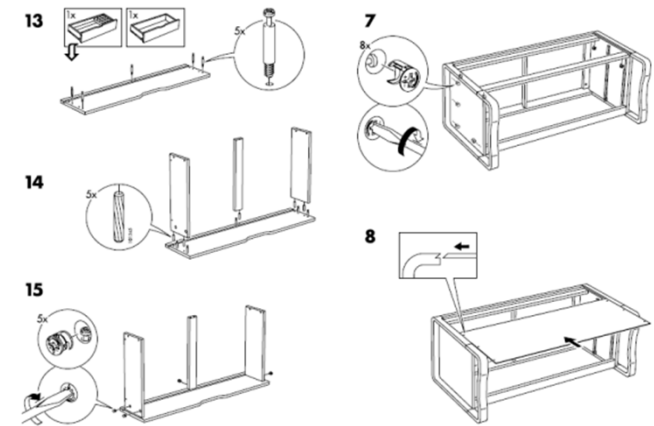
Building The Smallest Things

- 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*.

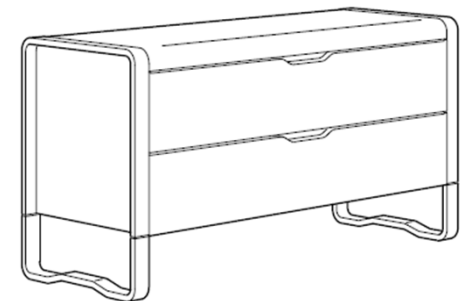


Molecular IKEA

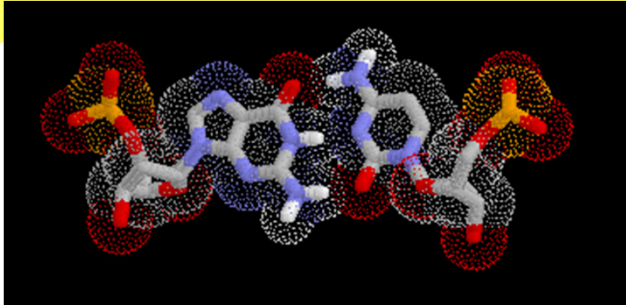
- Nature can self-assemble.
Can we?
- “Dear IKEA, please send me a chest of drawers that assembles itself.”
- We need a magical material where the pieces are pre-programmed to fit into to each other.
- At the molecular scale many such materials exist; let’s pick one...



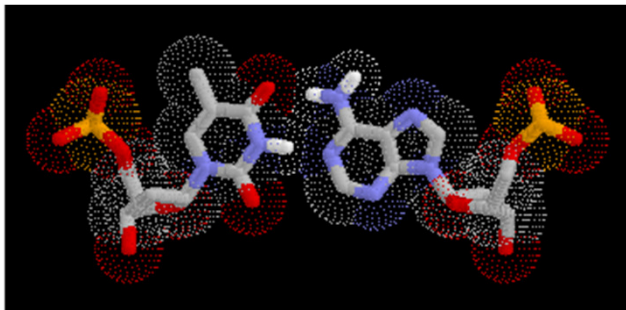
Add water



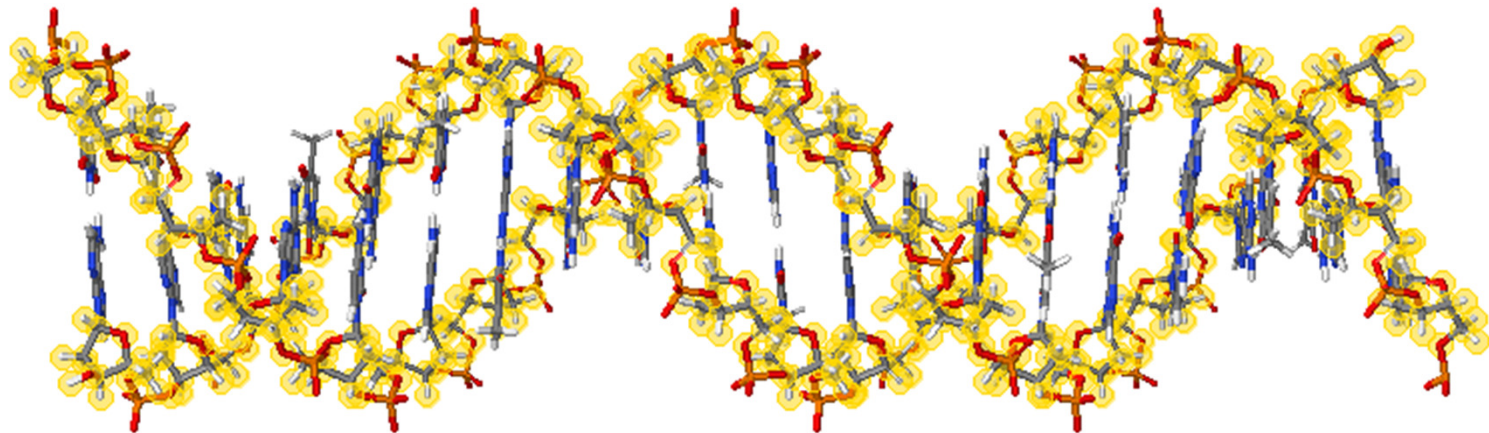
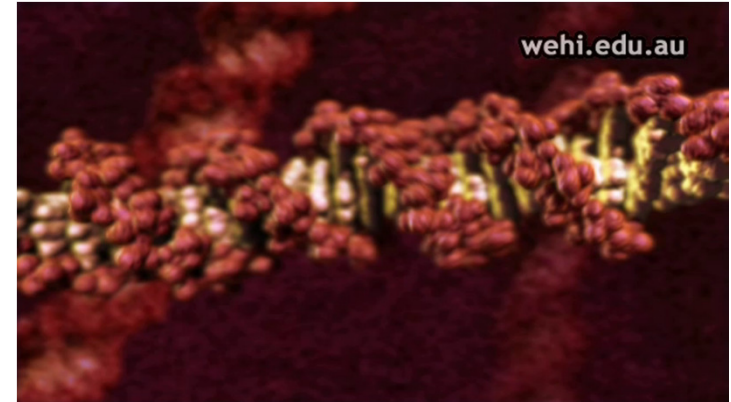
DNA



GC Base Pair
Guanine-Cytosine



TA Base Pair
Thymine-Adenine



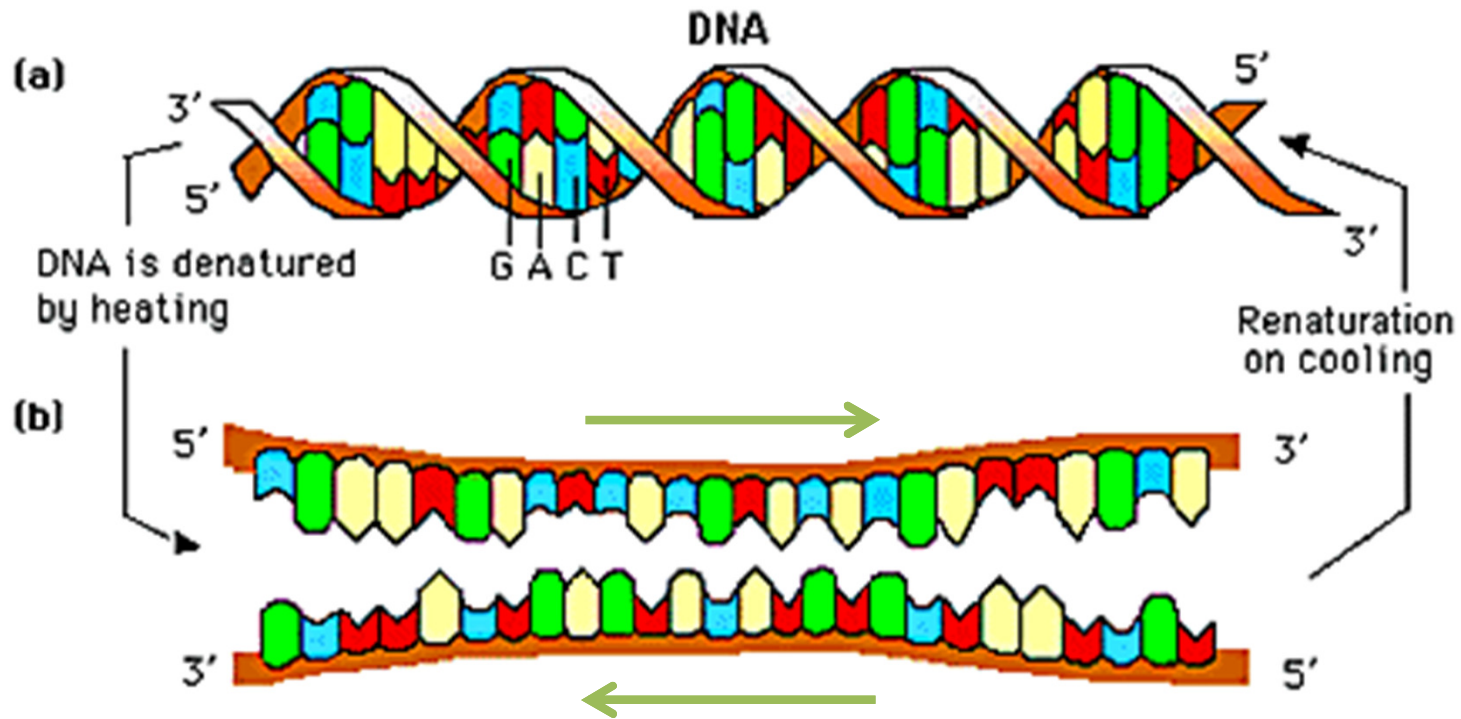
Sequence of Base Pairs (GACT alphabet)

[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>)

ssDNA

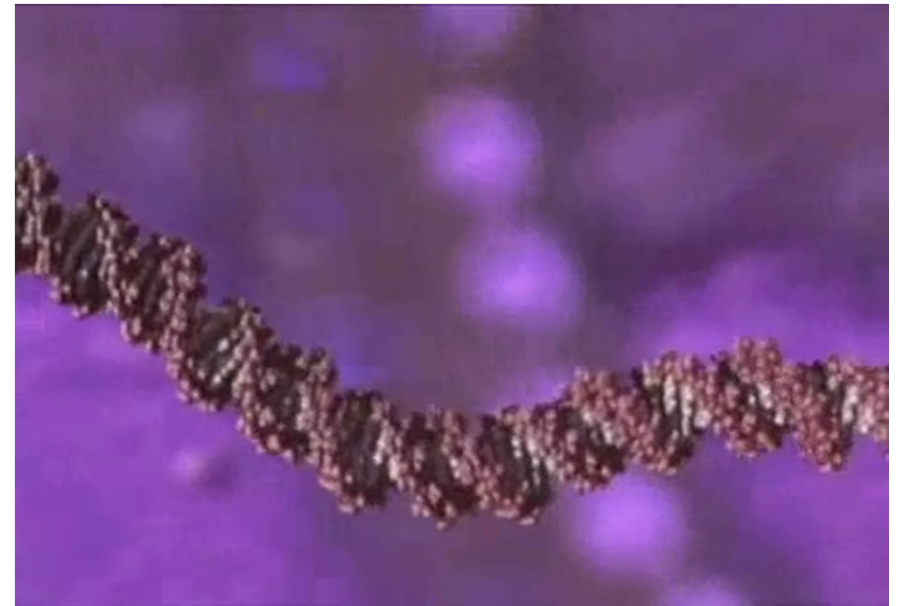
Double-stranded DNA



Single-stranded DNA has an orientation
Each strand spells a GACT sequence
The two strands have *opposite* orientations

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



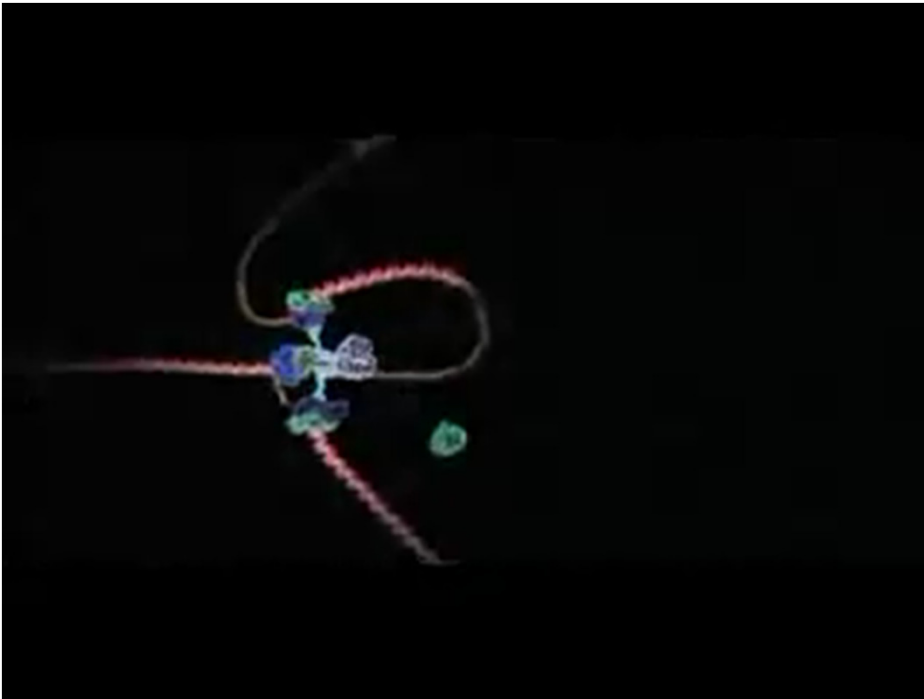
DNA wrapping into chromosomes



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)



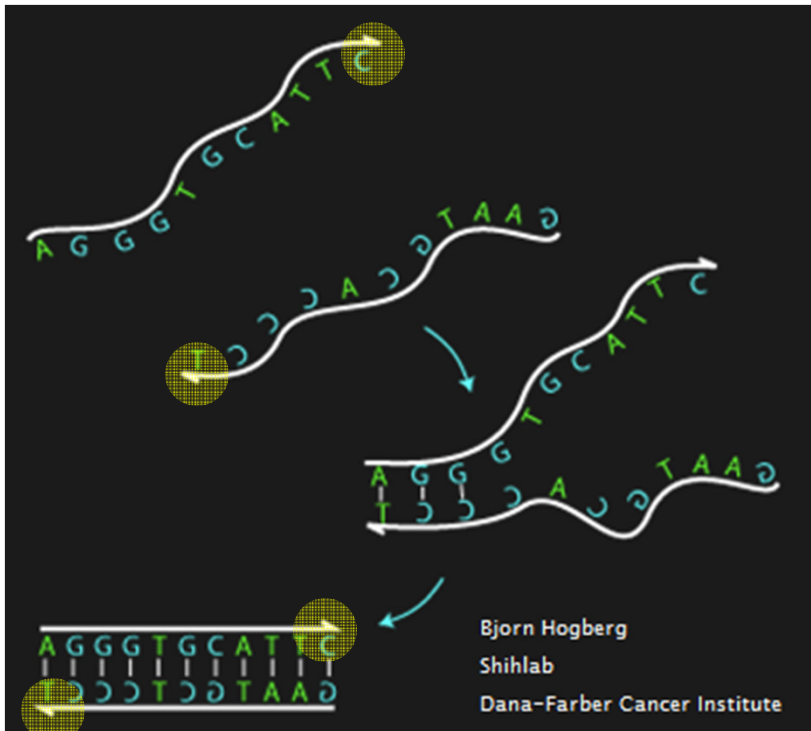
DNA transcription in *real time*

RNA polymerase II: 15-30 base/second

Drew Berry

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

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.

