# Speaking the Language of Molecules

**Luca Cardelli**
Microsoft Research

**Modelling Complex Biological Systems in the Context of Genomics**
**Sophia Antipolis 2010-05-26**
**http://lucacardelli.name**

# Outline

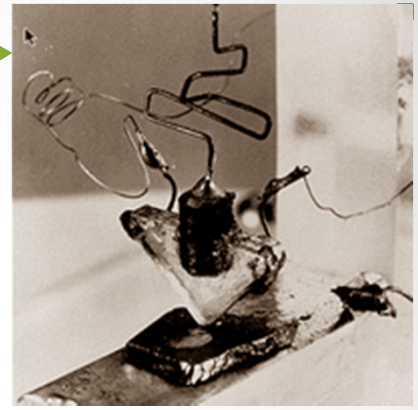
- Molecular Structures
  - Getting smaller
  - Self-assembly
- Molecular Languages
  - Natural languages: proteins, genes, membranes
  - Modeling languages (systems biology)
  - Executable languages (nano-engineering)
- Molecular Compilation
  - Intermediate Languages
  - Analysis Tools and Techniques
  - Nano-programming workflow

# Molecular Structures

# Smaller and Smaller

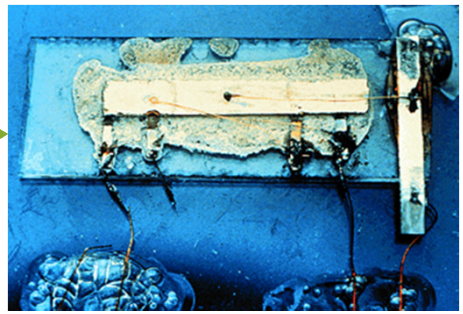
First working transistor

John Bardeen and Walter Brattain , Dec. 23, 1947.



First integrated circuit

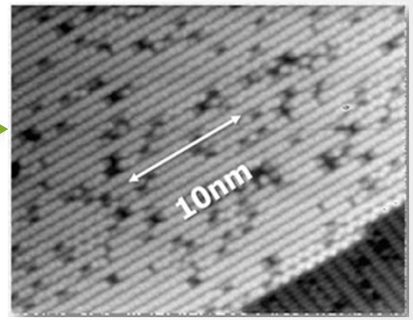
Jack Kilby, Sep. 1958.



50 years later

25nm NAND flash

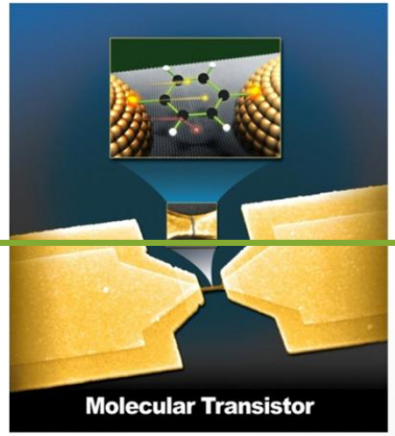
Intel&Micron, Jan. 2010. ~50atoms.



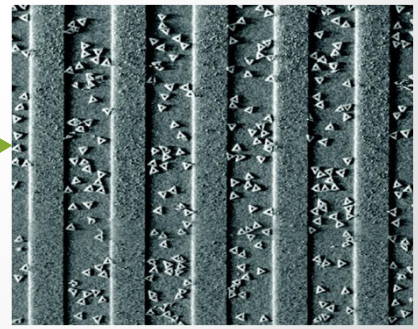

Scanning tunneling microscope image of a silicon surface showing 10nm is ~20 atoms across

Single molecule transistor

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




Molecules on a chip

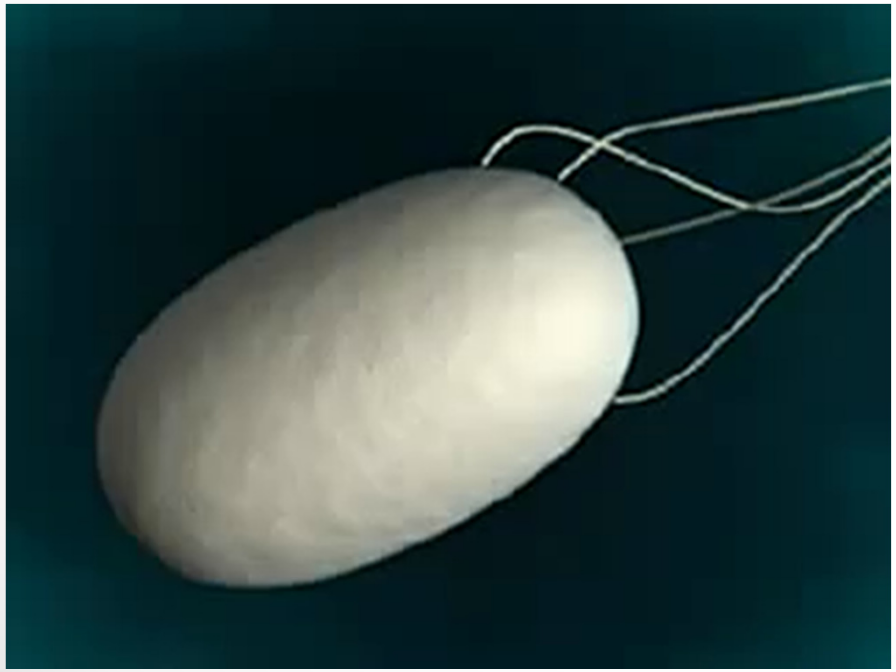


~10 Moore's Law cycles left!

