

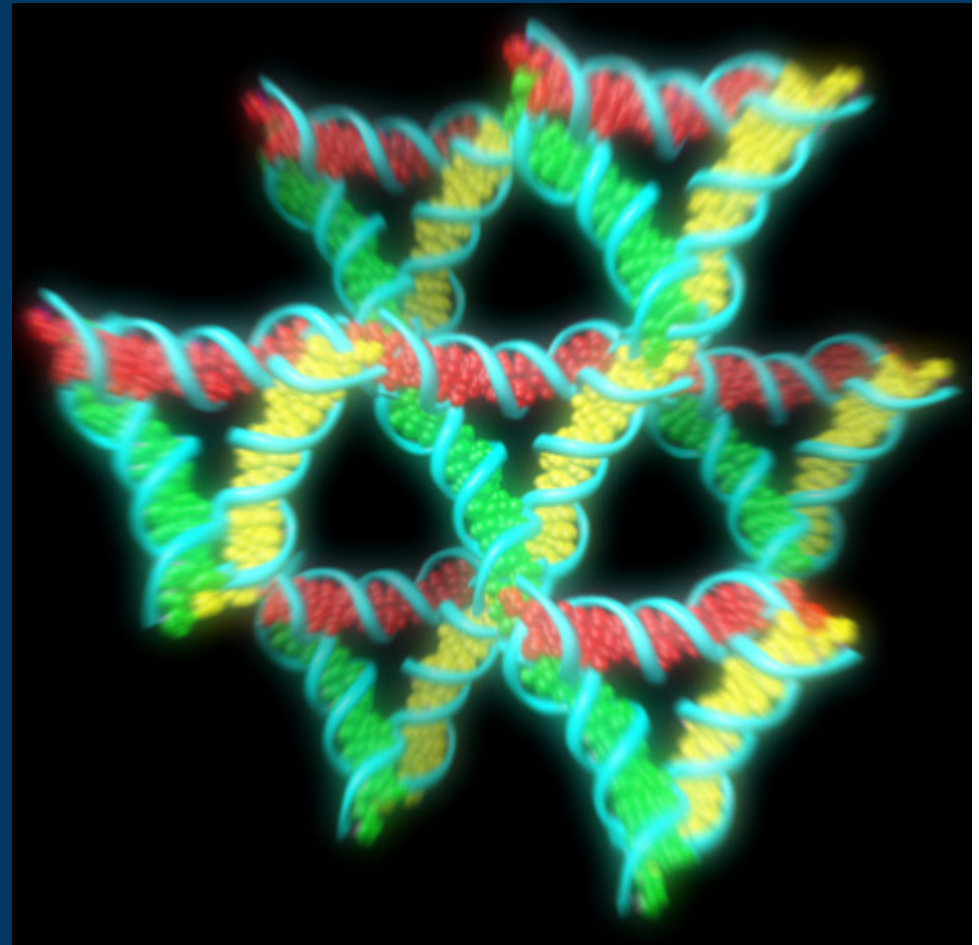
Molecular Programming

The systematic
manipulation of matter

Luca Cardelli

Microsoft Research

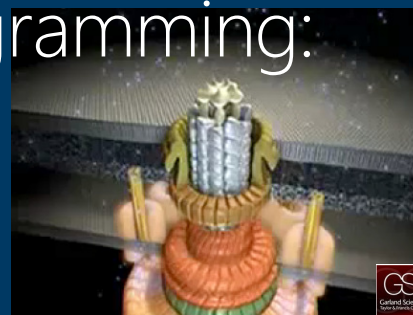
2014-07-30 ECOOP Keynote, Uppsala



Objectives

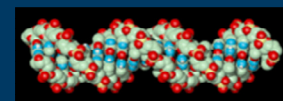
- The promises of Molecular Programming:

- 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

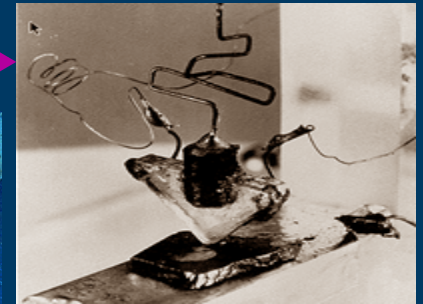
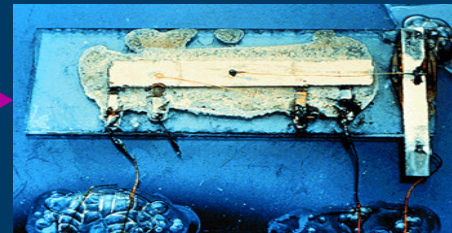
Intel&Micron, Jan. 2010. ~50atoms

Single molecule transistor

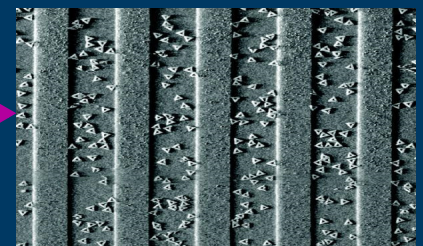
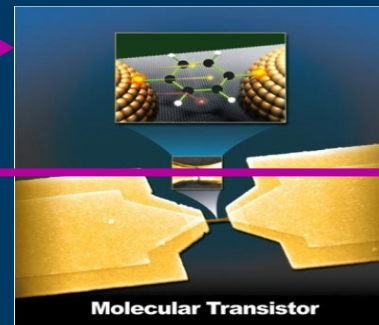
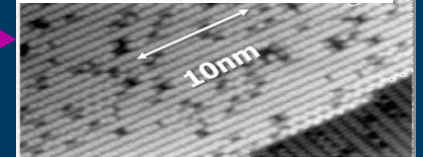
Observation of molecular orbital gating
Nature, 2009; 462 (7276): 1039

Molecules on a chip

~10 Moore's Law cycles left!



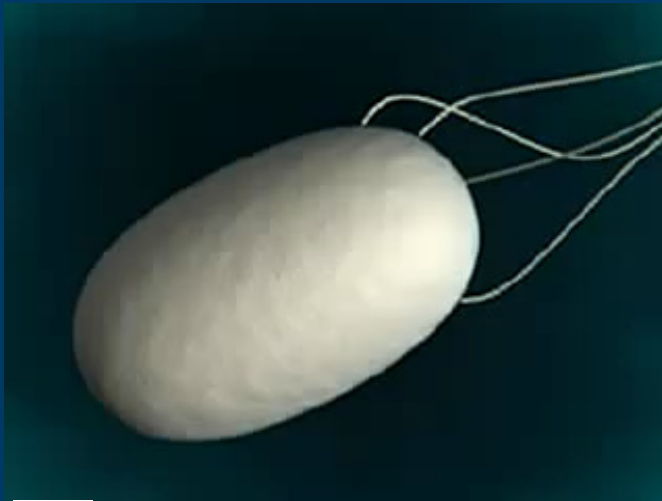
Scanning tunneling microscope image of a silicon surface showing 10nm is ~20 atoms across



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*

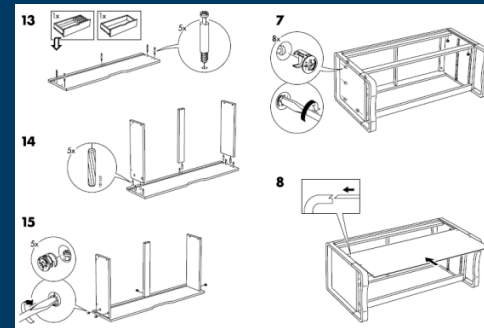


www.youtube.com/watch?v=Ey7Emddf7Y

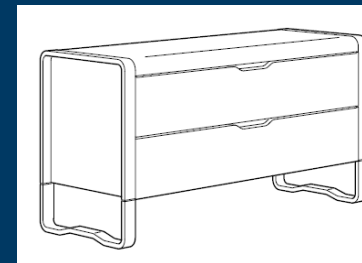


Molecular IKEA

- 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



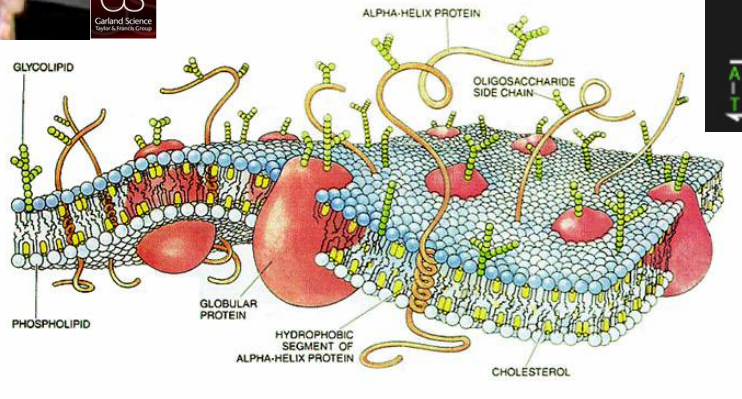
http://www.ikea.com/ms/en_US/customer_service/assembly_instructions.html

Programmed Self-Assembly

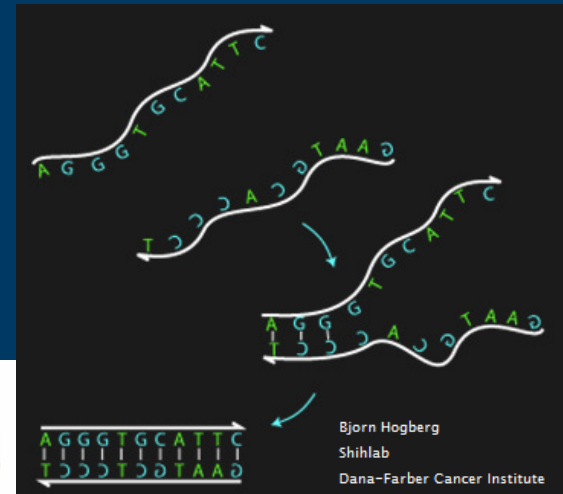
Proteins



Membranes



DNA/RNA



The Software Argument

Smaller and smaller things can be programmed

We can program...

- Computers.
- Completely!



Information



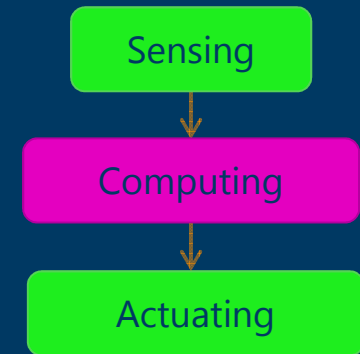
Computing



Information

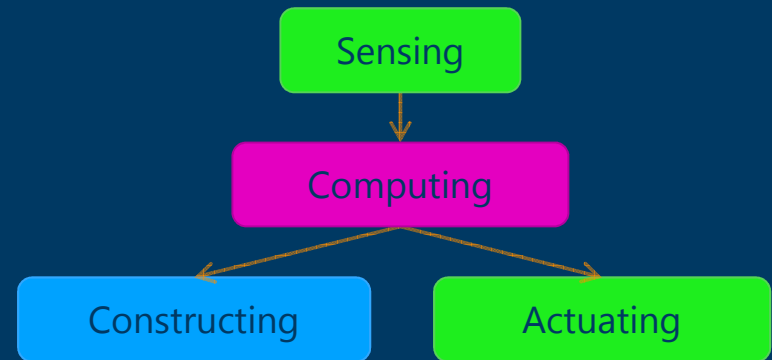
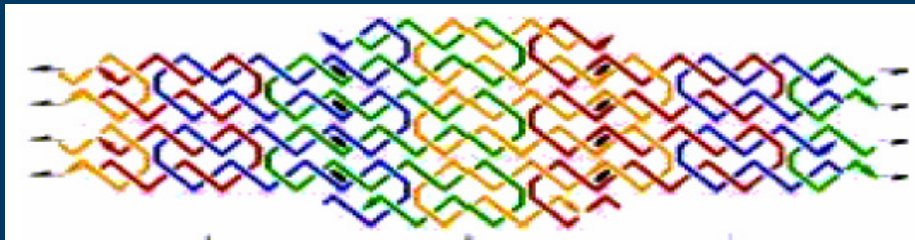
We can program...

- Physical systems.
 - Completely!
(Modulo sensors/actuators)



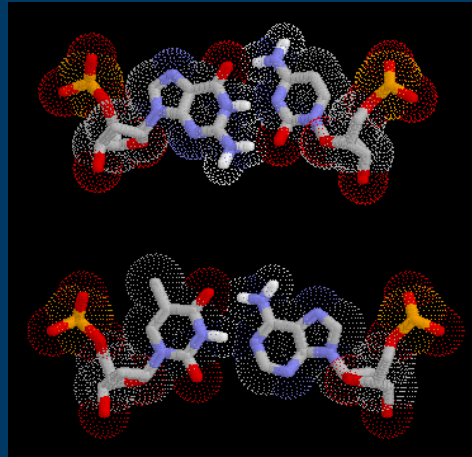
We can program...

- Matter
 - Completely and directly!
 - Currently: only DNA/RNA.



