Microsoft Engineering Excellence

# Molecular Programming

Luca Cardelli, Principal Researcher, MSR Cambridge

Programming Principles and Tools

2012-12-06

Microsoft

# Objectives

- The promises of Molecular Programming:
  - In Science & Medicine
  - In Engineering
  - In Computing
- The current practice of Molecular Programming
  - DNA technology
  - Molecular languages and tools
  - Example of a molecular algorithm

# The Hardware Argument

Smaller and smaller things can be built

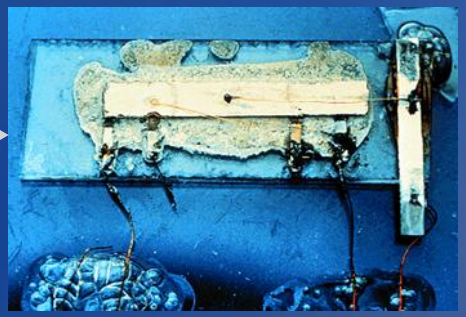
# Smaller and Smaller

First working transistor

John Bardeen and Walter Brattain , Dec. 23, 1947

First integrated circuit

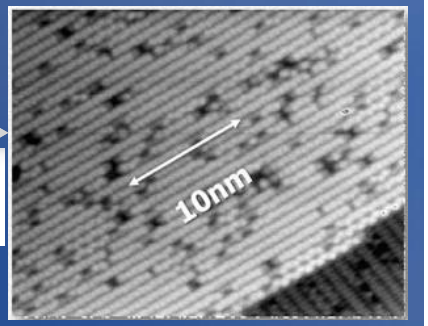
Jack Kilby, Sep. 1958.



**50 years later**

25nm NAND flash

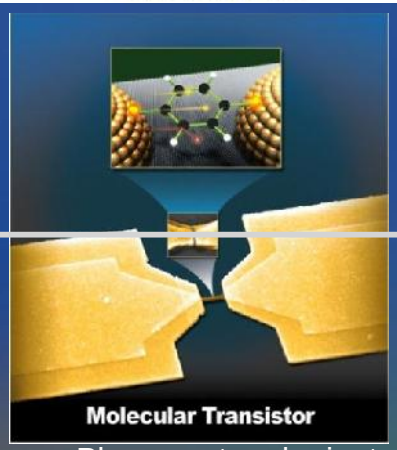
Intel&Micron, Jan. 2010. ~50atoms



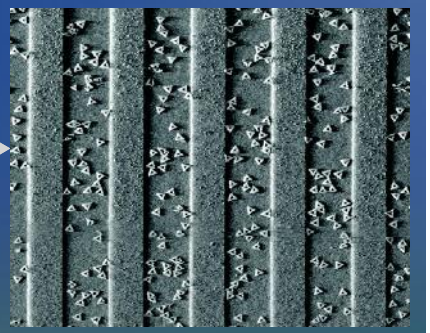

Scanning tunneling microscope image of a silicon surface showing 10nm is ~20 atoms across

Single molecule transistor

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




Molecules on a chip

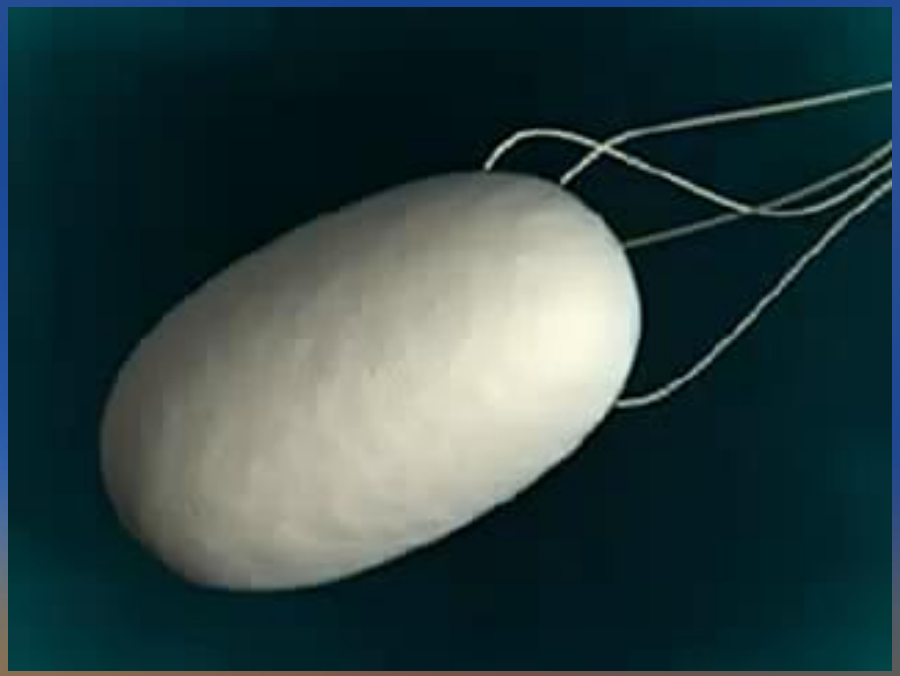


Placement and orientation of individual DNA shapes on lithographically patterned surfaces. Nature Nanotechnology 4, 557 - 561 (2009).

~10 Moore's Law cycles left!

# Building the *Smallest* Things

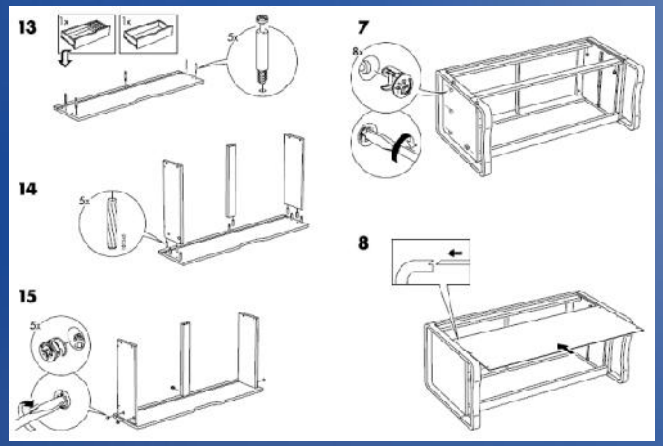
- How do we build structures that are by definition smaller than your tools?
- Basic answer: you can't. Structures (and tools) should build themselves!
- By *programmed self-assembly*



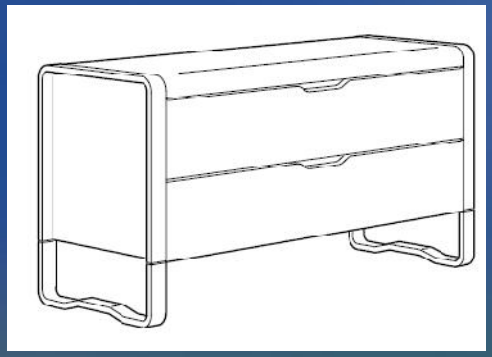
www.youtube.com/watch?v=Ey7Emmddf7Y

# 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…



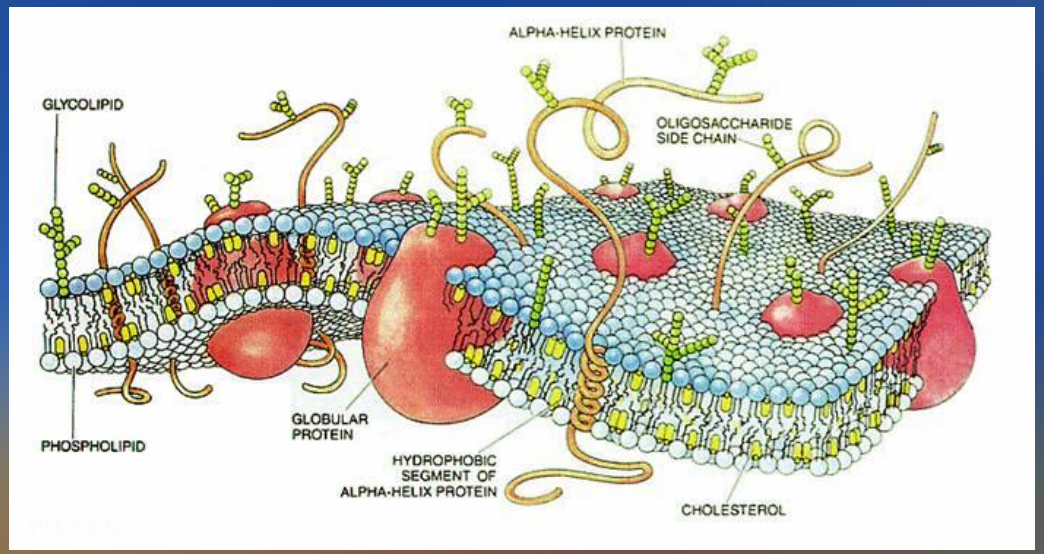

Add water



http://www.ikea.com/ms/en_US/customer_service/assembly_instructions.html

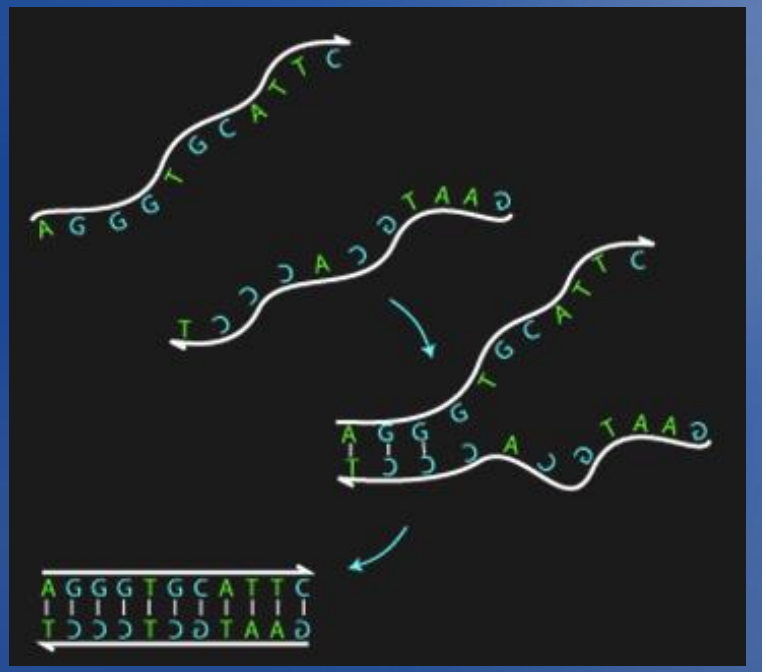
# Programmed Self-Assembly

## Proteins




Wikimedia

## DNA/RNA



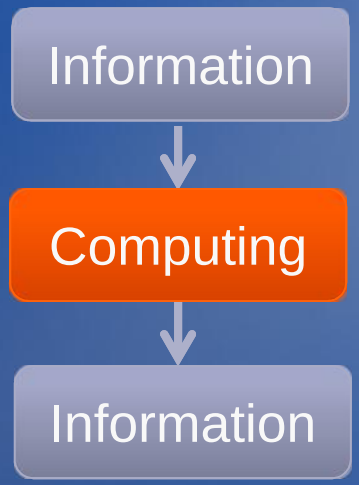

## Membranes

# The Software Argument

Smaller and smaller things can be programmed

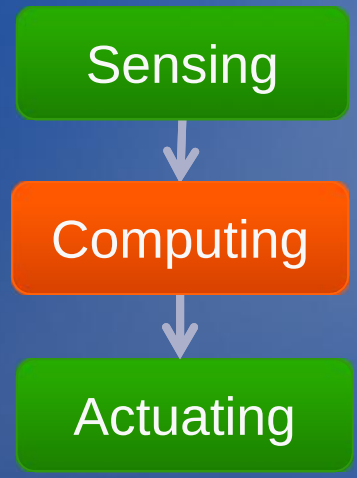
# We can program...

- Computers
  - Completely!



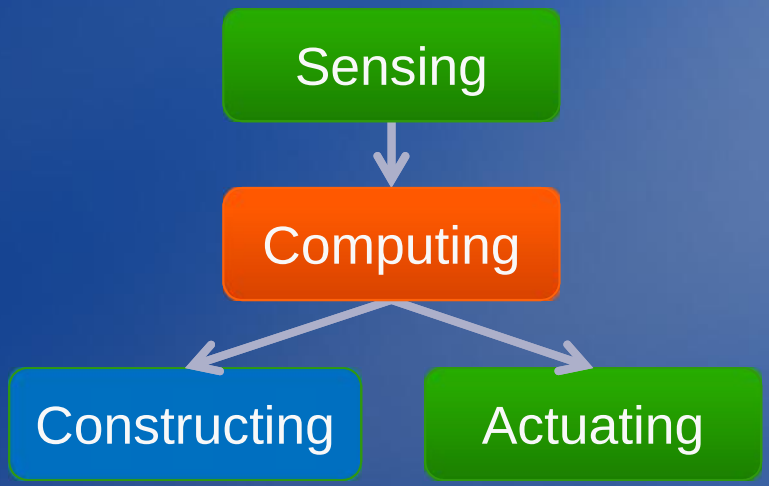
# We can program...

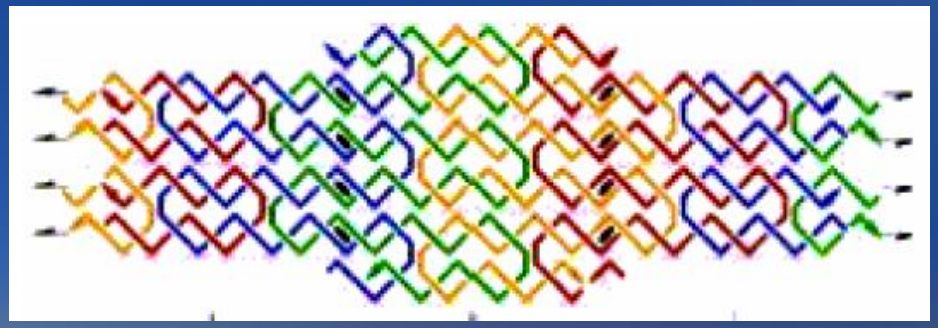
- Physical systems
  - Completely!
  - Modulo sensors/actuator capabilities



# We can program...

- Matter
  - Completely
  - Directly!
- Which matter?
  - Currently: only DNA/RNA
  - But this is not so limiting...



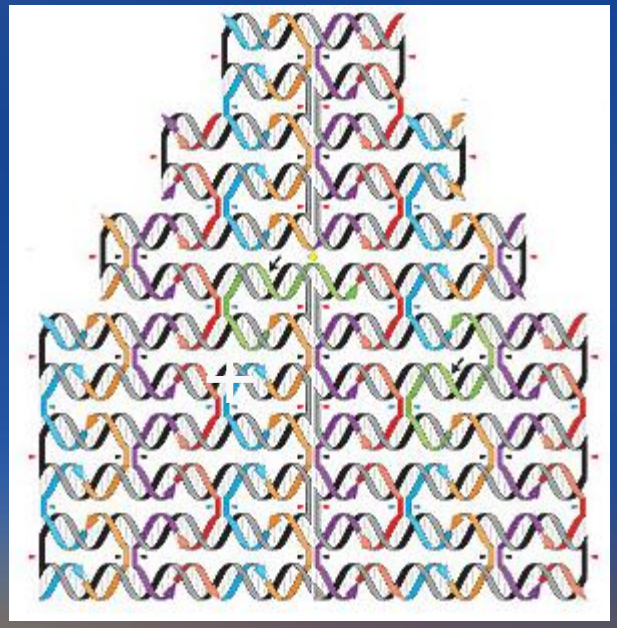



# What can we do with it?

- Organize ANY matter [caveats apply]
- Execute ANY kinetics [caveats: up to time scaling]
- Control molecular systems
- Interface to Biology

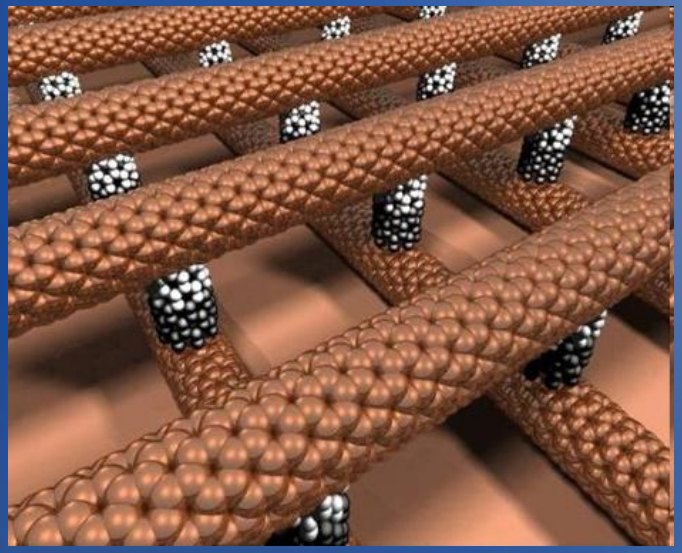
# Organizing Any Matter

- Use one kind of programmable matter (e.g. DNA)
- To organize (almost) ANY matter through it



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

6 nm grid of individually addressable DNA 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*

# Executing Any Desired Kinetics

- The kinetics of any finite network of chemical reactions among abstract species, can be executed (physically) with especially programmed DNA molecules

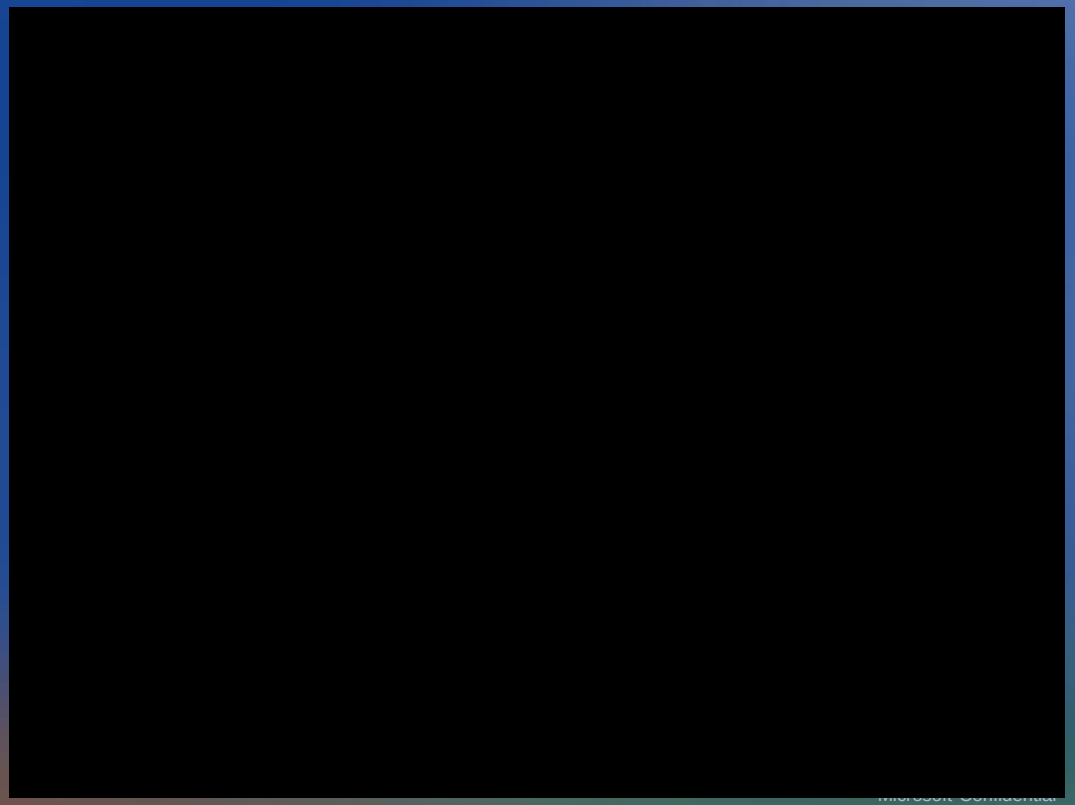
- Chemical reactions as an executable programming language for dynamical systems!