Nanoscale Engineering

- Sensing

- Reacting to forces
- Binding to molecules

- Actuating

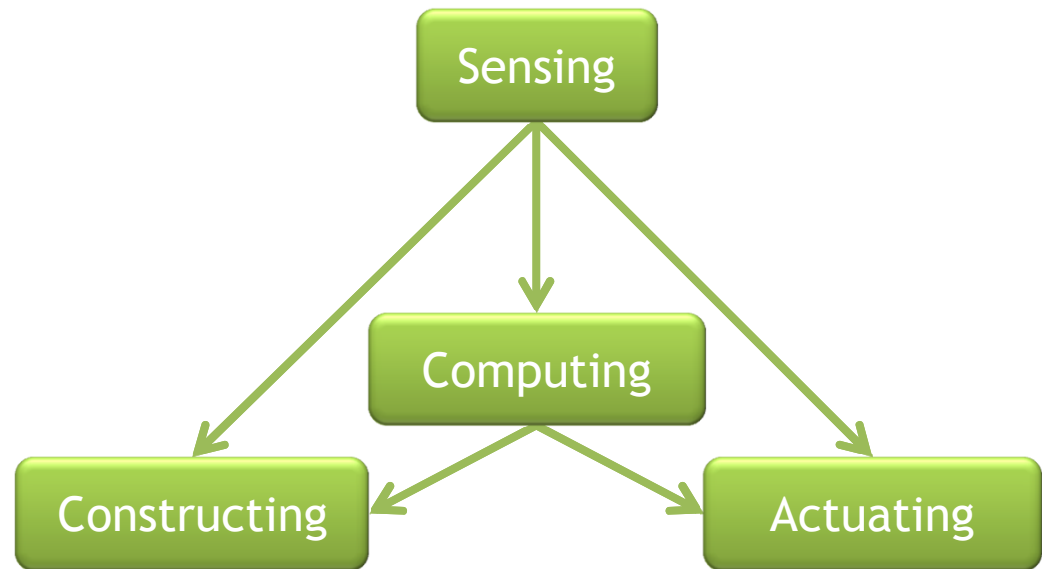
- Releasing molecules
- Producing forces

- Constructing

- Chassis
- Growth

- Computing

- Signal Processing
- Decision Making



Nucleic Acids (DNA/RNA) can do all this, and interface to biological structures.

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**
 - If the inputs are voltages, the outputs must be voltages
 - If the inputs are proteins, the outputs must be proteins
 - If the outputs are photons the inputs must be photons
 - If the inputs are DNA, the outputs must be DNA
- Central design question
 - What should our **signals** (not components!) be?
 - Design components that manipulate those signals.

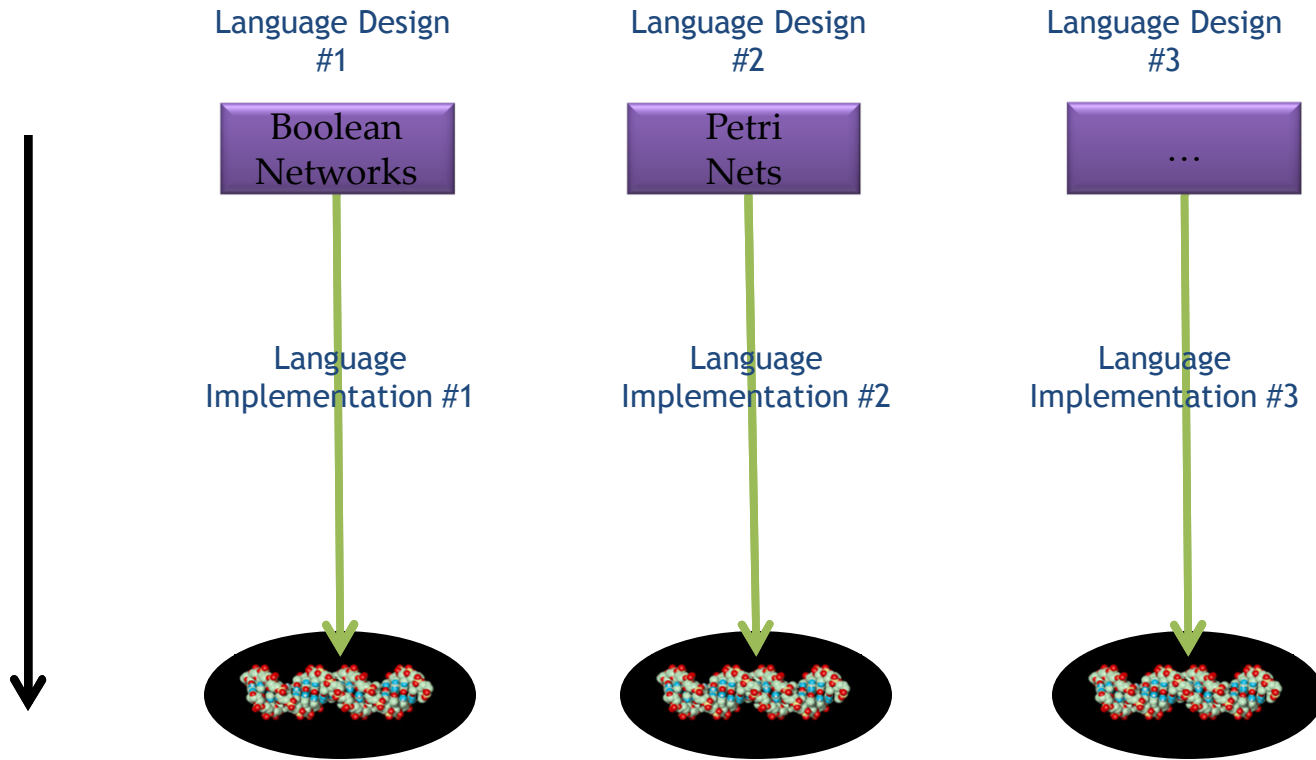
What does DNA Compute?

- Electronics has *electrons*
 - All electrons are the same
 - All you can do is see if you have *few* ('False') or *lots* ('True') of electrons
 - Hence Boolean logic is at the basis of digital circuit design
 - Symbolic and numeric computation has to be encoded above that
 - But mostly we want to compute with symbols and numbers, not with Booleans
- DNA computing has *symbols* (DNA words)
 - DNA words are not all the same
 - **Symbolic computation** can be done *directly*
 - We can also directly use molecular concurrency
- **Process Algebra** as the 'Boolean Algebra' of DNA Computing
 - What are the 'gates' of symbolic concurrent computation?
 - That's what Process Algebra is about
 - (Process Algebra comes from the theory of concurrent systems)

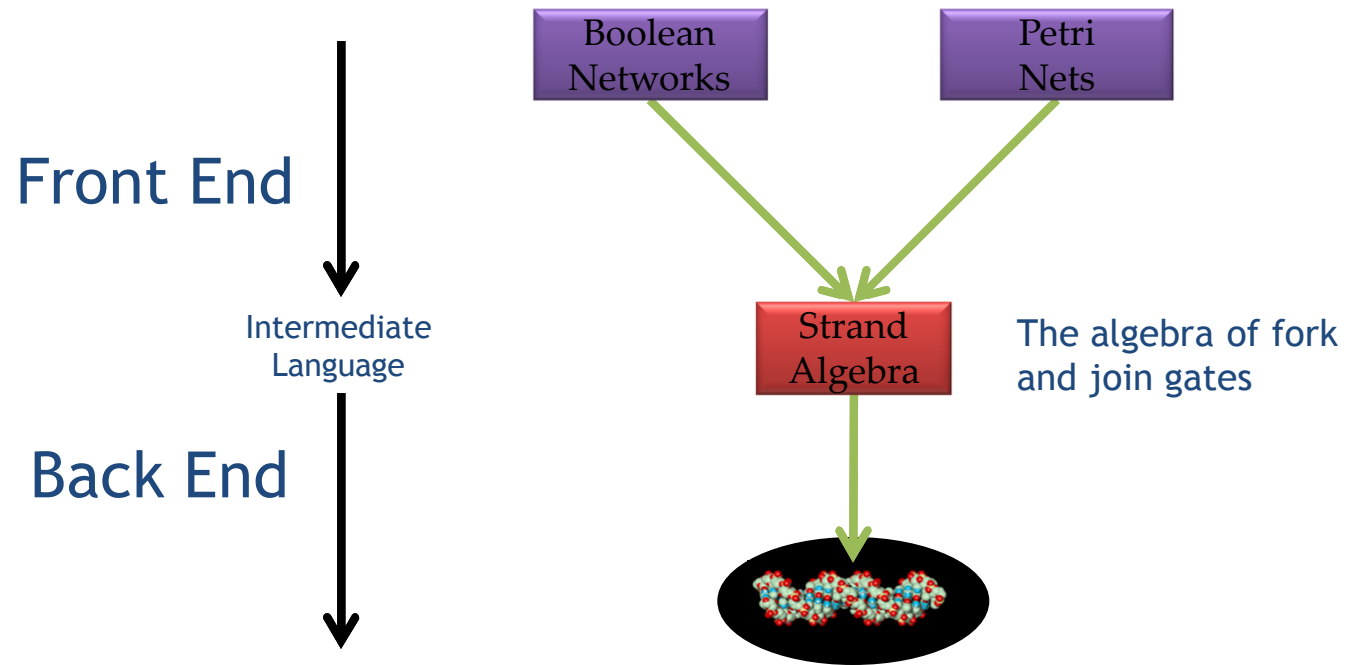
Implementing "Arbitrary" Computing Functions