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

# Building The Small*est* Things

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

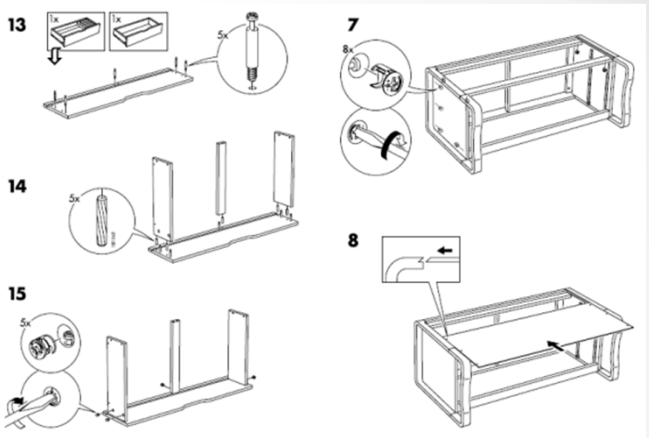


GS Garland Science Taylor & Francis Group

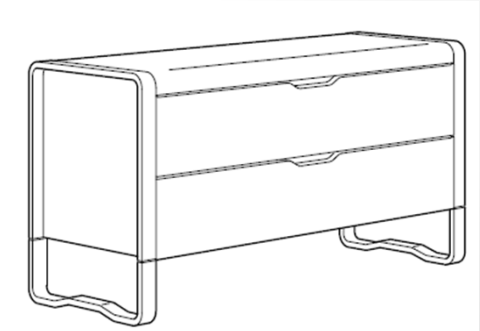
www.youtube.com/watch?v=Ey7Emmddf7Y

# 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

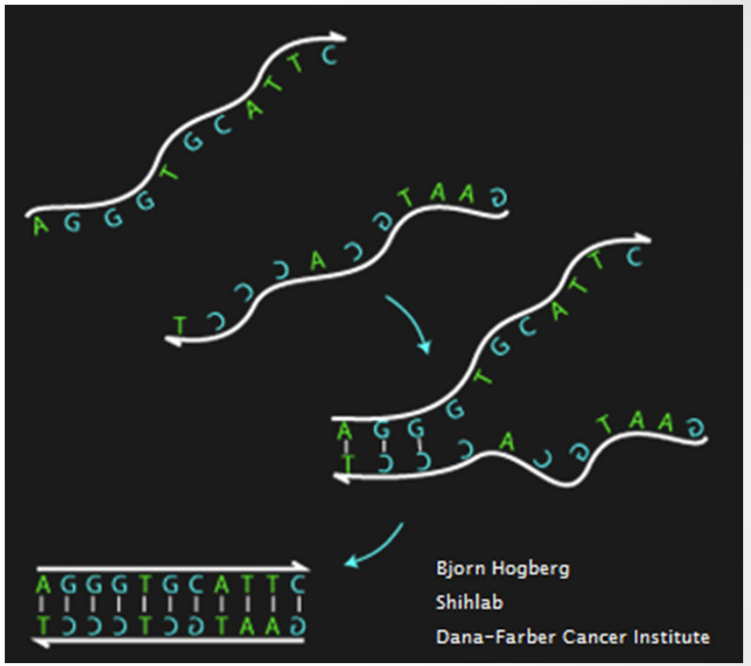