**Two-Domain DNA Strand Displacement**

Luca Cardelli (Microsoft Research)

**DNA as a universal substrate for chemical kinetics**

David Soloveichik[a,1], Georg Seelig[a,b,1], and Erik Winfree[c,1]



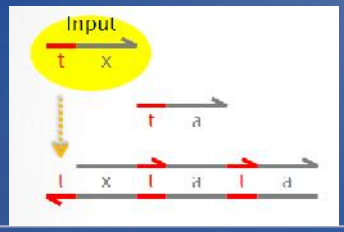
# Building Molecular Controllers

- All the components of nanocontrollers can already be built entirerly and solely with DNA, and interfaced to the environment.

DNA Aptamers

Sensing

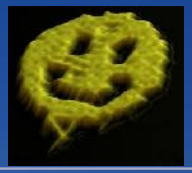
Computing



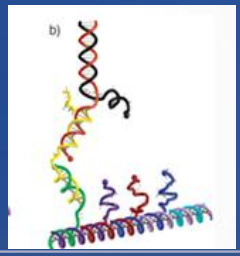

DNA Logical Gates

Constructing

Actuating



Self-assembling DNA Tiles




DNA Walkers & Tweezers

# Interfacing to Biology

## A doctor in each cell

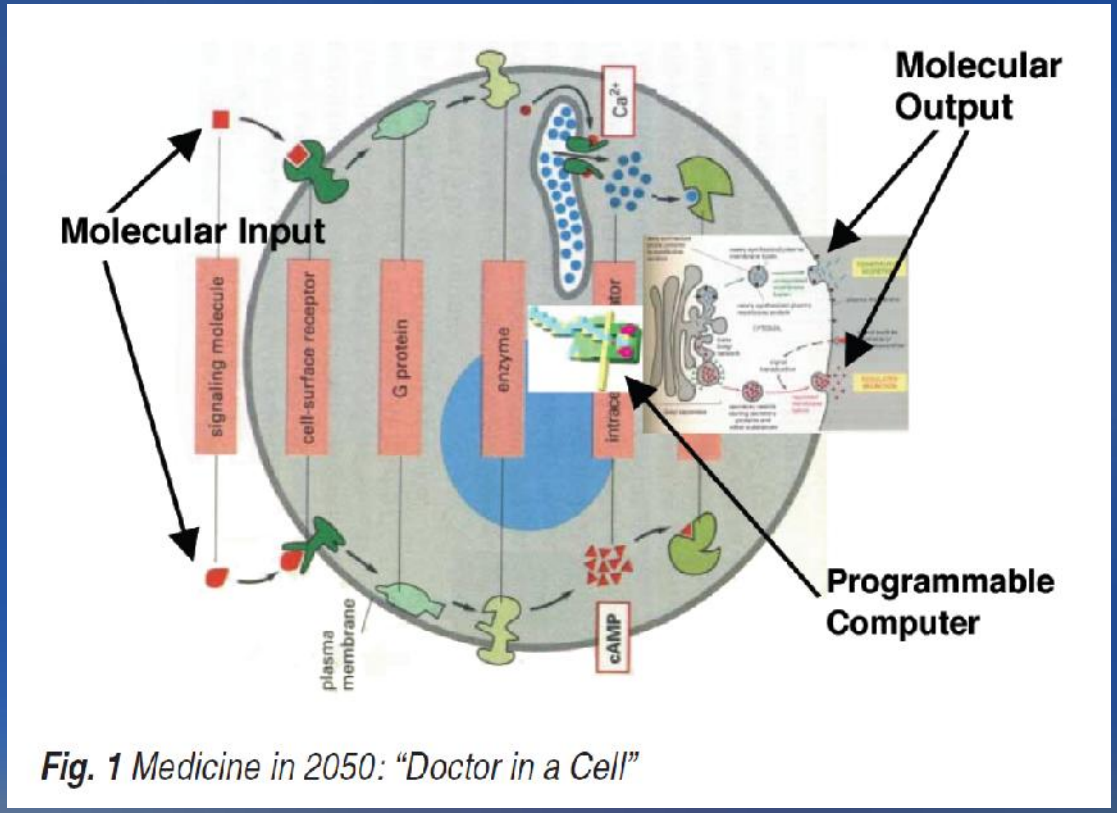



*Fig. 1* Medicine in 2050: "Doctor in a Cell"

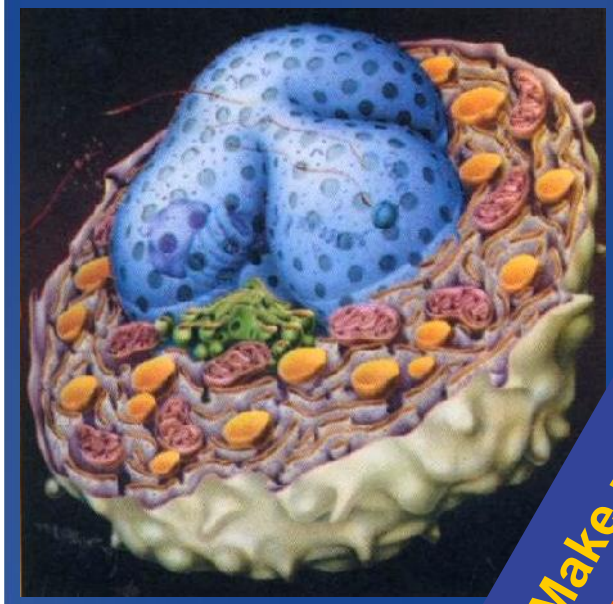
Ehud Shapiro
Rivka Adar
Kobi Benenson
Gregory Linshitz
Aviv Regev
William Silverman

**Molecules and computation**

# The Biological Argument

Biological systems are already ‘molecularly programmed’

# Abstract Machines of Biology



H.Lodish & al. Molecular Cell Biology 4th ed.

Regulation

**Gene Machine**
Nucleotides

Make proteins

Send signals

Direct construction

Confine genome and regulators

Hold receptors, host reactions

Enact fusion, fission

**Protein Machine**
Aminoacids

**Membrane Machine**
Phospholipids

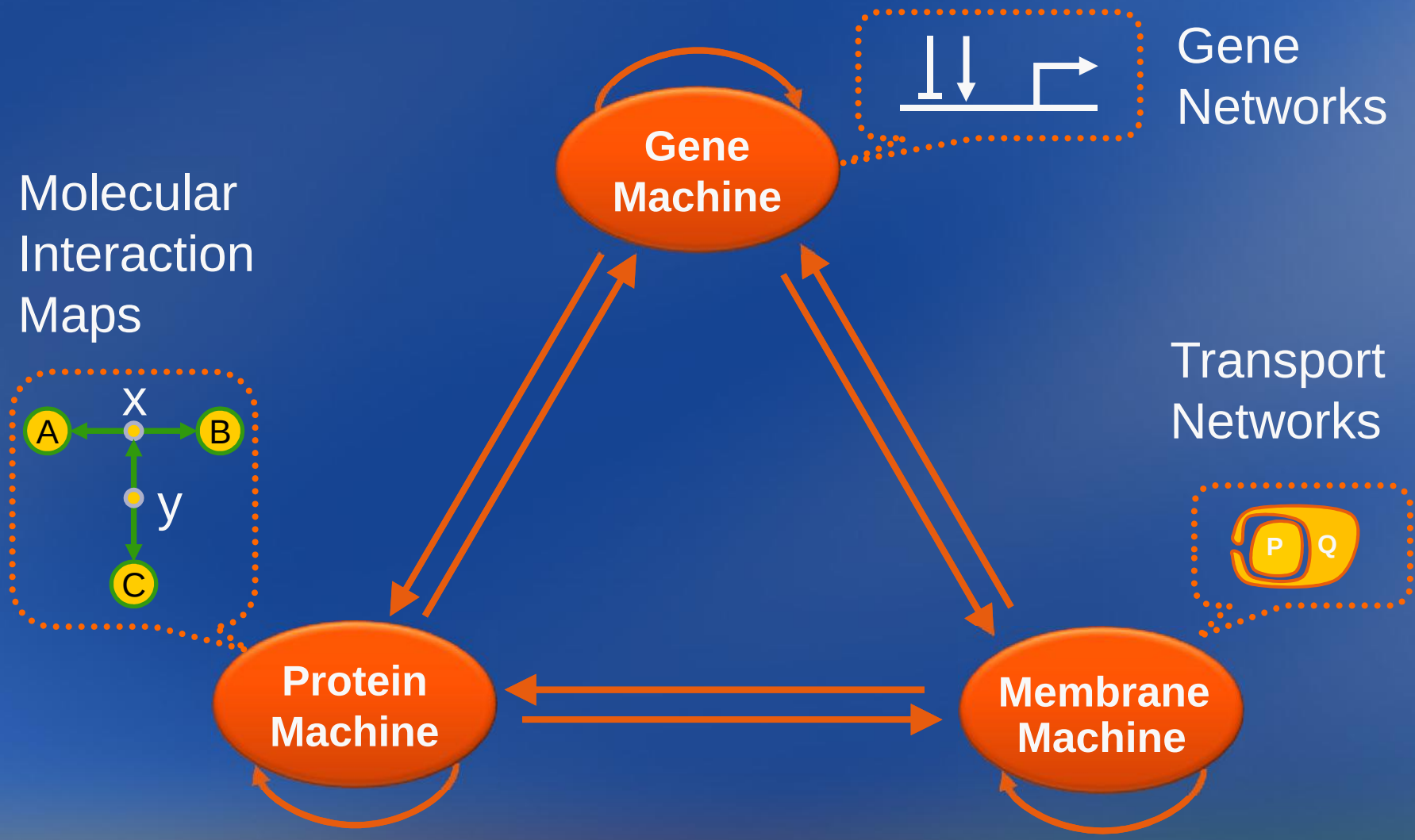
Metabolism, Propulsion
Signaling, Transport

**Glycan Machine**
Sugars

Surface and
Extracellular Features

Confinement, Storage
Bulk Transport

# Languages


Gene Networks
Gene Machine
Molecular Interaction Maps
x
A
B
y
C
Transport Networks
P
Q
Protein Machine
Membrane Machine

# But ...

- Biology is programmable, but not by us!

- Still work in progress:
  - Gene networks are being programmed in synthetic biology, but using existing ‘parts’
  - Protein networks are a good candidate, unfortunately we cannot yet effectively design proteins
  - Transport networks are being looked at for programming microfluidic devices manipulating vesicles

# Molecular Languages

... that we can deal with

# Long-Term Action Plan

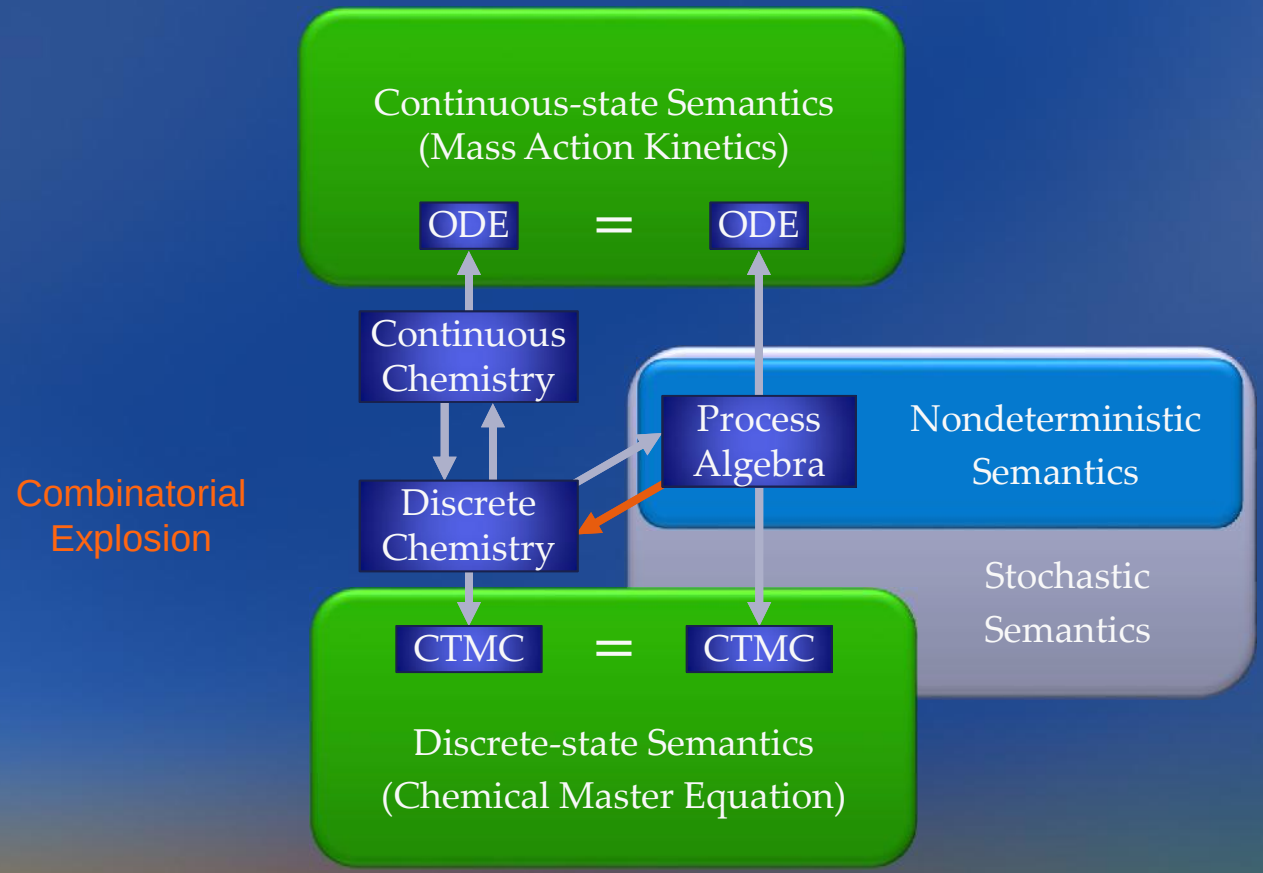
- Building a full pipeline
  - Mathematical Foundations [~ concurrency theory in the 80's]
  - Programming Languages [~ software engineering in the 70's]
  - Analytical Methods and Tools [~ formal methods in the 90's]
  - Device Architecture and Manufacturing [~ electronics in the 60's]
- Molecular Compilers
  - Front end: theory-backed analyzable programming languages
  - Back end: executable molecular systems
  - Requiring techniques for mastering complexity and analyzing system performance/safety ... mostly familiar to us
- No "alien technology"!
  - Do not use components (from Biology) we do not understand how to build ourselves. [David Soloveichik]

# Our Assembly Language: Chemistry

- A Lingua Franca between Biology, Dynamical Systems, and Concurrent Languages
- Chemical Reaction Networks
  - $A + B \rightarrow_r C + D$ (the program)
- Ordinary Differential Equations
  - $d[A]/dt = -r[A][B]$ … (the behavior)
- Rich analytical techniques based on Calculus
- But prone to combinatorial explosion
  - E.g., due to the peculiarities of protein interactions

# Chemistry as a Concurrent Language

- A connection with the theory of concurrency
  - Via Process Algebra and Petri Nets



# How do we “run” Chemistry?

- Chemistry is not easily executable
  - “Please Mr Chemist, execute me this bunch of reactions that I just made up”
- Most molecular languages are not executable
  - They are descriptive (modeling) languages
- How can we execute molecular languages?
  - With real molecules?
  - That we can design ourselves?
  - And that we can buy on the web?

Microsoft Engineering Excellence

# Molecular Programming with DNA

Building the cores of programmable molecular controllers

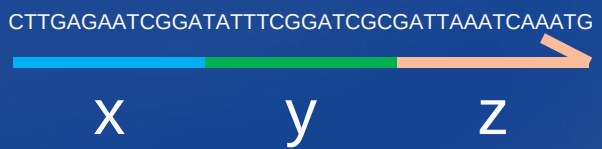
Microsoft

# DNA Computing

- Non-goals
  - Not to solve NP-complete problems with large vats of DNA
  - Not to replace silicon
- Bootstrapping a carbon-based technology
  - To precisely control the organization and dynamics of matter and information at the molecular level
  - DNA is our engineering material
    - Its biological origin is “accidental” (but convenient)
    - It is an information-bearing programmable material
    - It is possible that other such materials will be developed

# Domains

- Subsequences on a DNA strand are called domains
  - *provided* they are “independent” of each other




- That is, differently named domains must not hybridize
  - With each other, with each other’s complement, with subsequences of each other, with concatenations of other domains (or their complements), etc.