Compilers

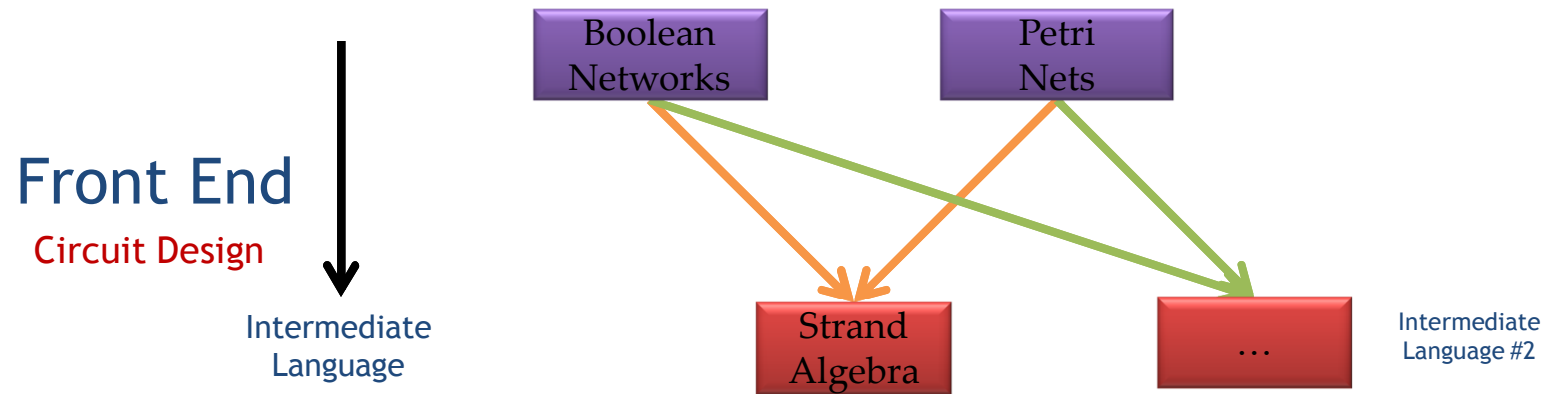
Monolithic
Compilers



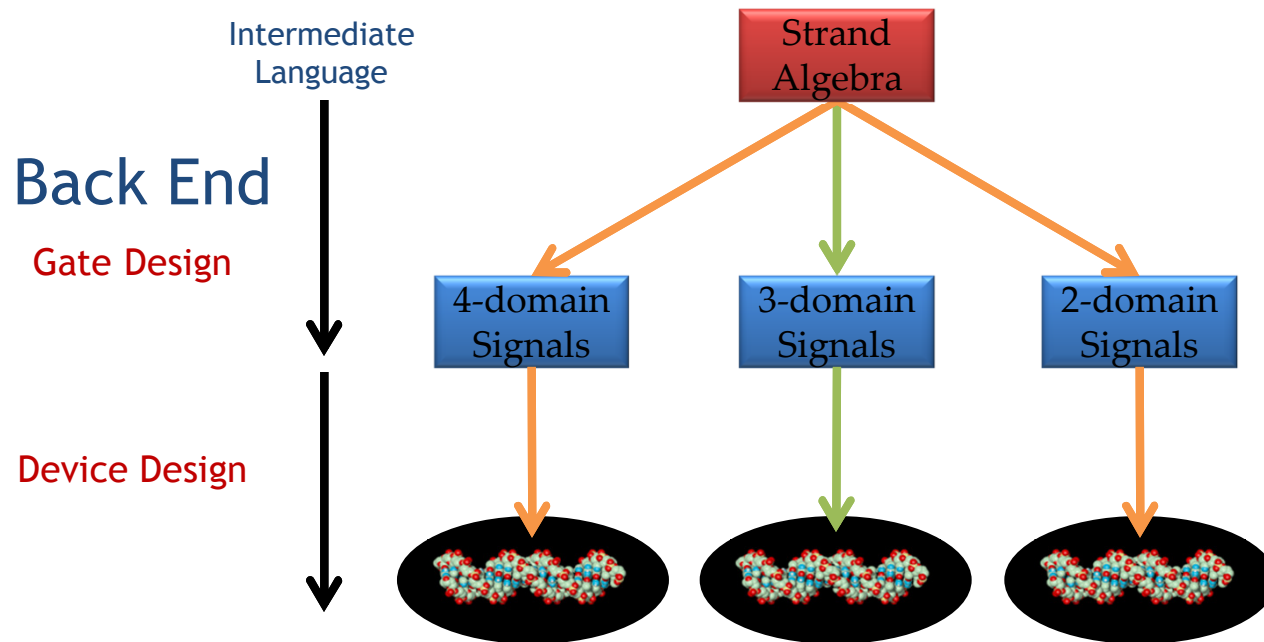
Intermediate Languages



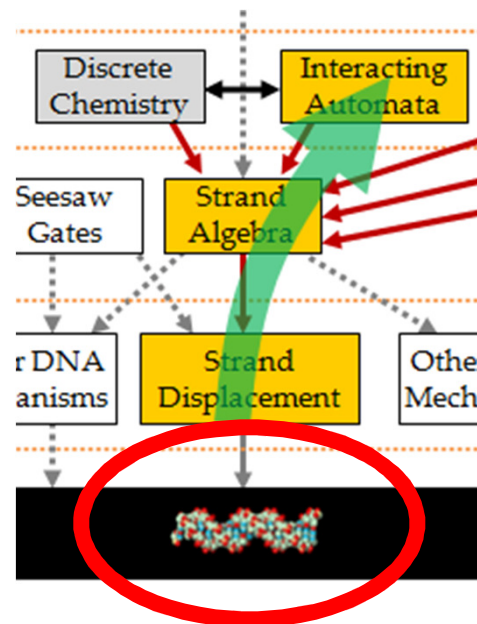
Front Ends



Back Ends



Toehold Mediated Strand Displacement



Rules of the Game

- Short complementary segments hybridize reversibly

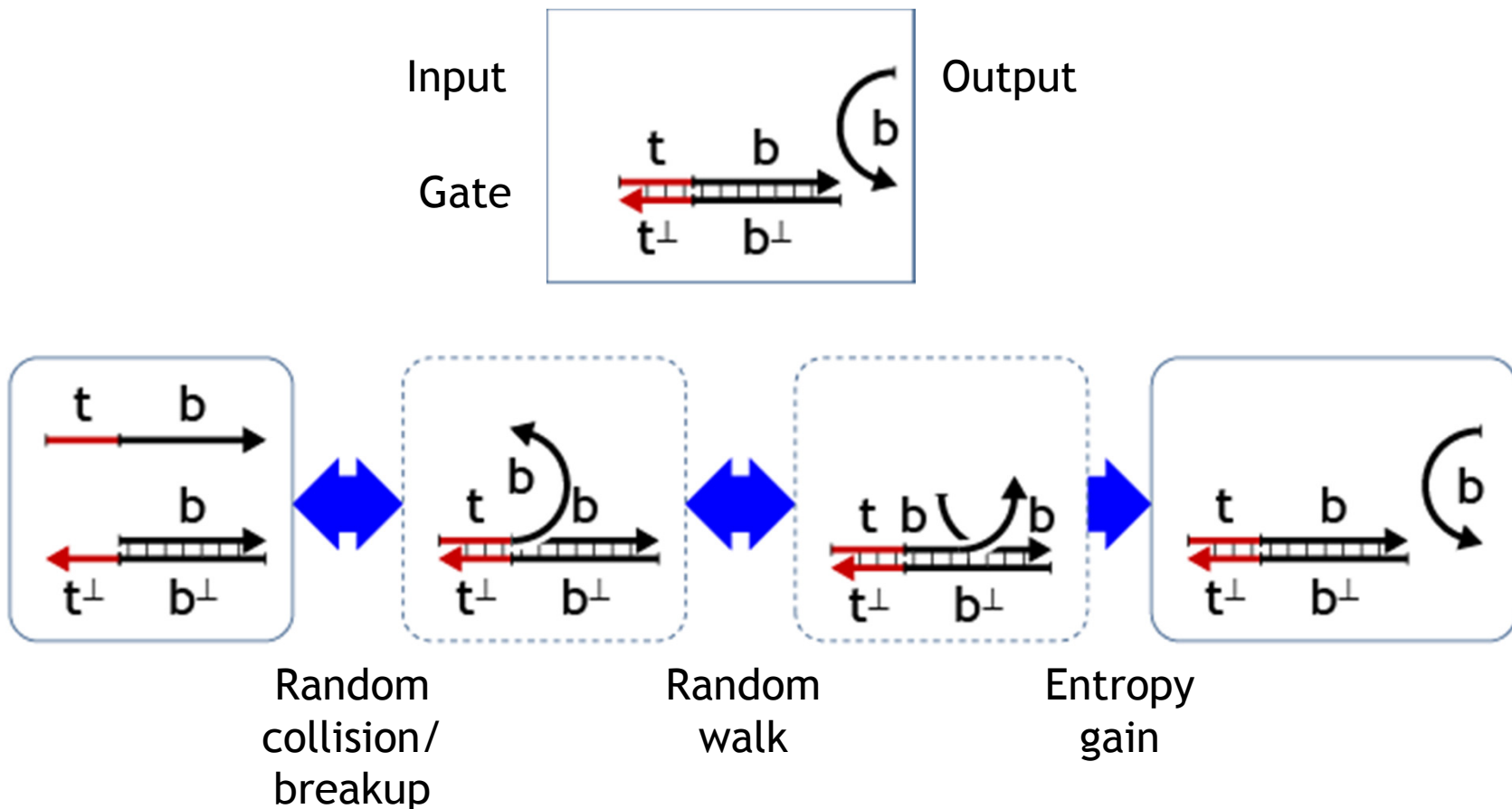


- Long complementary segments hybridize irreversibly



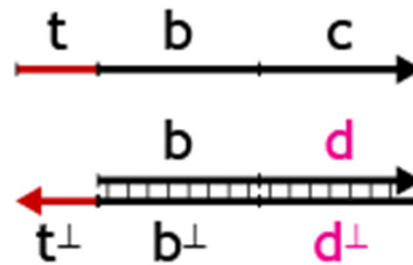
DNA Strand Displacement

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

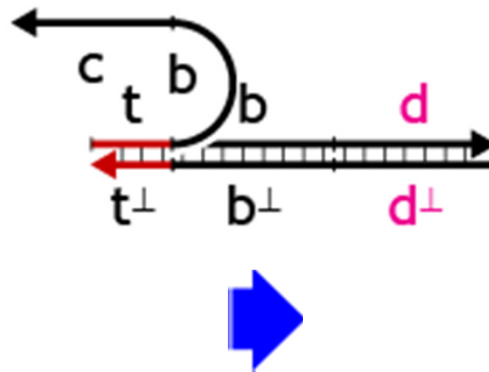


Failed Strand Displacement

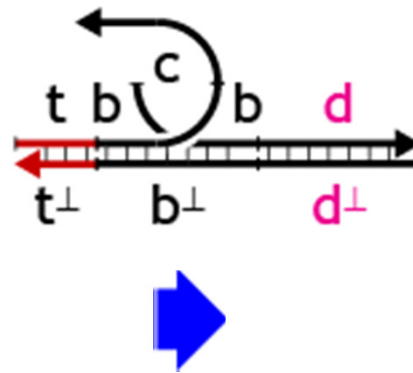
- What if the input does not match the gate?



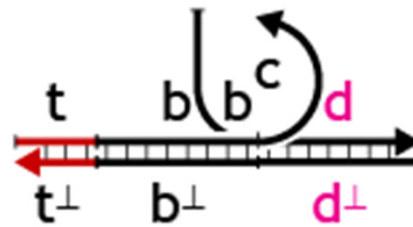
Failed Strand Displacement



Failed Strand Displacement

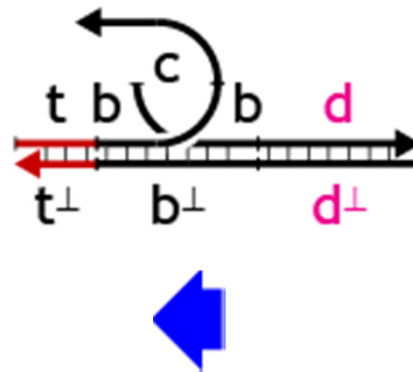


Failed Strand Displacement

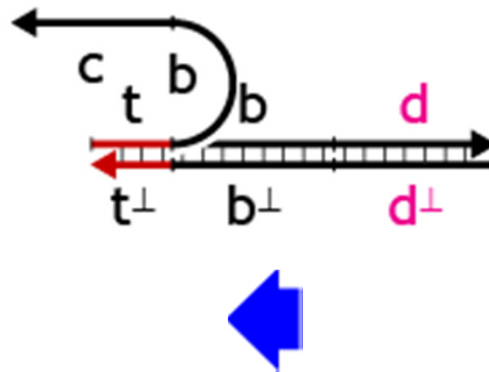


?

Failed Strand Displacement

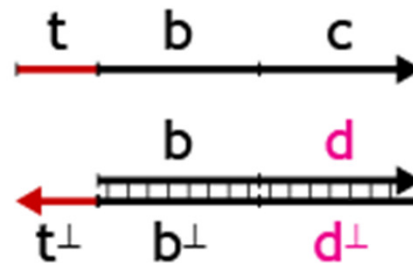


Failed Strand Displacement



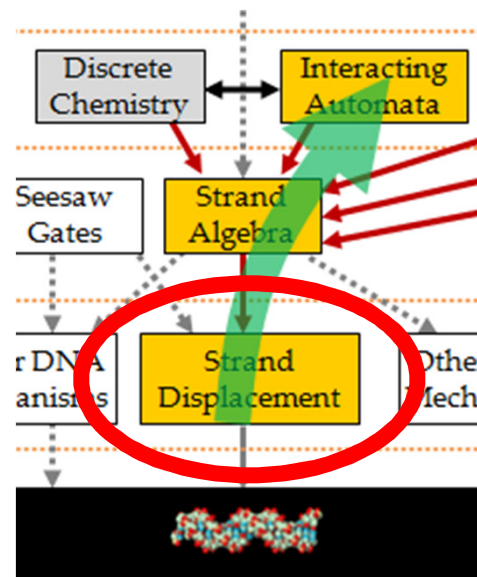
Failed Strand Displacement

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



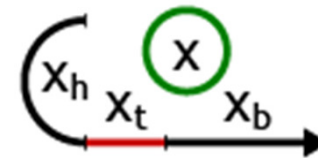
- Matching depends on the long segment only
 - Strand displacement succeeds iff the whole long segment matches
 - The address space is determined by the size of the long segment, which is unbounded (not by the size of the toehold)
 - The toehold is just a 'cache' of the address

Strand Displacement Signals and Gates



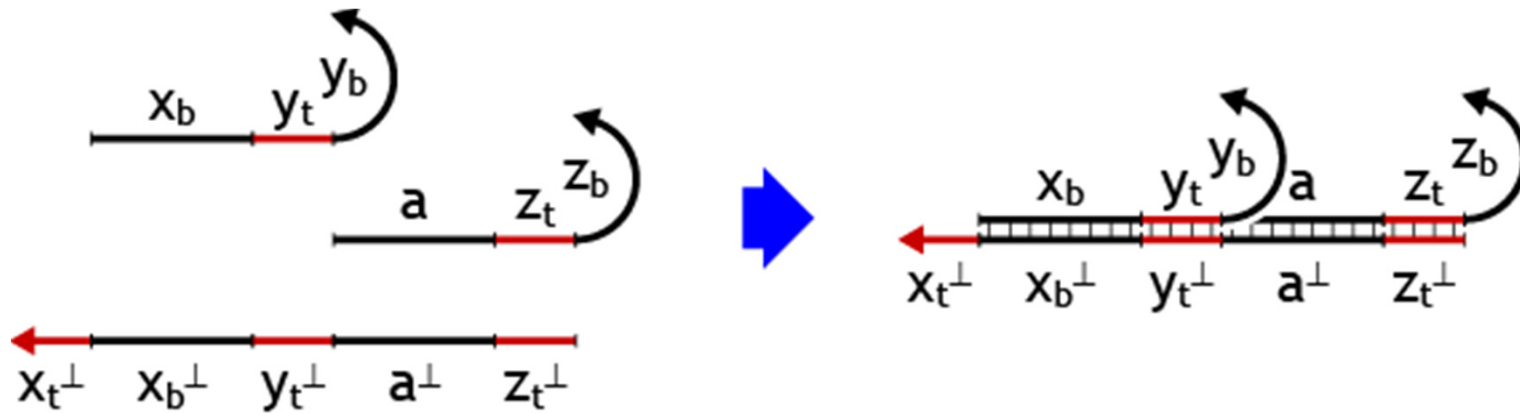
Signals

- 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

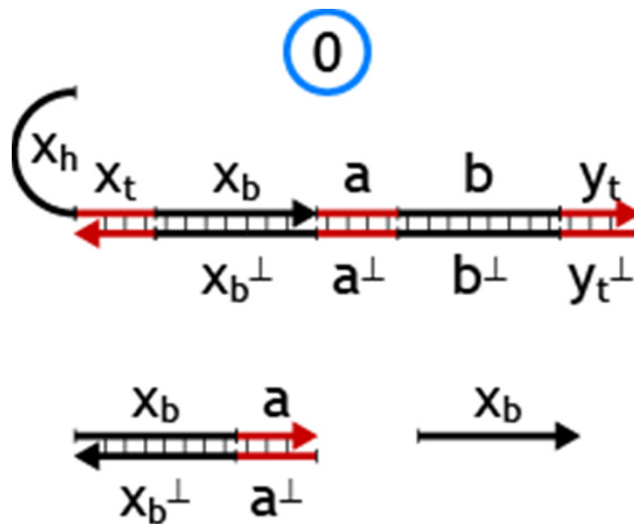
- Double-stranded structures with free toeholds



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

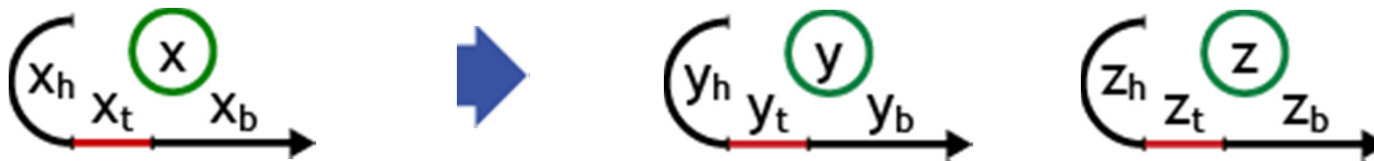
Waste

A system is considered *inert* (terminated) if it has no free toeholds.



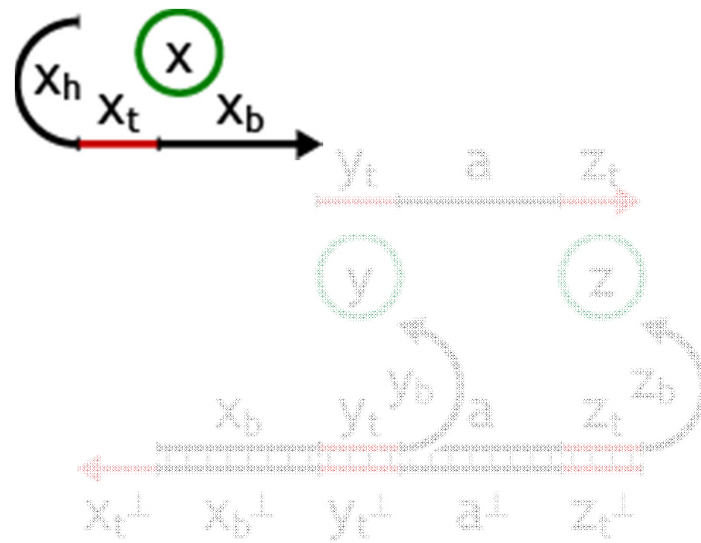
Fork Gate

- $x \rightarrow y + z$



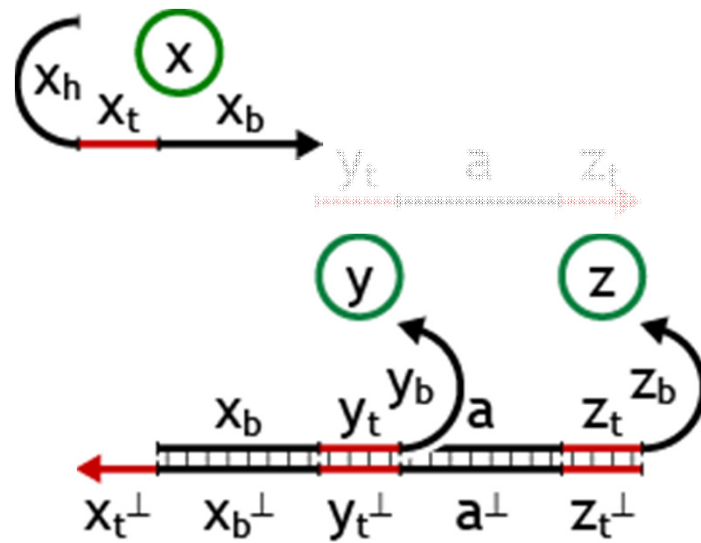
- $x \rightarrow y + 0$ transform x to y (transducer)
- $x \rightarrow x + y$ linear production of y (catalyst)
- $x \rightarrow x + x$ exponential production of x (amplifier)