http://www.ikea.com/ms/en_US/customer_service/assembly_instructions.html

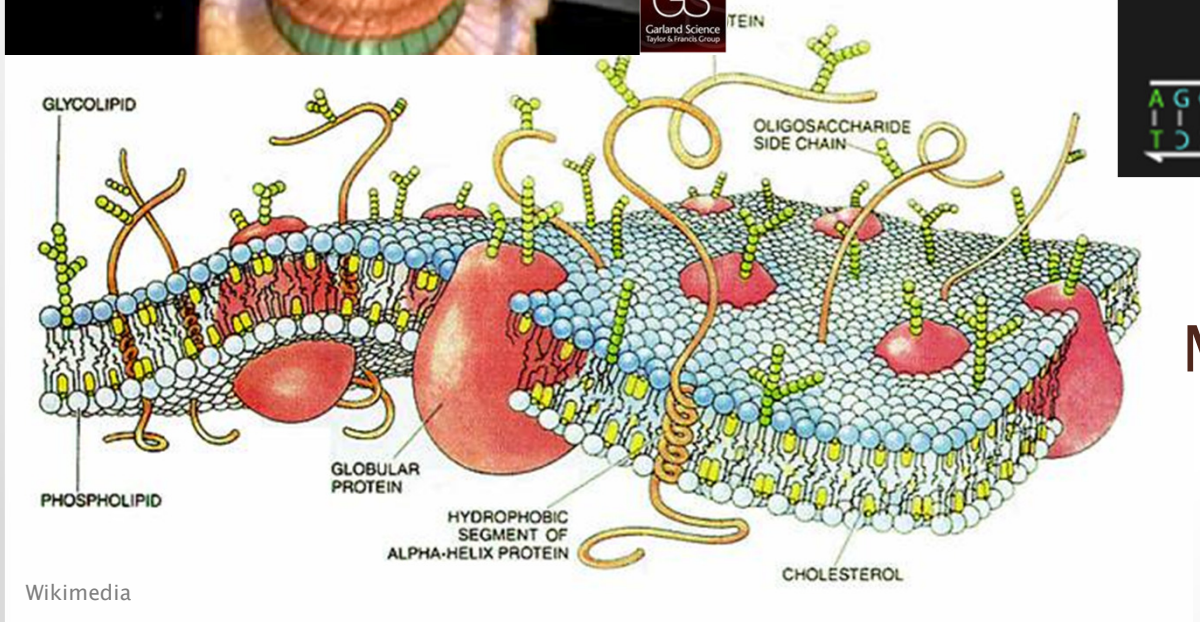
# Programmed Self-Assembly

## Proteins


## DNA/RNA




## Membranes

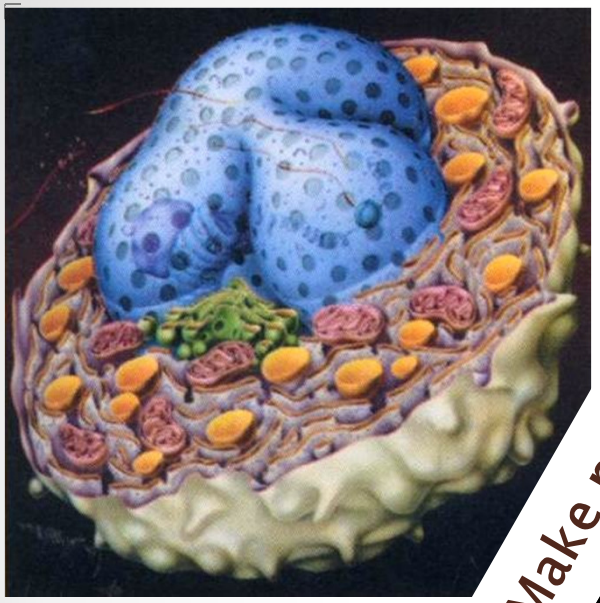



Wikimedia

# Molecular Languages

## - natural languages -

# Abstract Machines of Biochemistry



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

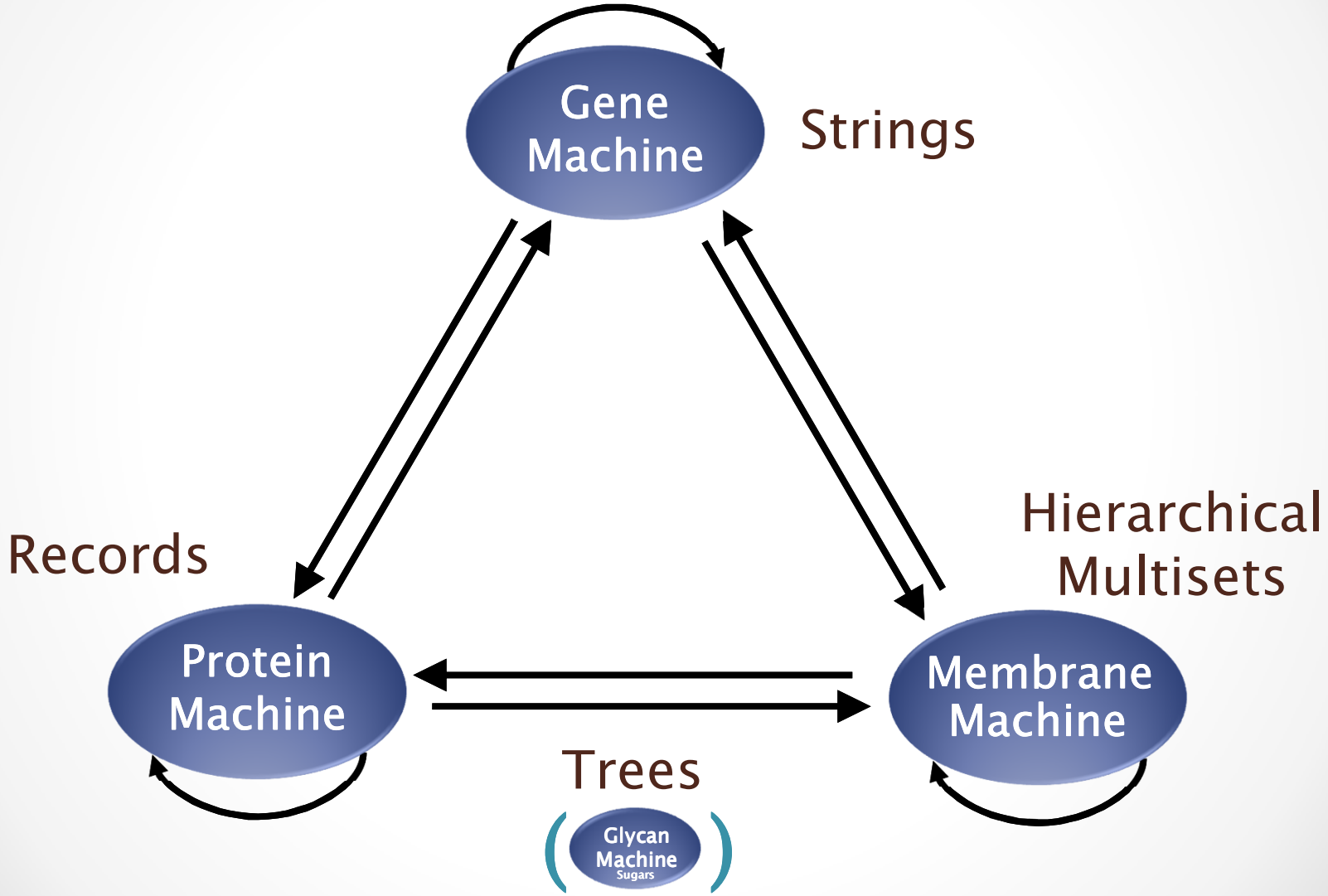
Metabolism, Propulsion
Signaling, Transport