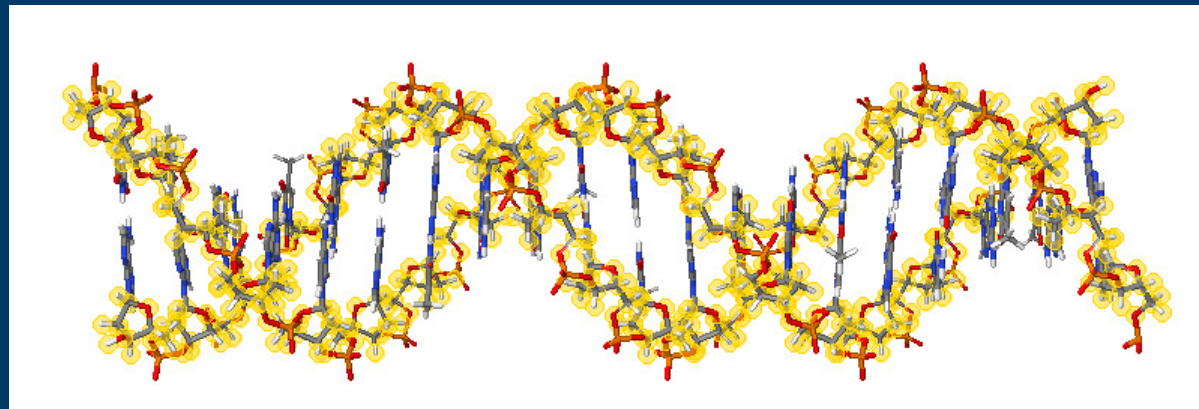
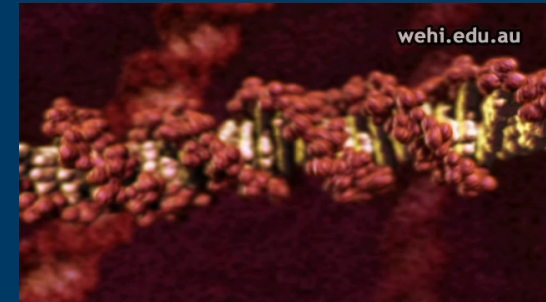
It's like a 3D printer without the printer!
[Andrew Hellington]

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>)

Robust, and *Long*

- DNA in each human cell:
 - 3 billion base pairs
 - 2 meters long, 2nm thick
 - folded into a $6\mu\text{m}$ 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



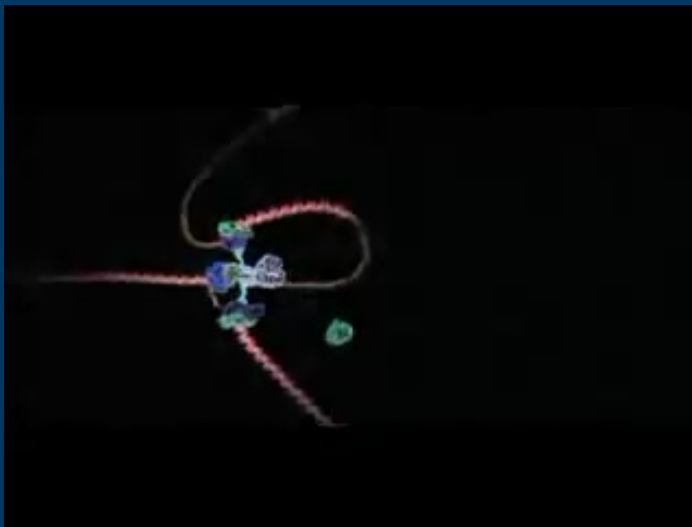
DNA wrapping into chromosomes



Andromeda Galaxy
2.5 million light years

Zippering Along

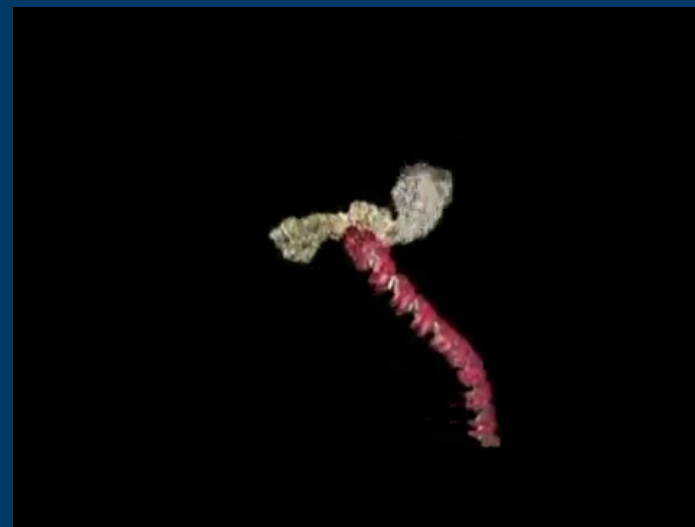
- DNA can support structural and computational complexity.



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)



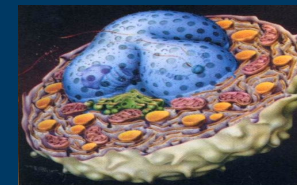
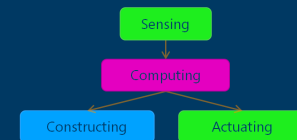
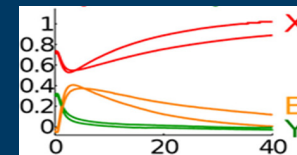
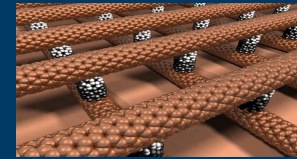
DNA transcription in *real time*

RNA polymerase II: 15-30 base/second

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

What can we do with “just” DNA?

- Organize ANY matter [caveats apply]
- Execute ANY kinetics [caveats: up to time scaling]
- Build Nano-Control Devices
- Interface to Biology

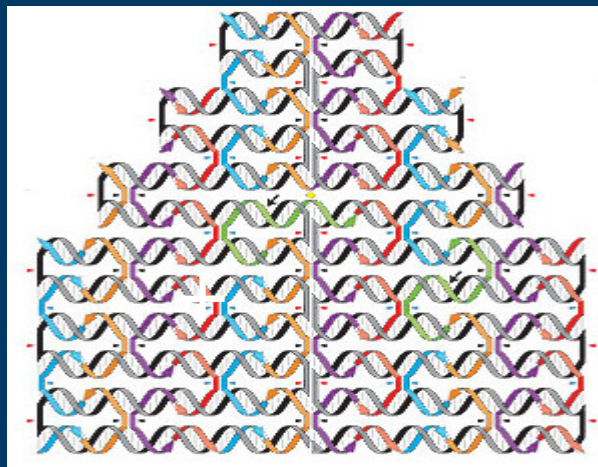


H.Lodish & al. Molecular Cell Biology 4th ed.

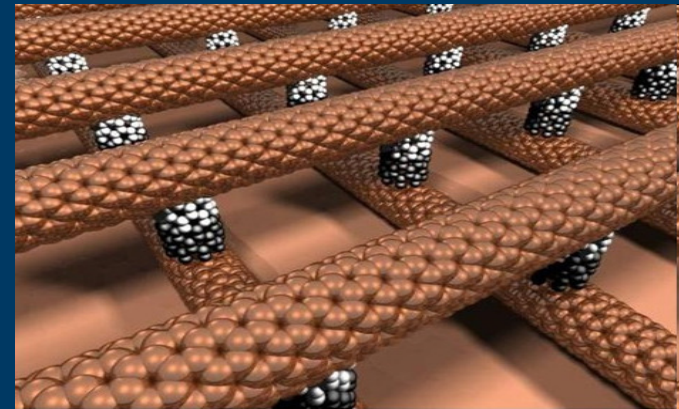
Organizing Any Matter

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

6 nm grid of individually addressable DNA pixels



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



European Nanoelectronics Initiative Advisory Council

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

Executing Any Kinetics

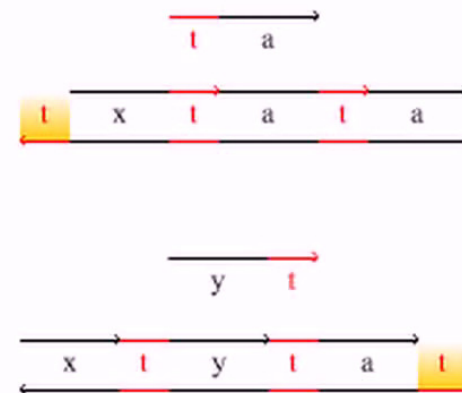
- The kinetics of any finite network of chemical reactions, can be implemented (physically) with especially programmed DNA molecules.
- Chemical reactions as an executable programming language for dynamical systems!

DNA as a universal substrate for chemical kinetics PNAS

David Soloveichik^{a,1}, Georg Seelig^{a,b,1}, and Erik Winfree^{c,1}

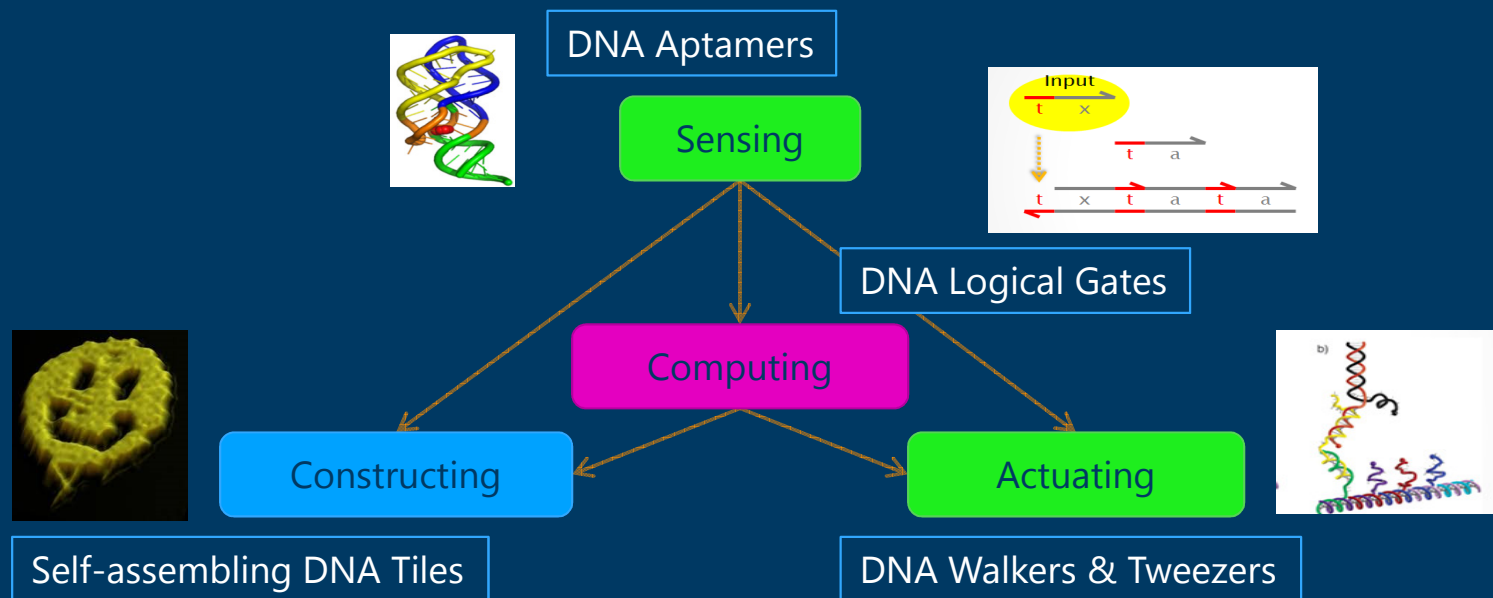
Powered by Sothink

Transducer $x \rightarrow y$



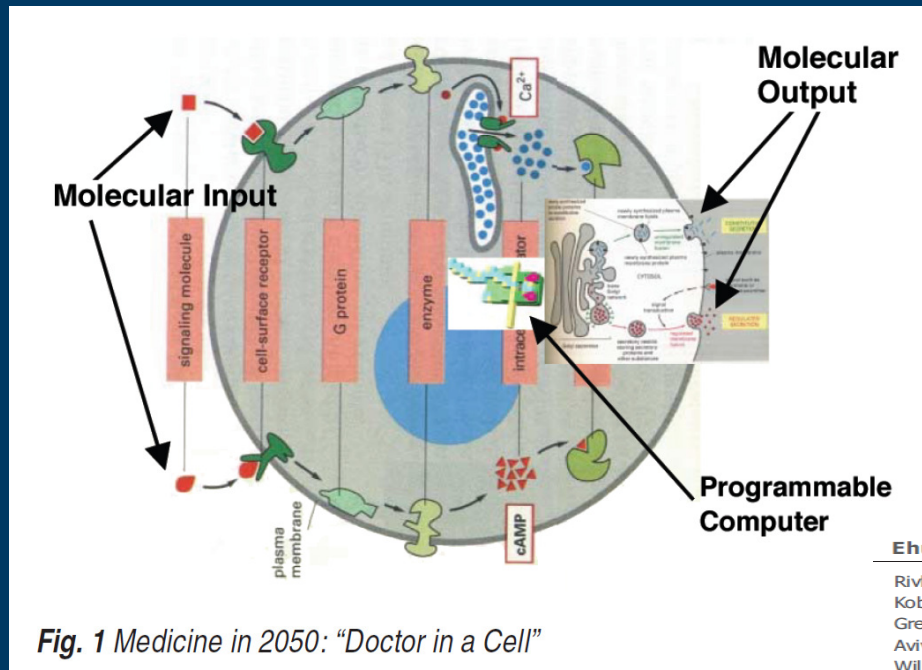
Building Nano-Control Devices

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



Interfacing to Biology

- A doctor in each cell



Ehud Shapiro

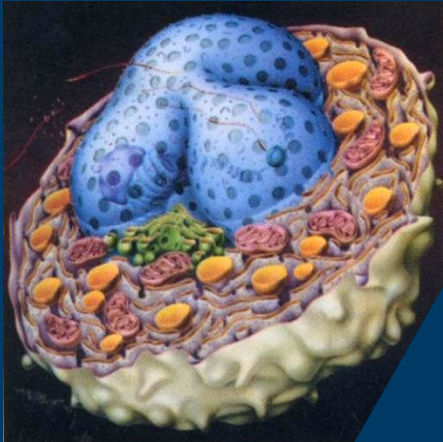
Rivka Adar
Kobi Benenson
Gregory Linshitz
Aviv Regev
William Silverman