- Still somewhat of an open problem
  - A large literature
  - Can work in practice
  - Domain sequences often designed “by hand”

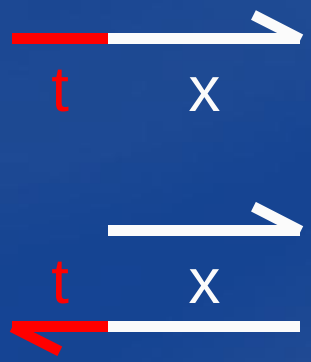
# Short Domains


Reversible Hybridization

# Long Domains

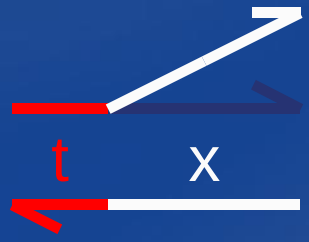

Irreversible Hybridization

# Strand Displacement



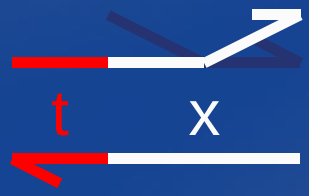

“Toehold Mediated”

# Strand Displacement



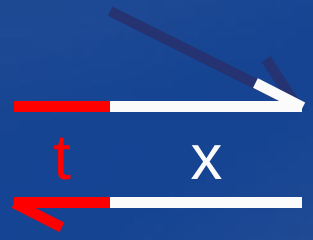

Toehold Binding

# Strand Displacement



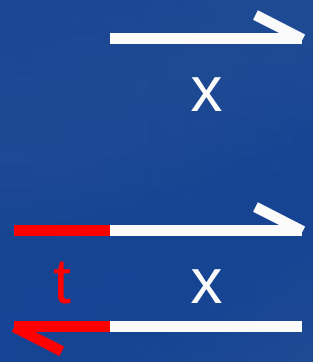

Branch Migration

# Strand Displacement



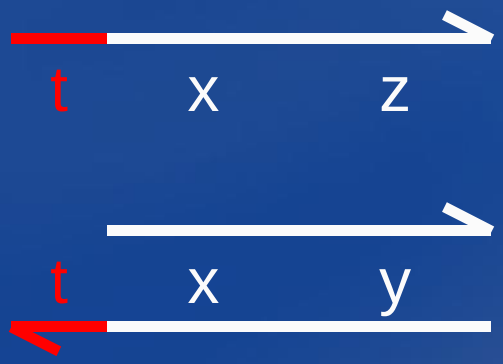

Displacement

# Strand Displacement

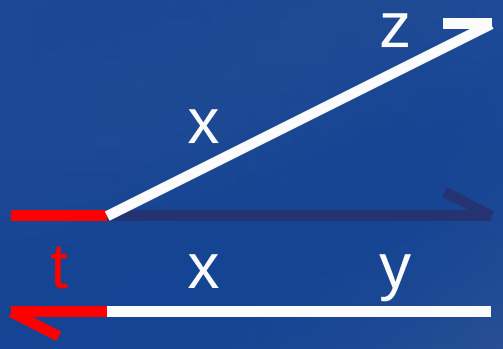



Irreversible release

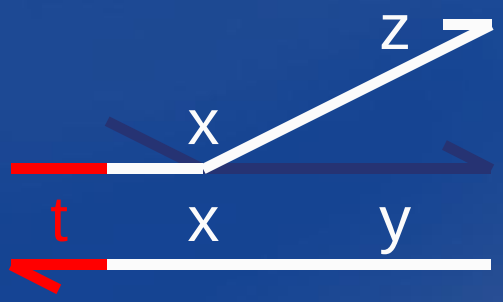
# Bad Match



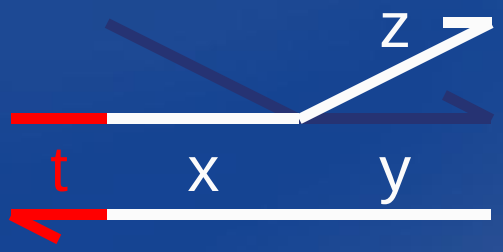
# Bad Match



# Bad Match



# Bad Match

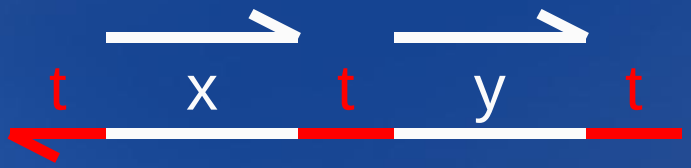



Cannot proceed
Hence will undo

# Two-Domain Architecture

- Signals: 1 toehold + 1 recognition region


- Gates: “top-nicked double strands” with open toeholds



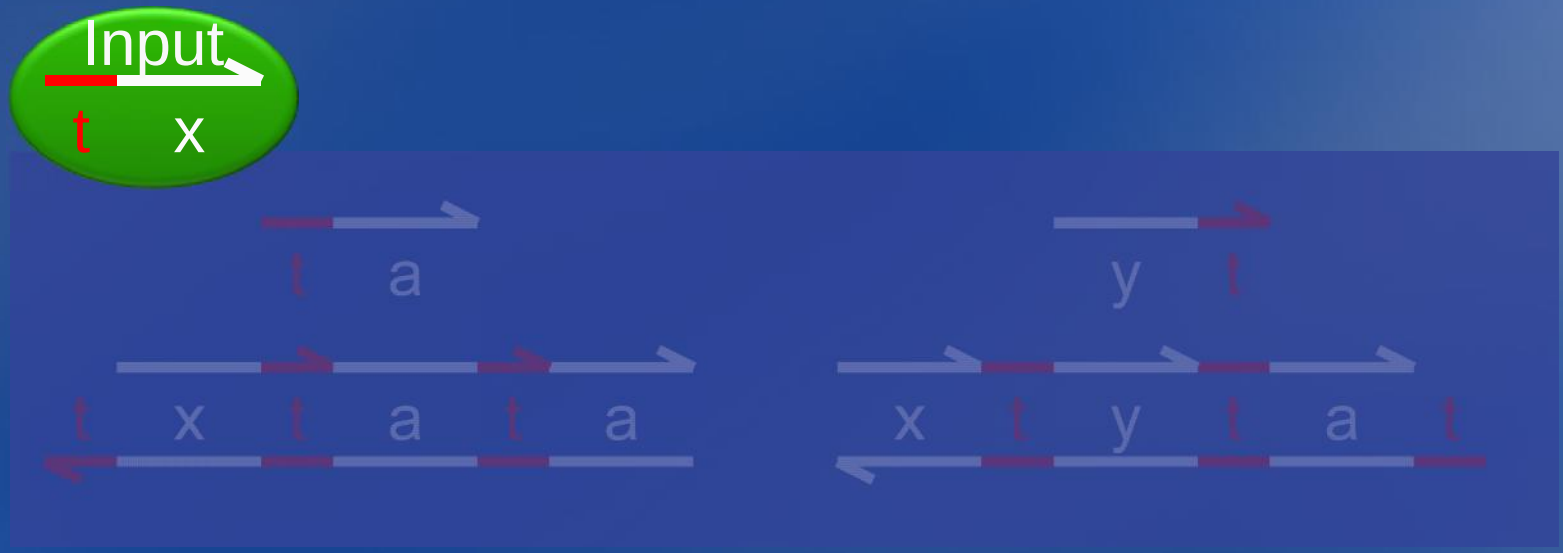

Garbage collection
“built into” the gates

Two-Domain DNA Strand Displacement

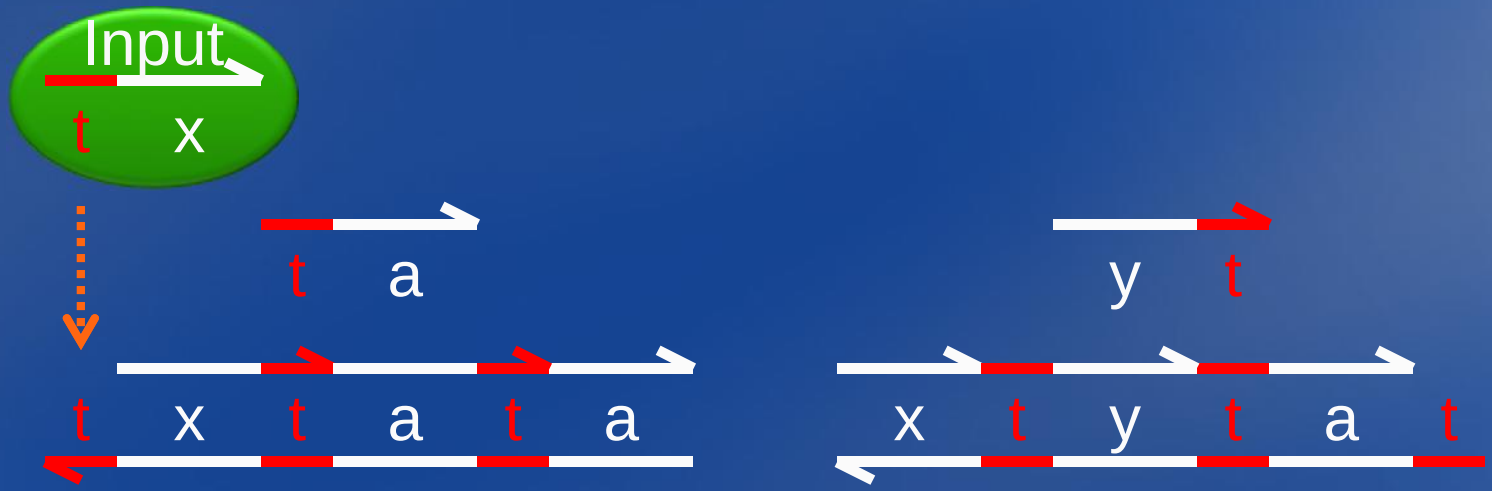
*Luca Cardelli*

In S. B. Cooper, E. Kashefi, P. Panangaden (Eds.):
Developments in Computational Models (DCM 2010).
EPTCS 25, 2010, pp. 33-47. May 2010.

# Transducer x→y



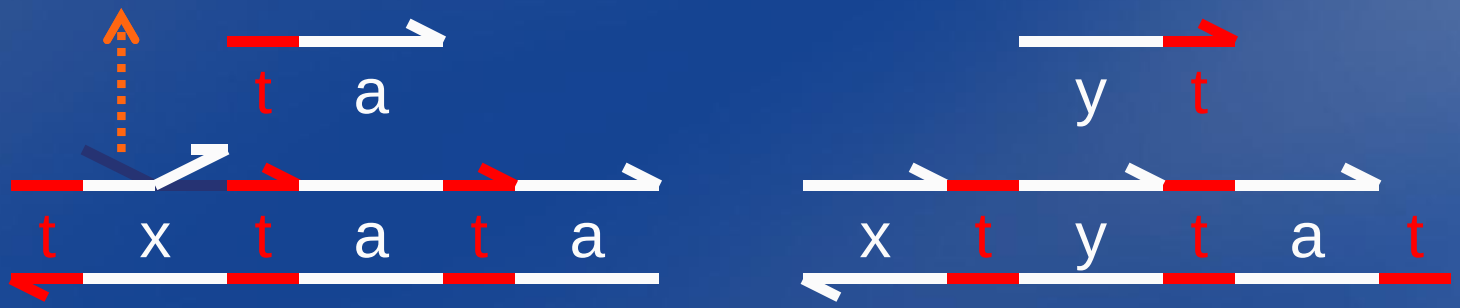
# Transducer x→y



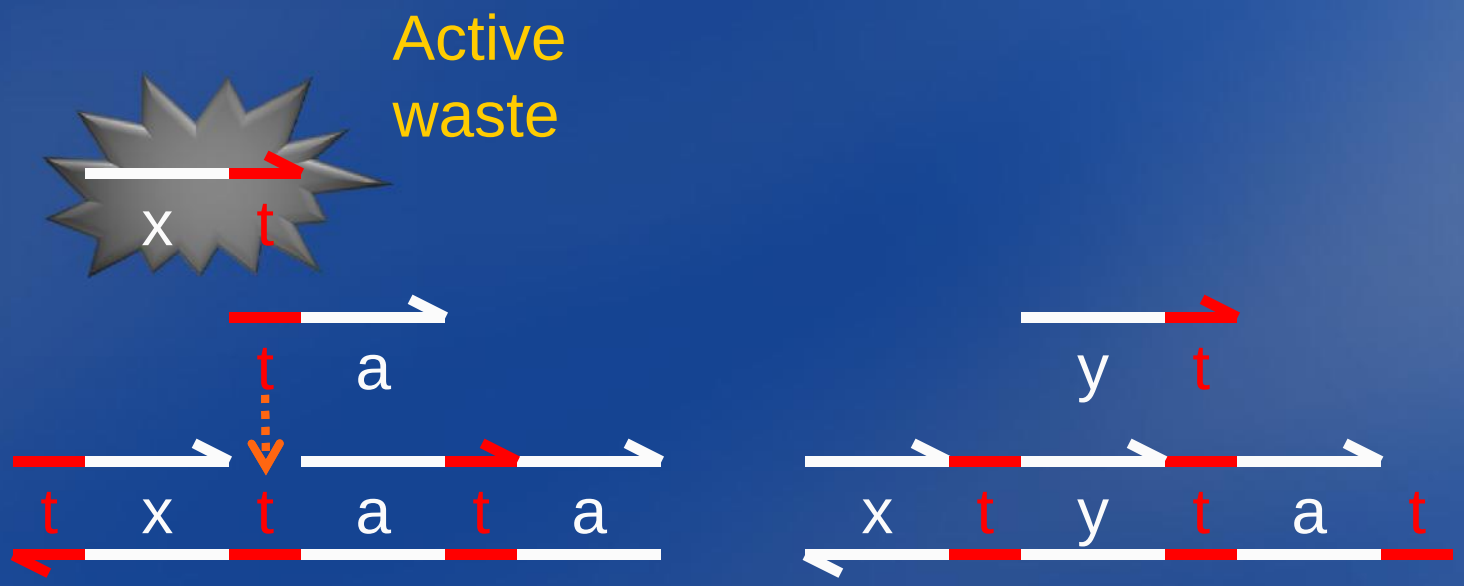

Built by self-assembly!