Glycan Machine
Sugars

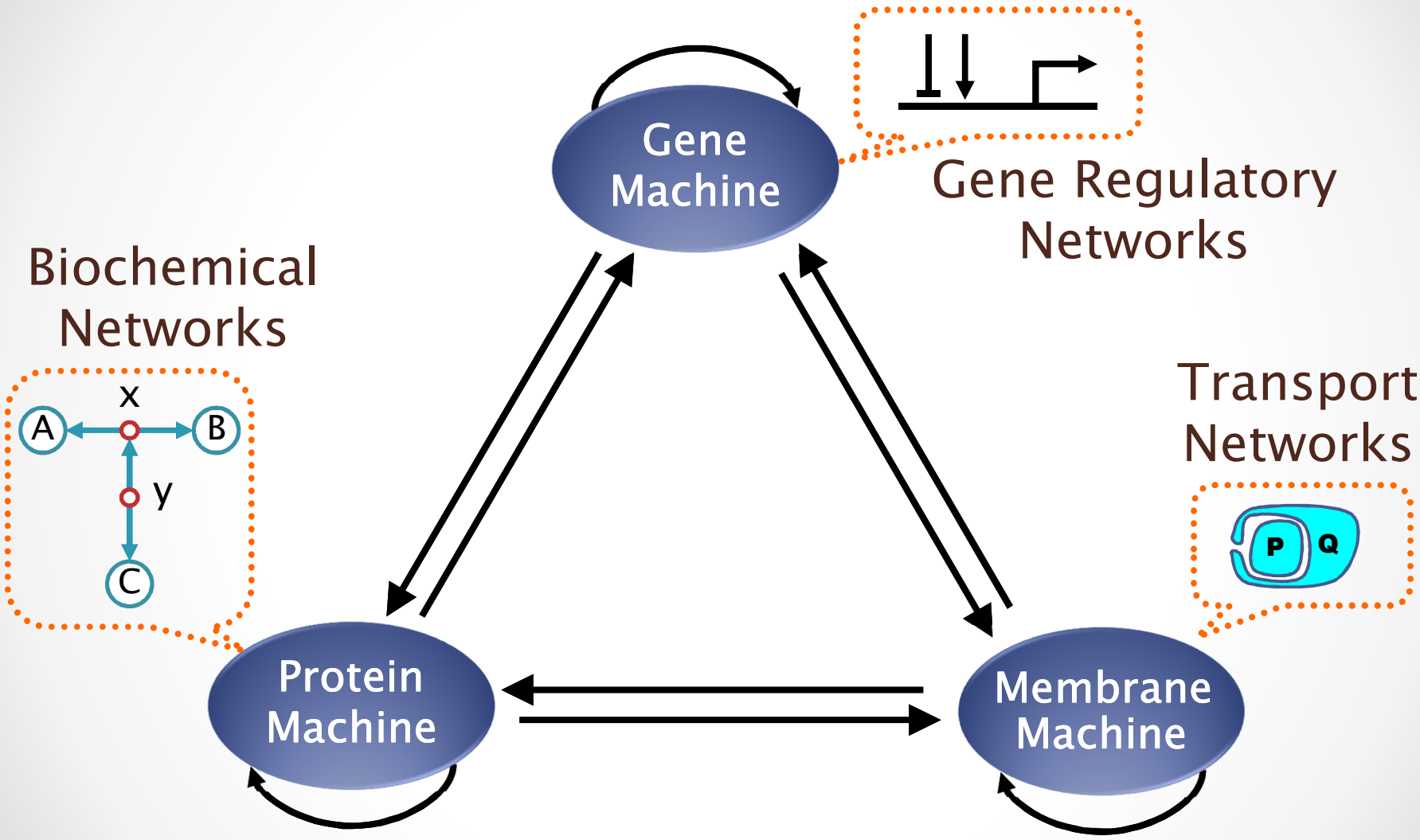
Surface and Extracellular Features

Confinement, Storage
Bulk Transport

# Data Structures of Biochemistry



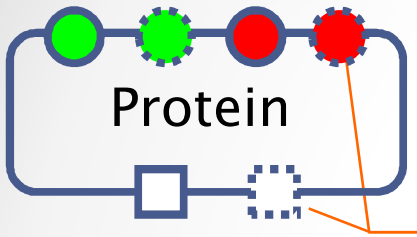
# Languages of Biochemistry



# The Protein Machine

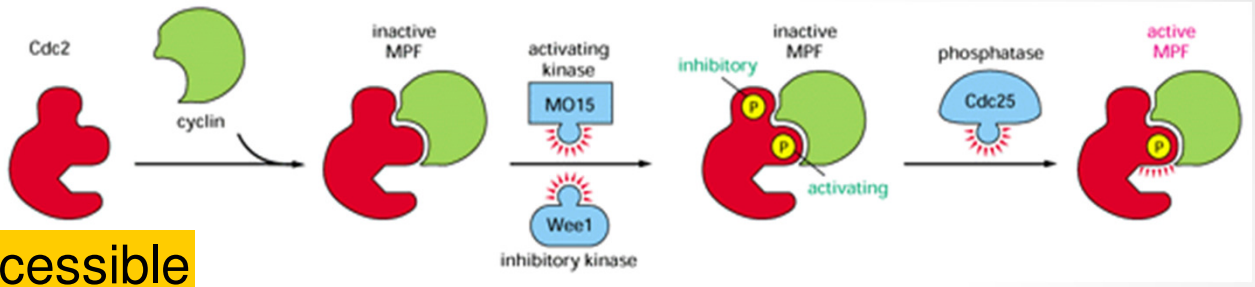
*cf.* BioCalculus [Kitano&Nagasaki], κ-calculus [Danos&Laneve]