**Molecules and
computation**

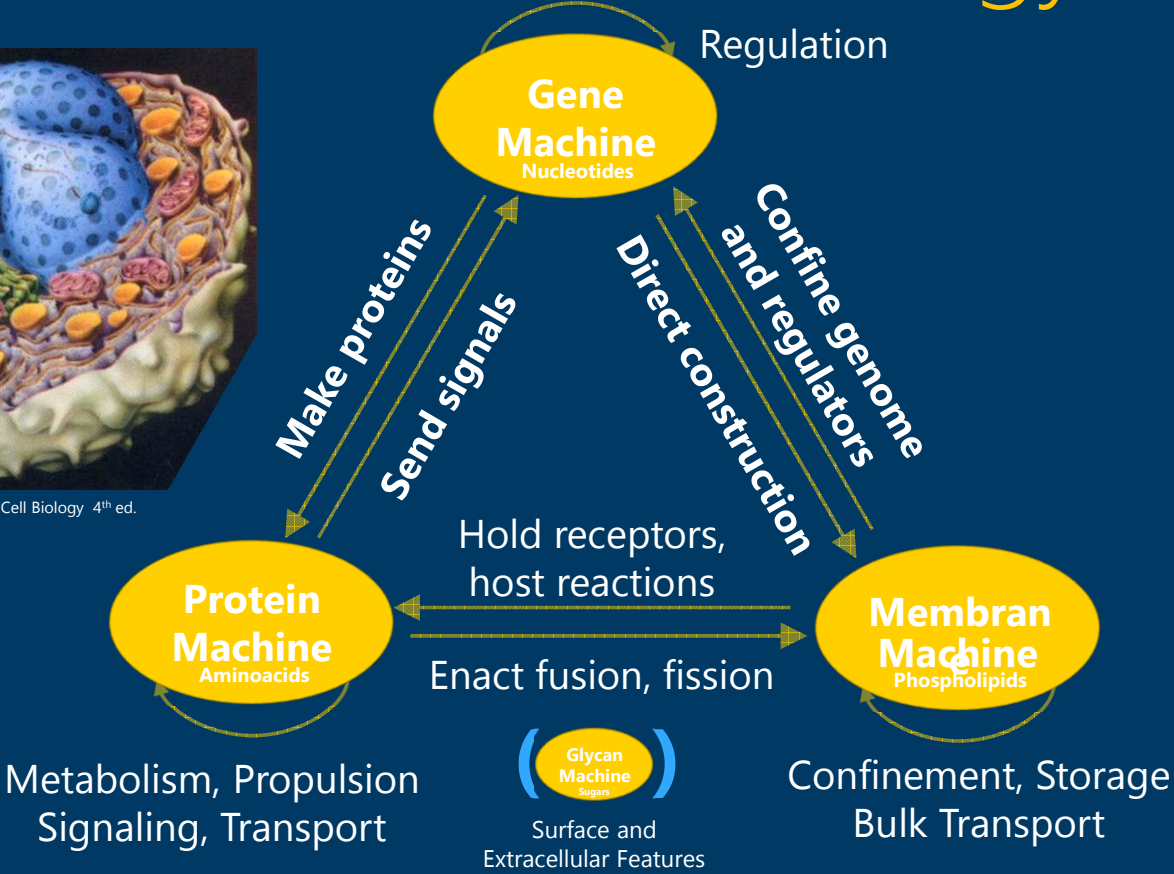
The Biological Argument

Biological systems are already
'molecularly programmed'

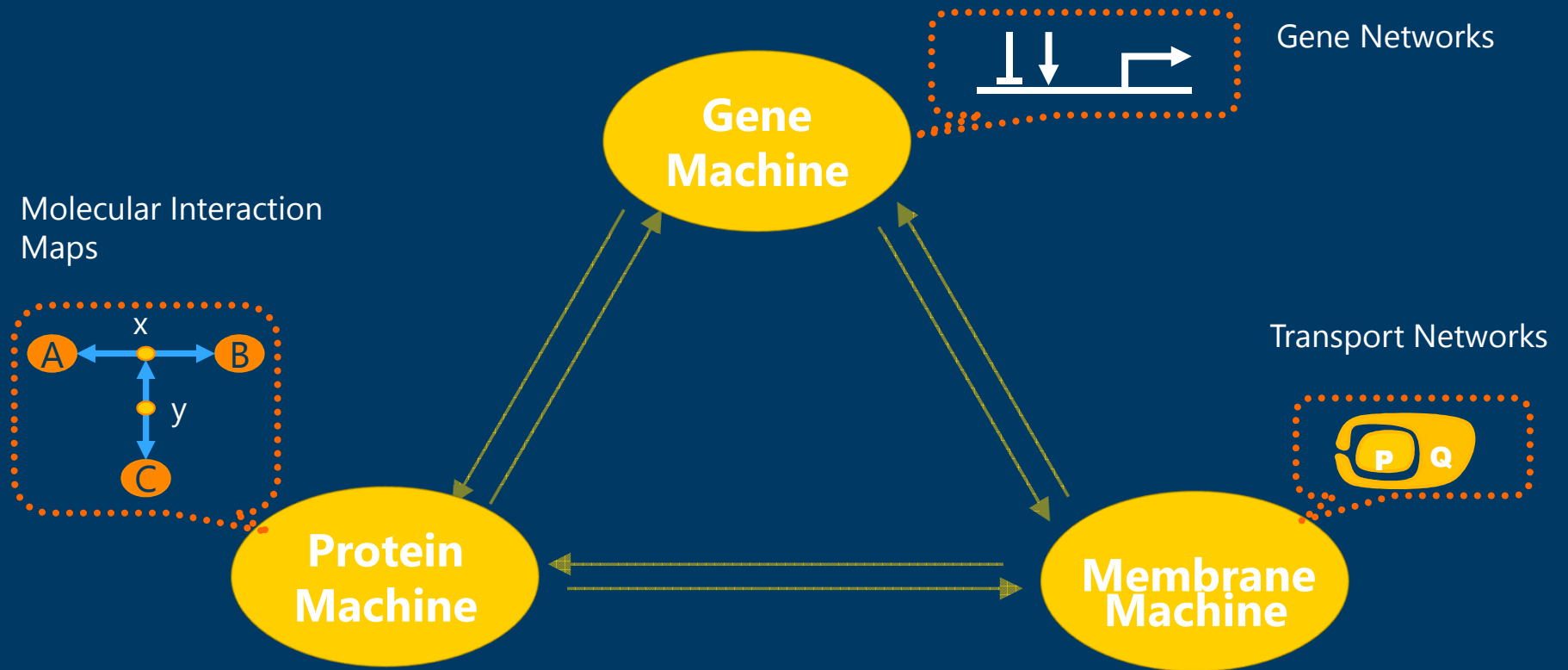
Abstract Machines of Biology



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



Biological Languages



But ...

- Biology is programmable, but (mostly) 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, but we cannot yet effectively design proteins
 - Transport networks are being looked at for programming microfluidic devices manipulating vesicles

Molecular Languages

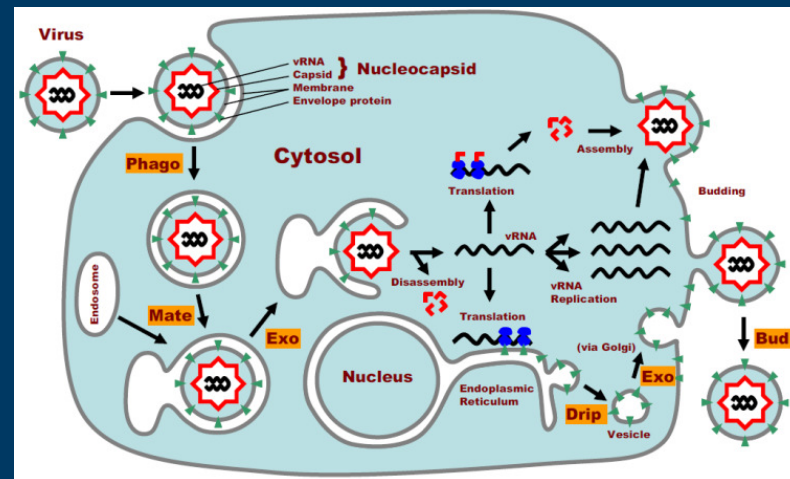
... that **we** can execute

Action Plan

- Building a full software/hardware pipeline for a new fundamental technology
 - **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]
- To realize the potential of Molecular Programming
- “With *no alien technology*” [David Soloveichik]
- This is largely a ‘software problem’ even when working on device design

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...



Our Assembly Language: Chemistry

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

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?

Molecular Programming with DNA

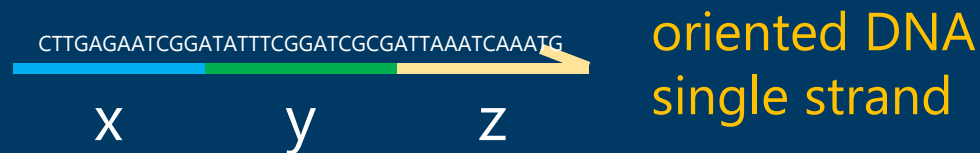
Building the cores of programmable
molecular controllers

The role of 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
 - Other such materials will be (are being) developed

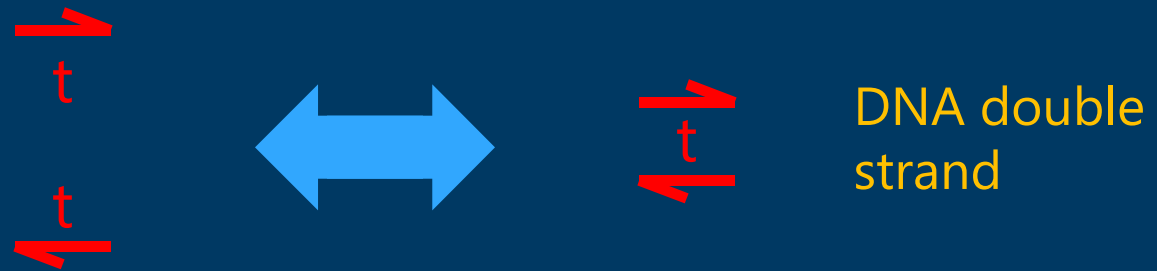
Domains

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



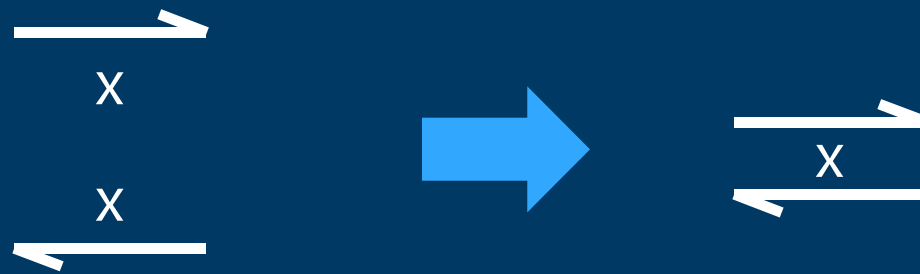
- Differently named domains must not **hybridize**
 - With each other, with each other's complement, with subsequences of each other, with concatenations of other domains (or their complements), etc.