**ta** is a *private* signal (a different ‘a’ for each xy pair)

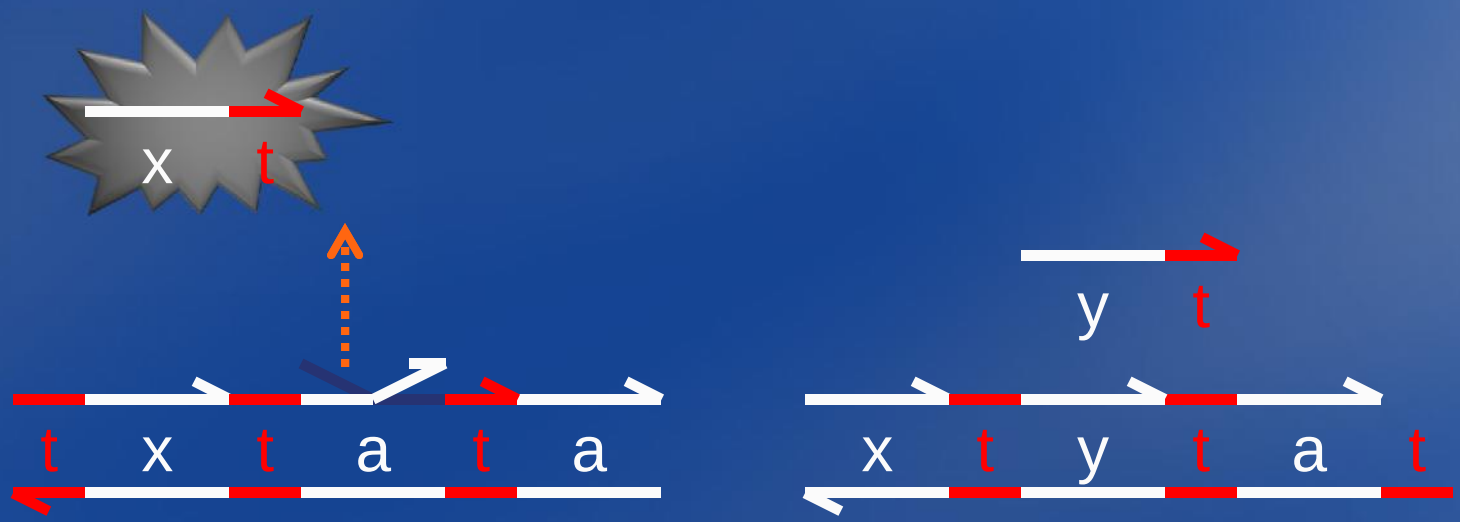
# Transducer x→y



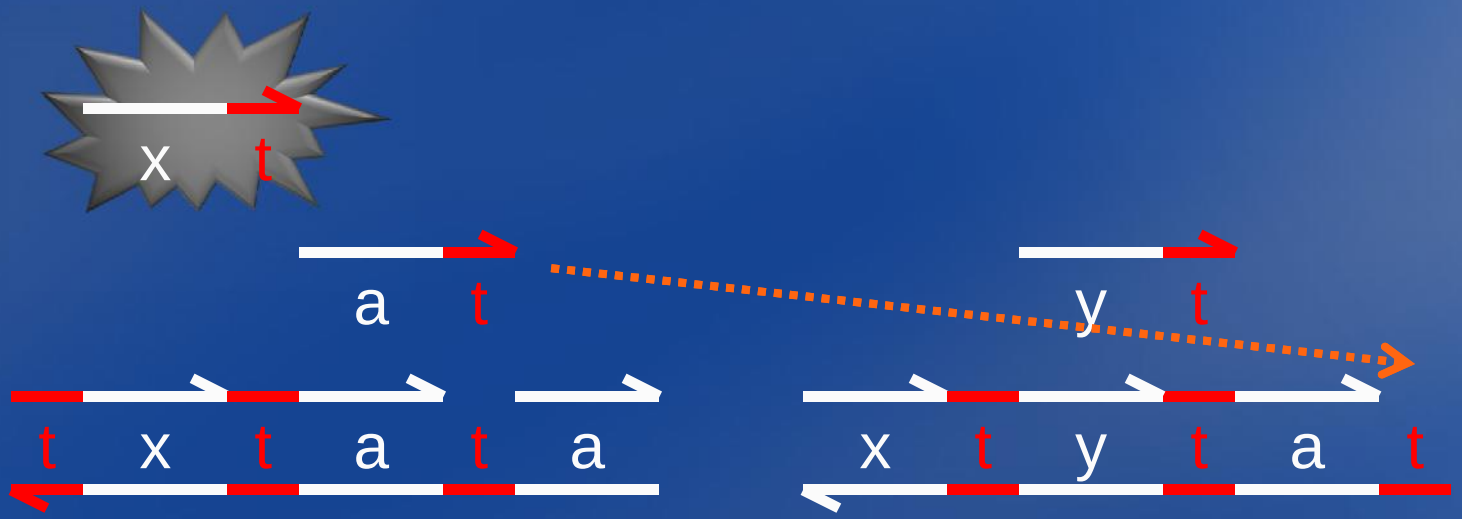
# Transducer x→y



# Transducer x→y



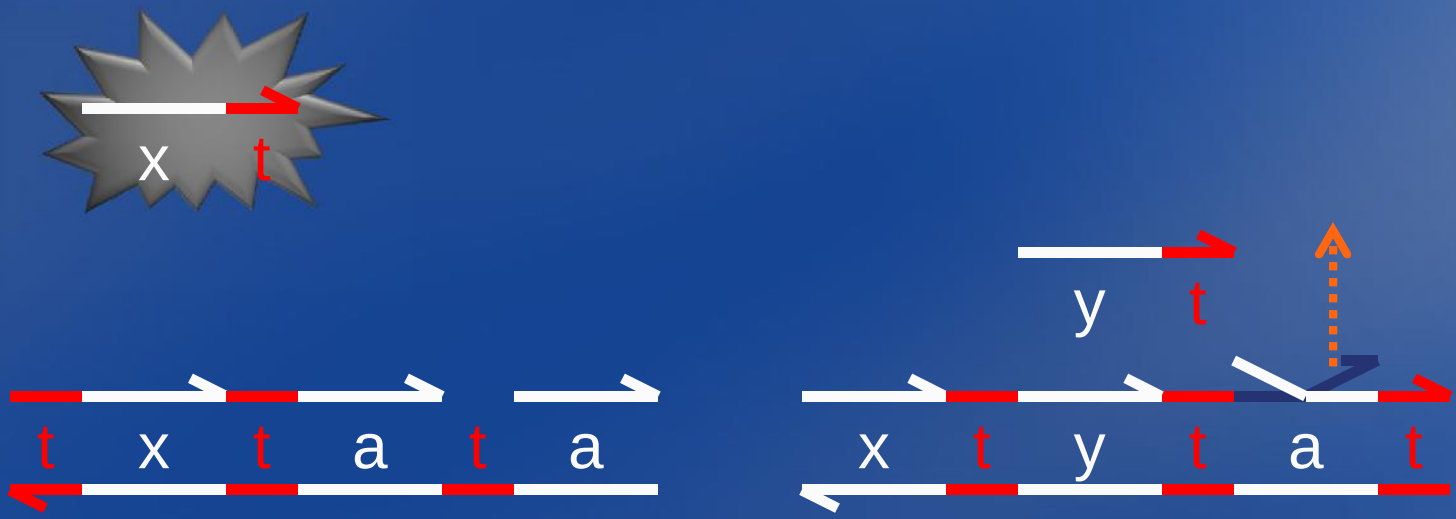
# Transducer x→y



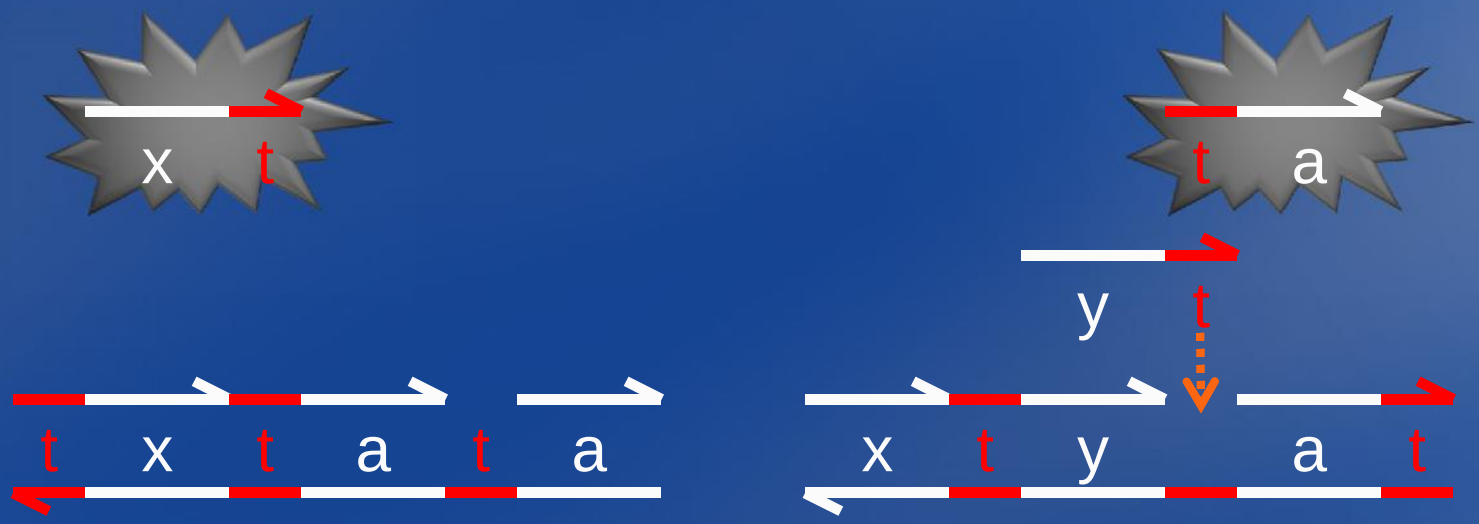

So far, a **tx** *signal* has produced an **at** *cosignal.*

But we want signals as output, not cosignals.

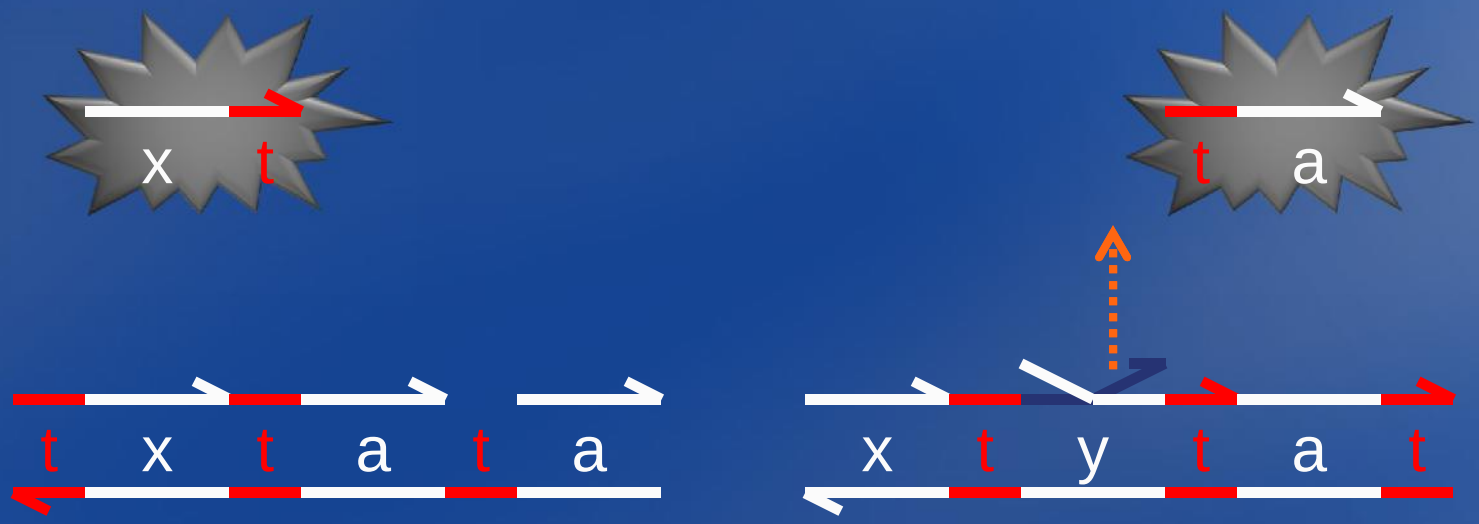
# Transducer x→y



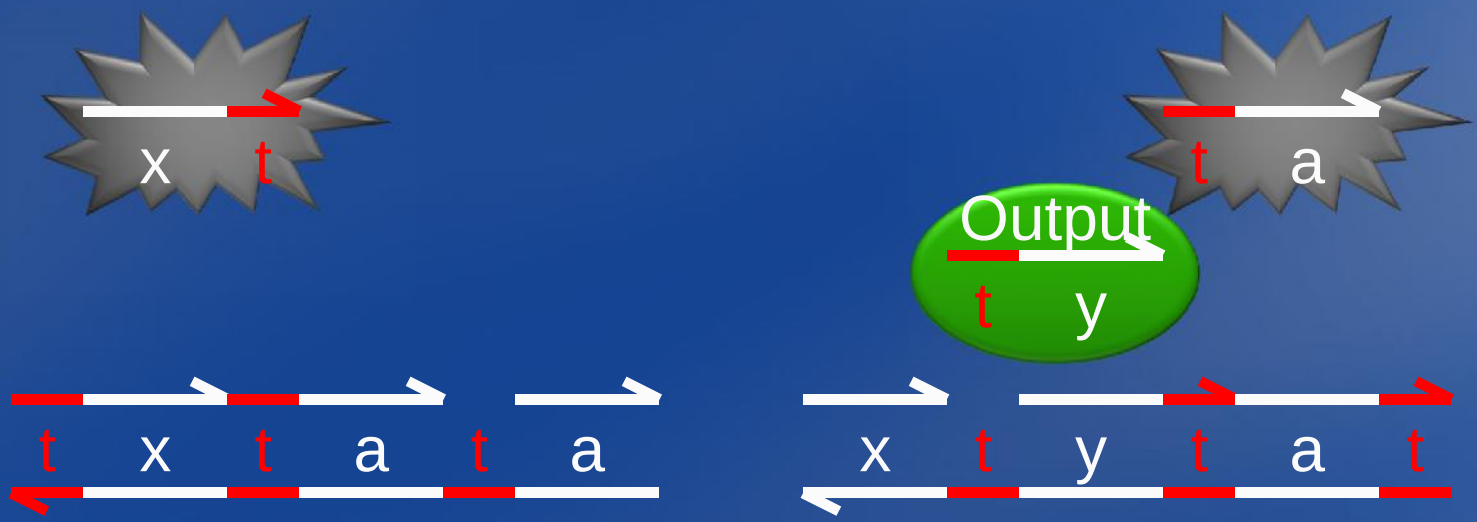
# Transducer x→y



# Transducer x→y



# Transducer x→y




Here is our output **ty** *signal*.

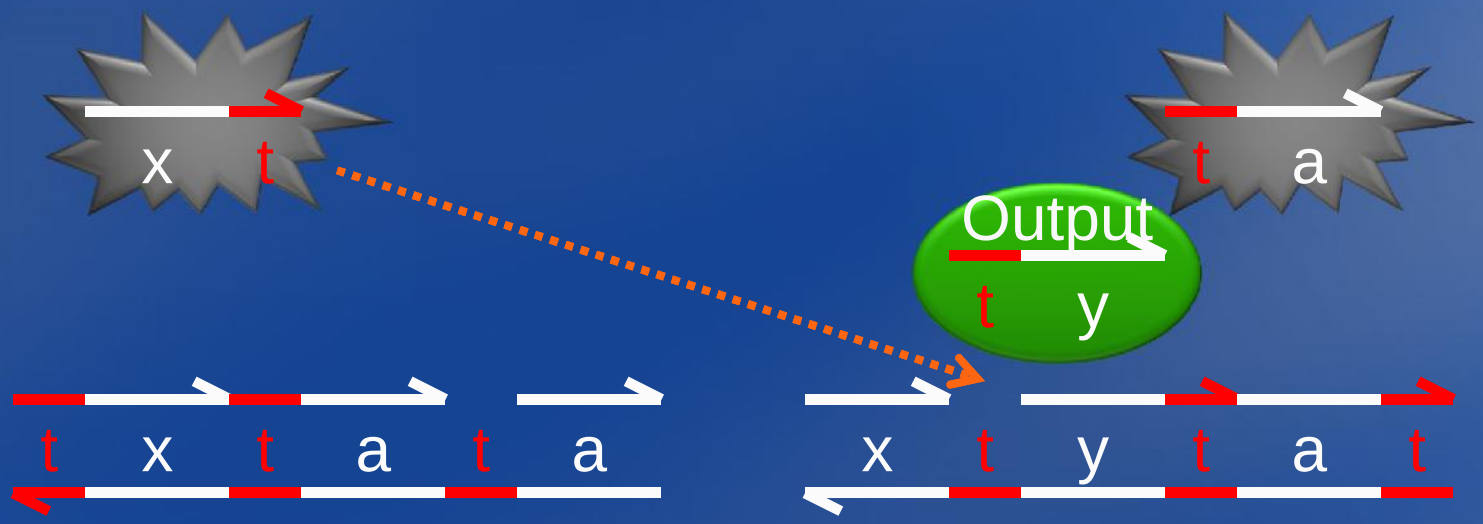
But we are not done yet:

1) We need to make the output irreversible.

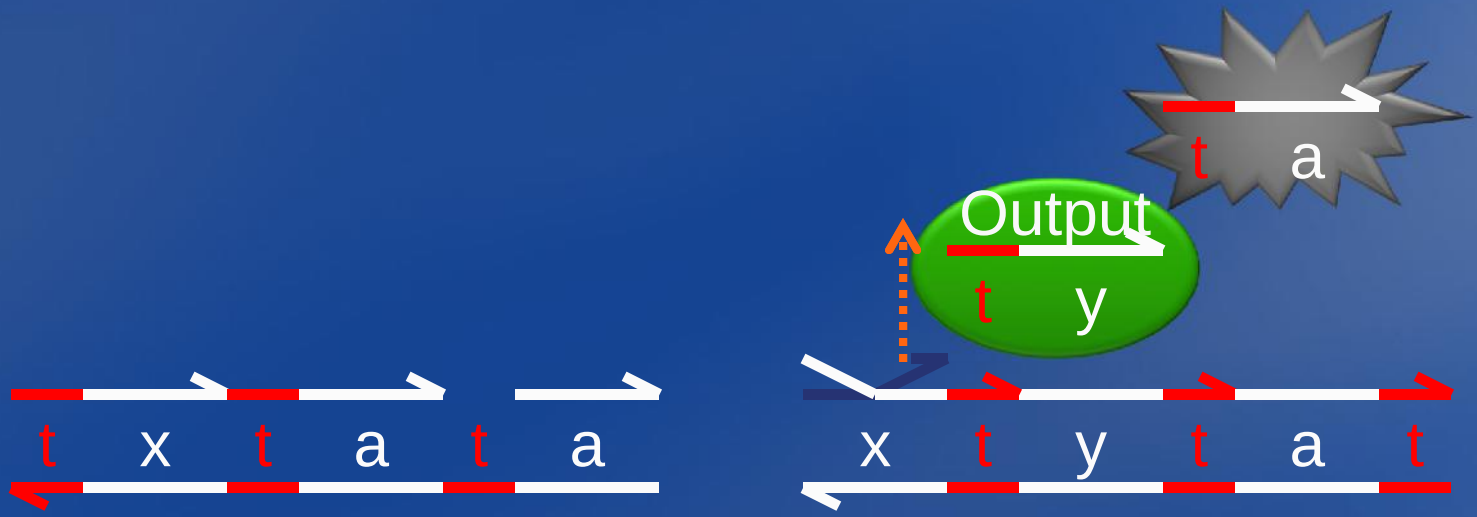
2) We need to remove the garbage.

We can use (2) to achieve (1).