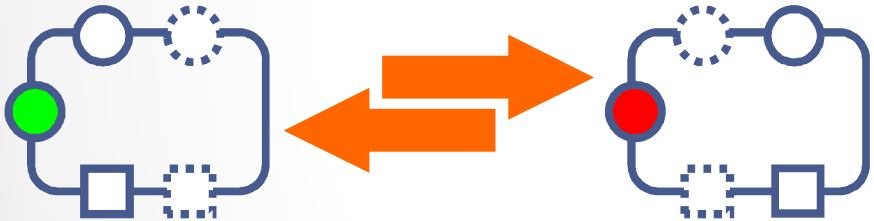
## On/Off switches




Inaccessible

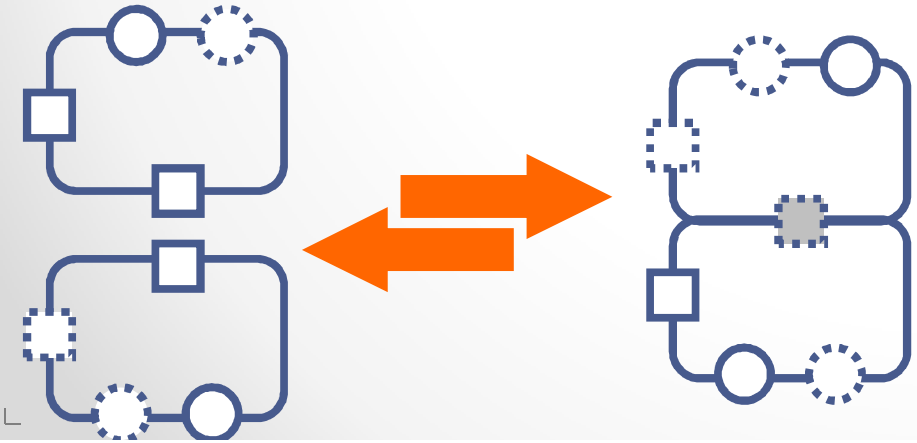
Binding Sites



## Switching accessible switches

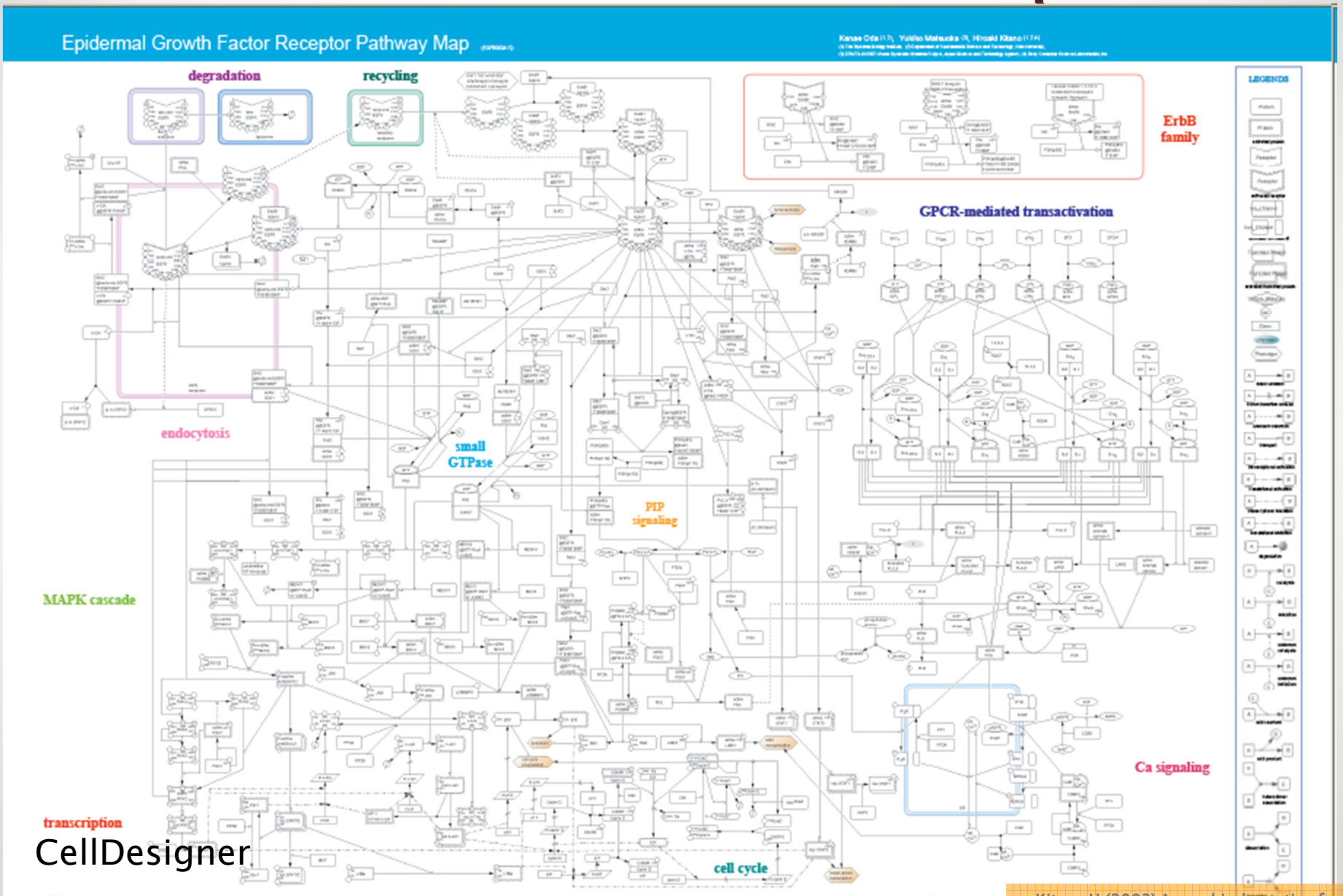
– May cause other switches and binding sites to become (in)accessible.