Short Domains



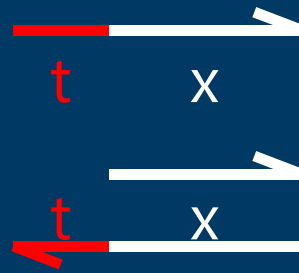
Reversible Hybridization

Long Domains



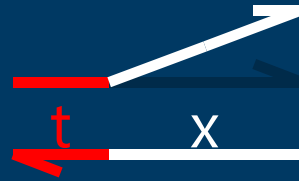
Irreversible Hybridization

Strand Displacement



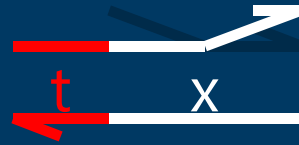
“Toehold Mediated”

Strand Displacement



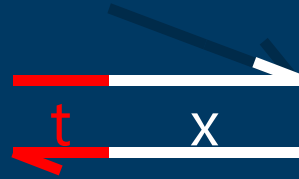
Toehold Binding

Strand Displacement



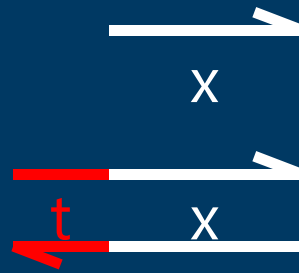
Branch Migration

Strand Displacement



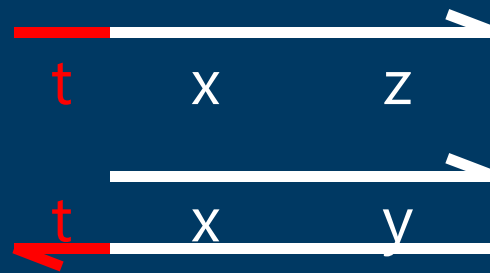
Displacement

Strand Displacement

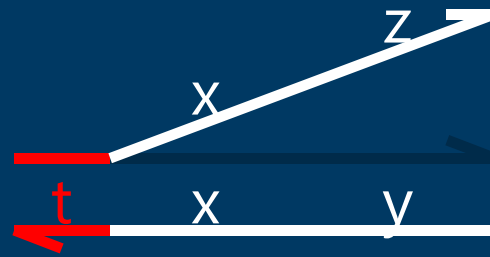


Irreversible release

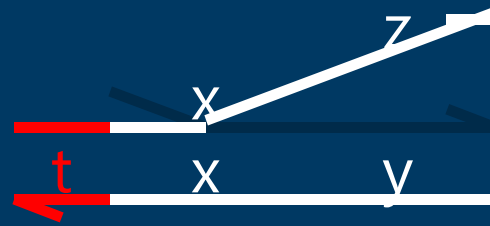
Bad Match



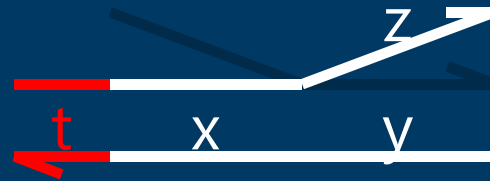
Bad Match



Bad Match



Bad Match



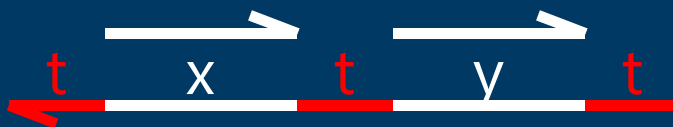
Cannot proceed
Hence will undo

Two-Domain Architecture

- Signals: 1 toehold + 1 recognition region



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



Garbage collection
“built into” the gate
operation

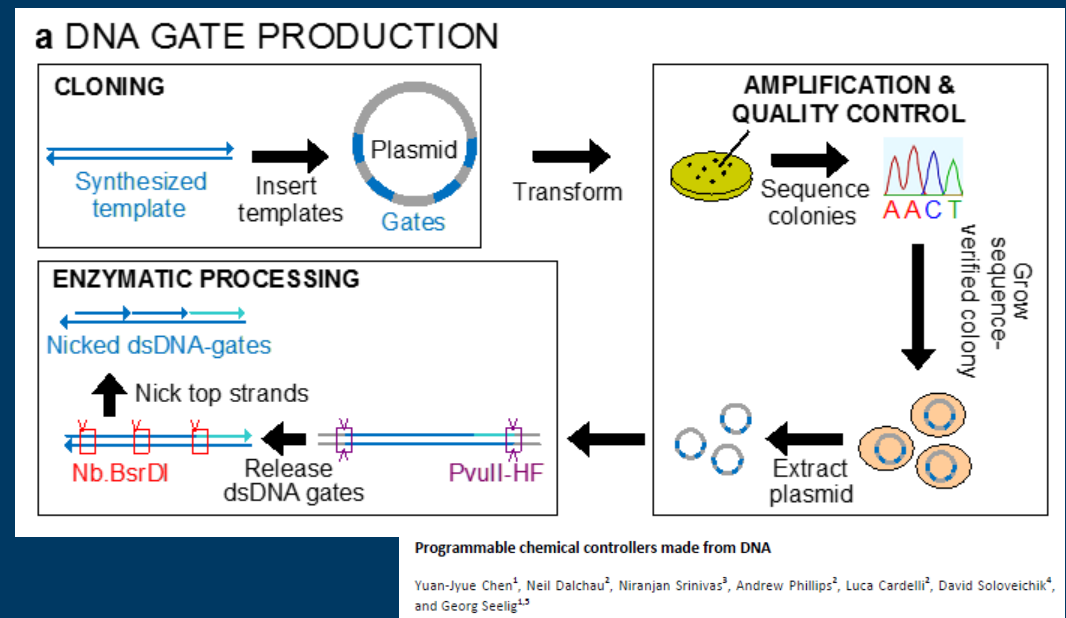
Two-Domain DNA Strand Displacement

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.

Plasmidic Gate Technology

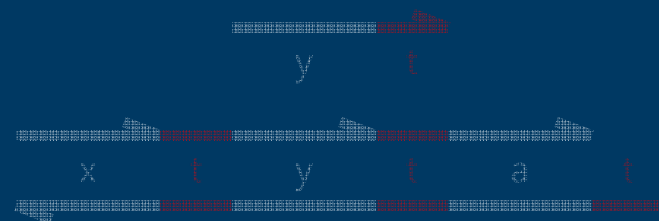
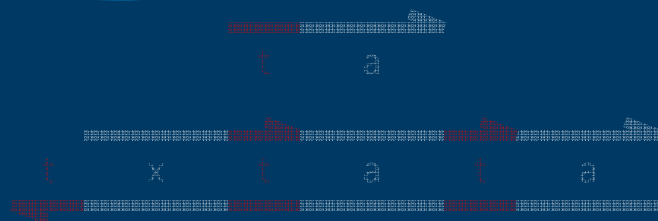
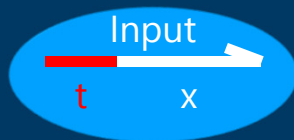
- Synthetic DNA is length-limited
 - Finite error probability at each nucleotide addition, hence ~ 200nt max
- Bacteria can replicate plasmids for us
 - Loops of DNA 1000's nt, with extremely high fidelity
 - Practically no structural limitations on gate fan-in/fan-out



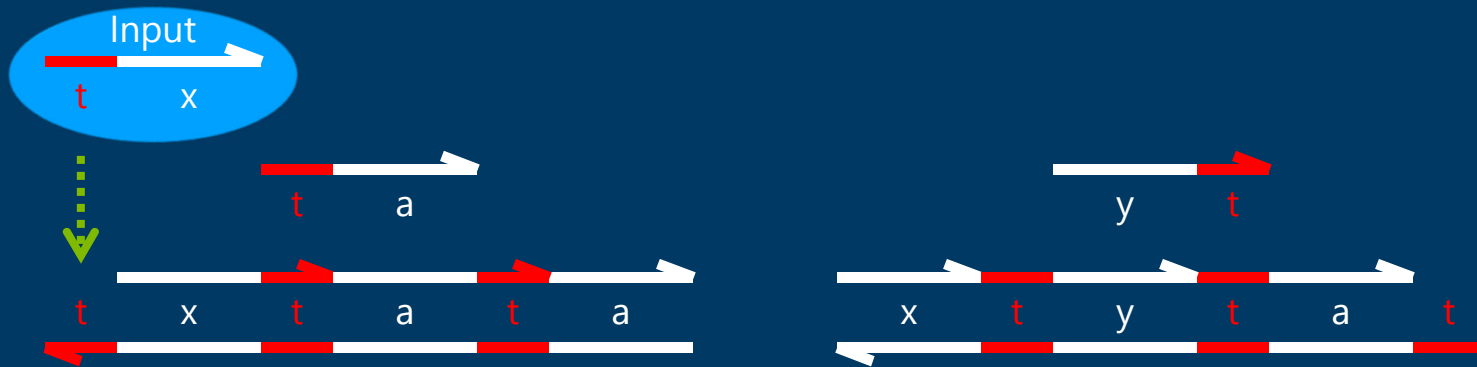
Only possible with two-domain architecture

Transducer

Transducer $x \rightarrow y$



Transducer $x \rightarrow y$



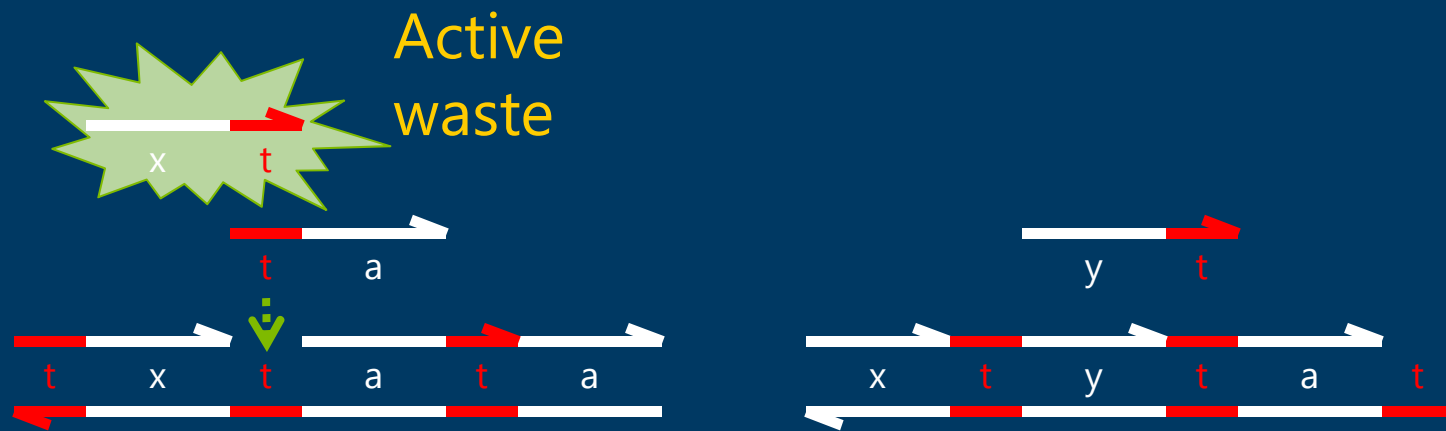
Built by self-assembly!