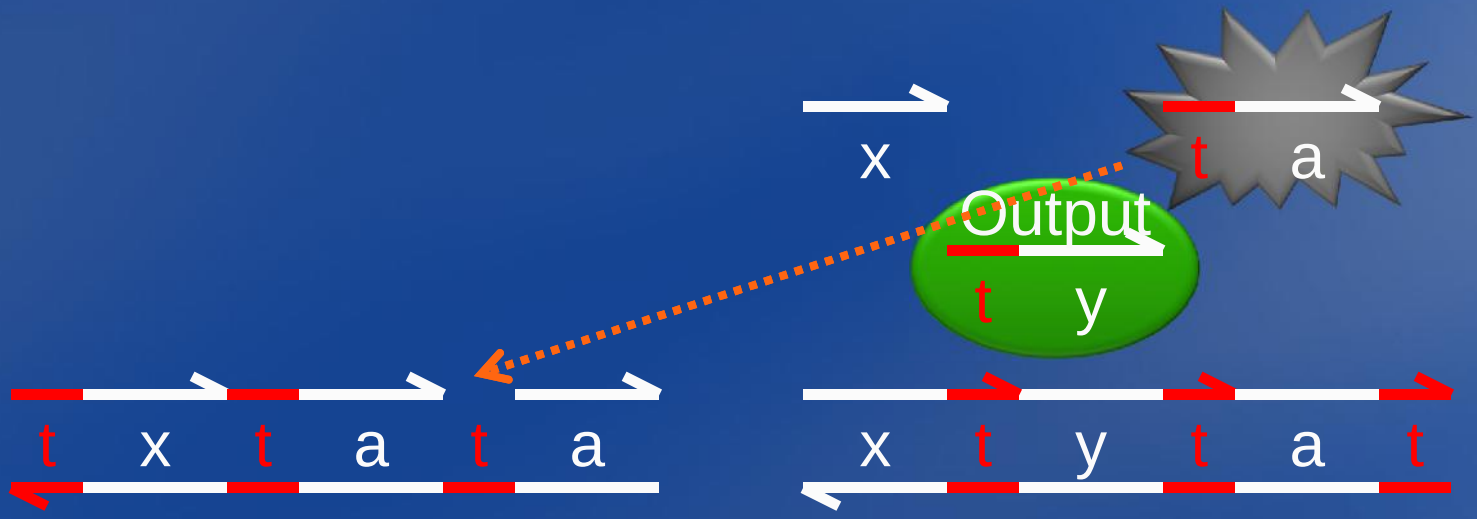
# Transducer x→y



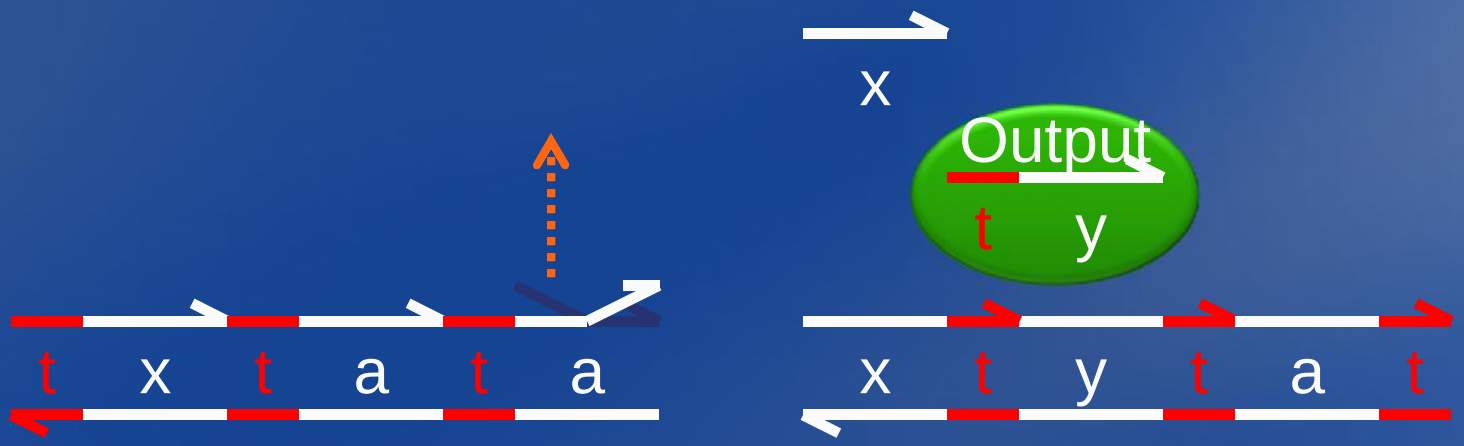
# Transducer x→y



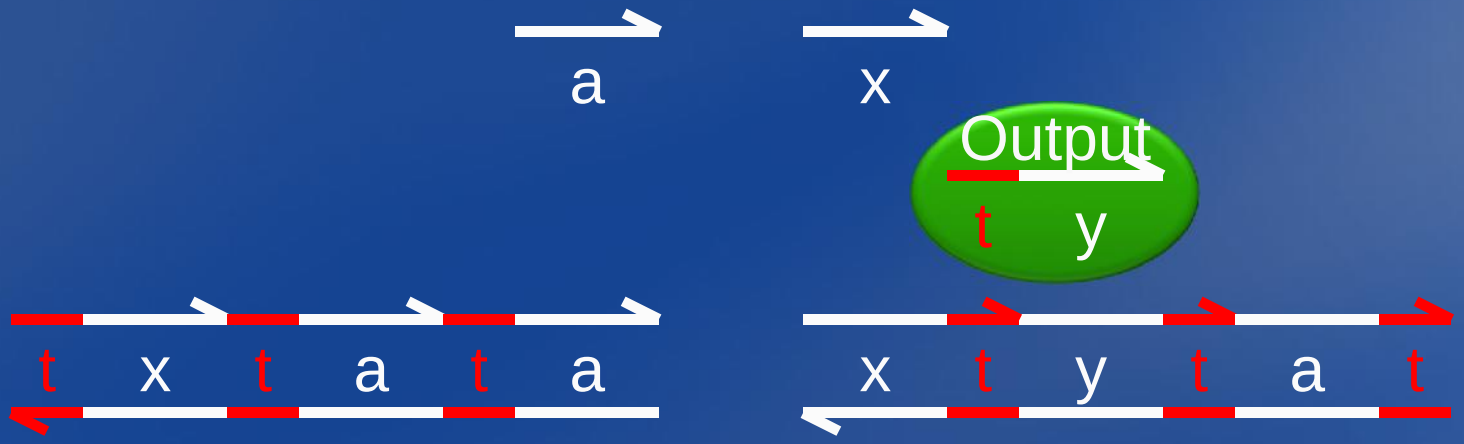
# Transducer x→y



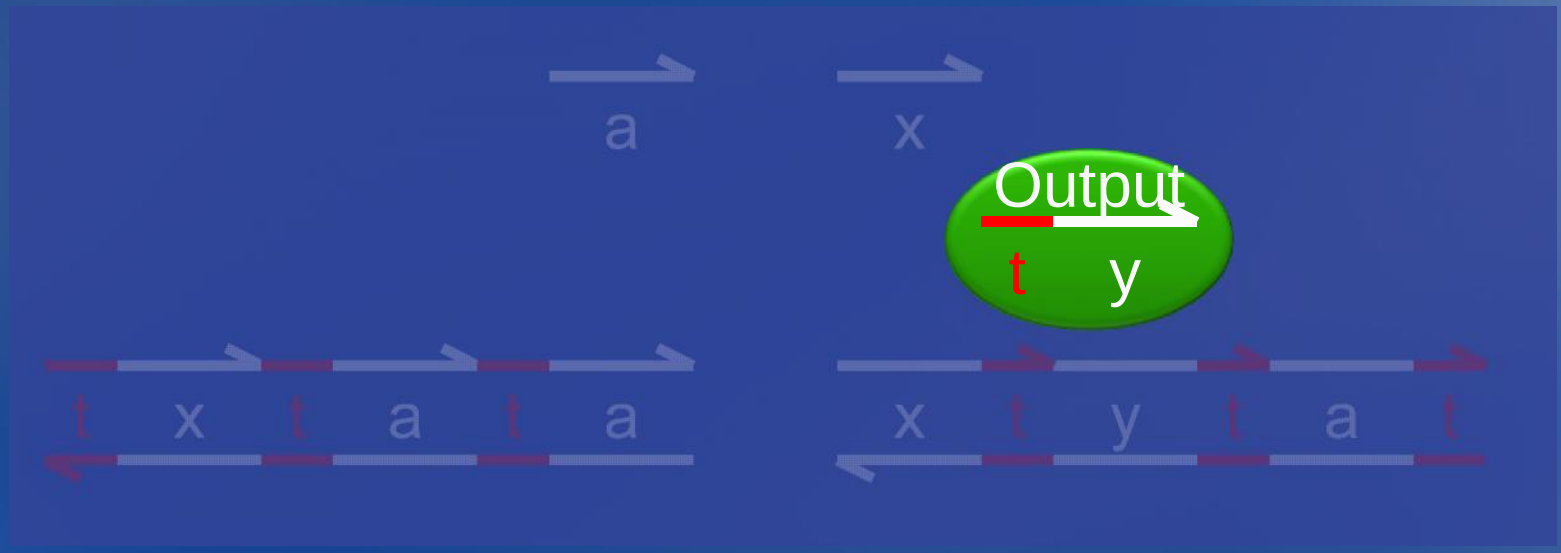
# Transducer x→y



# Transducer x→y

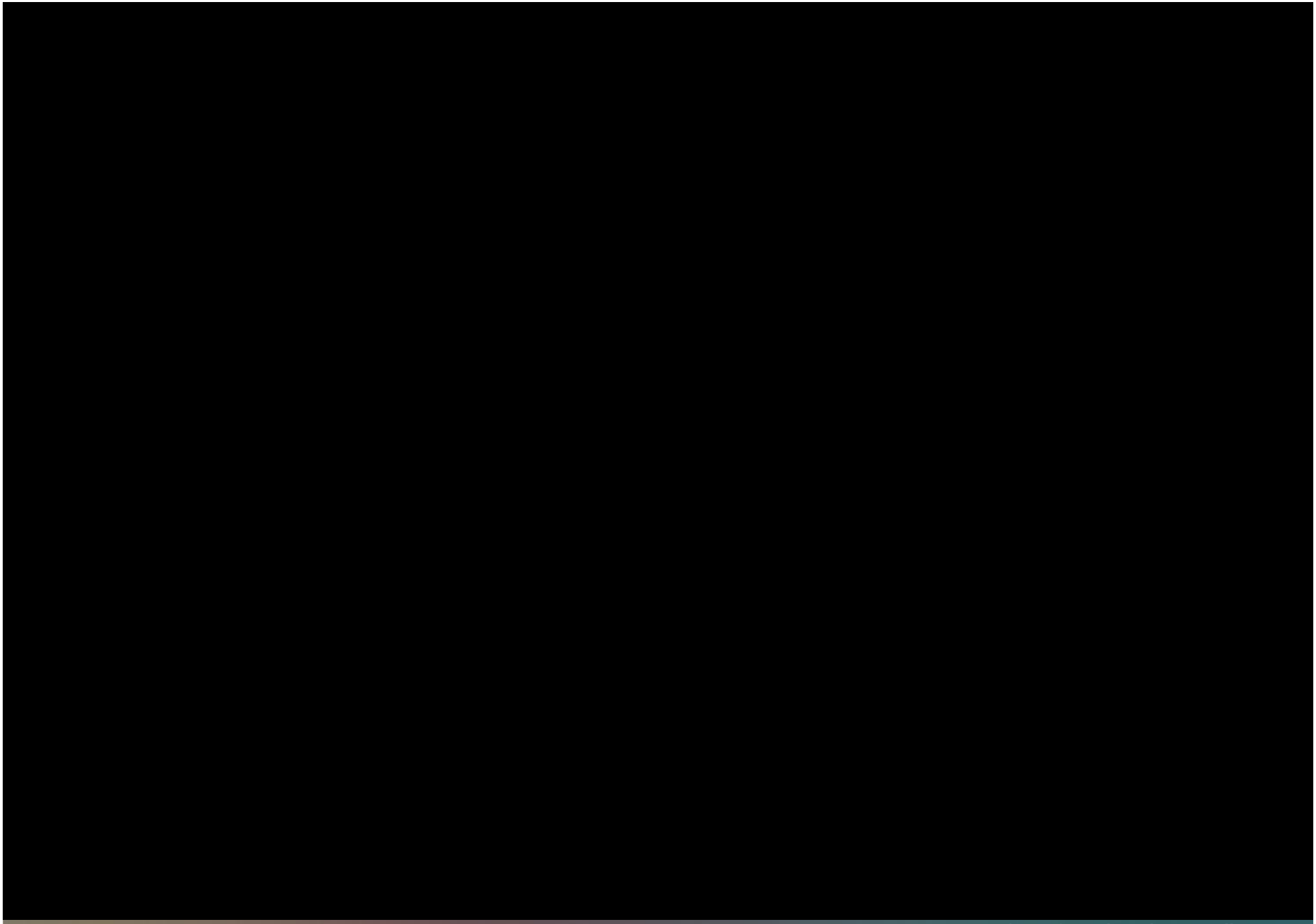


# Transducer x→y




Done.

N.B. the gate is consumed: it is the energy source



Microsoft Engineering Excellence

# Tools and Techniques

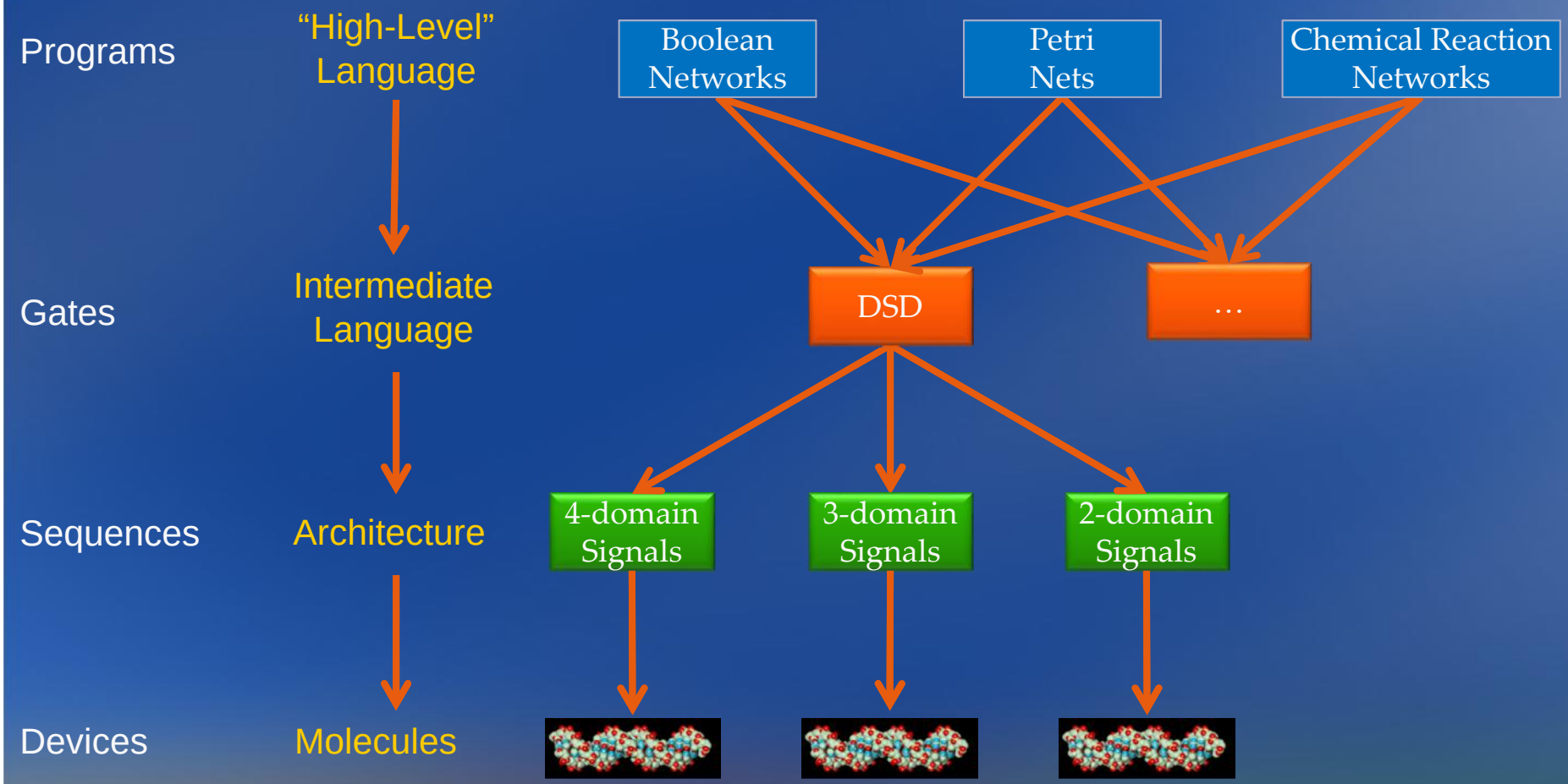
A software pipeline for Molecular Programming

Microsoft

# High(er)-Level Languages

- Gene Networks
  - Synchronous Boolean networks
    - Stewart Kauffman, etc.
  - Asynchronous Boolean networks
    - René Thomas, etc.
- Protein Networks
  - Process Algebra (stochastic π-calculus etc.)
    - Priami, Regev-Shapiro, etc.
  - Graph Rewriting (kappa, BioNetGen etc.)
    - Danos-Laneve, Fontana & al., etc.
- Membrane Networks
  - Membrane Computing
    - Gheorghe Păun, etc.
  - Brane Calculi
    - Luca Cardelli, etc.
- Waiting for an architecture to run on...

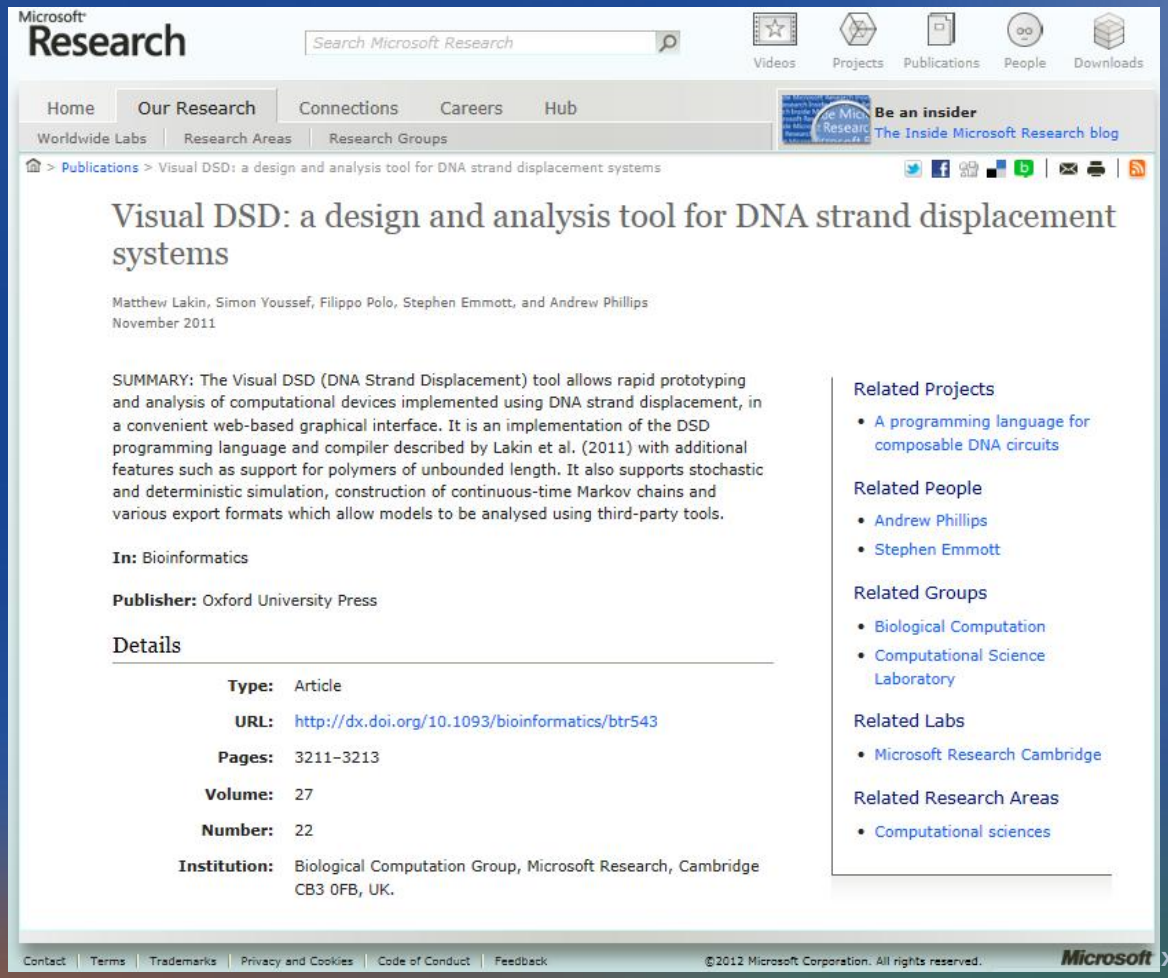
# Molecular Compilation



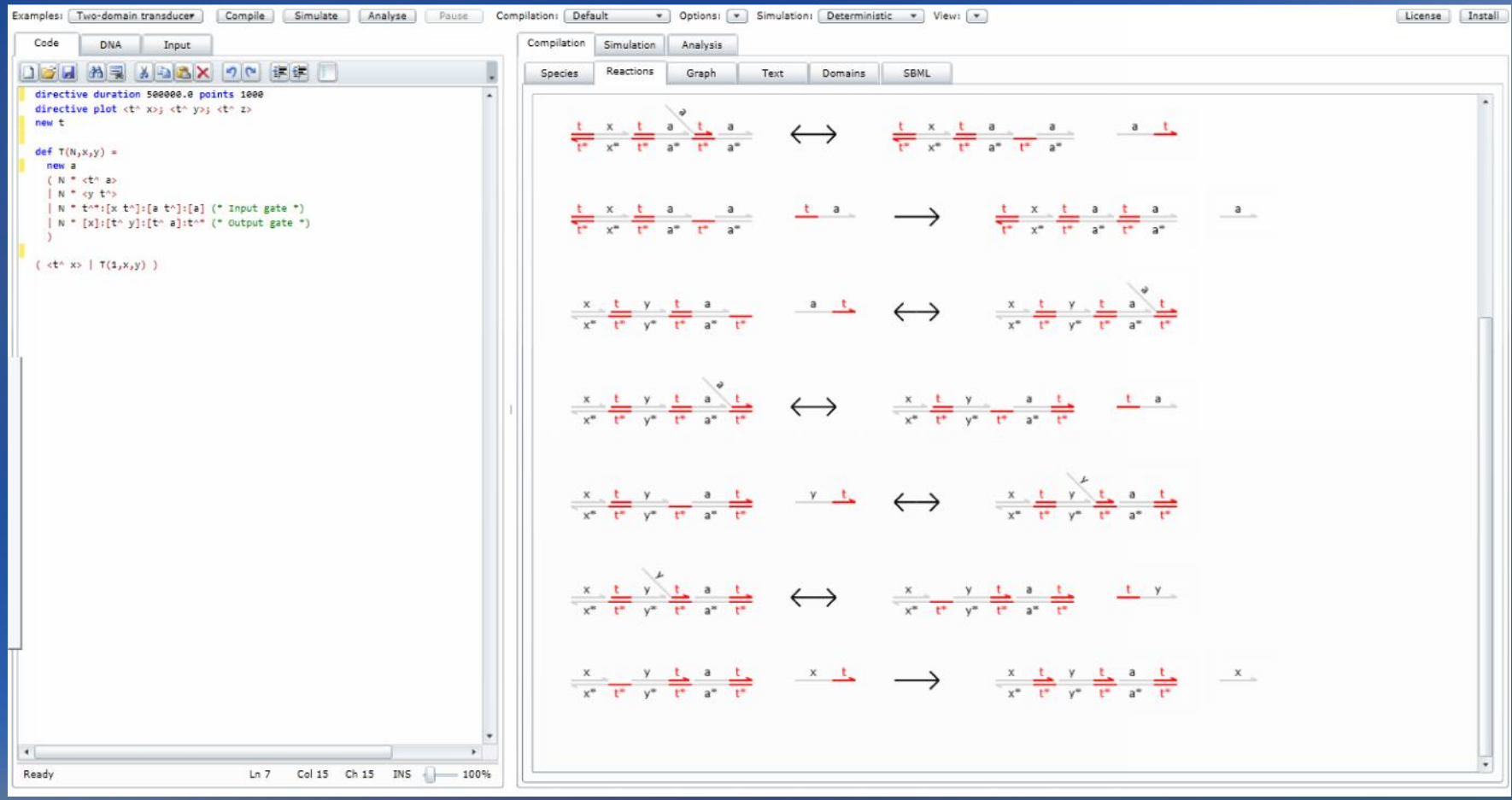
# Visual DSD (DNA Strand Displacement)