## Binding accessible sites

– May cause other switches and binding sites to become (in)accessible.

# Molecular Interaction Maps (Kohn/Kitano)

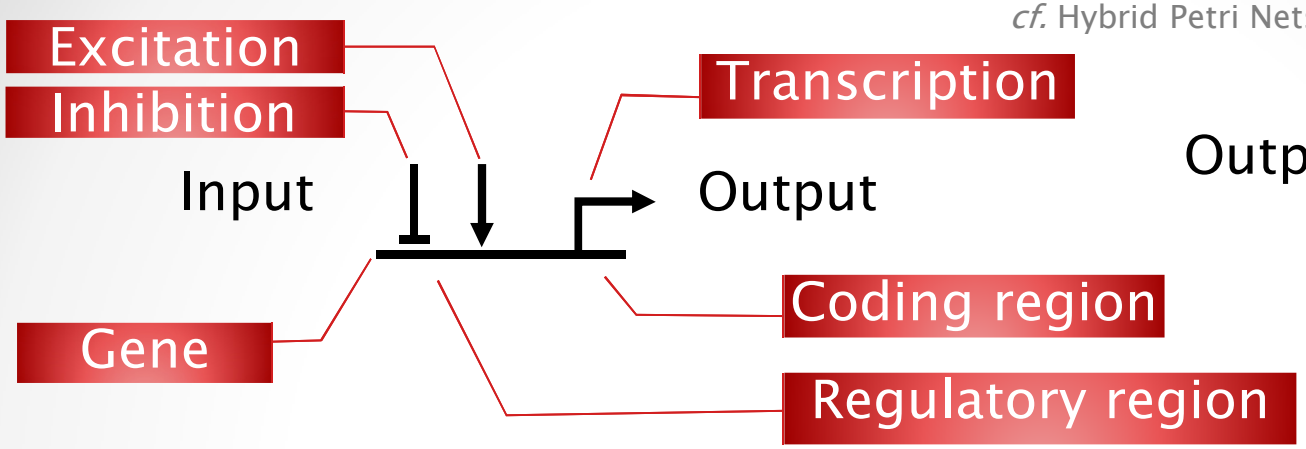


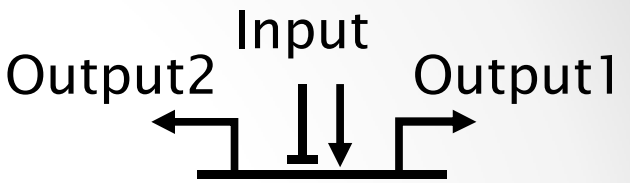

CellDesigner

Kitano H (2003) A graphical notation for biochemical networks. *BioSilico* 1: 169–176

# The Gene Machine

*cf.* Hybrid Petri Nets [Matsuno, Doi, Nagasaki, Miyano]