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

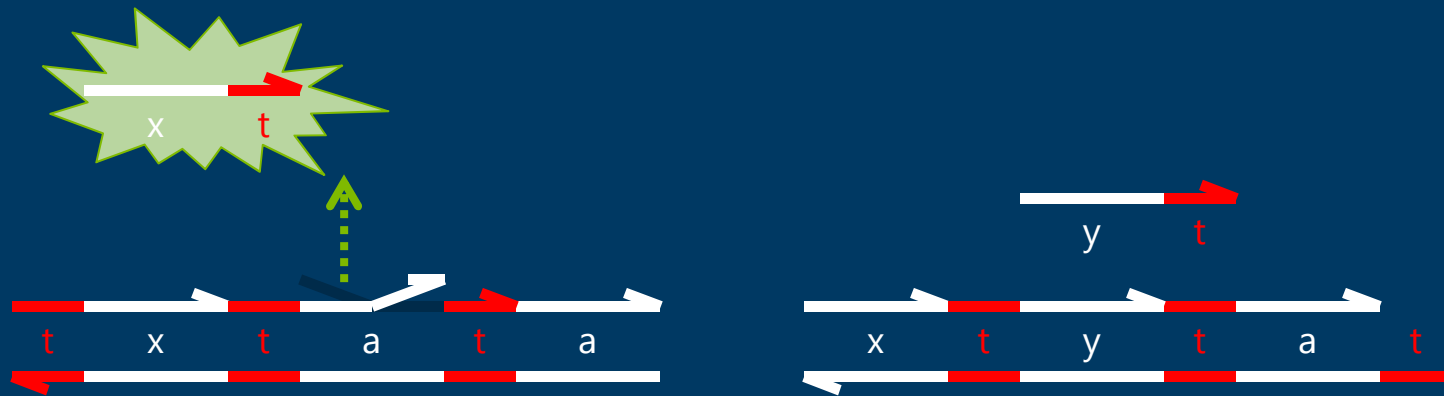
Transducer $x \rightarrow y$



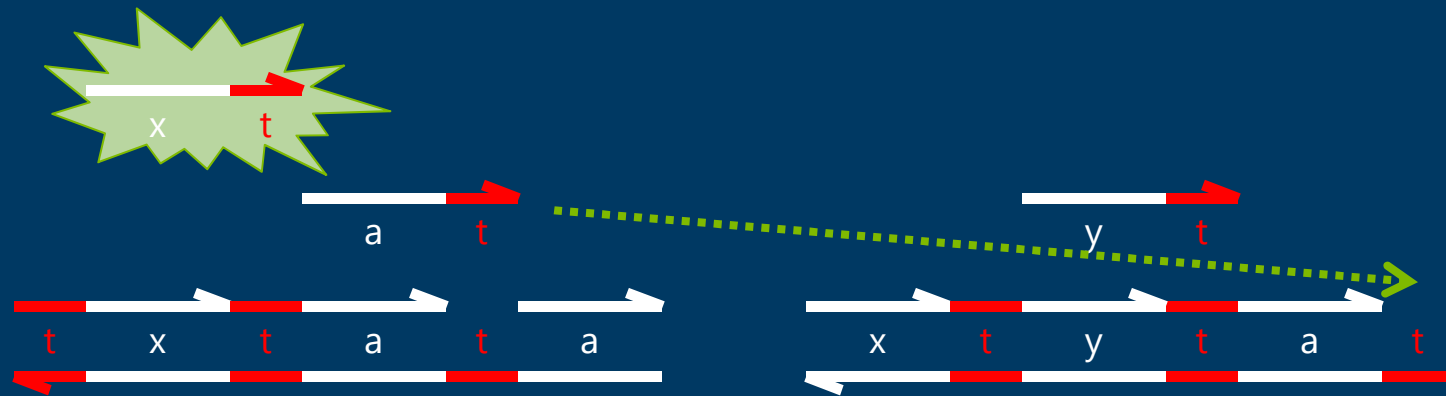
Transducer $x \rightarrow y$



Transducer $x \rightarrow y$

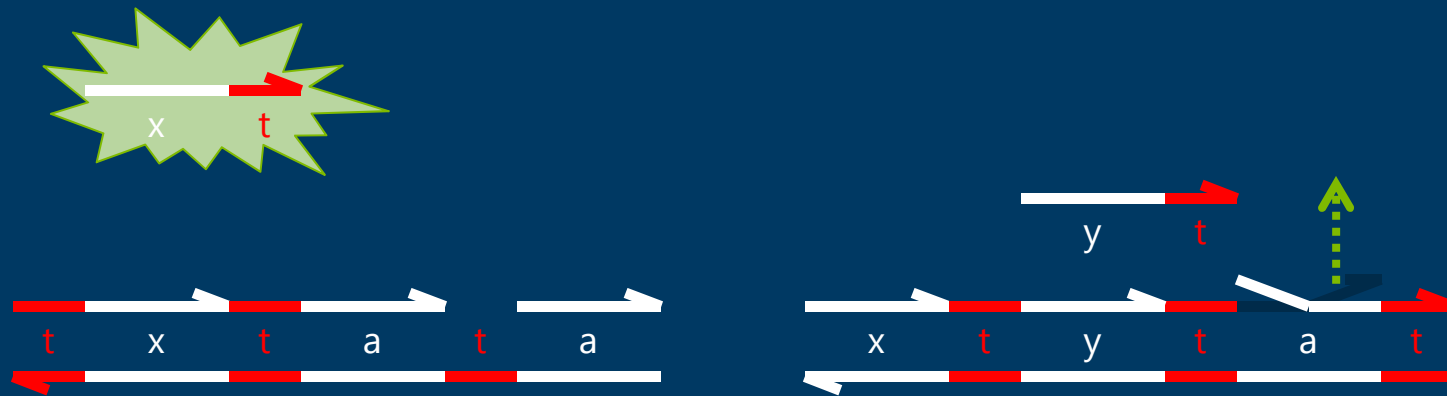


Transducer $x \rightarrow y$



So far, a **tx** signal has produced an **at** cosignal.
But we want signals as output, not cosignals.

Transducer $x \rightarrow y$



Transducer $x \rightarrow y$



Transducer $x \rightarrow y$



Transducer $x \rightarrow y$



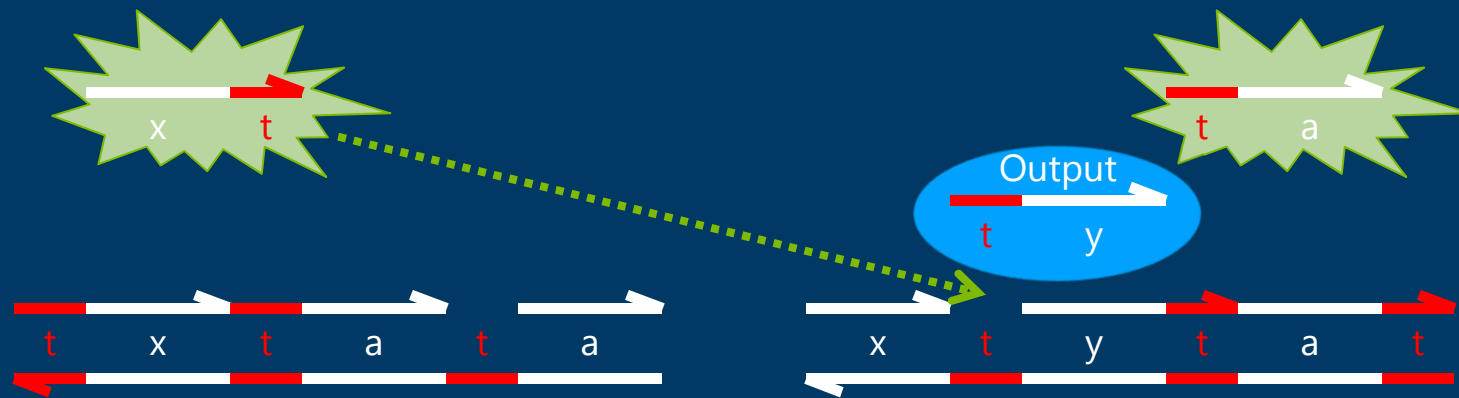
Here is our output **ty** signal.

But we are not done yet:

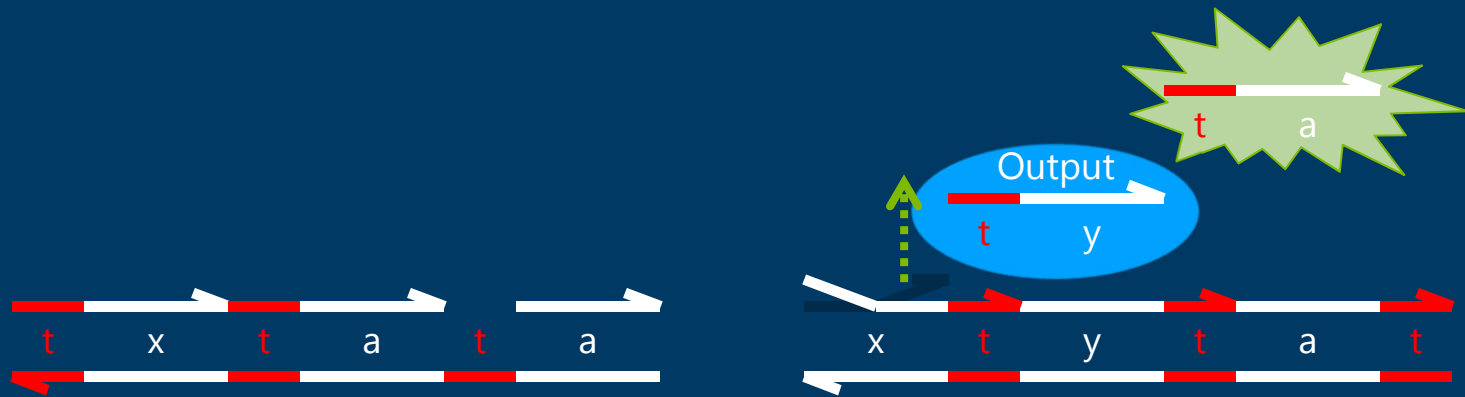
- 1) We need to make the output irreversible.
- 2) We need to remove the garbage.

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

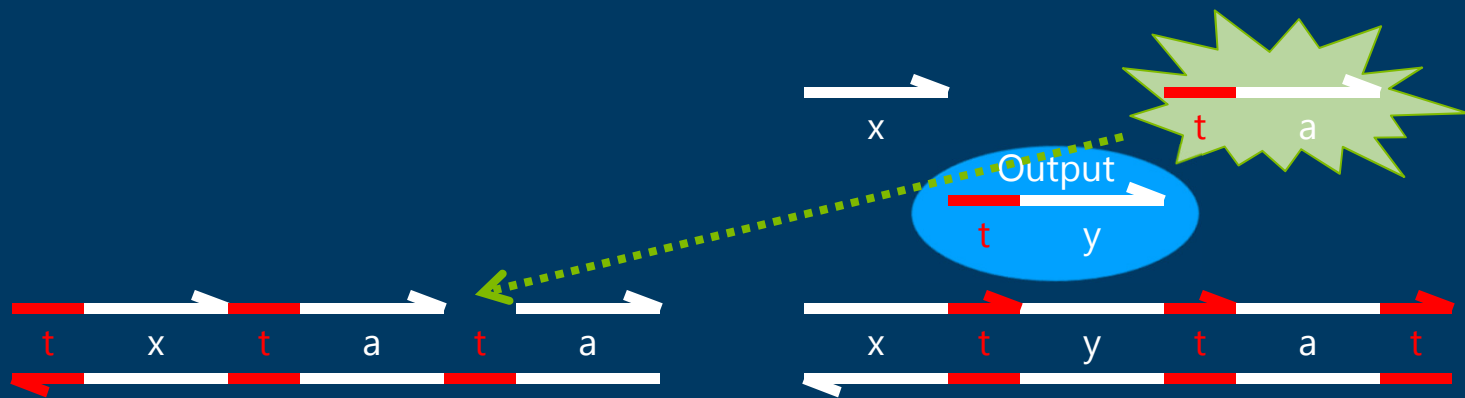
Transducer $x \rightarrow y$



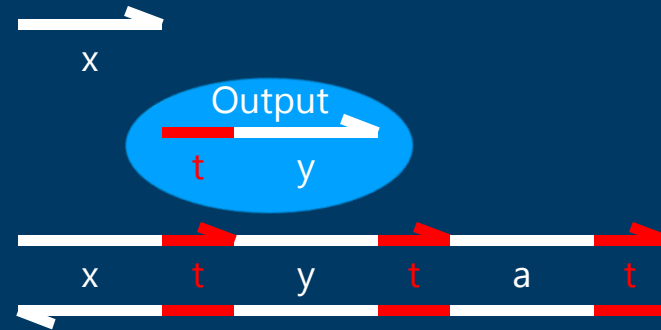
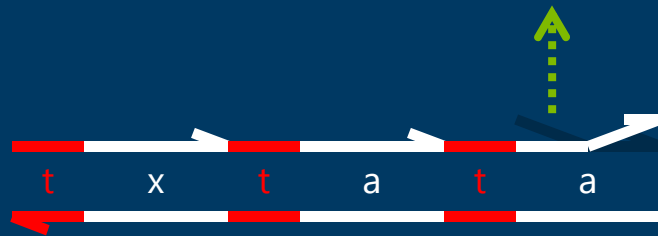
Transducer $x \rightarrow y$



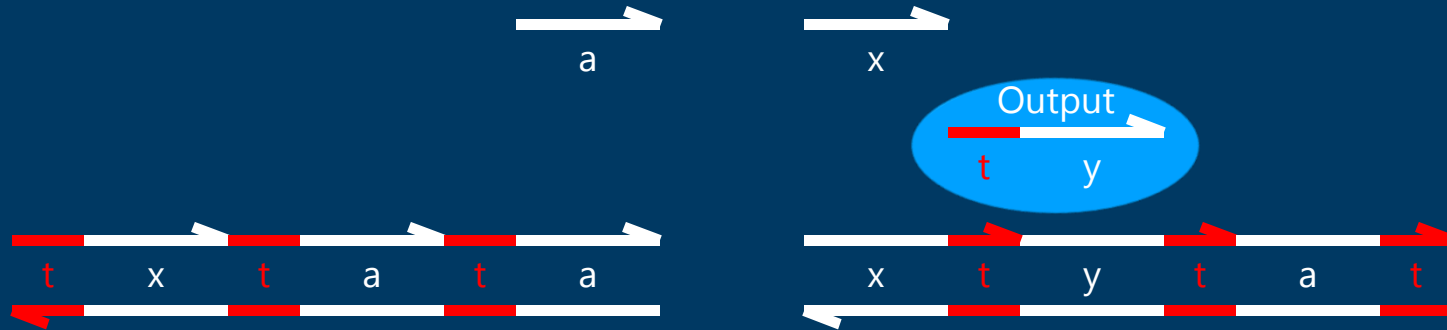
Transducer $x \rightarrow y$



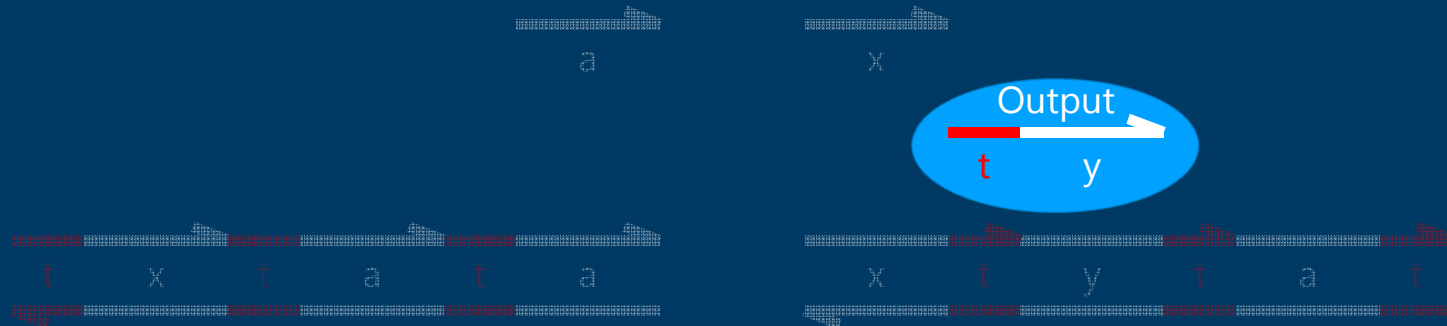
Transducer $x \rightarrow y$



Transducer $x \rightarrow y$



Transducer $x \rightarrow y$



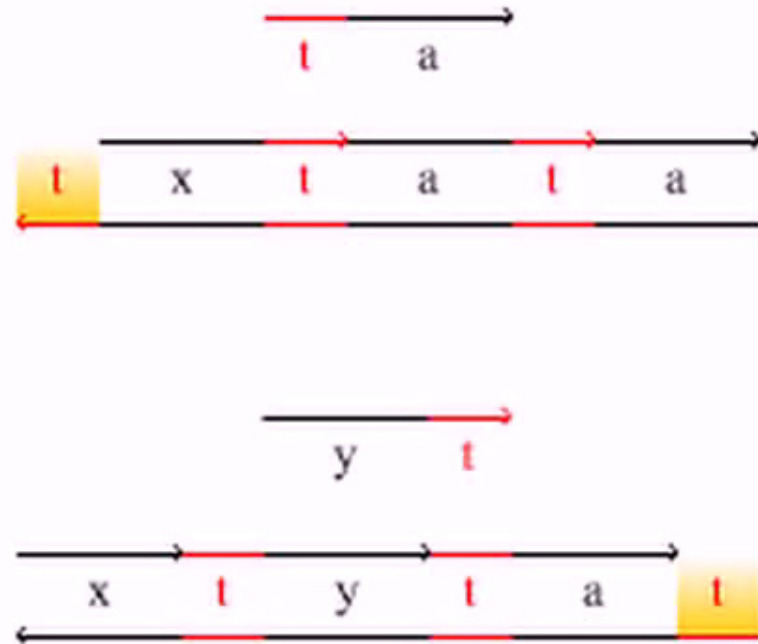
Done.