- http://research.microsoft.com/apps/pubs/default.aspx?id=157262

  Andrew Phillips
  MSR Cambridge



# A Development Environment for DNA Gates

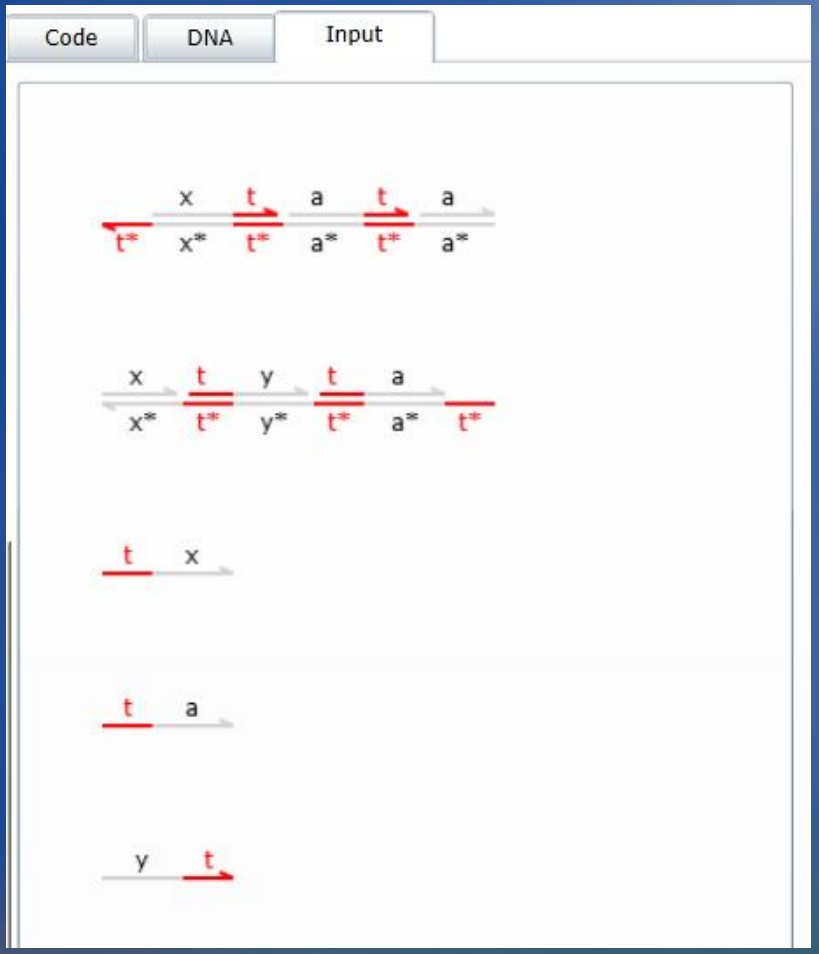


# A Language for DNA Structures

- Describe the initial structures

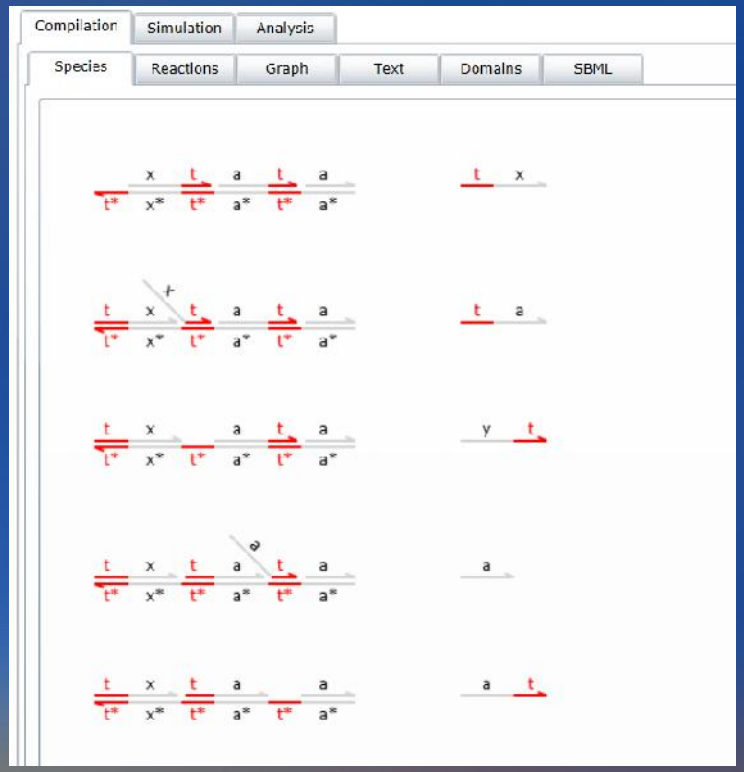
Code | DNA | Input

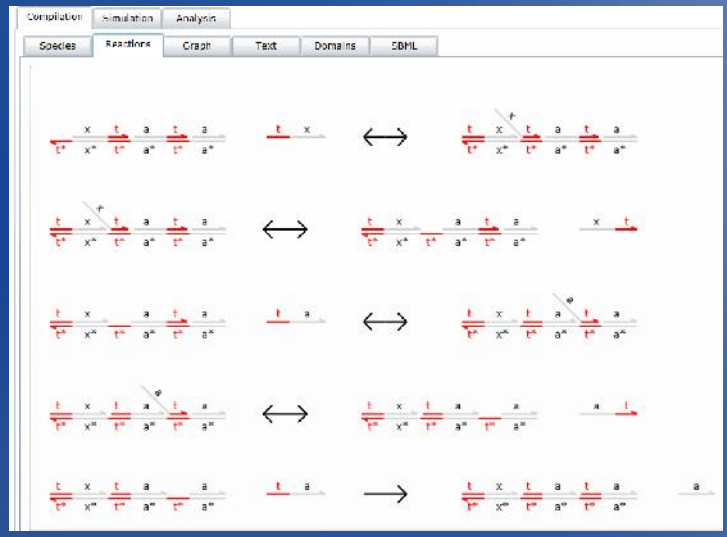
```
directive duration 10000.0 points 1000
directive plot <t^ x>; <t^ y>; <t^ z>
new t

def T(N,x,y) =
  new a
  ( N * <t^ a>
  | N * <y t^>
  | N * t^*:[x t^]:[a t^]:[a] (* Input gate *)
  | N * [x]:[t^ y]:[t^ a]:t^* (* Output gate *)
  )

( <t^ x> | T(1,x,y) )
```

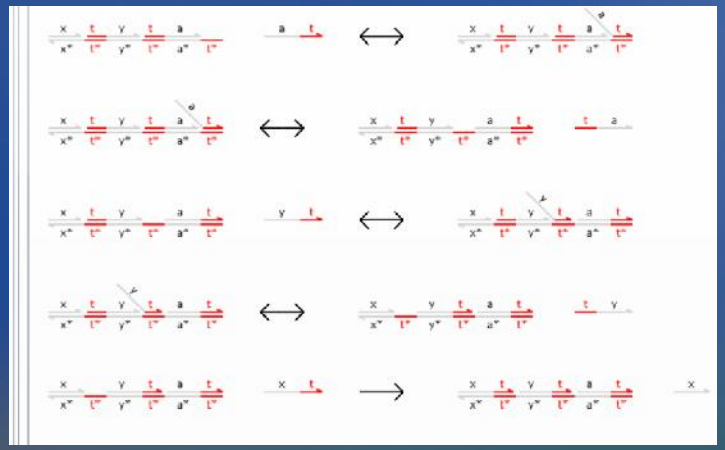


# Compute Species and Reactions

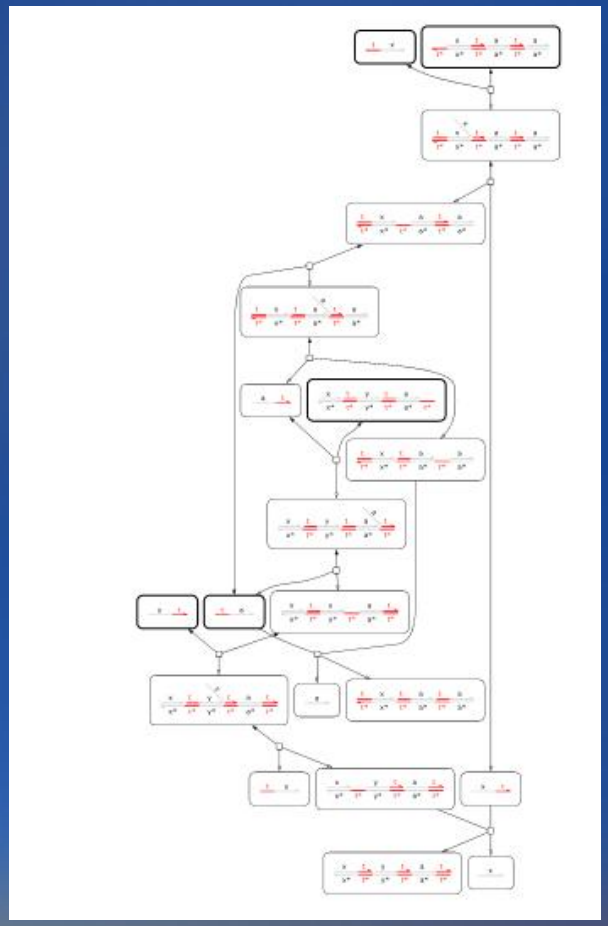
- Recursively computed from the initial structures



# Reaction Graph and Export



Compilation | Simulation | Analysis

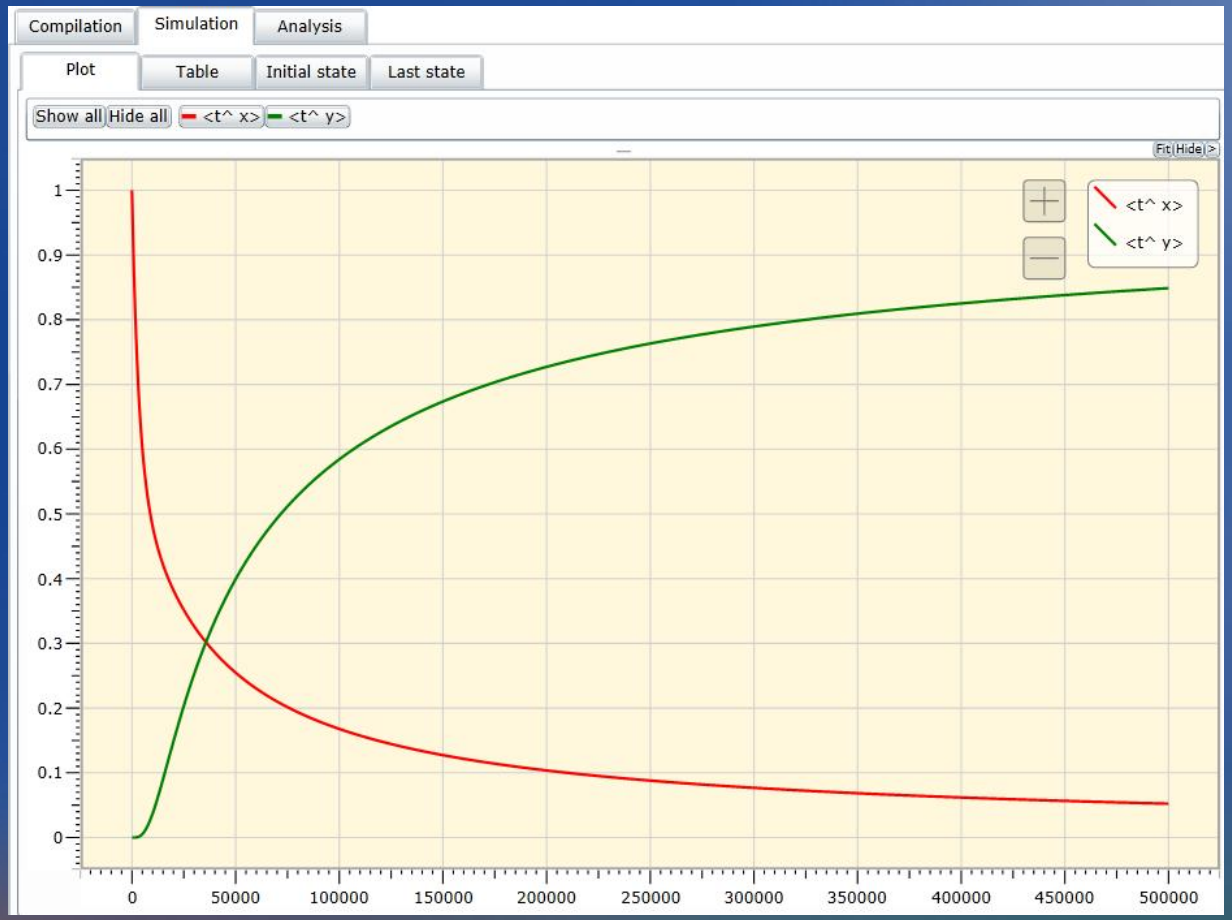
Species | Reactions | Graph | Text | Domains | SBML

Save as XML

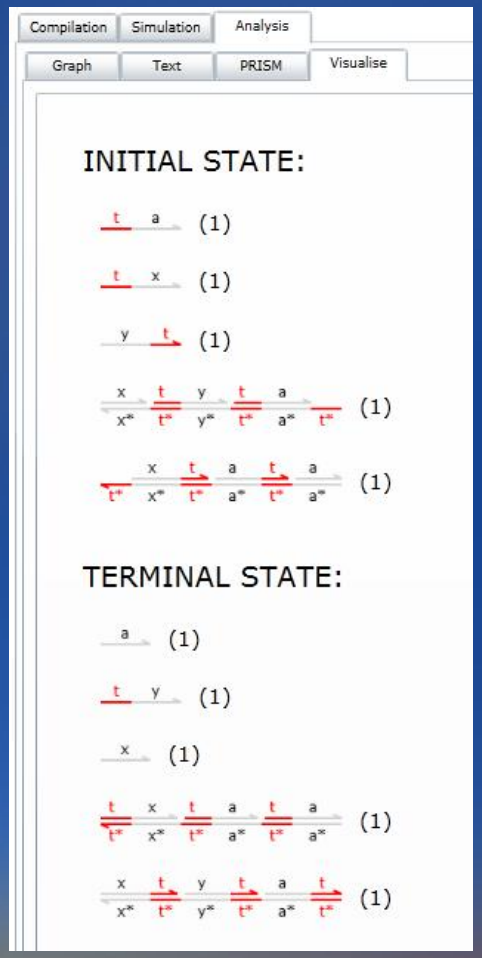
```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2/version1" level="2" version="1">
 <model>
  <listOfCompartments>
   <compartment id="c" size="1"/>
  </listOfCompartments>
  <listOfSpecies>
   <species id="s_id0" name="&lt;t^ x>" compartment="c" initialAmount="1" constant="false"/>
   <species id="s_id1" name="&lt;t^ a>" compartment="c" initialAmount="1" constant="false"/>
   <species id="s_id2" name="&lt;y t^>" compartment="c" initialAmount="1" constant="false"/>
   <species id="s_id3" name="{t^*}[x t^]:[a t^]:[a]" compartment="c" initialAmount="1" constant="false"/>
   <species id="s_id4" name="[t^ x]:&lt;x>[t^]:[a t^]:[a]" compartment="c" initialAmount="0" constant="false"/>
   <species id="s_id5" name="[t^ x]{t^*}:[a t^]:[a]" compartment="c" initialAmount="0" constant="false"/>
   <species id="s_id6" name="[t^ x]:[t^ a]:&lt;a>[t^]:[a]" compartment="c" initialAmount="0" constant="false"/>
   <species id="s_id7" name="[t^ x]:[t^ a]{t^*}:[a]" compartment="c" initialAmount="0" constant="false"/>
   <species id="s_id8" name="[t^ x]:[t^ a]:[t^ a]" compartment="c" initialAmount="0" constant="false"/>
   <species id="s_id9" name="&lt;a>" compartment="c" initialAmount="0" constant="false"/>
   <species id="s_id10" name="&lt;a t^>" compartment="c" initialAmount="0" constant="false"/>
   <species id="s_id11" name="&lt;x t^>" compartment="c" initialAmount="0" constant="false"/>
   <species id="s_id12" name="[x]:[t^ y]:[t^ a]{t^*}" compartment="c" initialAmount="1" constant="false"/>
   <species id="s_id13" name="[x]:[t^ y]:[t^ a]:&lt;a>[t^]" compartment="c" initialAmount="0" constant="false"/>
   <species id="s_id14" name="[x]:[t^ y]{t^*}:[a t^]" compartment="c" initialAmount="0" constant="false"/>
   <species id="s_id15" name="[x]:[t^ y]:&lt;y>[t^]:[a t^]" compartment="c" initialAmount="0" constant="false"/>
   <species id="s_id16" name="[x]{t^*}:[y t^]:[a t^]" compartment="c" initialAmount="0" constant="false"/>
   <species id="s_id17" name="[x t^]:[y t^]:[a t^]" compartment="c" initialAmount="0" constant="false"/>
   <species id="s_id18" name="&lt;x>" compartment="c" initialAmount="0" constant="false"/>
   <species id="s_id19" name="&lt;t^ y>" compartment="c" initialAmount="0" constant="false"/>
  </listOfSpecies>
  <listOfReactions>
   <reaction id="r_id20" reversible="false">
    <listOfReactants>
     <speciesReference species="s_id3"/>
     <speciesReference species="s_id0"/>
```