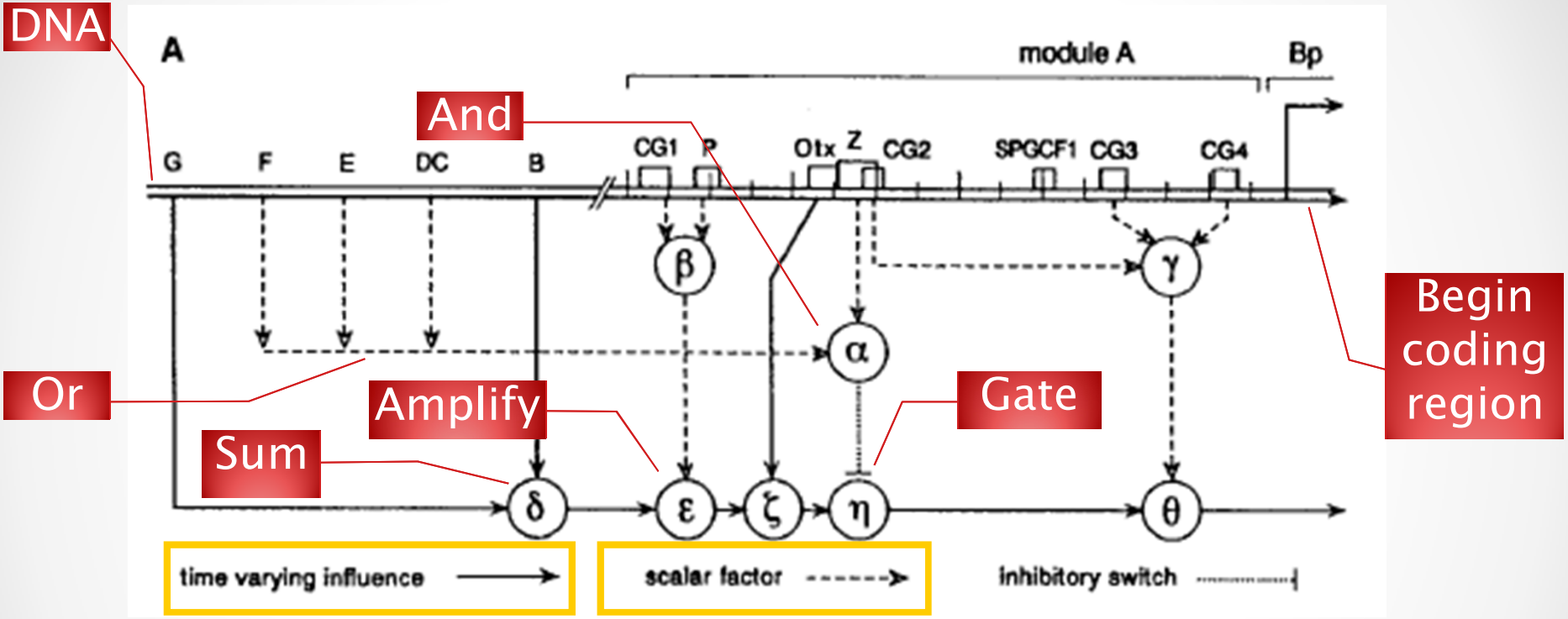

Regulation of a gene influences transcription. The regulatory region has precise DNA sequences meant for binding regulators.

Transcription produces molecules (RNA or, through RNA, proteins) that bind to regulatory region of other genes (or that are end-products).

Human (and mammalian) Genome Size
3Gbp (Giga base pairs) 750MB @ 4bp/Byte (CD)
Non-repetitive: 1Gbp 250MB
In genes: 320Mbp 80MB
Coding: 160Mbp 40MB
Protein-coding genes: 30,000-40,000

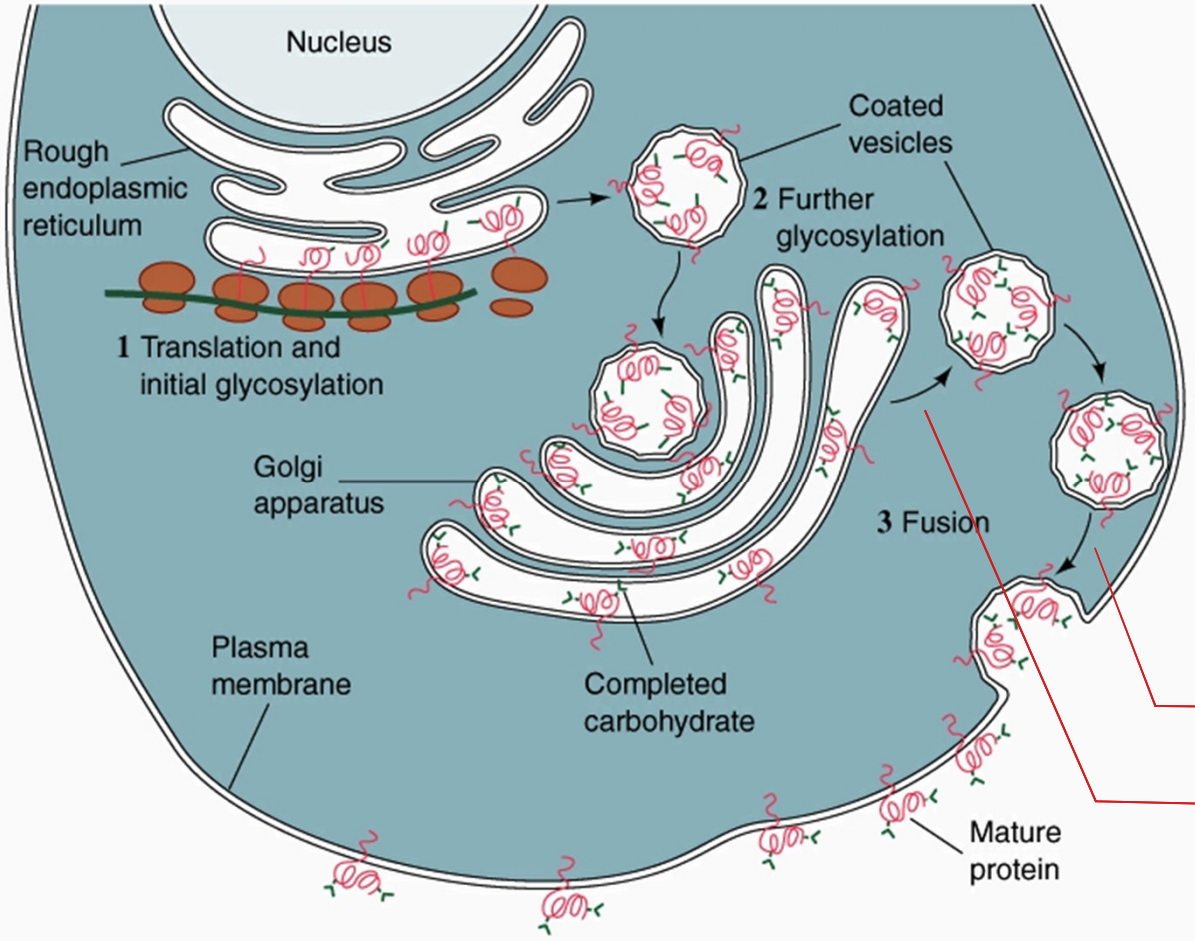
M.Genitalium (smallest true organism)
580,073bp 145KB (eBook)
E.Coli (bacteria): 4Mbp 1MB (floppy)
Yeast (eukarya): 12Mbp 3MB (MP3 song)
Wheat 17Gbp 4.25GB (DVD)

# Function of a Regulatory Region



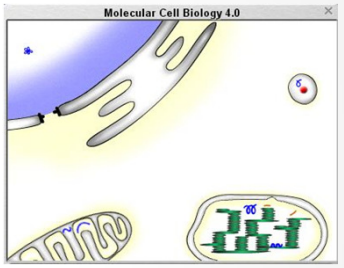

C-H.Yuh, H.Bolouri, E.H.Davidson. Genomic Cis-Regulatory Logic: Experimental and Computational Analysis of a Sea Urchin Gene. Science 279:1896-1902, 1998

# The Membrane Machine






Molecular transport and transformation through dynamic compartment fusion and fission.