Fork Gate

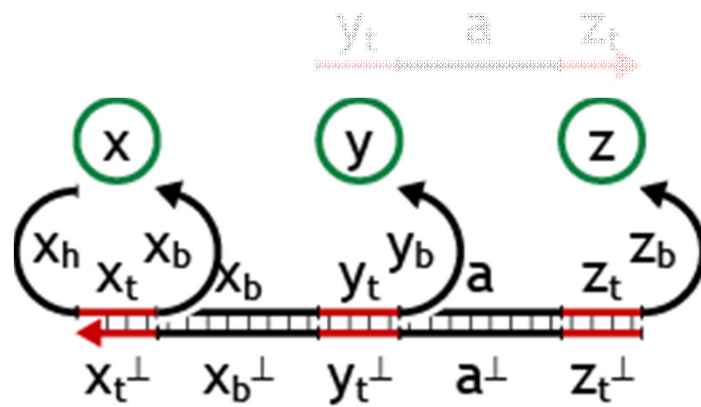


This is the
Fork Gate
structure

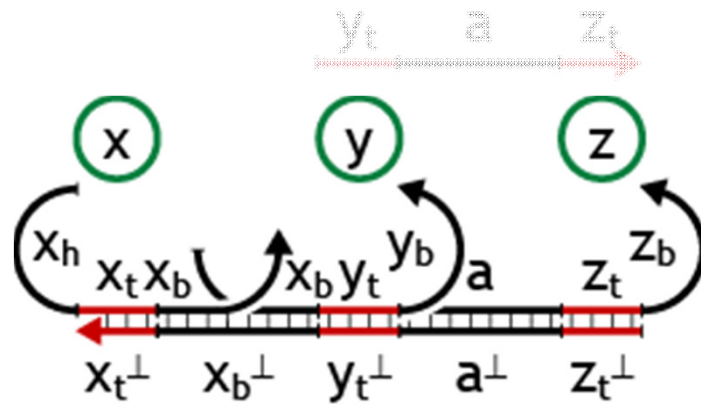
Fork Gate



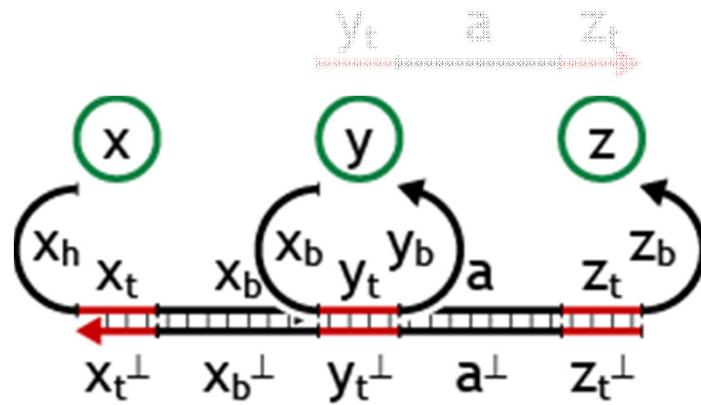
Fork Gate



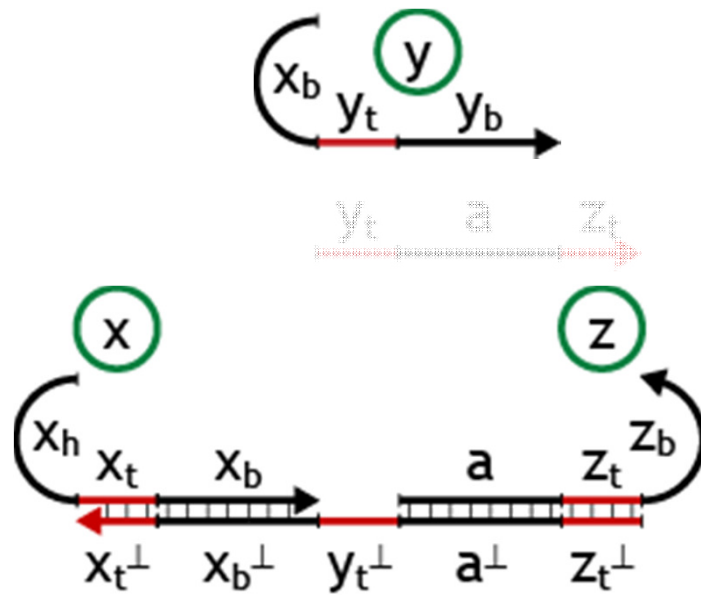
Fork Gate



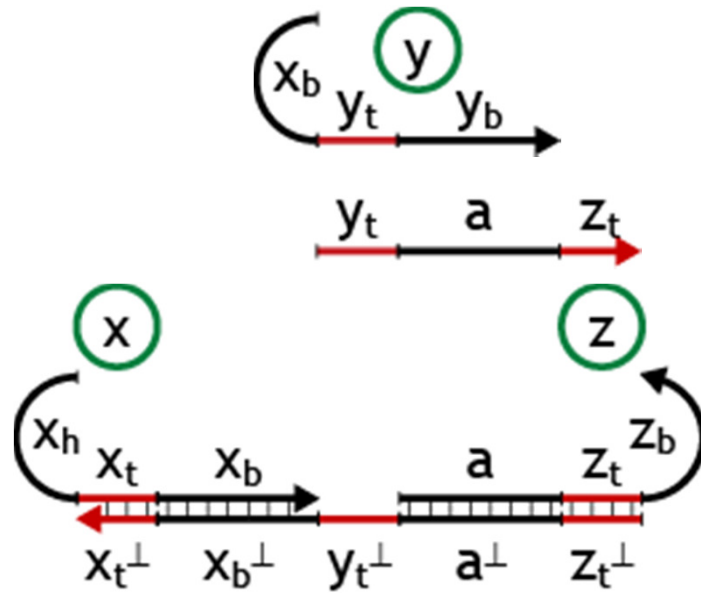
Fork Gate



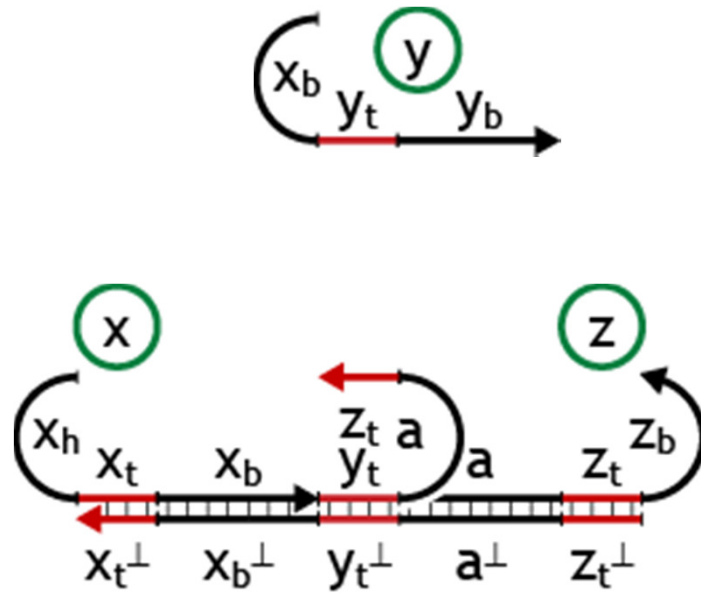
Fork Gate



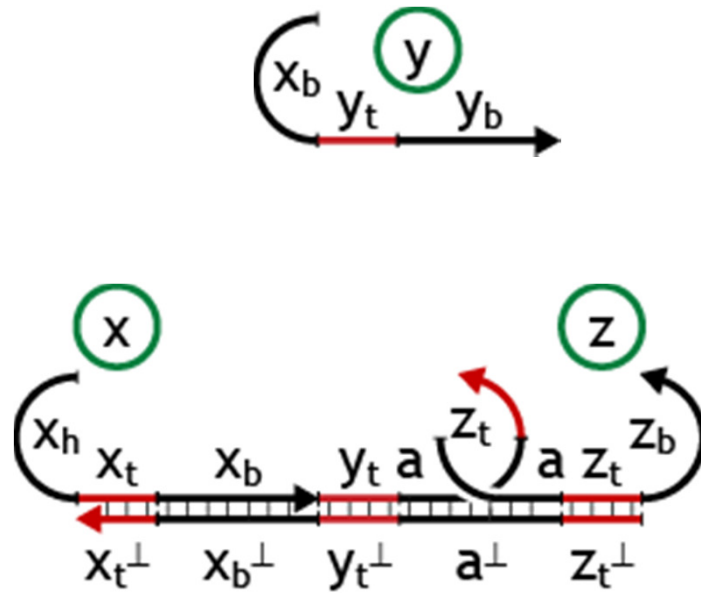
Fork Gate



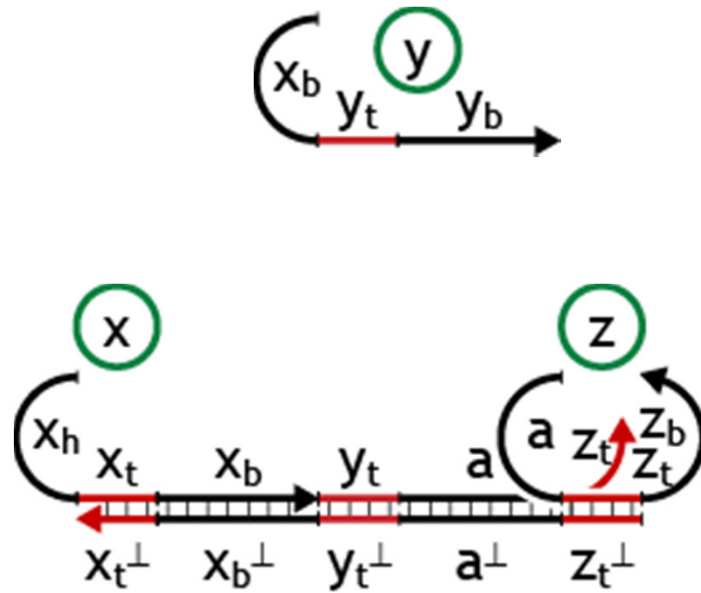
Fork Gate



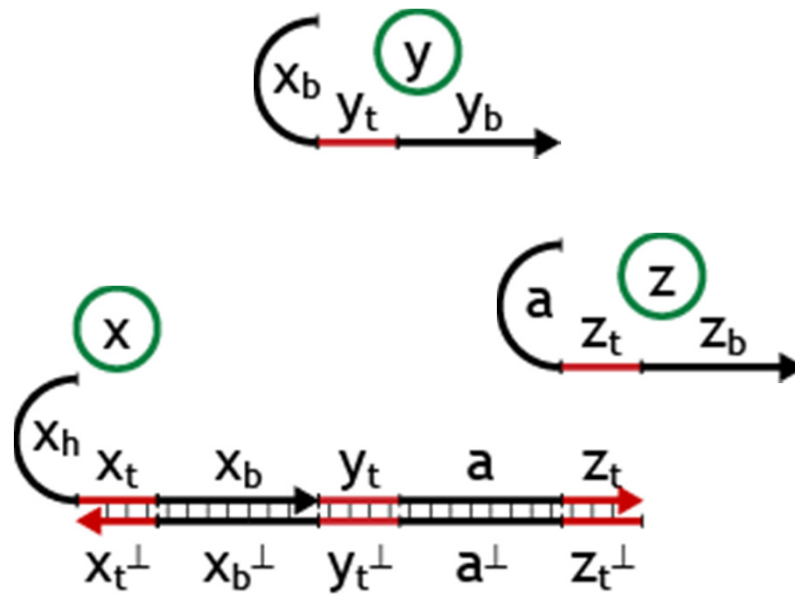
Fork Gate



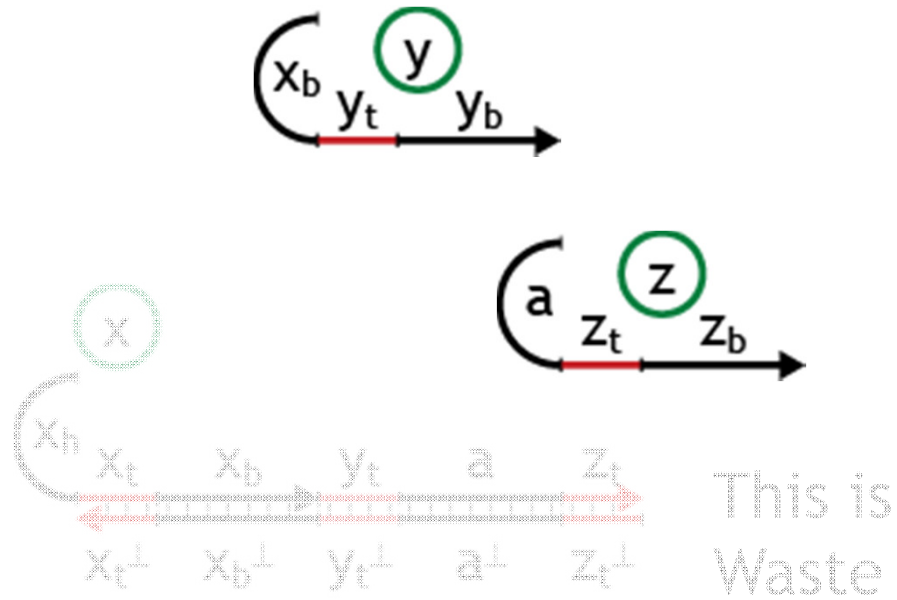
Fork Gate



Fork Gate



Fork Gate

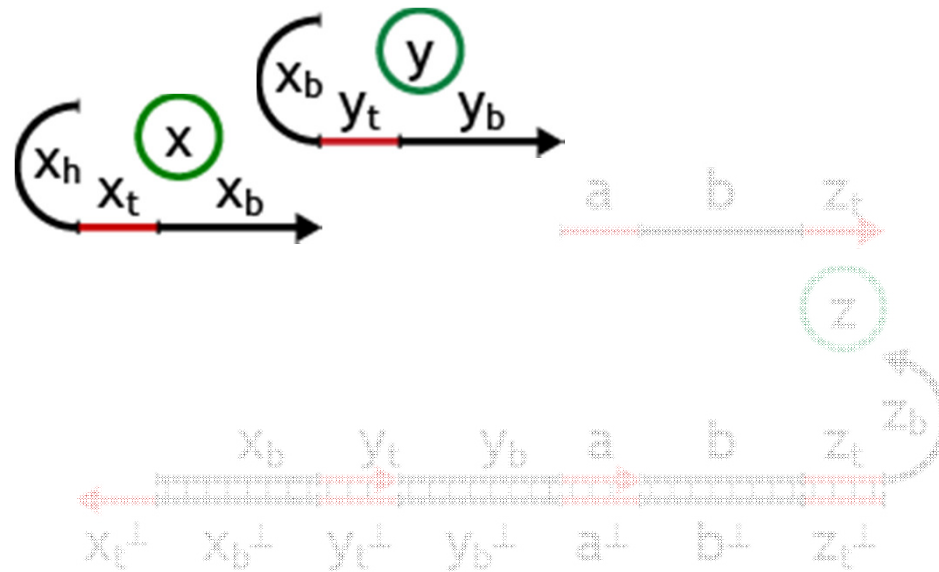


Join Gate

- $x + y \rightarrow z$