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

(no proteins, no enzymes, no heat-cycling, etc.; just DNA in salty water)

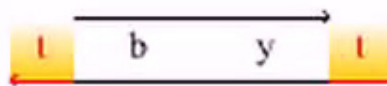
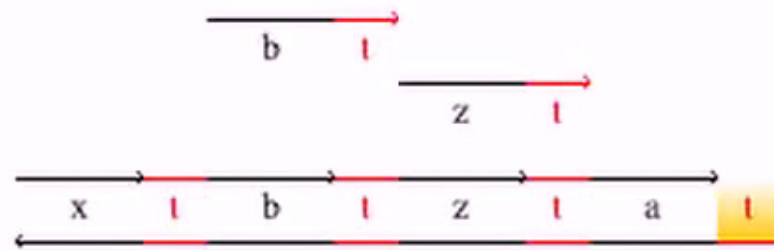
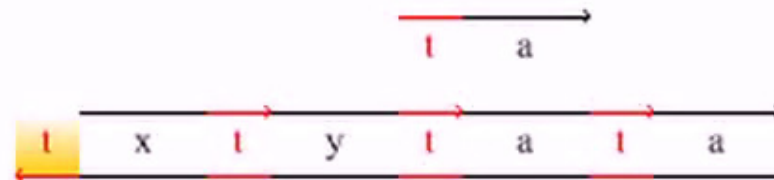
Powered by Sothink

Transducer $x \rightarrow y$



Powered by Sothink

Join $x+y \rightarrow z$



Tools and Techniques

A software pipeline for Molecular Programming

Development Tools

MSRC Biological Computation Group

Visual DSD

A Development Environment for DNA Strand Displacement

The screenshot displays the Visual DSD software interface. On the left, a code editor shows a program for a catalytic reaction. A red box highlights the `Cat` function definition:

```
def Cat(N, x, y, z) =  
  new a  
  ( (1.5*N) * t^*: [x t^]: [y u^]: [a]  
    | (1.5*N) * [x]: [t^ z]: [t^ y]: u^*  
    | (2.0*N) * <u^ a>  
    | (2.0*N) * <z t^>  
  )  
  
def Rep(N, x, f1) =  
  ((3.0*N) * t^*: [x] <f1^>)  
  
( onex * <Calibration>  
  | Cat(onex, X, Y, B)  
  | Rep(onex, B, f1)  
  | onex * <t^ X>  
  | onex * <t^ Y>  
)
```

The central panel shows a graphical representation of DNA strand displacement reactions. It features several reaction schemes with DNA strands represented as horizontal lines with colored segments (red, green, blue) and arrows indicating binding and unbinding. Equilibrium arrows (\leftrightarrow) connect different states of the system.

On the right, a plot window shows a graph of the reaction progress over time. The x-axis represents time from 0 to 5000, and the y-axis represents a quantity from 0 to 50. A green curve shows a sigmoidal increase, leveling off at a value of approximately 45. The plot includes a legend with entries for `<Calibration>` and `<B f1^>`.

At the bottom left, there is a logo for the **JOURNAL OF THE ROYAL SOCIETY Interface** and a text box containing the title and authors:

A programming language for composable DNA circuits
Andrew Phillips and Luca Cardelli

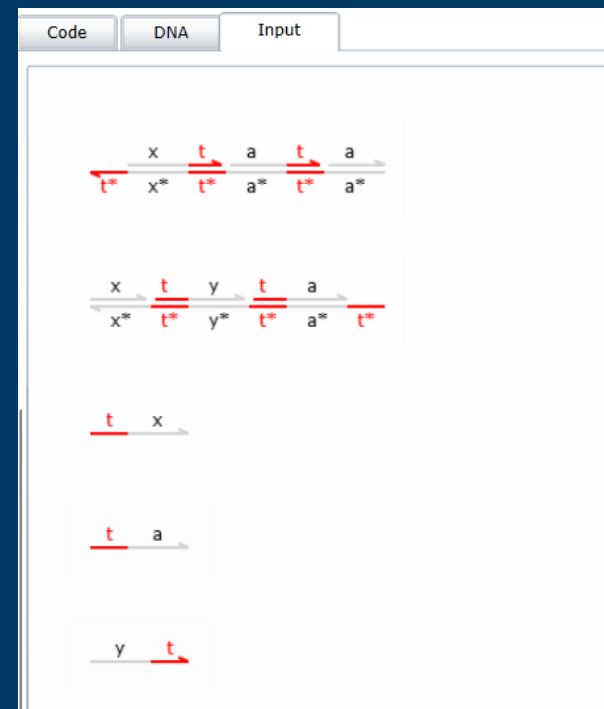
A Language for DNA Structures

- Describe the initial structures

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

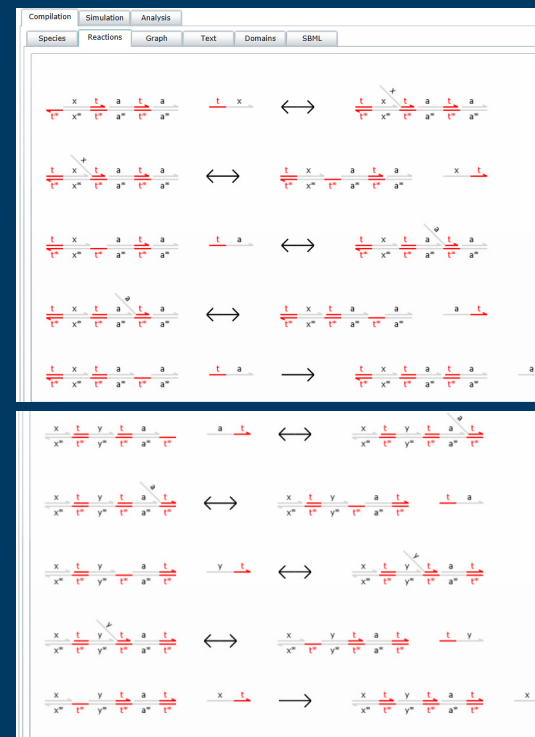
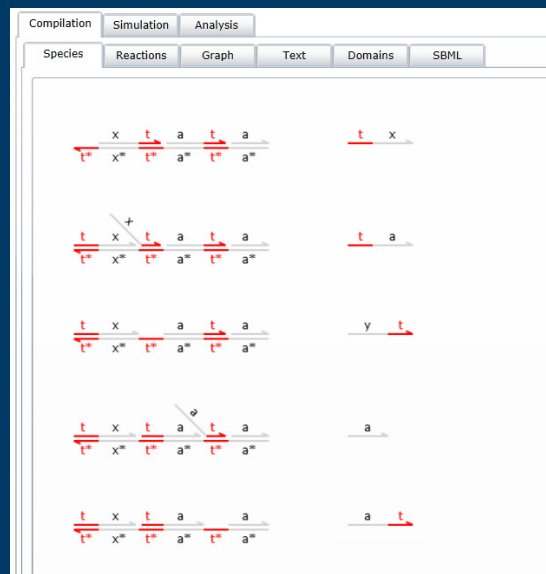
def T(N,x,y) =
  new a
  ( N * <t^ a>
  | N * <y t^>
  | N * t^*: [x t^]: [a t^]: [a] (* Input gate *)
  | N * [x]: [t^ y]: [t^ a]: t^* (* Output gate *)
  )