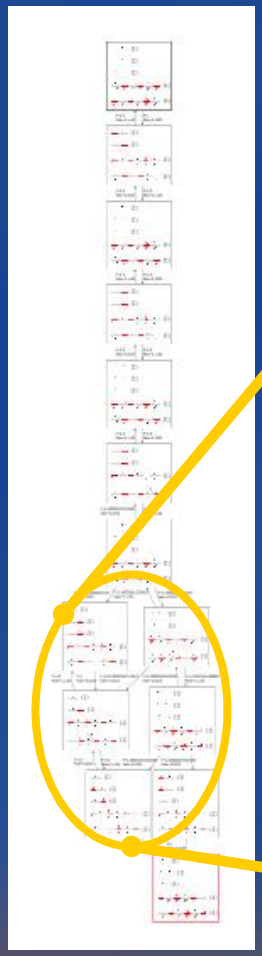
# Simulation

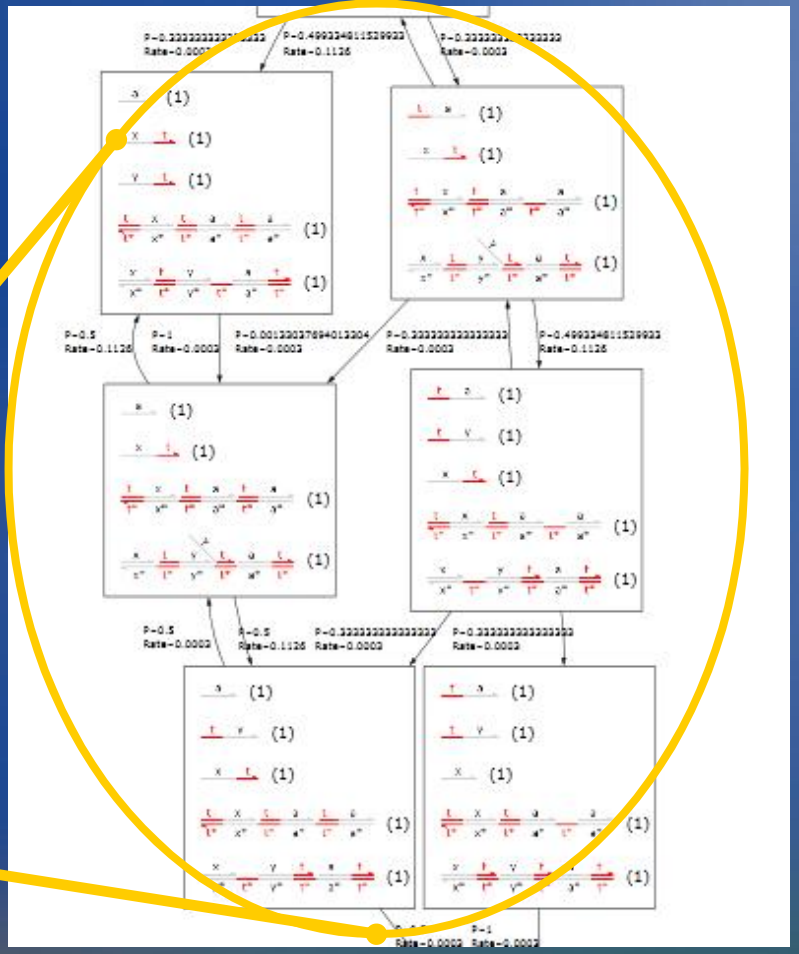
- Stochastic
- Deterministic
- “JIT”



# Analysis

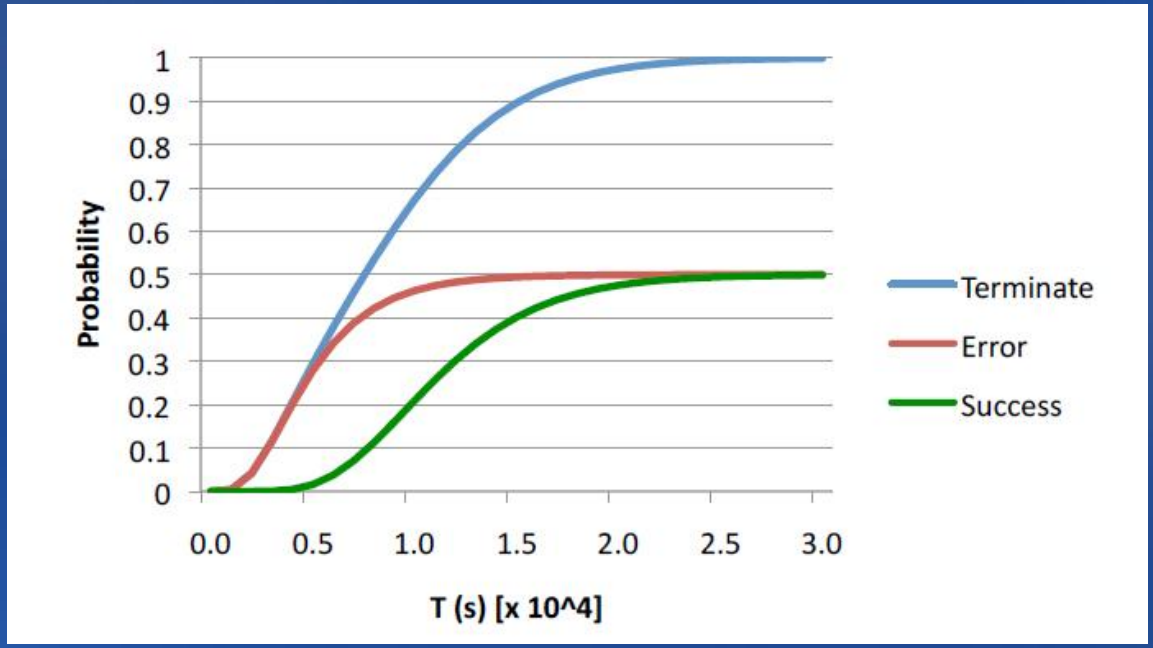

Compilation
Simulation
Analysis
Graph
Text
PRISM
Visualise
INITIAL STATE:
(1)
(1)
(1)
(1)
(1)
TERMINAL STATE:
(1)
(1)
(1)
(1)
(1)





# Modelchecking

- Export to PRISM probabilistic modelchecker




Design and Analysis of DNA Strand Displacement Devices using Probabilistic Model Checking

Matthew R. Lakin *† David Parker ‡†

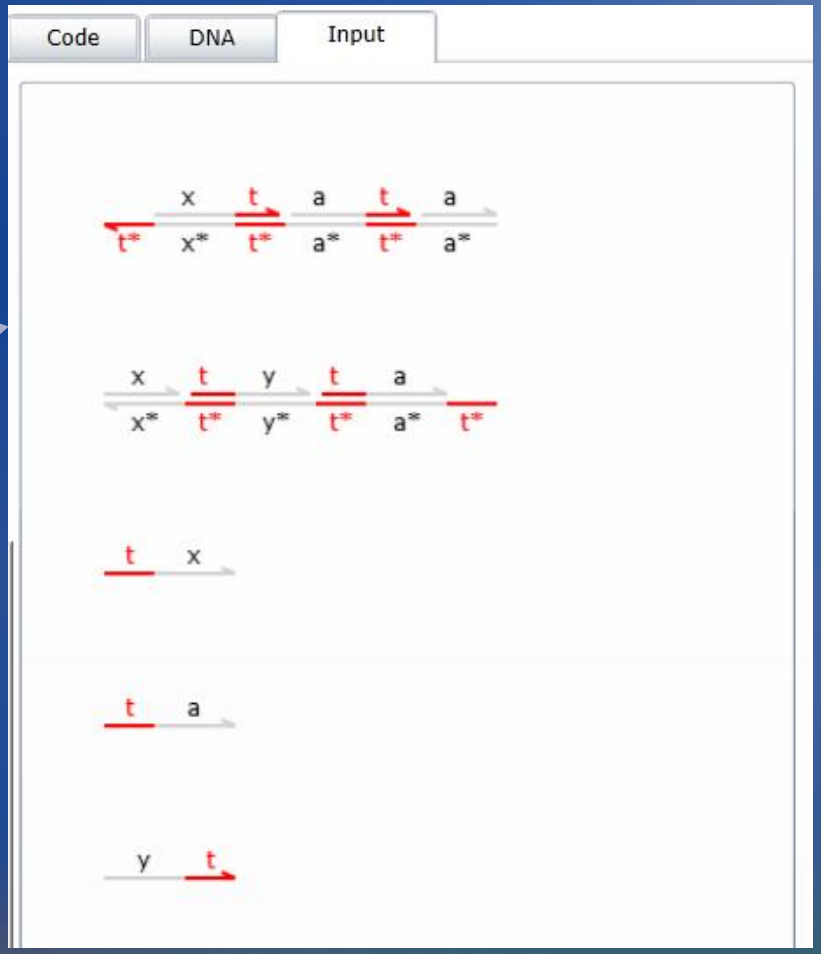
Luca Cardelli* Marta Kwiatkowska ‡

Andrew Phillips*§

# Tool Output: Domain Structures

- Abstract structures
  - (no DNA sequences)

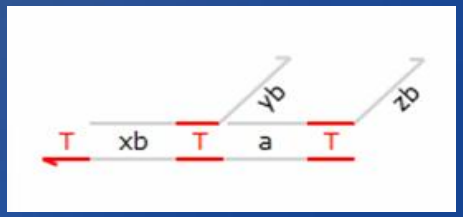
"Ok, I want to run this for real"

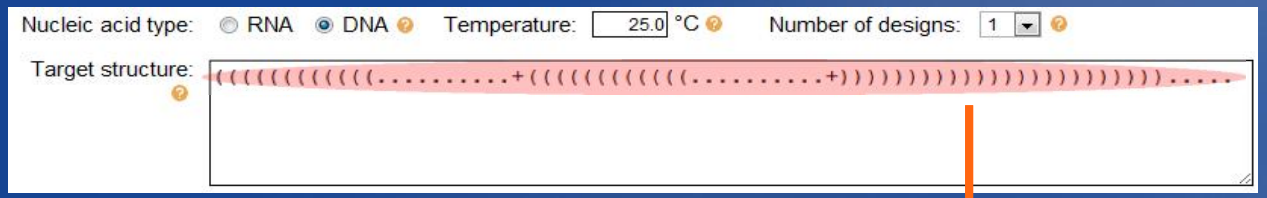


# From Structures to Sequences

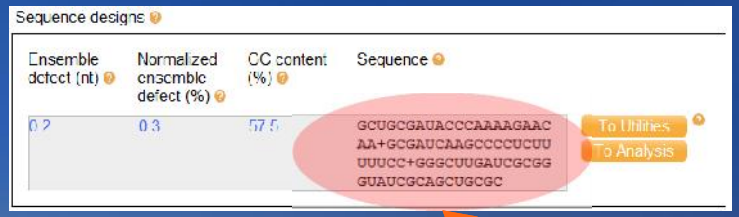

www.nupack.org

DSD Structure → "Dot-Paren" representation

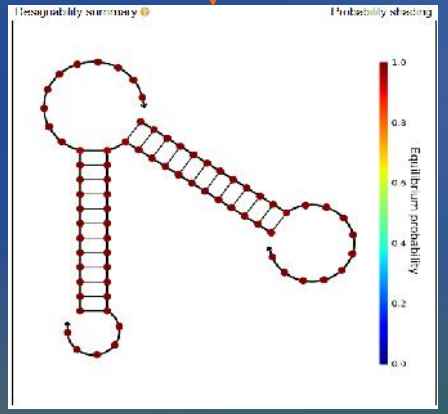



Output Sequences

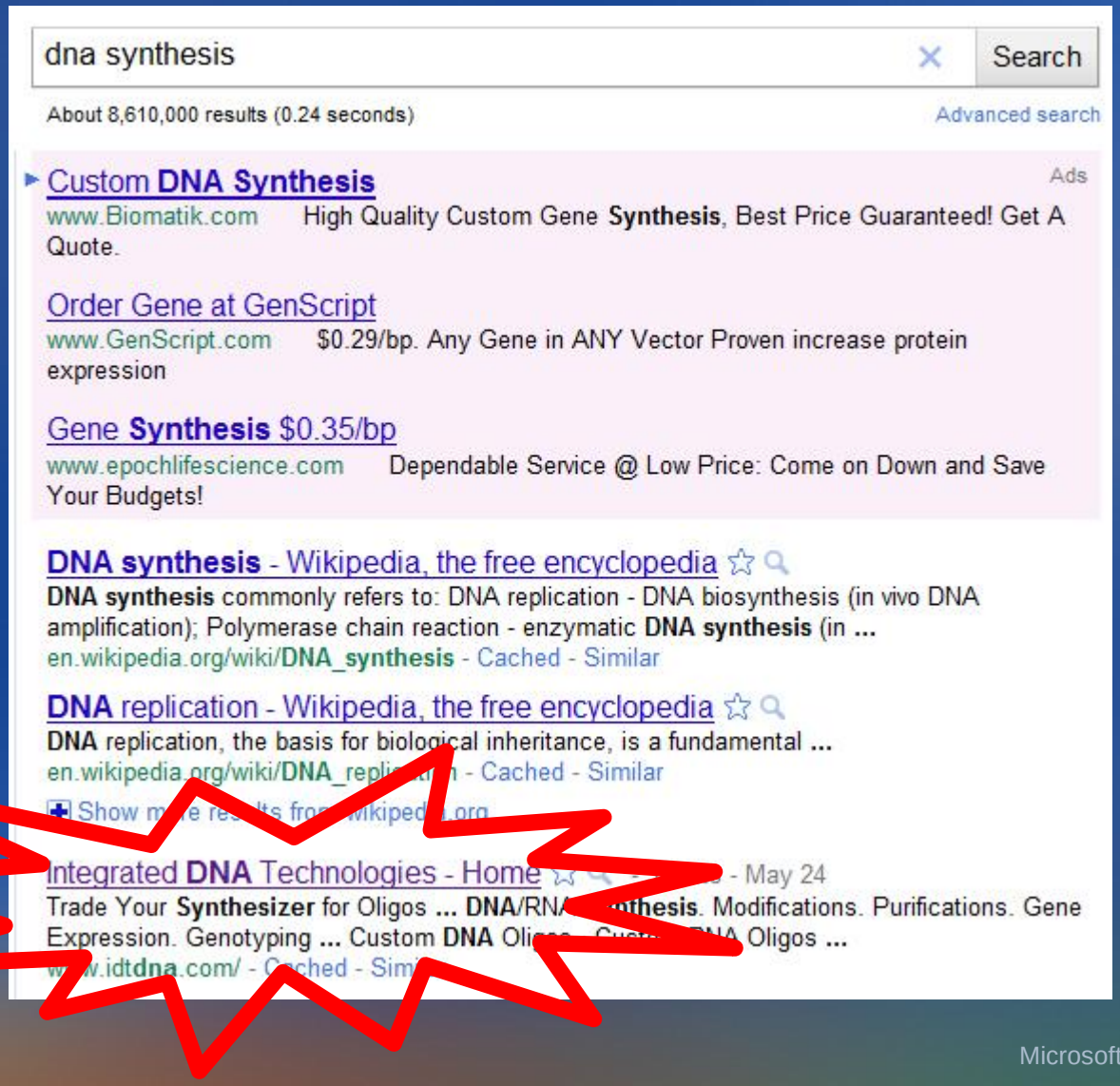



Thermodynamic Synthesis



"Ok, where do I buy these"

# DNA Synthesis



# From Sequences to Molecules

- Copy&Paste from nupack

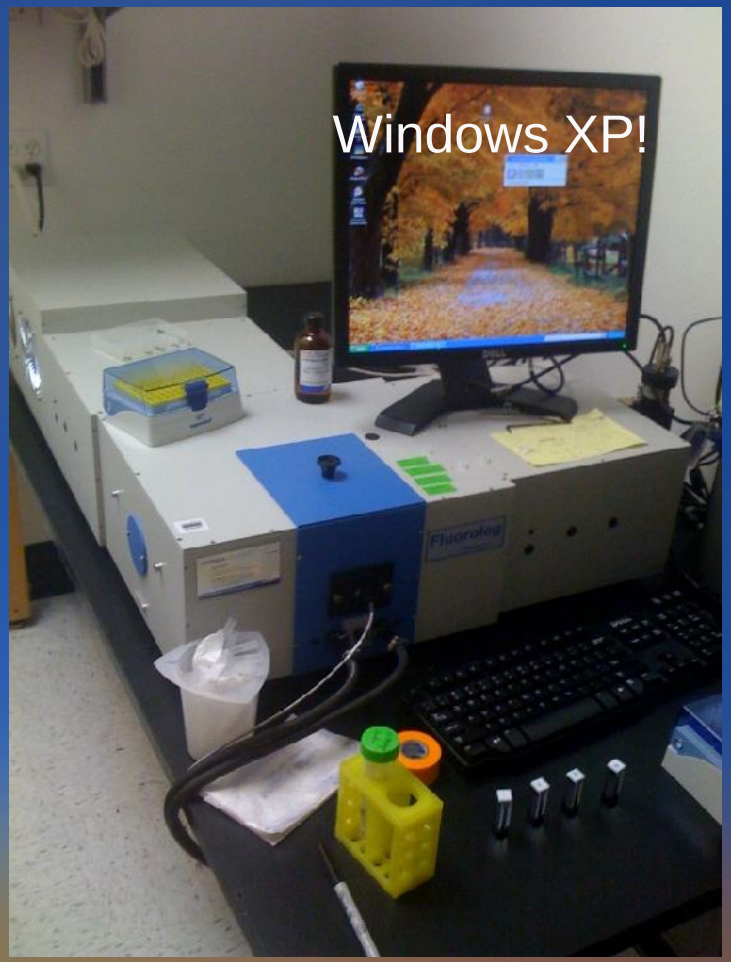
# Molecules by FedEx

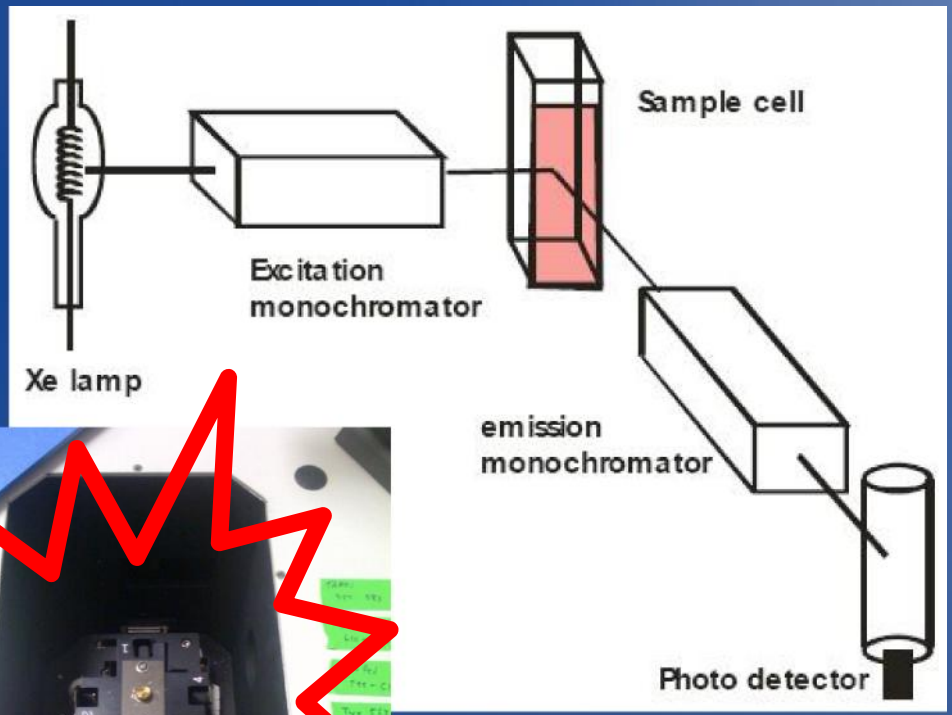

"Ok, how do I run these?"