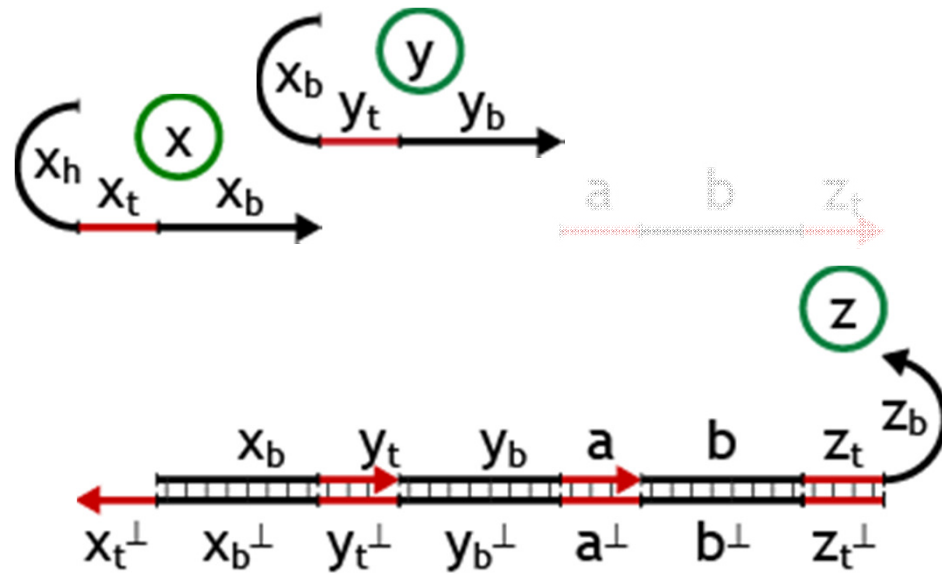


Join Gate

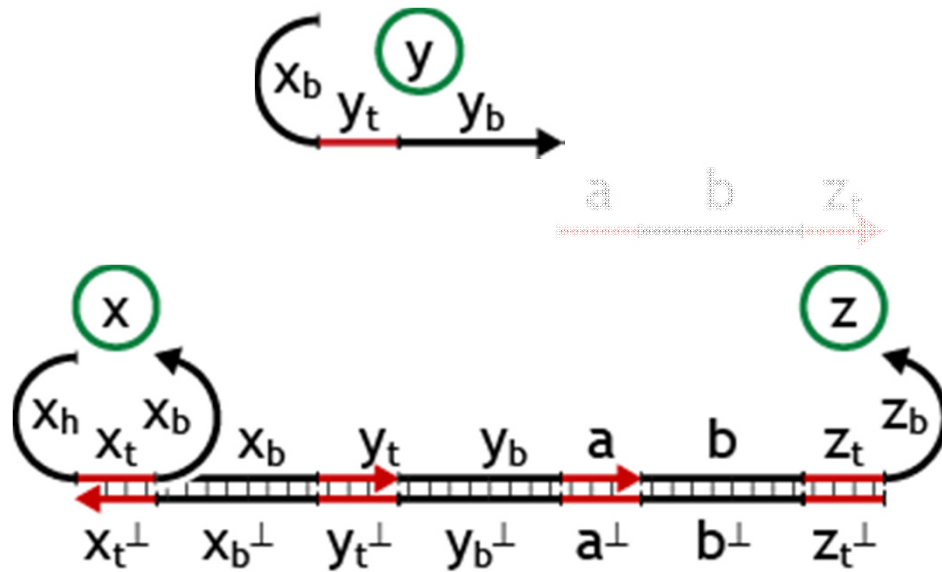


This is the
Join Gate
structure

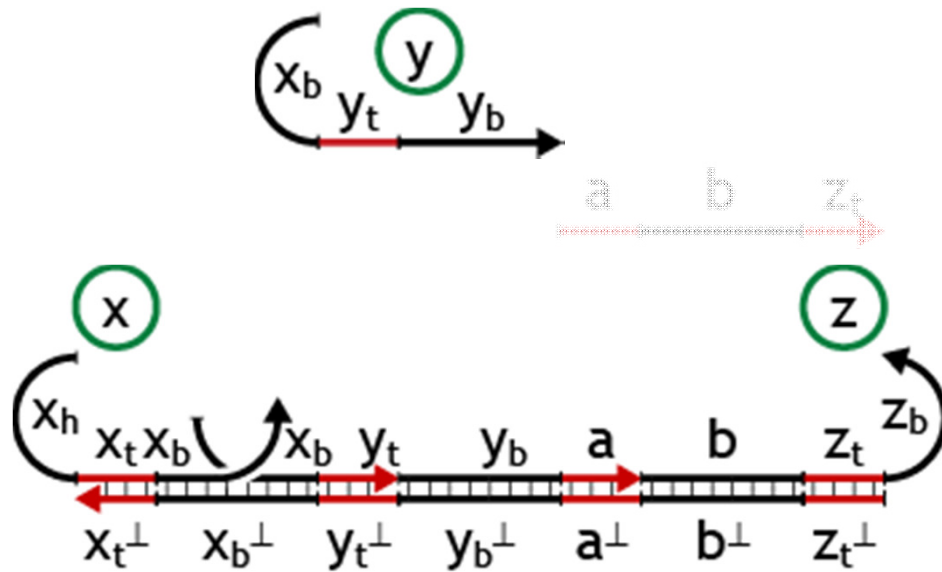
Join Gate



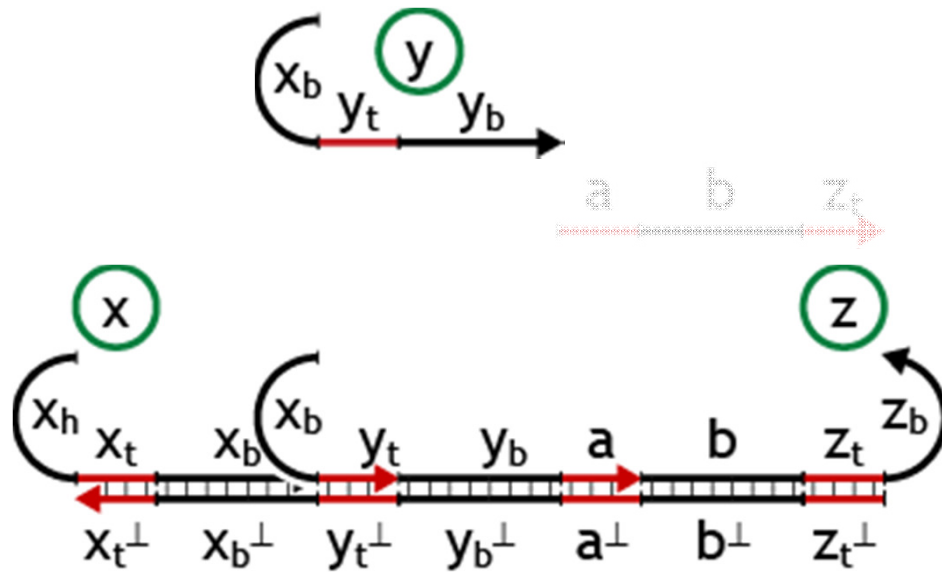
Join Gate



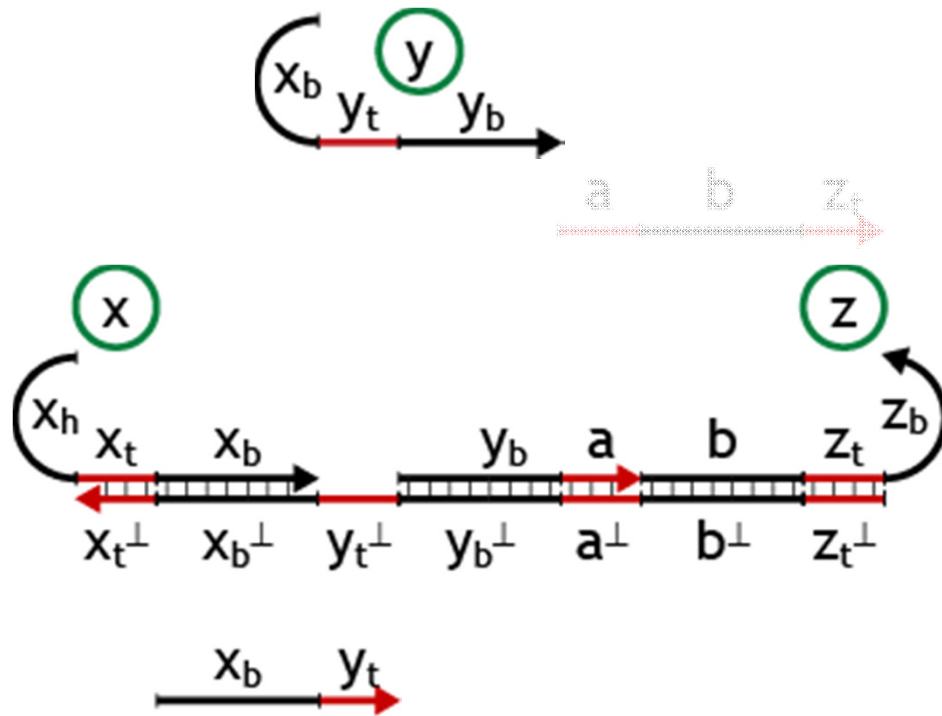
Join Gate



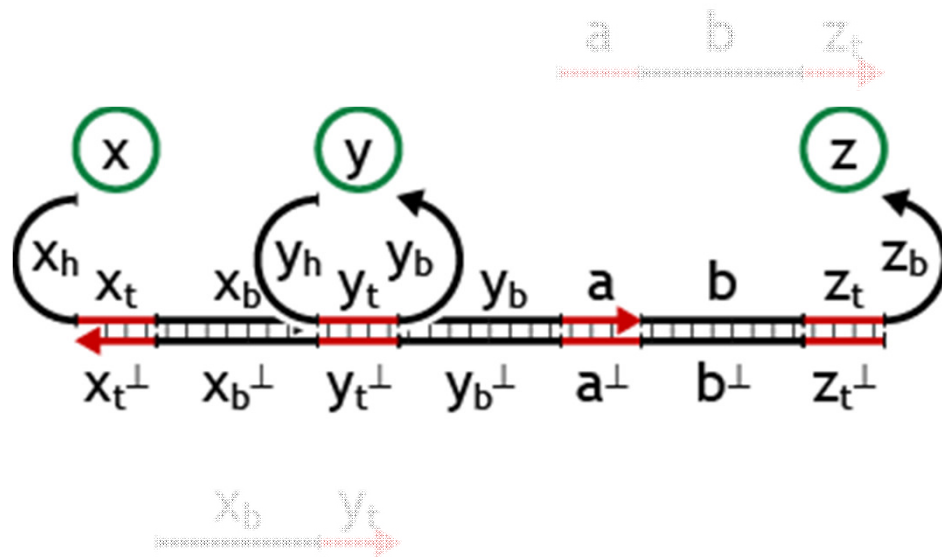
Join Gate



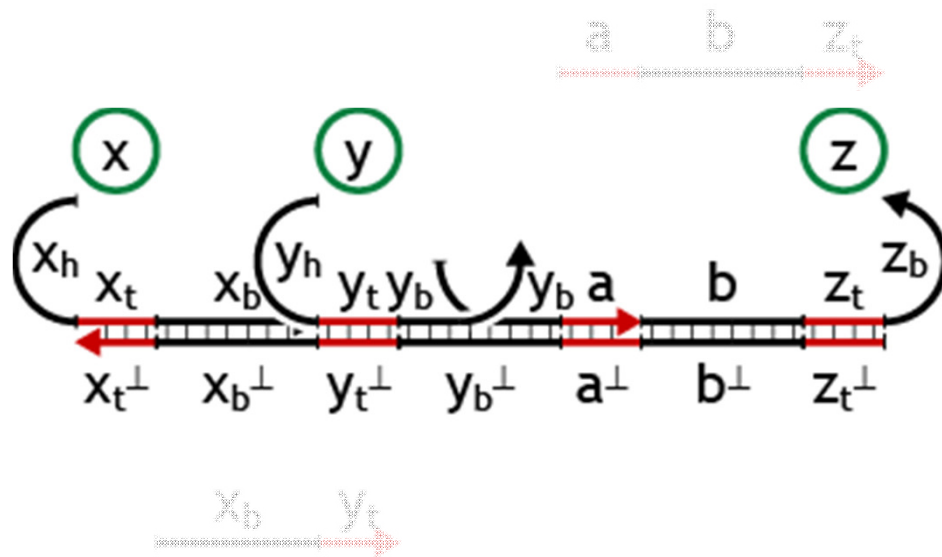
Join Gate



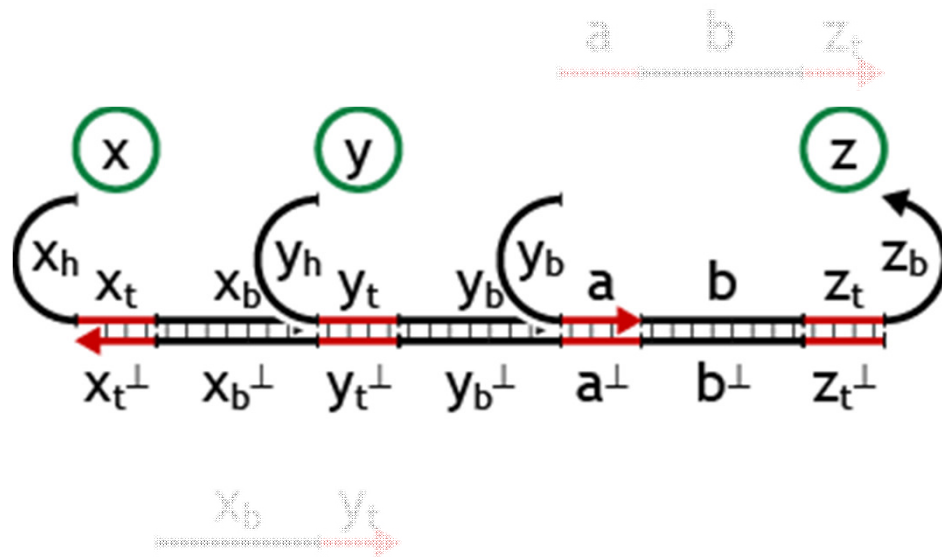
Join Gate



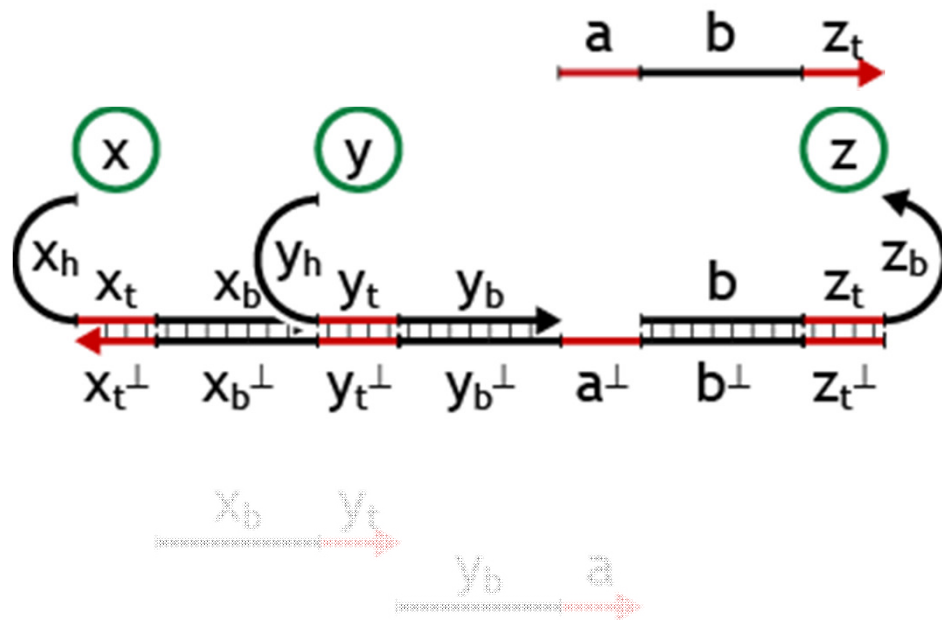
Join Gate



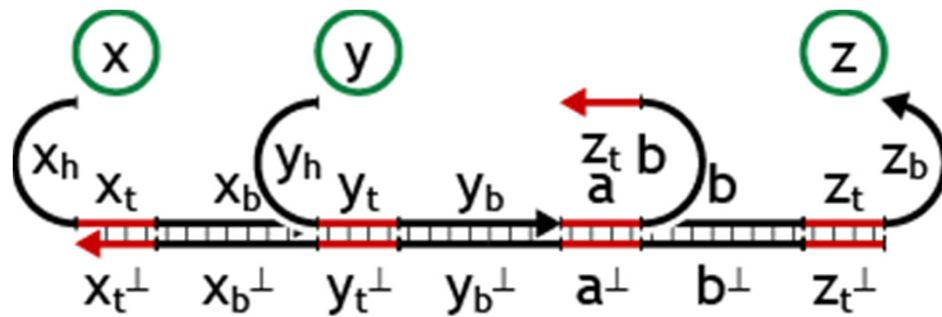
Join Gate



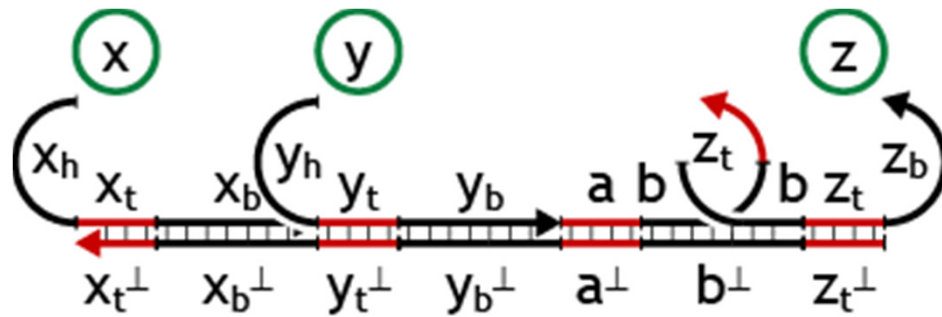
Join Gate



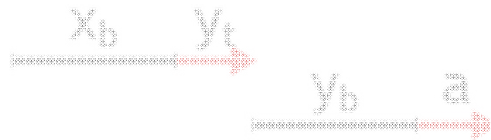
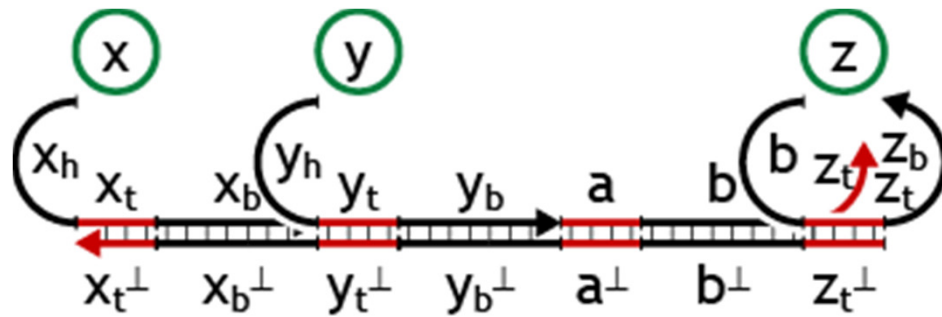
Join Gate