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

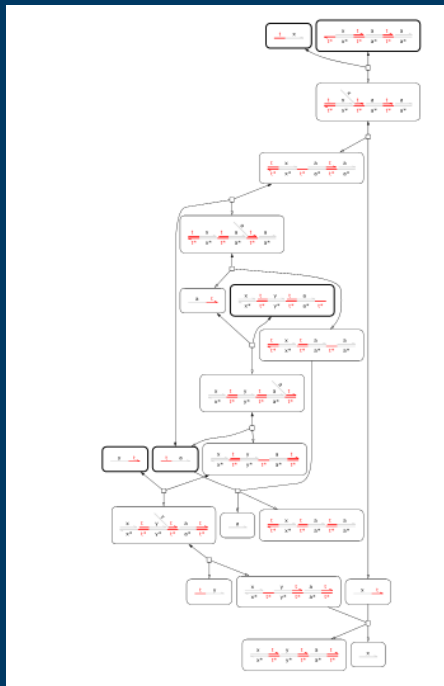


Compute Species and Reactions

- Recursively computed from the initial structures



Reaction Graph and Export

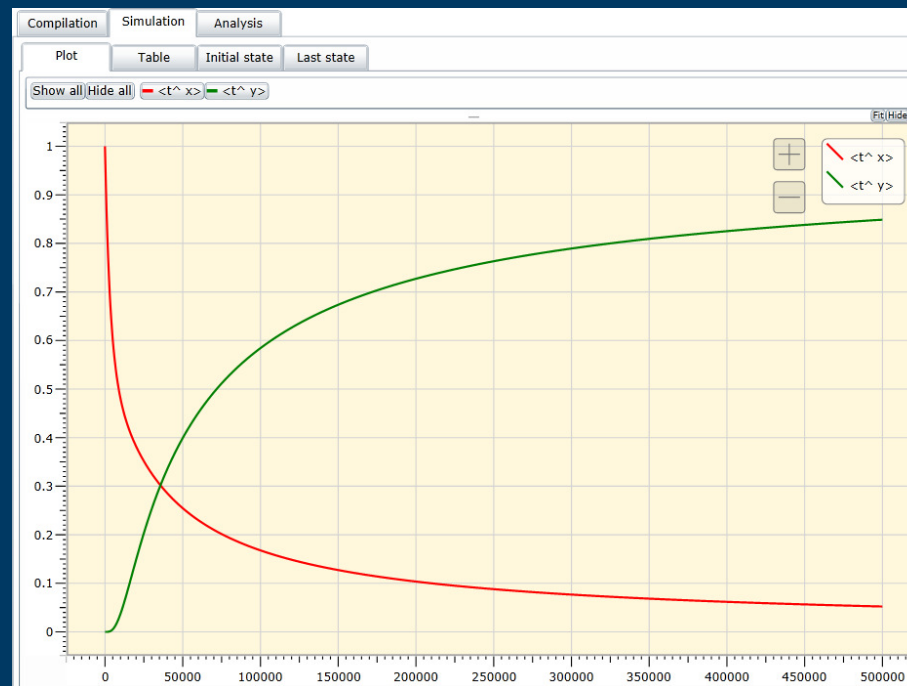


```
Compilation Simulation Analysis
Species Reactions Graph Text Domains SBML
Save as XML
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2/version1" level="2" version="1">
<model>
<listOfCompartments>
<compartment id="c" size="1"/>
</listOfCompartments>
<listOfSpecies>
<species id="s_id0" name="&t^ x^" compartment="c" initialAmount="1" constant="false"/>
<species id="s_id1" name="&t^ a^" compartment="c" initialAmount="1" constant="false"/>
<species id="s_id2" name="&t^ y^" 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]:&t;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]:&t;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="&t;a;" compartment="c" initialAmount="0" constant="false"/>
<species id="s_id10" name="&t;a t^>" compartment="c" initialAmount="0" constant="false"/>
<species id="s_id11" name="&t;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]:&t;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]:&t;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="&t;x;" compartment="c" initialAmount="0" constant="false"/>
<species id="s_id19" name="&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"



State Space Analysis

Compilation Simulation Analysis
Graph Text PRISM Visualise

INITIAL STATE:

$\frac{t}{x^m} a (1)$

$\frac{t}{x} (1)$

$\frac{y}{t} (1)$

$\frac{x}{x^m} \frac{t}{t^m} \frac{y}{y^m} \frac{t}{t^m} \frac{a}{a^m} \frac{t}{t^m} (1)$

$\frac{x}{t^m} \frac{t}{x^m} \frac{t}{t^m} \frac{a}{a^m} \frac{t}{t^m} \frac{a}{a^m} (1)$

TERMINAL STATE:

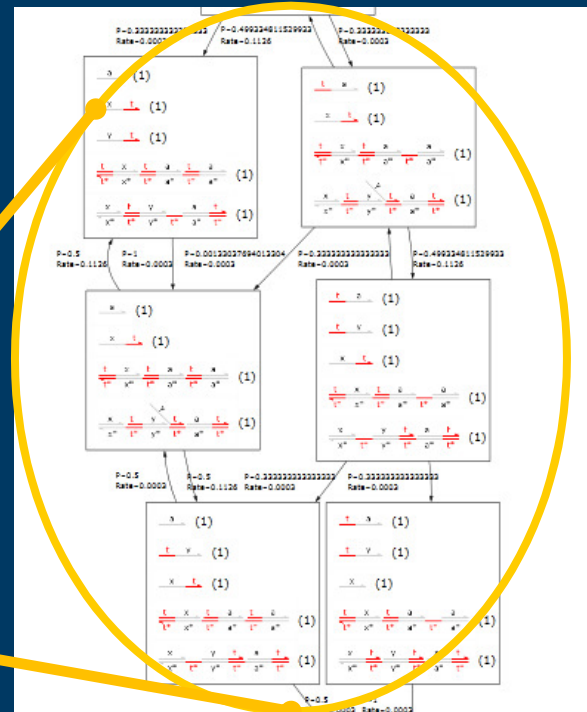
$\frac{a}{t} (1)$

$\frac{t}{y} (1)$

$\frac{x}{t} (1)$

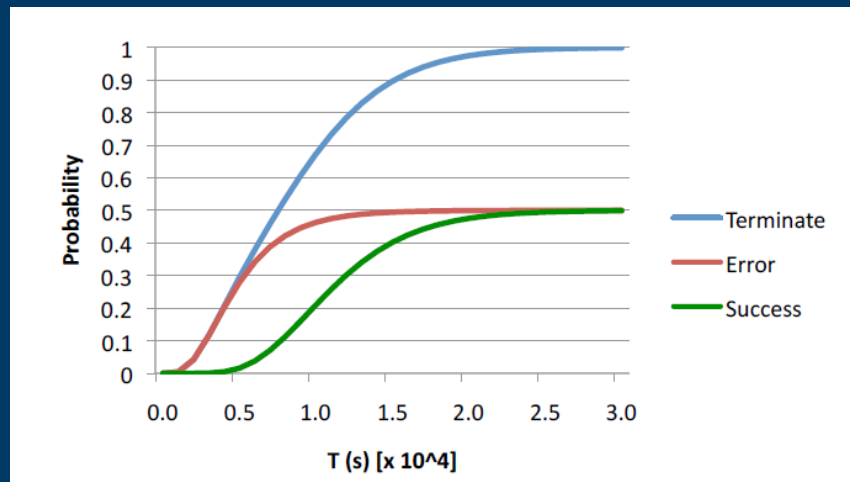
$\frac{t}{t^m} \frac{x}{x^m} \frac{t}{t^m} \frac{a}{a^m} \frac{t}{t^m} \frac{a}{a^m} (1)$

$\frac{x}{x^m} \frac{t}{t^m} \frac{y}{y^m} \frac{t}{t^m} \frac{a}{a^m} \frac{t}{t^m} (1)$



Modelchecking

- Export to PRISM probabilistic modelchecker



JOURNAL
OF
THE ROYAL
SOCIETY

Interface

Design and analysis of DNA strand displacement devices using probabilistic model checking

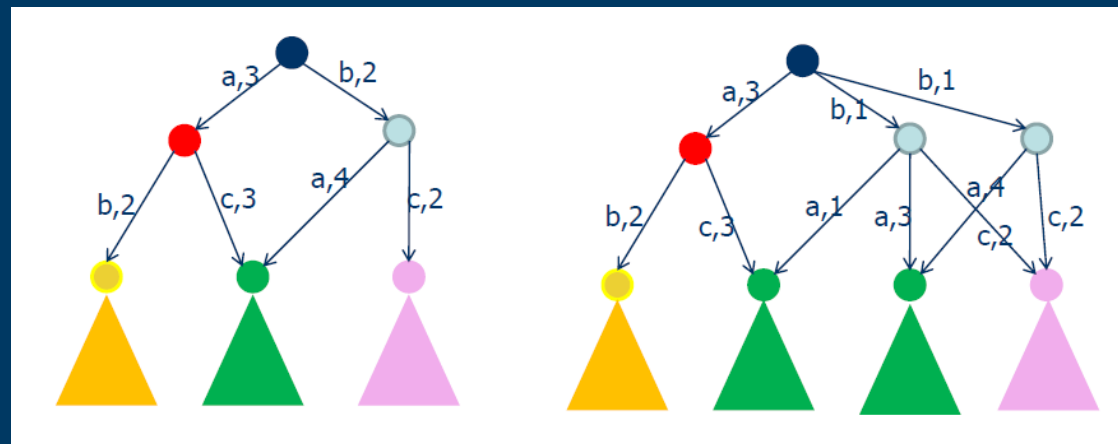
Matthew R. Lakin^{1,3,†}, David Parker^{2,†}, Luca Cardelli¹,
Marta Kwiatkowska² and Andrew Phillips^{1,*}

Verification

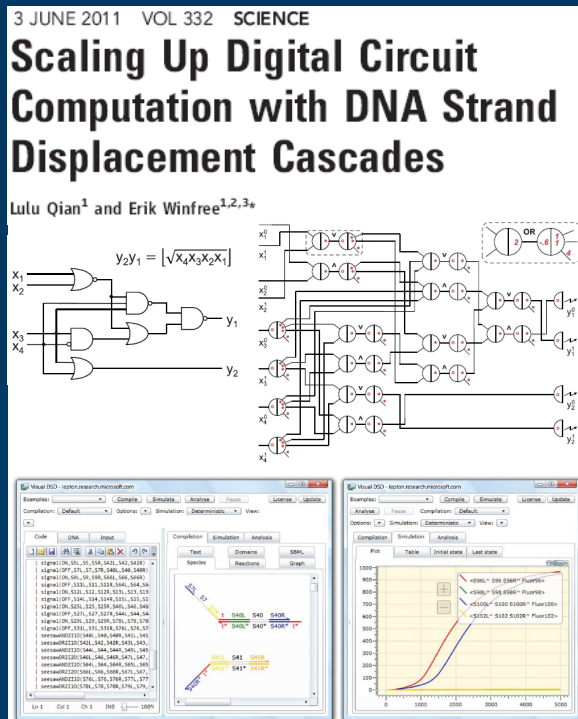
- Quantitative theories of system equivalence and approximation.

CONTINUOUS MARKOVIAN LOGICS
AXIOMATIZATION AND QUANTIFIED METATHEORY

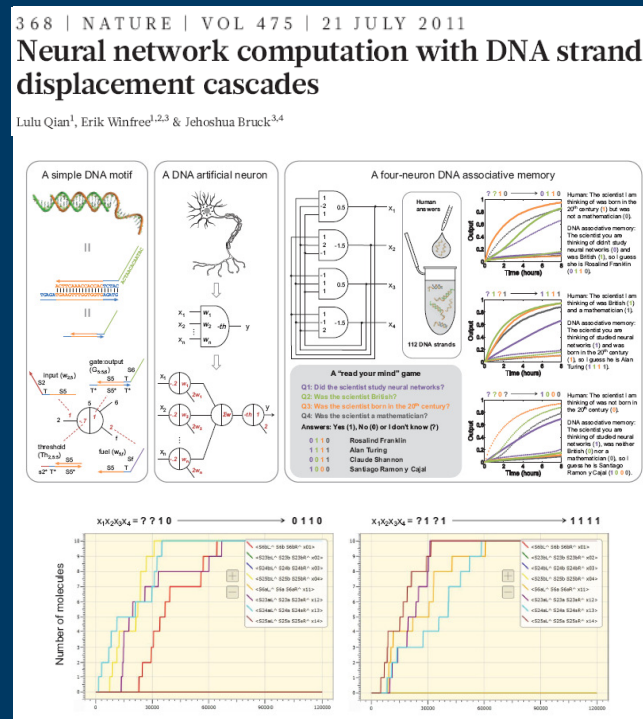
RADU MARDARE, LUCA CARDELLI, AND KIM G. LARSEN



Related Work Supporter by our Tools



Square root of a 4-bit number



Associative memory

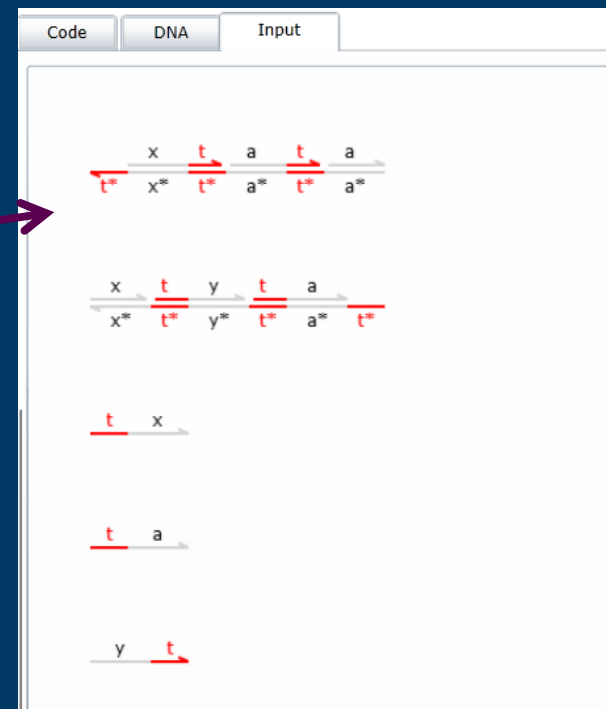
Execution

A wetlab pipeline for Molecular Programming

Output of Design Process

- Domain structures
 - (DNA sequences to be determined)

"Ok, how do I run this for real"

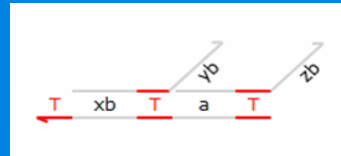


From Structures to Sequences



www.nupack.org

DSD Structure



"Dot-Paren" representation

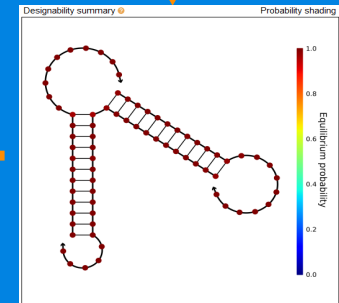
Nucleic acid type: RNA DNA Temperature: °C Number of designs:

Target structure:

Output Sequences

Ensemble defect (nt)	Normalized ensemble defect (%)	GC content (%)	Sequence	
0.2	0.3	57.5	<pre>gccccgaatcccccauuuagAAC AA+gccccgaatccccccucuu UUUC+gccccgaatccccgcg GUUUGcagcccccgc</pre>	<input type="button" value="To Utilities"/> <input type="button" value="To Analysis"/>

Thermodynamic Synthesis



"Ok, where do I buy these?"



"DNA Synthesis"

dna synthesis ×

About 8,610,000 results (0.24 seconds) [Advanced search](#)

▶ **Custom DNA Synthesis** Ads
www.Biomatik.com High Quality Custom Gene **Synthesis**, Best Price Guaranteed! Get A Quote.