# Add Water

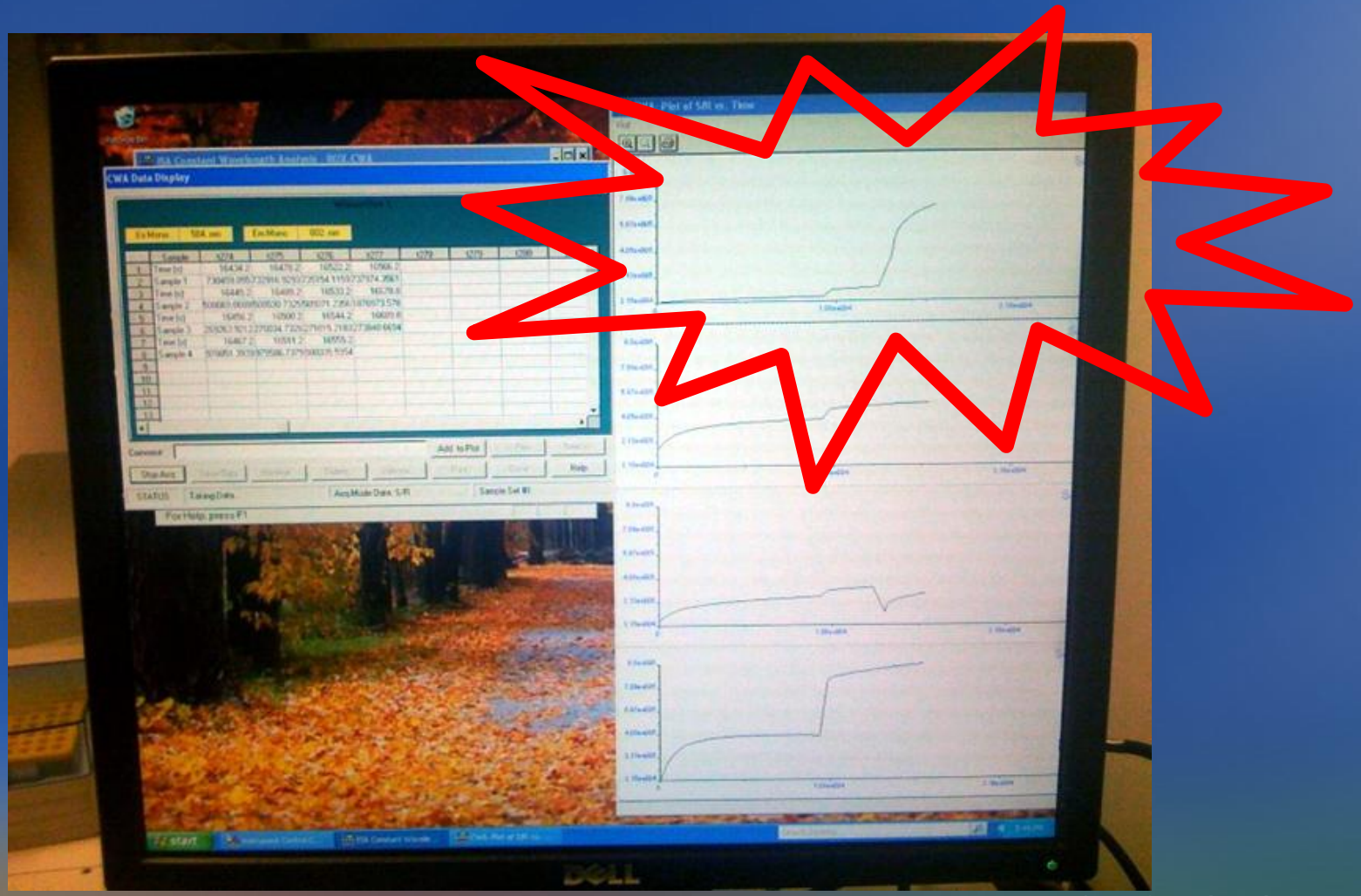
# Execute (finally!)

- Fluorescence is your one-bit ‘print’ statement



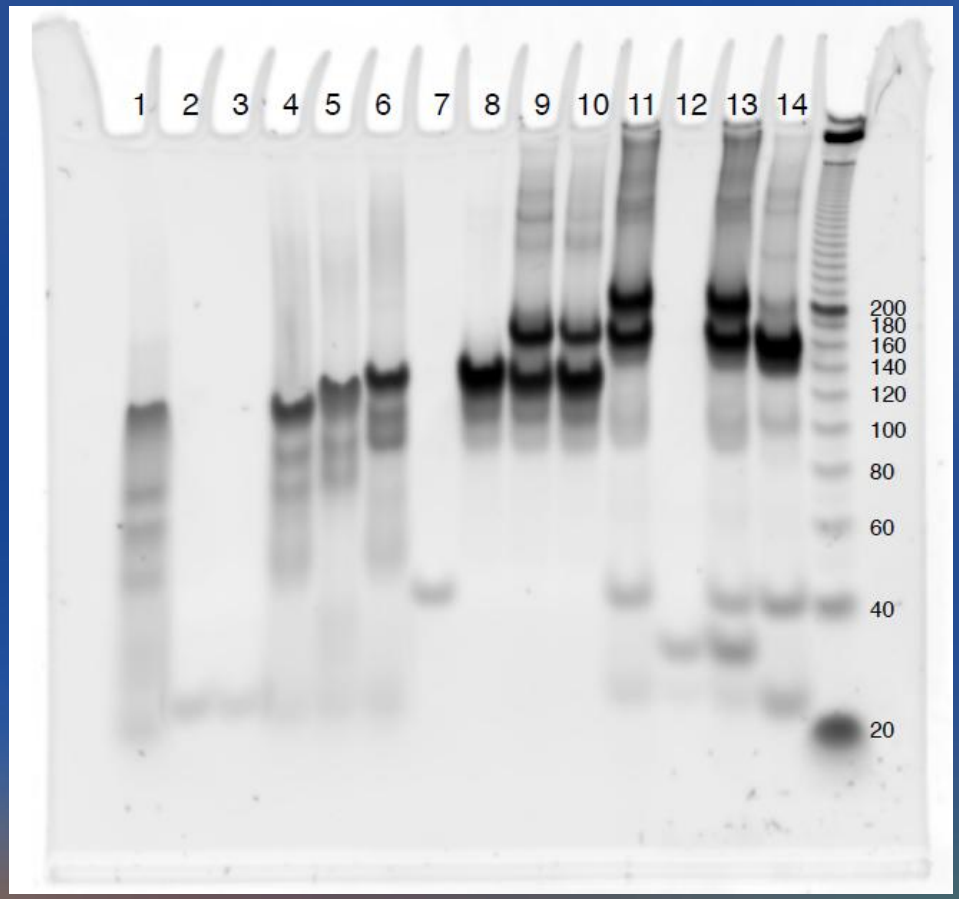




# Output



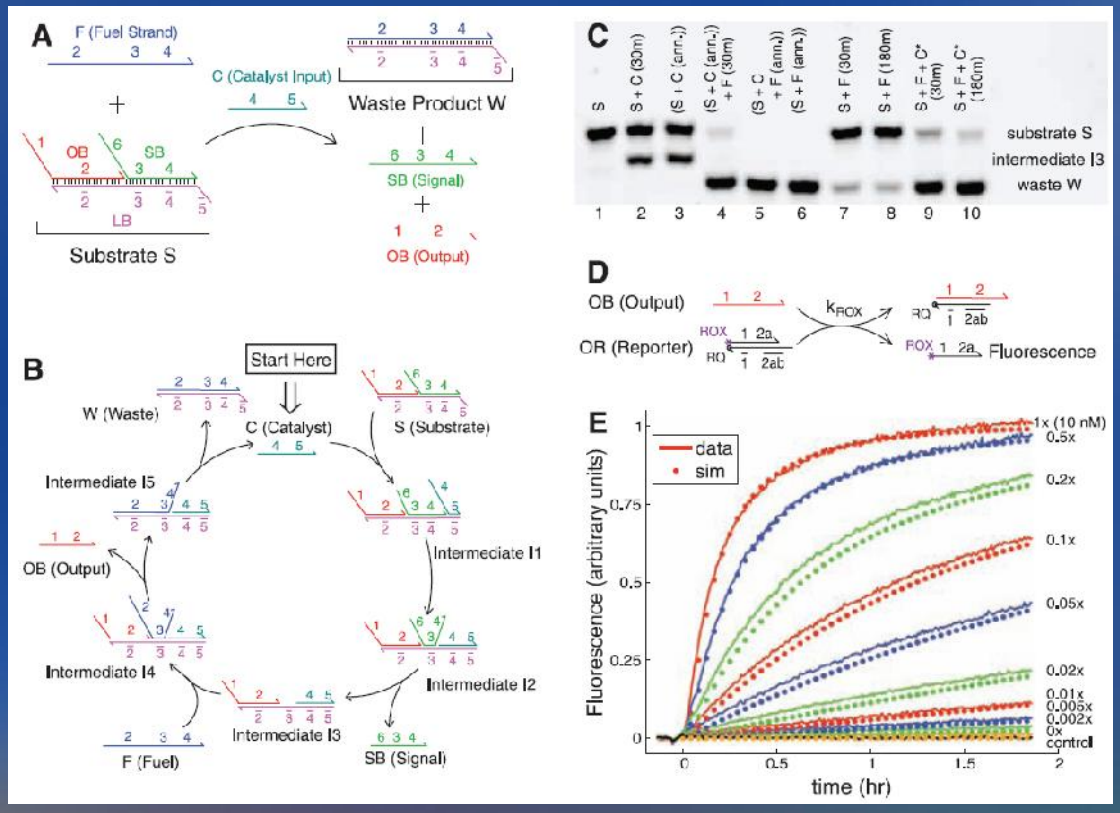
# Debugging

- A core dump



# Delivering!

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



Microsoft Engineering Excellence

# A Molecular Algorithm

Running something interesting with DNA

Microsoft

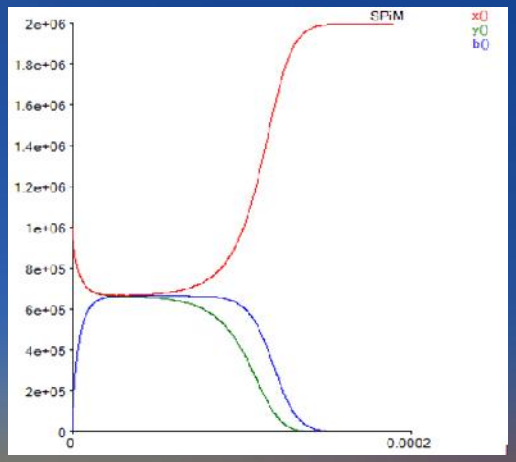
# Approximate Majority Algorithm

- Given two populations of agents (or molecules)
  - Randomly communicating by radio (or by collisions)
  - Reach an agreement about which population is in majority
  - By converting all the minority to the majority
    [Angluin et al., Distributed Computing, 2007]
- Could be used to restore a signal to full strength

- A chemical implementation
  - X + Y → B + B
  - B + X → X + X
  - B + Y → Y + Y

# Surprisingly good (in fact, optimal)

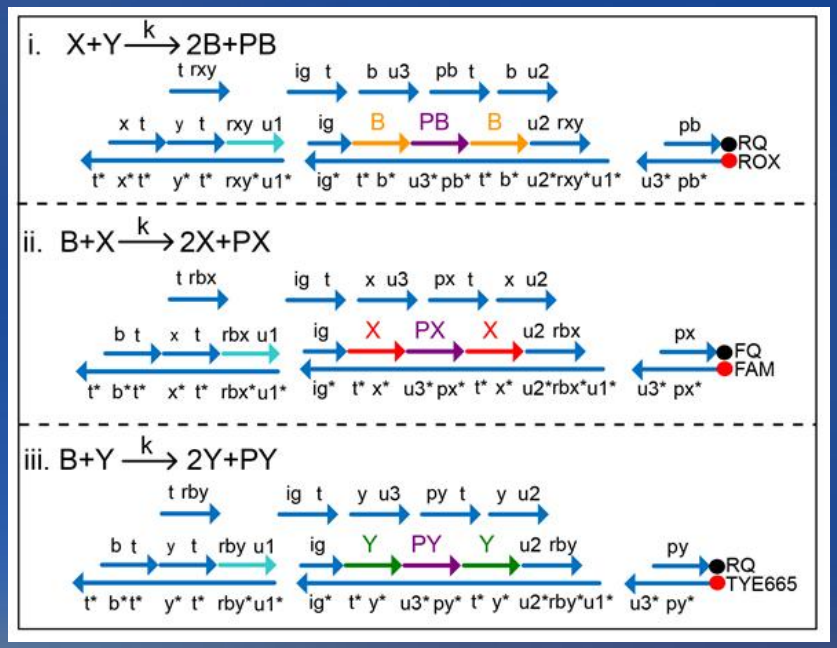
- Fast: reaches agreement in O(log n) time w.h.p.
  - O(n log n) communications/collisions
  - Even when initially #X = #Y! (stochastic symmetry breaking)
- Robust: true majority wins w.h.p.
  - If initial majority exceeds minority by $\omega(\sqrt{n} \log n)$
  - Hence the agreement state is stable

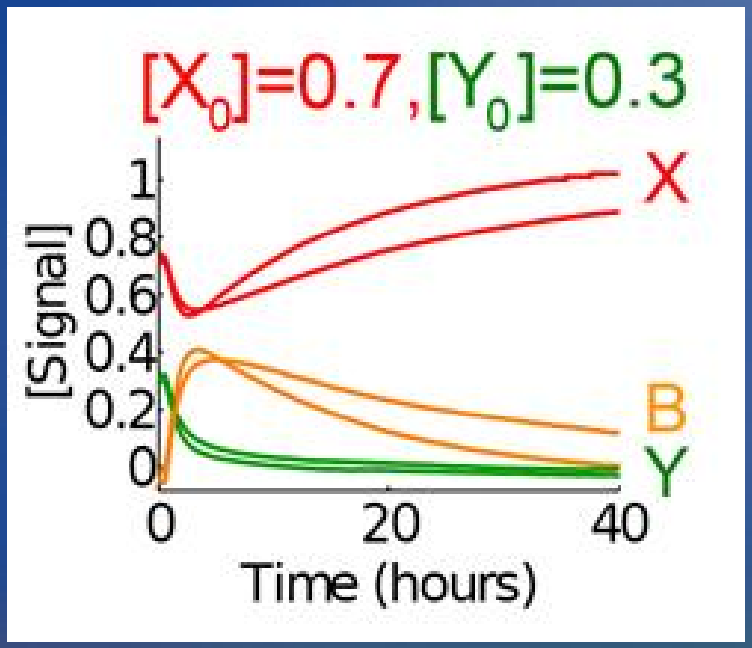



Stochastic simulation of worst-case, with initially #X = #Y

# DNA Implementation, at U.W.

- **A DNA Realization of Chemical Reaction Networks** [Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik and Georg Seelig]

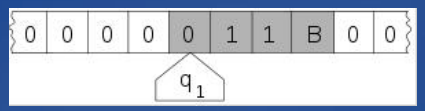


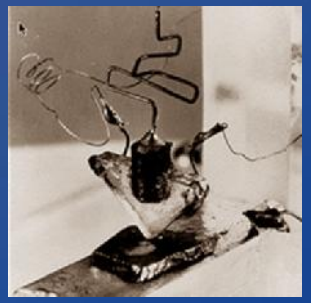
# Final Remarks

# A Brief History of DNA

Turing Machine, 1936

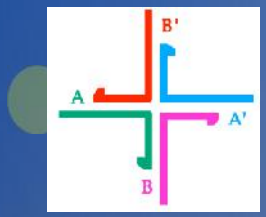



Transistor, 1947

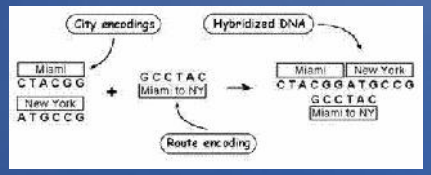


DNA, -3,800,000,000

Structural DNA Nanotech, 1982




DNA Algorithm, 1994




Digital ~~Computers~~

Computer programming

**Software**

*systematic manipulation of information*

20th century

**??**

*systematic manipulation of matter*

21th century

DNA ~~Computers~~

Molecular programming

# Acknowledgments


- Microsoft Research
  - Andrew Phillips
- Caltech
  - Winfree Lab
- U.Washington
  - Seelig Lab

Microsoft Engineering Excellence

# Questions?

Microsoft

# Resources

- Visual DSD at MSR
  http://research.microsoft.com/apps/pubs/default.aspx?id=157262
- Molecular Programming Project at Caltech
  http://molecular-programming.org/
- Georg Seelig's DNA Nanotech Lab at U.W. CS&E
  http://homes.cs.washington.edu/~seelig/

- This slide deck and related resources:
  http://lucacardelli.name/Talks/2012-12-06 Molecular Programming (Redmond).pptx