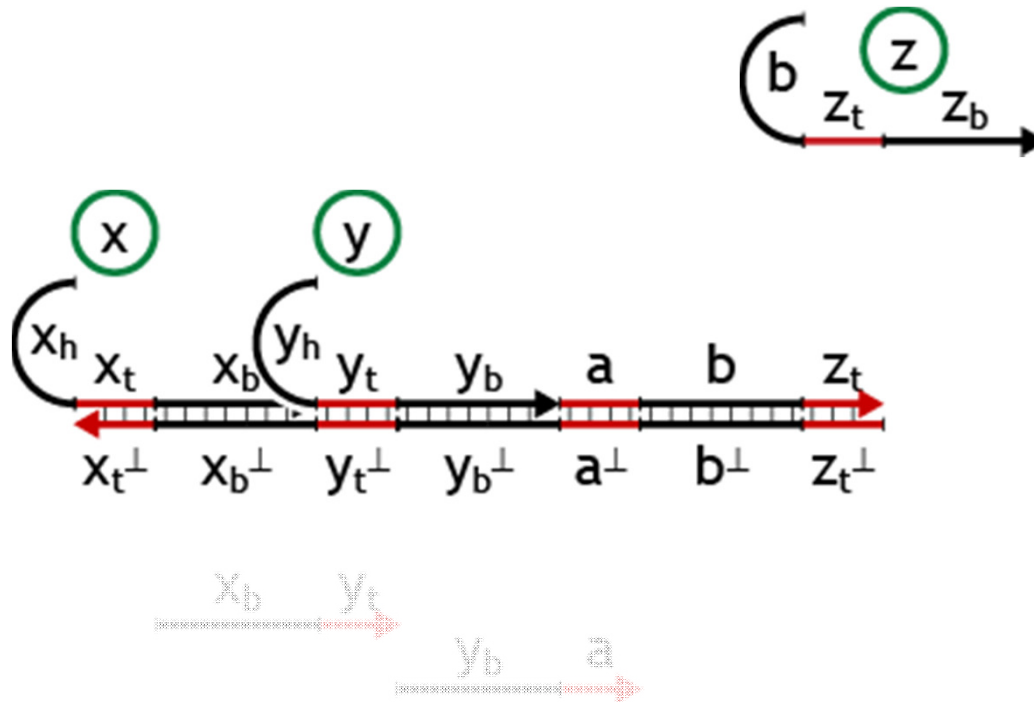
Join Gate



Join Gate



Join Gate

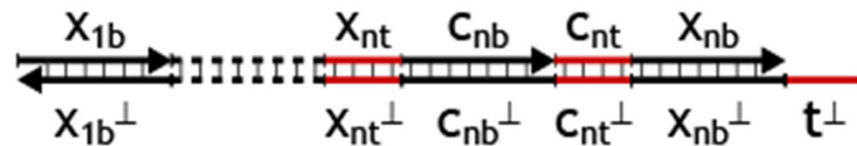
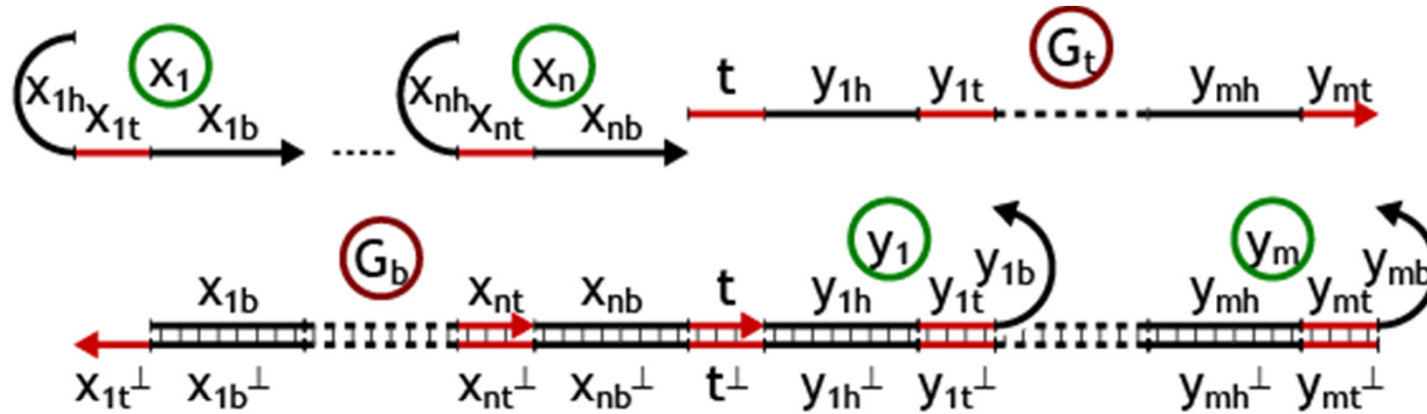


Gate Design Verification

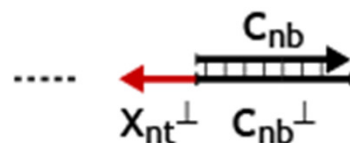
- Active garbage
 - The active join residuals slow down the performance of following joins.
 - → Add a garbage collector to remove the active residuals.
- Interference between gates
 - The join garbage collector interferes with the fork gate.
 - → Modify the fork gate to remove the interference.
- What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

$[x_1, \dots, x_n] \cdot [y_1, \dots, y_m]$ General Join/Fork Gate

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

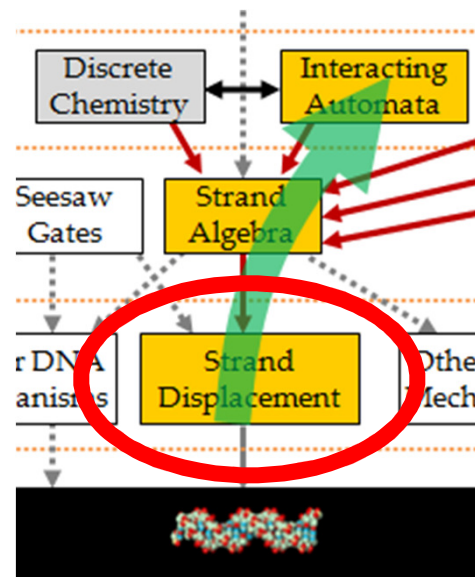


Garbage collection



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

Strand Displacement Intermediate Language



Matthew Lakin
Simon Youssef
Andrew Phillips

Syntax

A programming language for composable DNA circuits

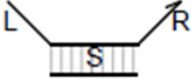
Andrew Phillips* and Luca Cardelli

Microsoft Research, Cambridge CB3 0FB, UK

A. Syntax of DNA molecules D

Upper strand with sequence complementary to S	\xrightarrow{S} $\langle S \rangle$
Molecule with segments G_1, \dots, G_k	$\xleftarrow{G_1 \ G_2 \ \dots \ G_k}$ $G_1 : G_2 : \dots : G_k$
Parallel molecules D_1, \dots, D_k	$D_1 \ D_2 \ \dots \ D_k$ $D_1 \ \ D_2 \ \ \dots \ \ D_k$
Molecules D with private domains N_1, \dots, N_k	$\text{new } (N_1, \dots, N_k) \ D$

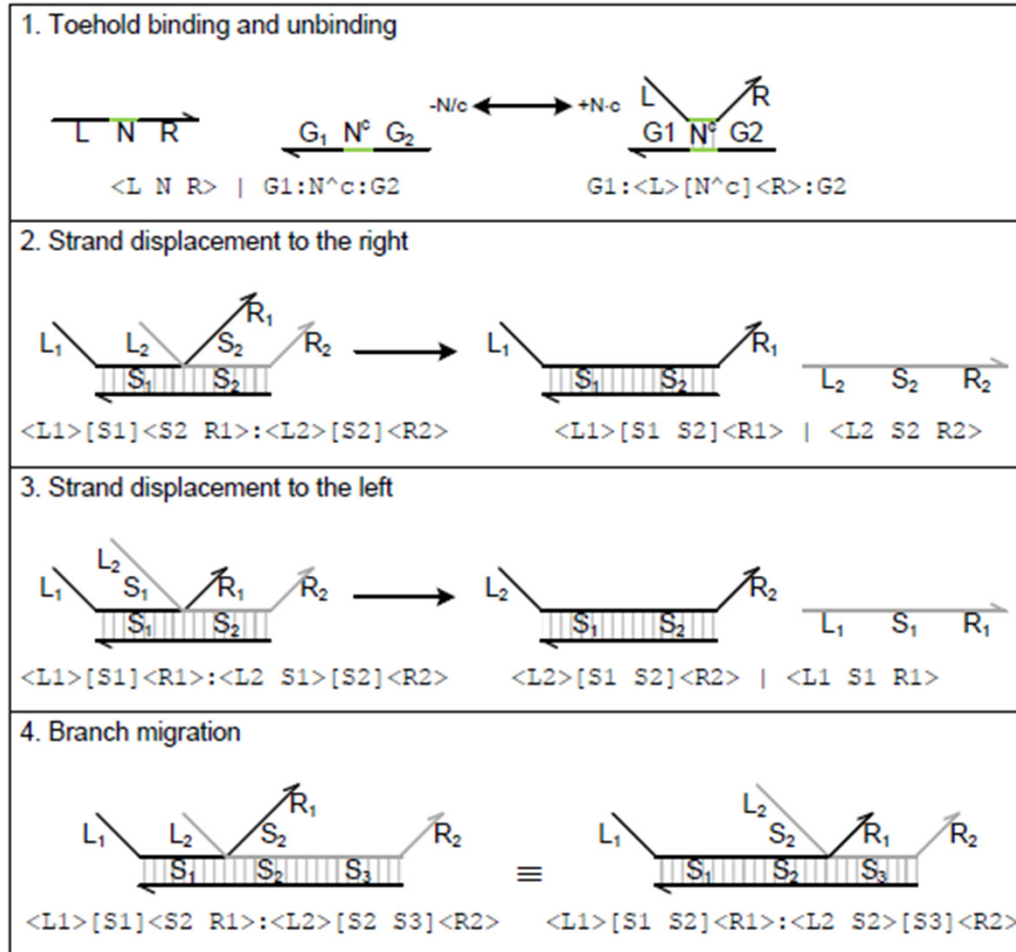
B. Syntax of DNA segments G

Lower strand with toehold N^c	$\underline{N^c}$ N^c
Double strand with sequence S and overhangs L, R	 $\langle L \rangle [S] \langle R \rangle$

C. Syntax of DNA sequences S, L, R

Sequence of domains O_1, \dots, O_k	$O_1 \ O_2 \ \dots \ O_k$ $o_1 \ o_2 \ \dots \ o_k$
---------------------------------------	---

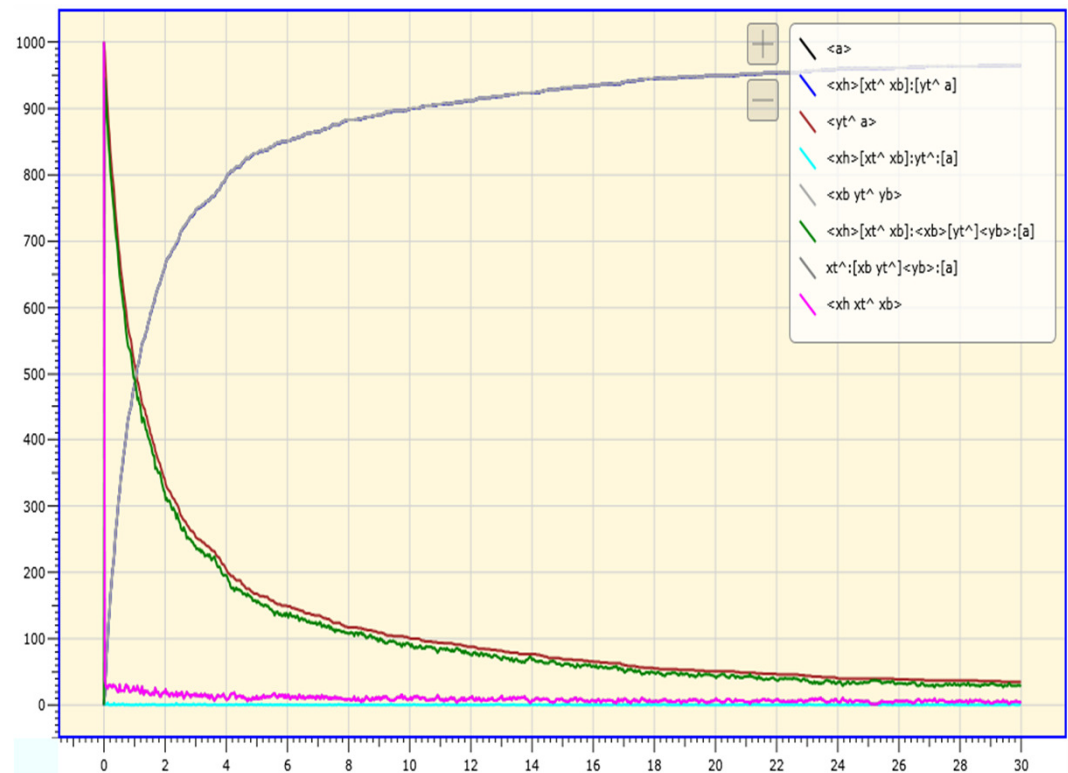
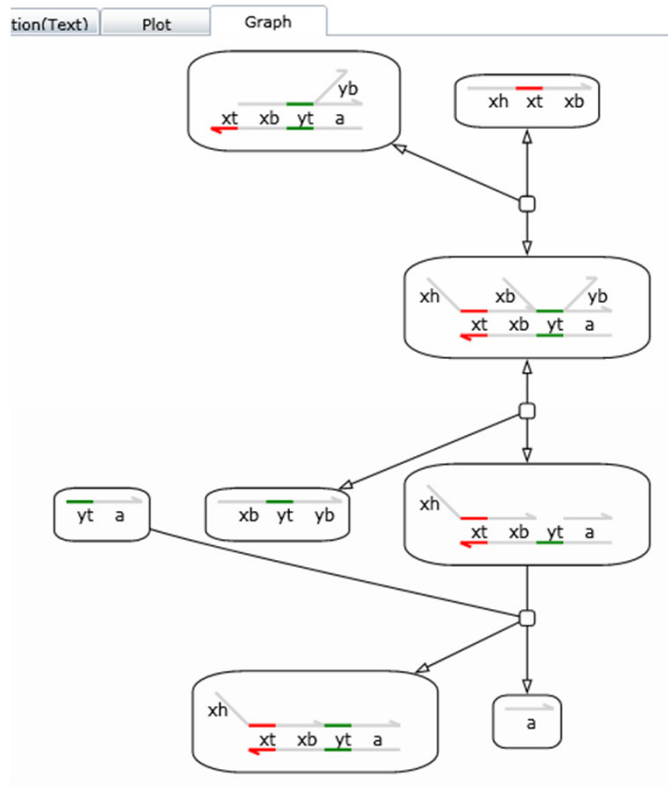
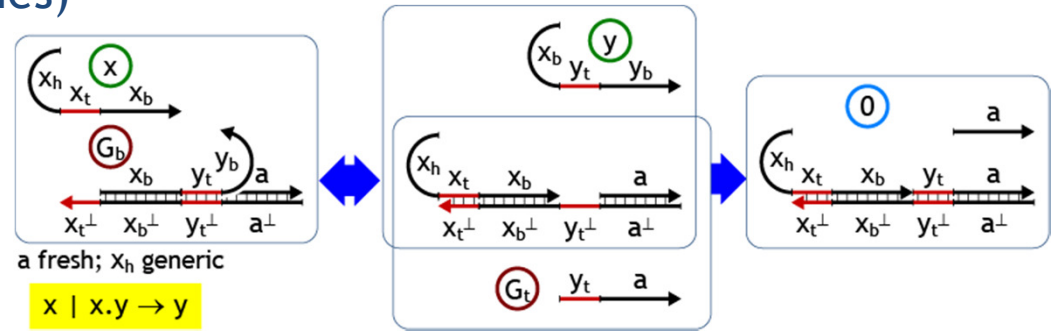
Dynamics