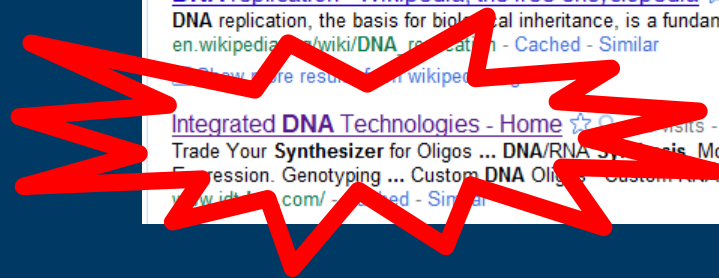
[Order Gene at GenScript](#)
www.GenScript.com \$0.29/bp. Any Gene in ANY Vector Proven increase protein expression

[Gene Synthesis \\$0.35/bp](#)
www.epochlifescience.com Dependable Service @ Low Price: Come on Down and Save Your Budgets!

[DNA synthesis - Wikipedia, the free encyclopedia](#) ☆ 🔍
DNA **synthesis** commonly refers to: DNA replication - DNA biosynthesis (in vivo DNA amplification); Polymerase chain reaction - enzymatic **DNA synthesis** (in ...
en.wikipedia.org/wiki/DNA_synthesis - Cached - Similar

[DNA replication - Wikipedia, the free encyclopedia](#) ☆ 🔍
DNA replication, the basis for biological inheritance, is a fundamental ...
en.wikipedia.org/wiki/DNA_replication - Cached - Similar

[Integrated DNA Technologies - Home](#) ☆ 🔍 Visits - May 24
Trade Your **Synthesizer** for Oligos ... **DNA/RNA Synthesis**, Modifications, Purifications, Gene Expression, Genotyping ... Custom **DNA Oligos** ... Custom **RNA Oligos** ...
www.idt.com/ - Cached - Similar



From Sequences to Molecules

- Copy&Paste from nupack



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

Scale: 5 nmole DNA oligo Purification: Standard

Sequence Name: 5'-ACT GCA CCA TAA GCA ACT TTT

ADD TO ORDER
ADD TO WISH LIST

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

Customized Labels (more detail)
 Stock IDT Label FREE

Molecules by FedEx



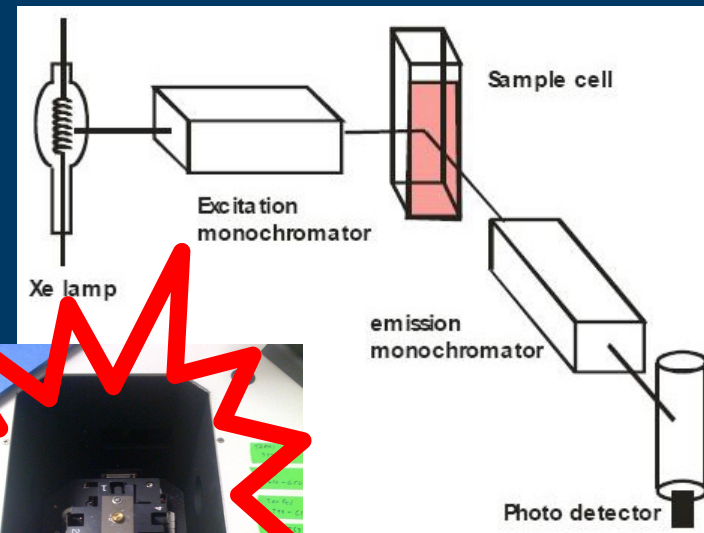
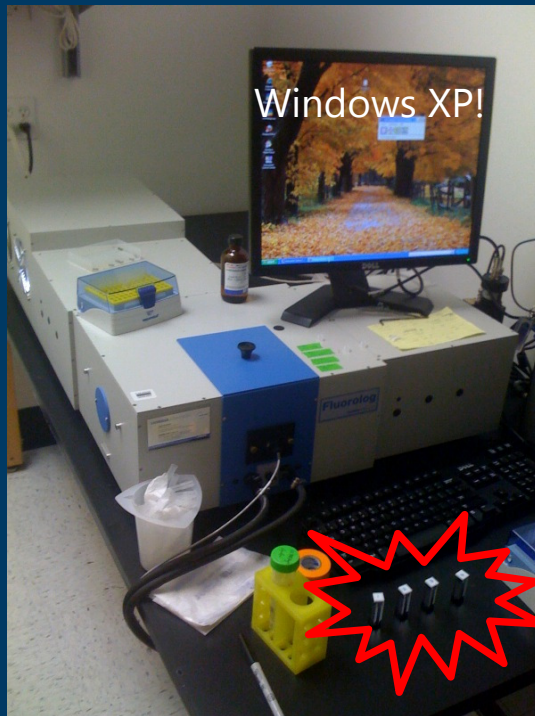
"Ok, how do I run these?"

Add Water

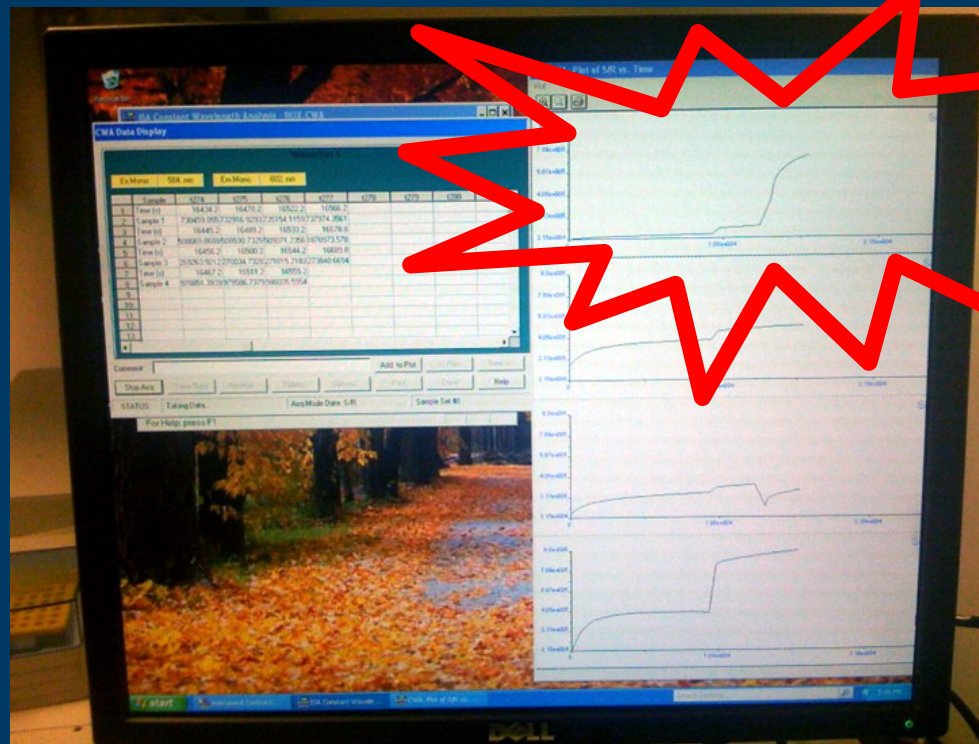


Execute (finally!)

- Fluorescence is your one-bit 'print' statement



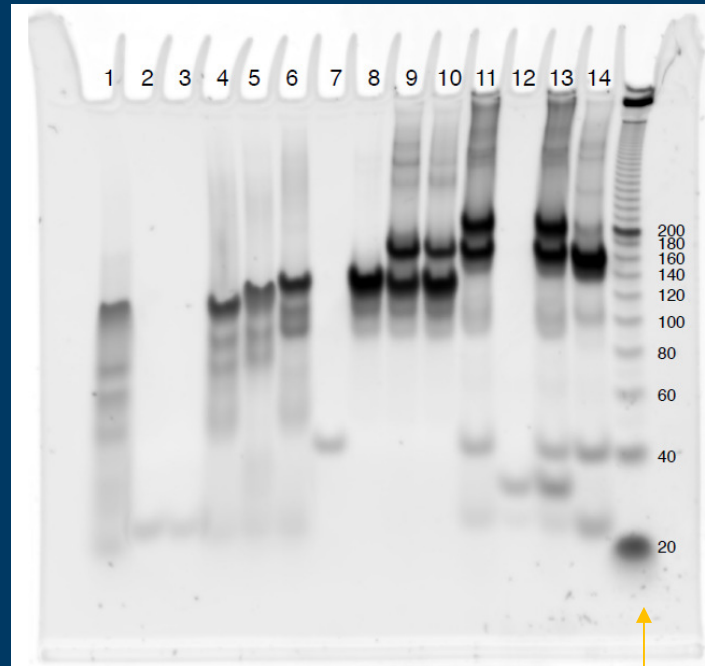
Output



Debugging

- A core dump

DNA
strand
length



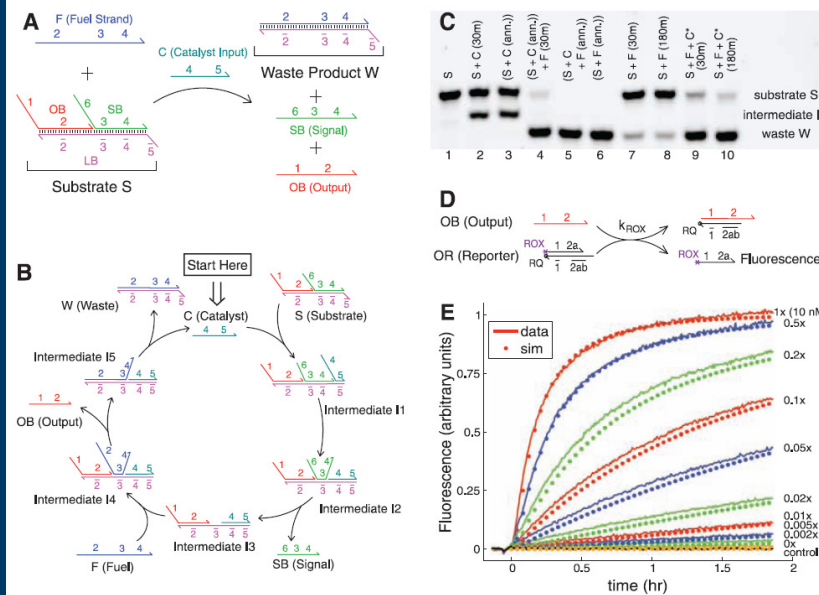
← Various processing stages →

Calibration
scale

Delivery!

Engineering Entropy-Driven Reactions and Networks Catalyzed by DNA

David Yu Zhang, *et al.*
Science **318**, 1121 (2007);
 DOI: 10.1126/science.1148532



A Molecular Algorithm

Running something interesting with DNA

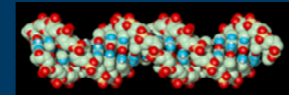
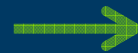
Approximate Majority Algorithm

- 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]

- 3 rules of agent (or molecule) interaction



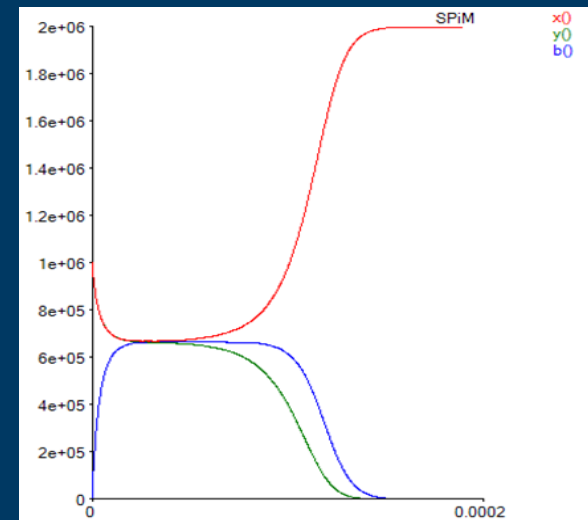
"our program"



Surprisingly good (in fact, optimal)

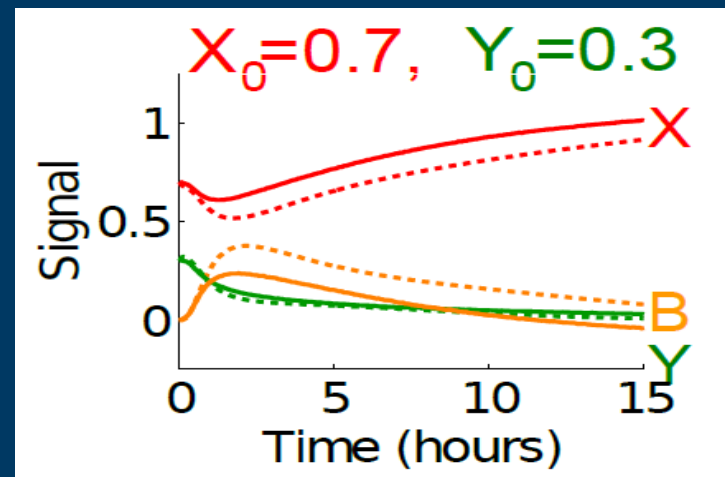
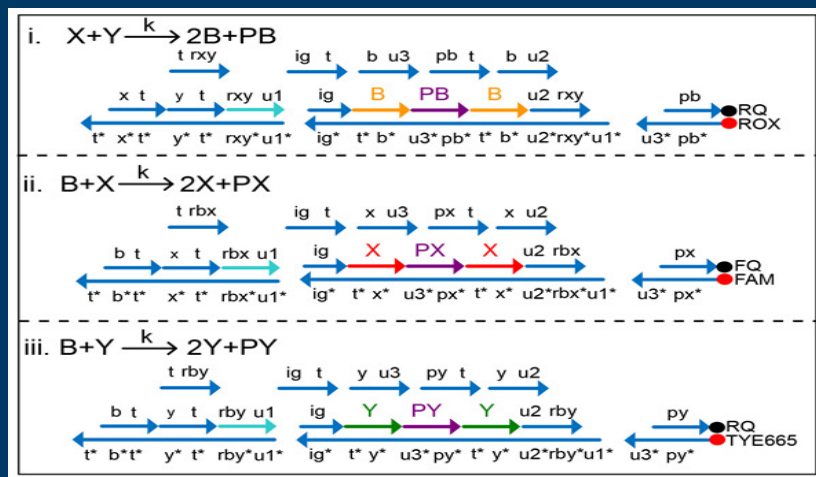
- Fast: reaches agreement in $O(\log n)$ time w.h.p.
 - $O(n \log n)$ communications/collisions
 - Even when initially $\#X = \#Y!$ (stochastic symmetry breaking)
- Robust: true majority wins w.h.p.
 - If initial majority exceeds minority by $\omega(\sqrt{n} \log n)$
 - Hence the agreement state is stable

Stochastic simulation of worst-case scenario with initially $\#X = \#Y$



DNA Implementation, at U.W.

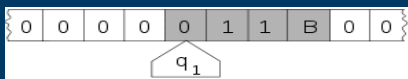
- Programmable chemical controllers made from DNA
 [Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik and Georg Seelig]



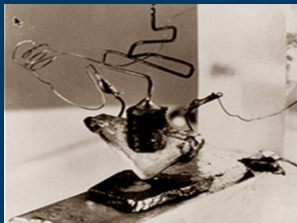
Final Remarks

A Brief History of DNA

Turing Machine, 1936



Transistor, 1947



~~Digital Computers~~

Computer programming

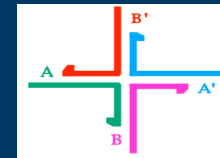
20th century

DNA, -3,800,000,000

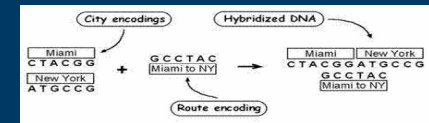


Systematic manipulation of information

Structural DNA Nanotech, 1982



DNA Algorithm, 1994



~~DNA Computers~~

Molecular programming

21th century

Acknowledgments

- Microsoft Research
 - Andrew Phillips, Biological Computation Group
- Caltech
 - Winfree Lab
- U.Washington
 - Seelig Lab

Questions?

Resources

- Visual DSD at MSR

<http://research.microsoft.com/en-us/projects/dna/>

- 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/>