Strand Displacement Analysis Tool

1 Transducer gate $x.y$ (3 initial species)

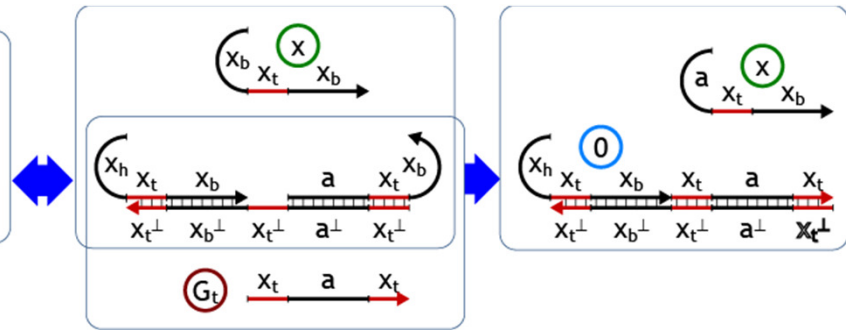
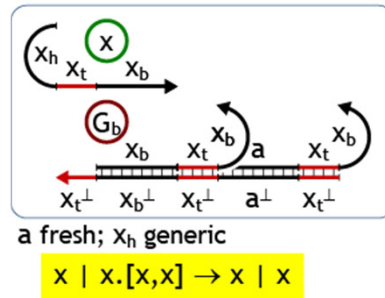
```
directive sample 30.0 1000
new xt@1.0,1.0
new yt@1.0,1.0
( 1000 <xh xt^ xb>
| 1000 * xt^:[xb yt^]<yb>:[a]
| 1000 * <yt^ a>
)
```



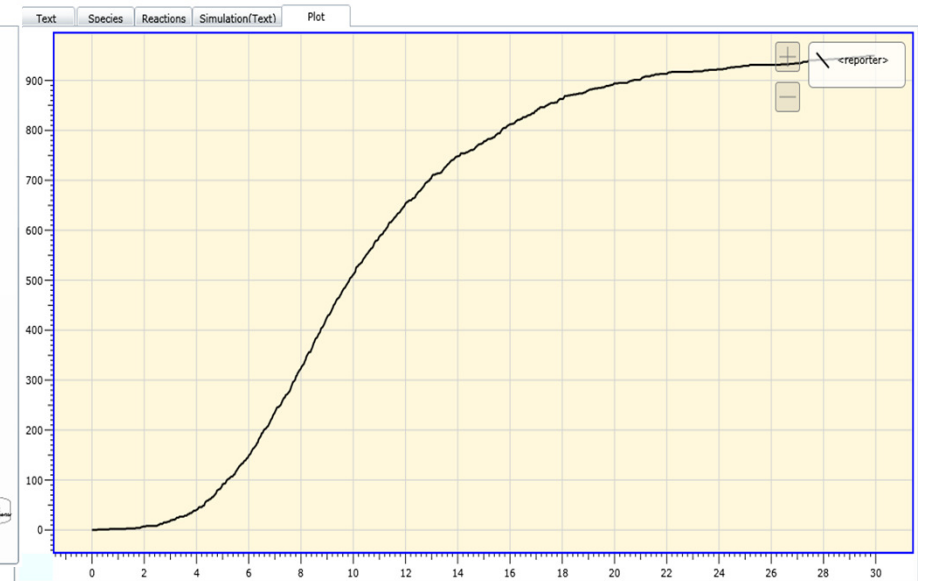
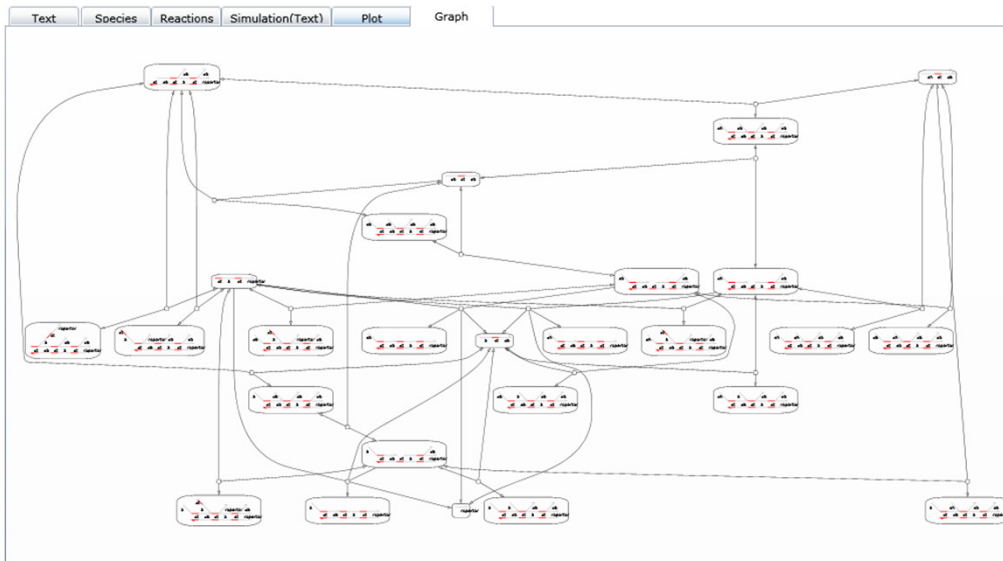
Strand Displacement Analysis Tool

Fork Chain Reaction $x.[x,x]$ (3 initial species)

```
directive sample 30.0 1000
directive plot "<reporter>"
new xt@ 1.0 , 1.0
( 1 * <xh xt^ xb>
| 1000 * xt^:[xb xt^]<xb>:[a xt^]<xb>:[reporter]
| 1000 * <xt^ a xt^ reporter>
)
```



26 Species, 20 Reactions



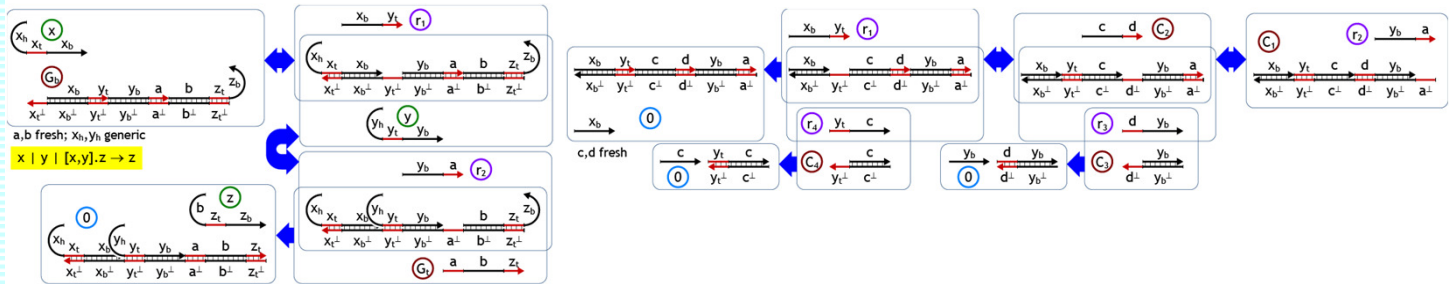
Strand Displacement Analysis Tool

1 Join gate with garbage collection [x,y].z (8 initial species)

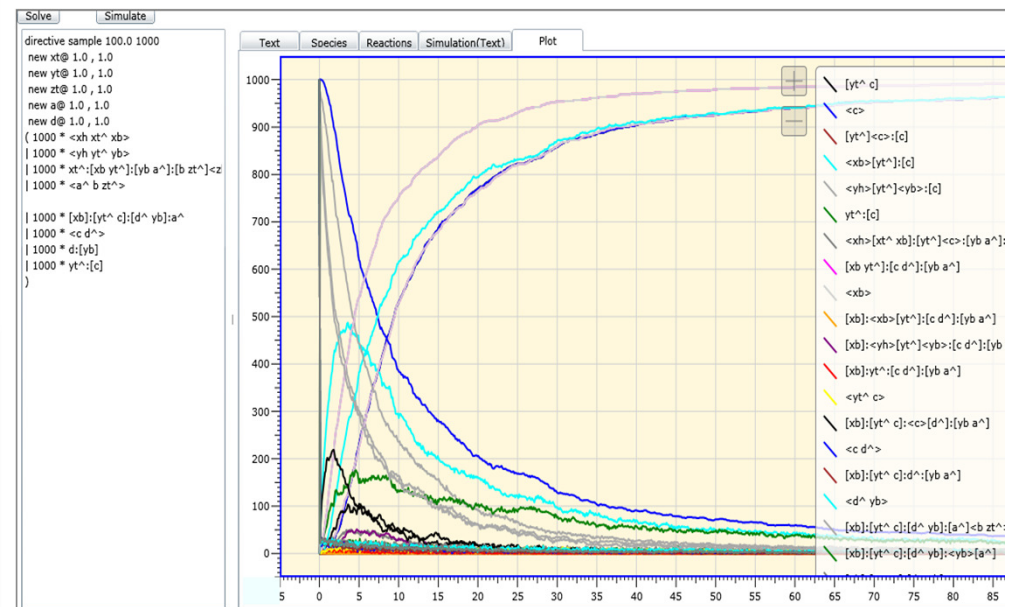
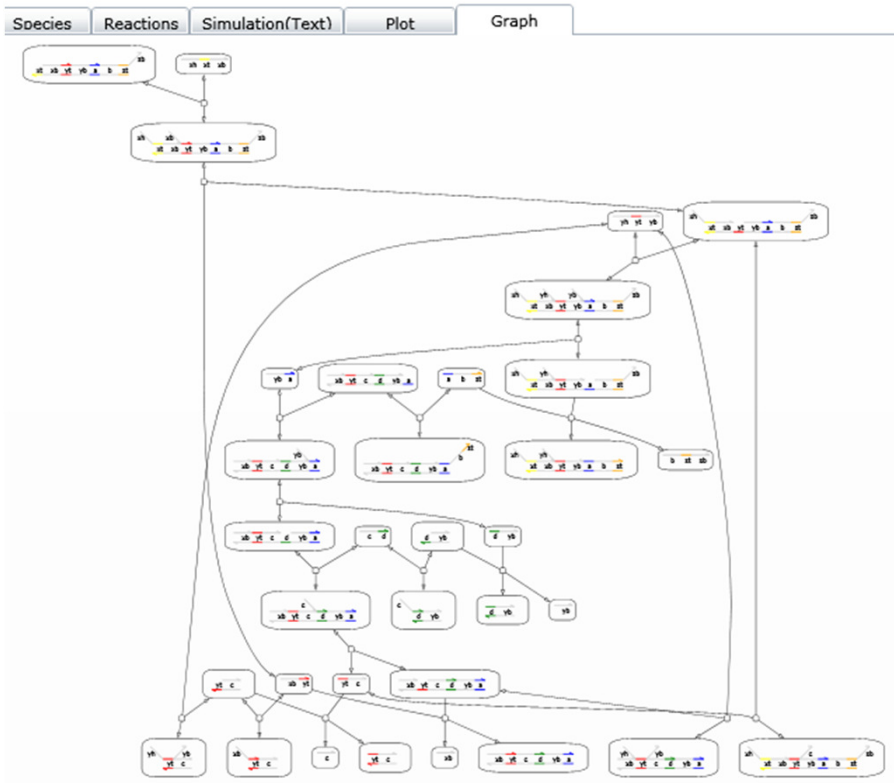
```

directive sample 1000.0 1000
new xt@ 1.0, 1.0
new yt@ 1.0, 1.0
new zt@ 1.0, 1.0
new a@ 1.0, 1.0
new d@ 1.0, 1.0
( 1000 * <xh xt^ xb>
| 1000 * <yh yt^ yb>
| 1000 * xt^:[xb yt^]:[yb a^]:[b zt^]<z>
| 1000 * <a^ b zt^>
)

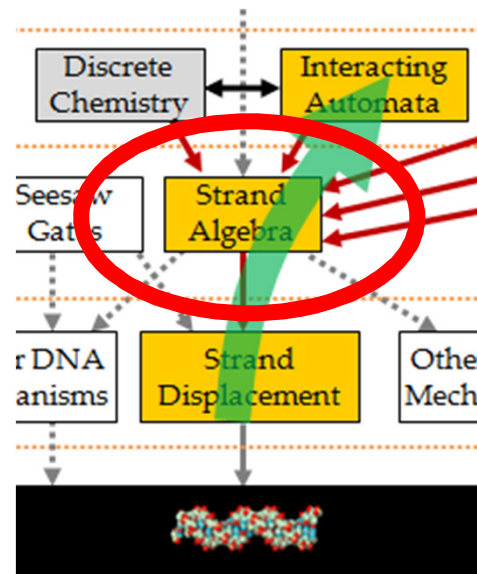
| 1000 * [xb]:[yt^ c]:[d^ yb]:a^
| 1000 * <c d^>
| 1000 * d^:[yb]
| 1000 * yt^:[c]
)
    
```



34 Species, 18 Reactions



Strand Algebra



Strand Algebra

$n \times m$ gates

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

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

Reaction Rule

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


Auxiliary rules (axioms of diluted well-mixed solutions)

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

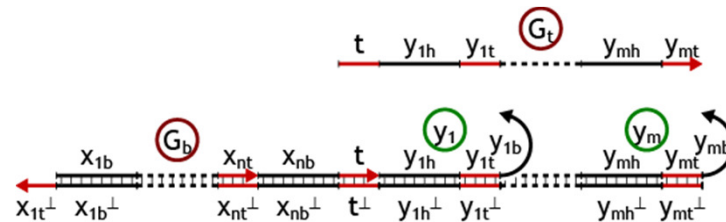
Where \equiv is a congruence relation (syntactical ‘chemical mixing’) with $P^* \equiv P | P^*$ for unbounded populations.

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

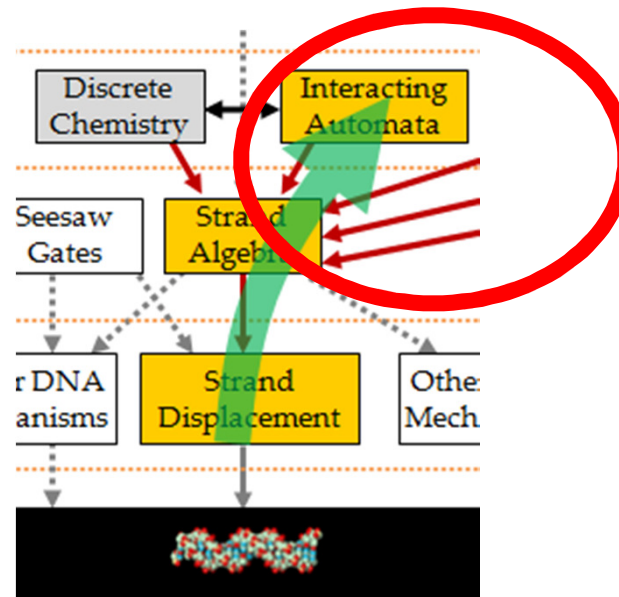
• $\text{compile}(P \mid P') = \text{mix}(\text{compile}(P), \text{compile}(P'))$

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

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

Abstract Machines



Chemical Reaction Networks

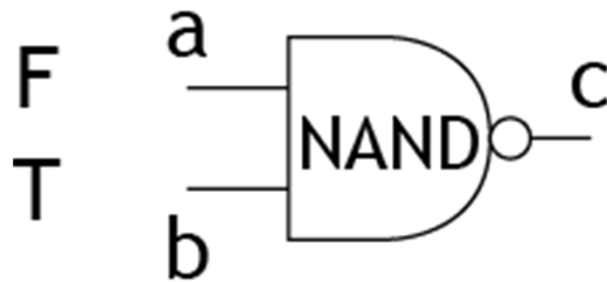
Implementing an arbitrary finite chemical system in DNA
with asymptotically correct kinetics
Soloveichick & al. DNA 15

Species become signals
Reactions become gates



Boolean Networks

Boolean Networks to Strand Algebra



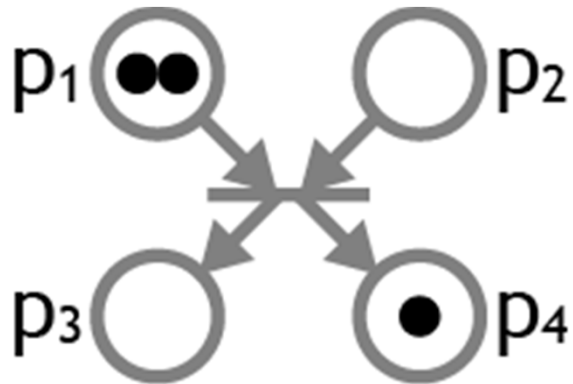
$$\begin{aligned} & ([a_F, b_F] \cdot c_T)^* \mid \\ & ([a_F, b_T] \cdot c_T)^* \mid \\ & ([a_T, b_F] \cdot c_T)^* \mid \\ & ([a_T, b_T] \cdot c_F)^* \mid \\ & a_F \mid b_T \end{aligned}$$

- 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

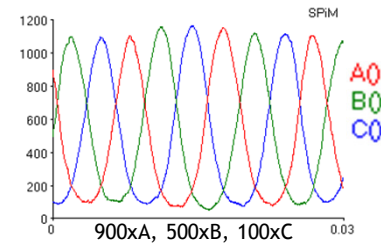
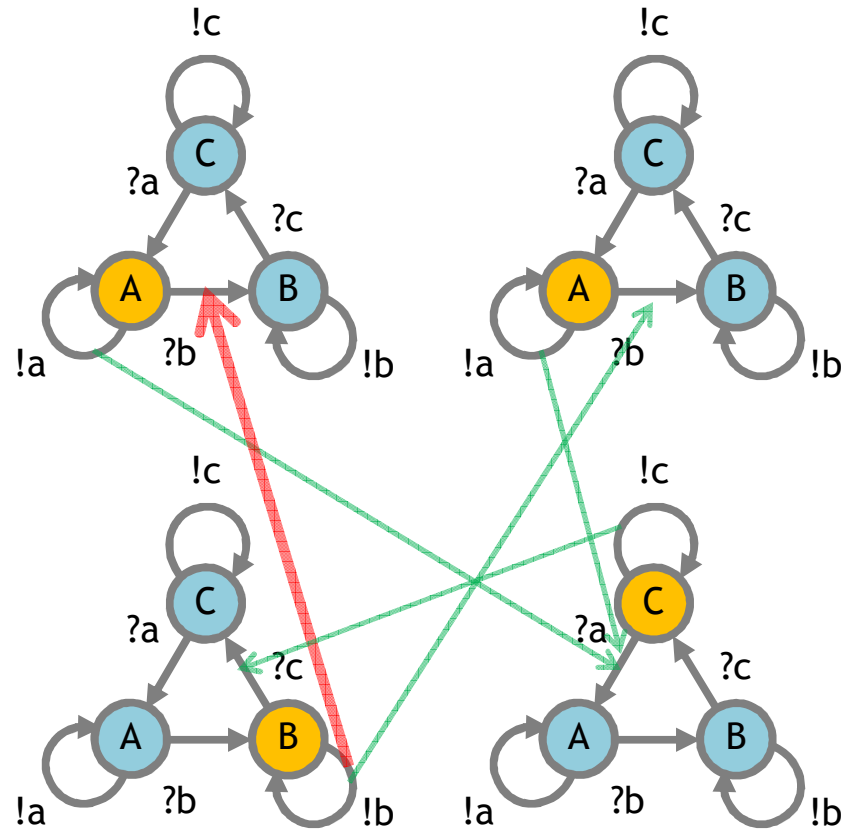
Petri Nets to Strand Algebra

Transitions as Gates
Place markings as Signals



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

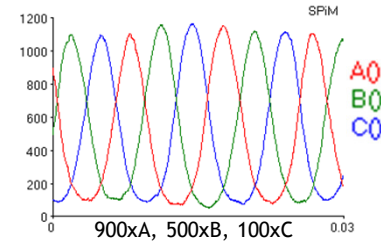
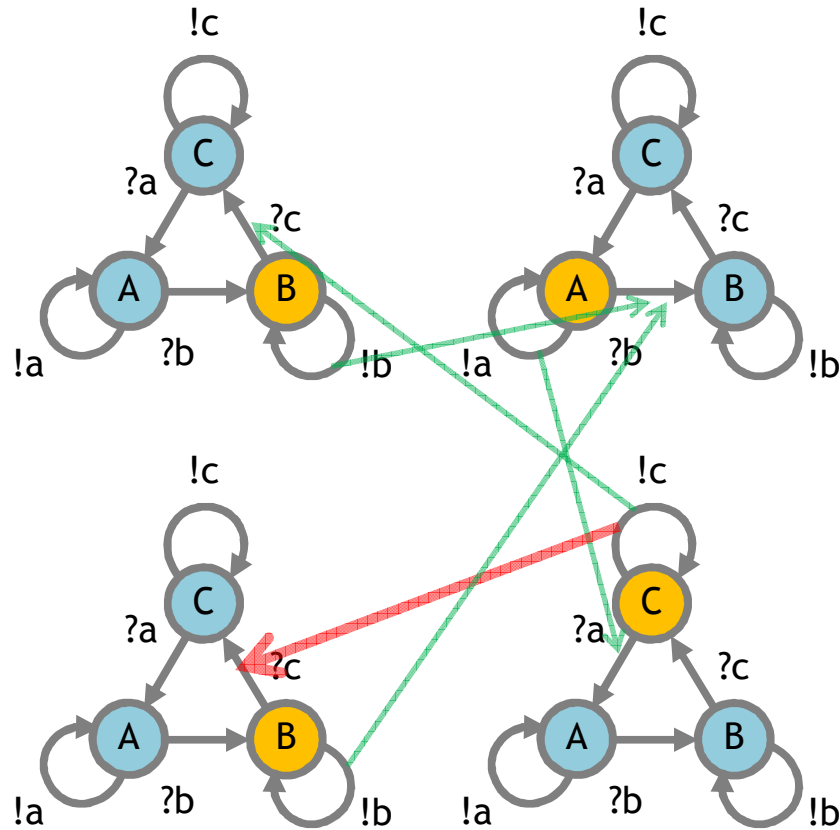
Interacting Automata



$$\begin{aligned}
 &([A, B]. [B, B])^* \mid \\
 &([B, C]. [C, C])^* \mid \\
 &([C, A]. [A, A])^* \mid \\
 &A \mid A \mid B \mid C
 \end{aligned}$$

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

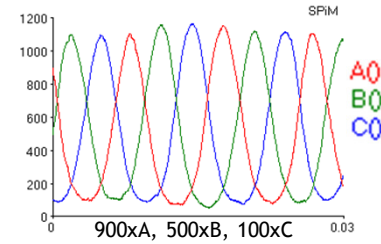
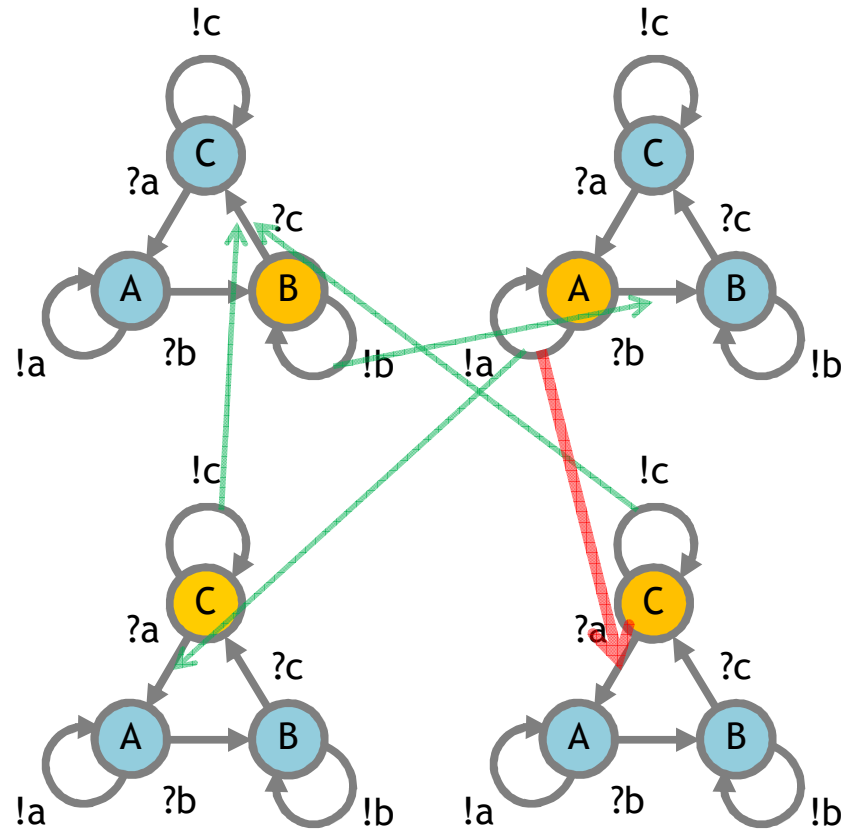
Interacting Automata



$$\begin{aligned}
 &([A, B]. [B, B])^* \mid \\
 &([B, C]. [C, C])^* \mid \\
 &([C, A]. [A, A])^* \mid \\
 &A \mid B \mid B \mid C
 \end{aligned}$$

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

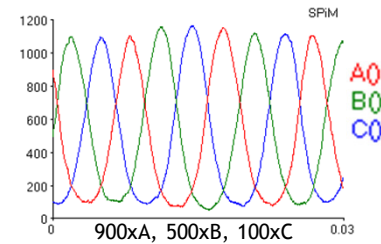
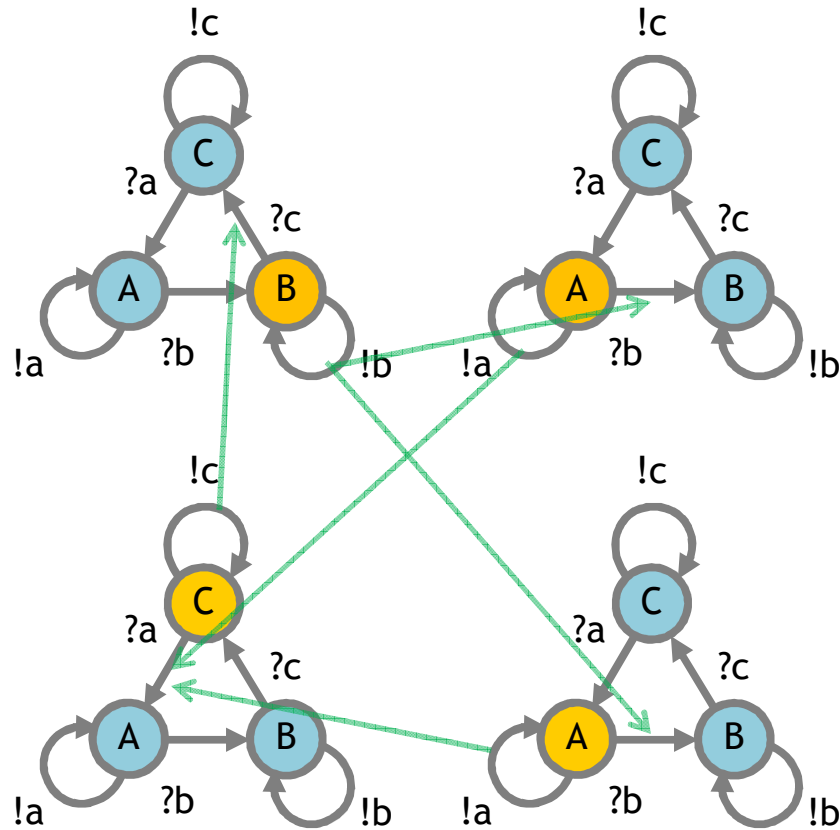
Interacting Automata



$$\begin{aligned}
 &([A, B]. [B, B])^* \quad | \\
 &([B, C]. [C, C])^* \quad | \\
 &([C, A]. [A, A])^* \quad | \\
 &A \quad | \quad B \quad | \quad C \quad | \quad C
 \end{aligned}$$

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

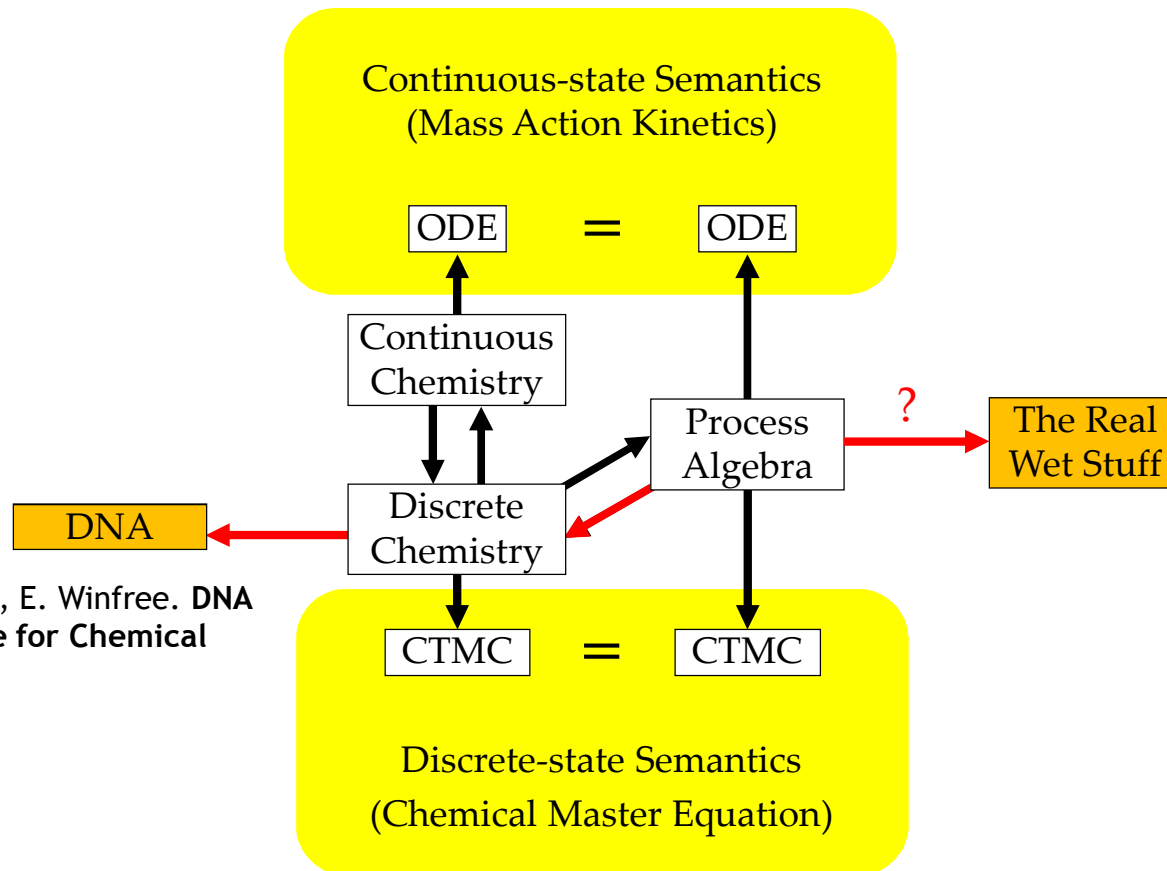
Interacting Automata



$$\begin{aligned}
 &([A, B]. [B, B])^* \quad | \\
 &([B, C]. [C, C])^* \quad | \\
 &([C, A]. [A, A])^* \quad | \\
 &A \quad | \quad A \quad | \quad B \quad | \quad C
 \end{aligned}$$

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

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)

Conclusions

Conclusion

- 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).