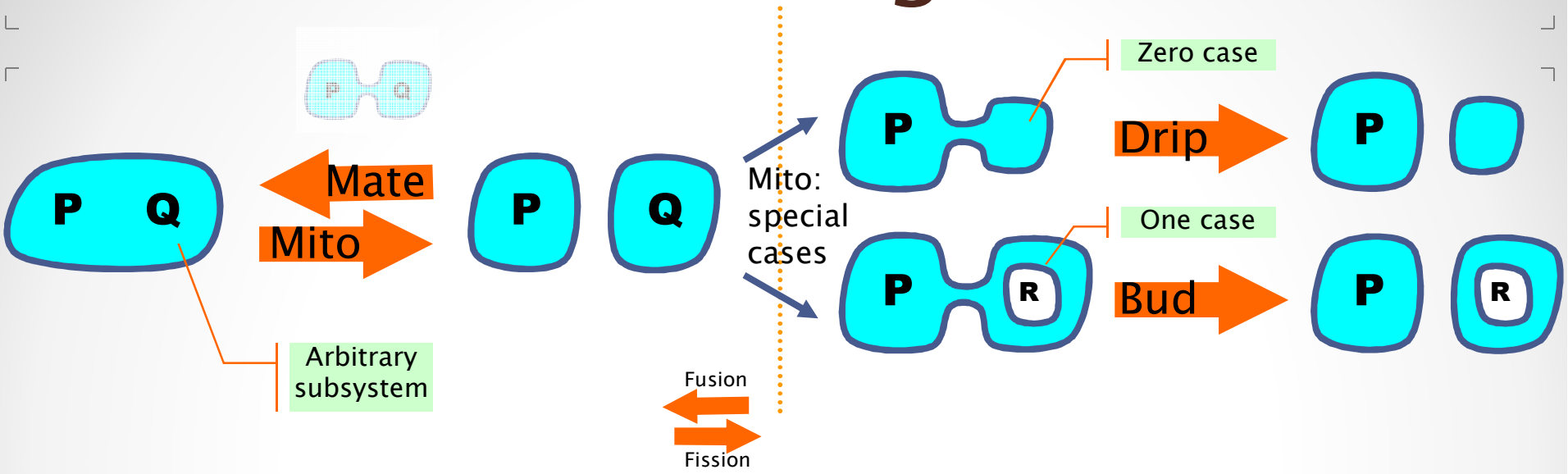

Taken from MCB CD

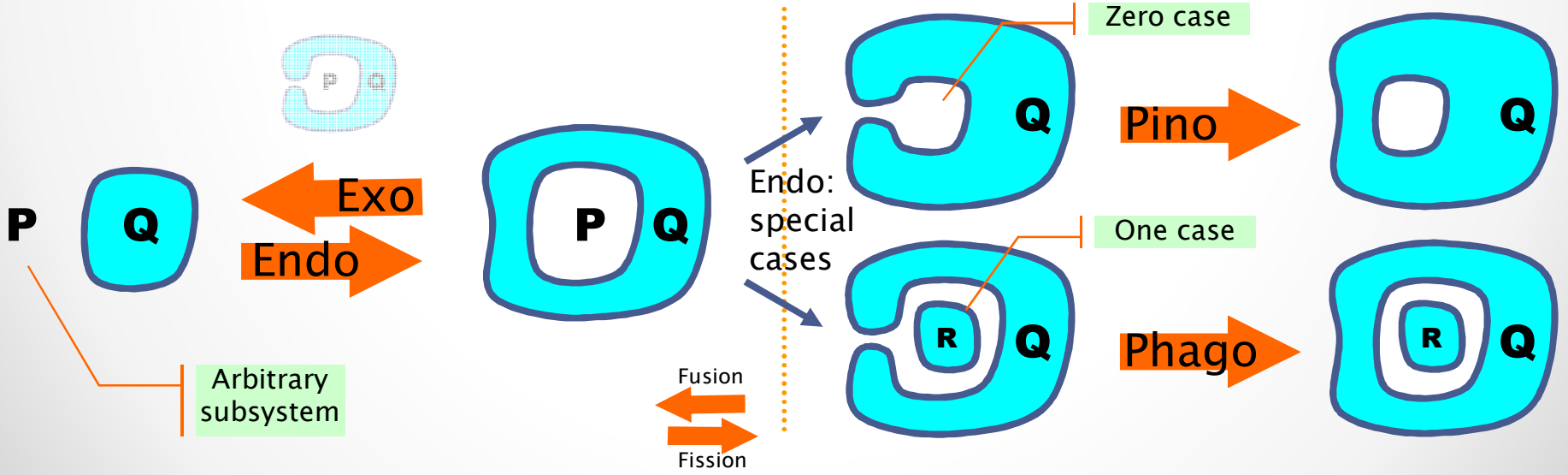
Fusion

Fission

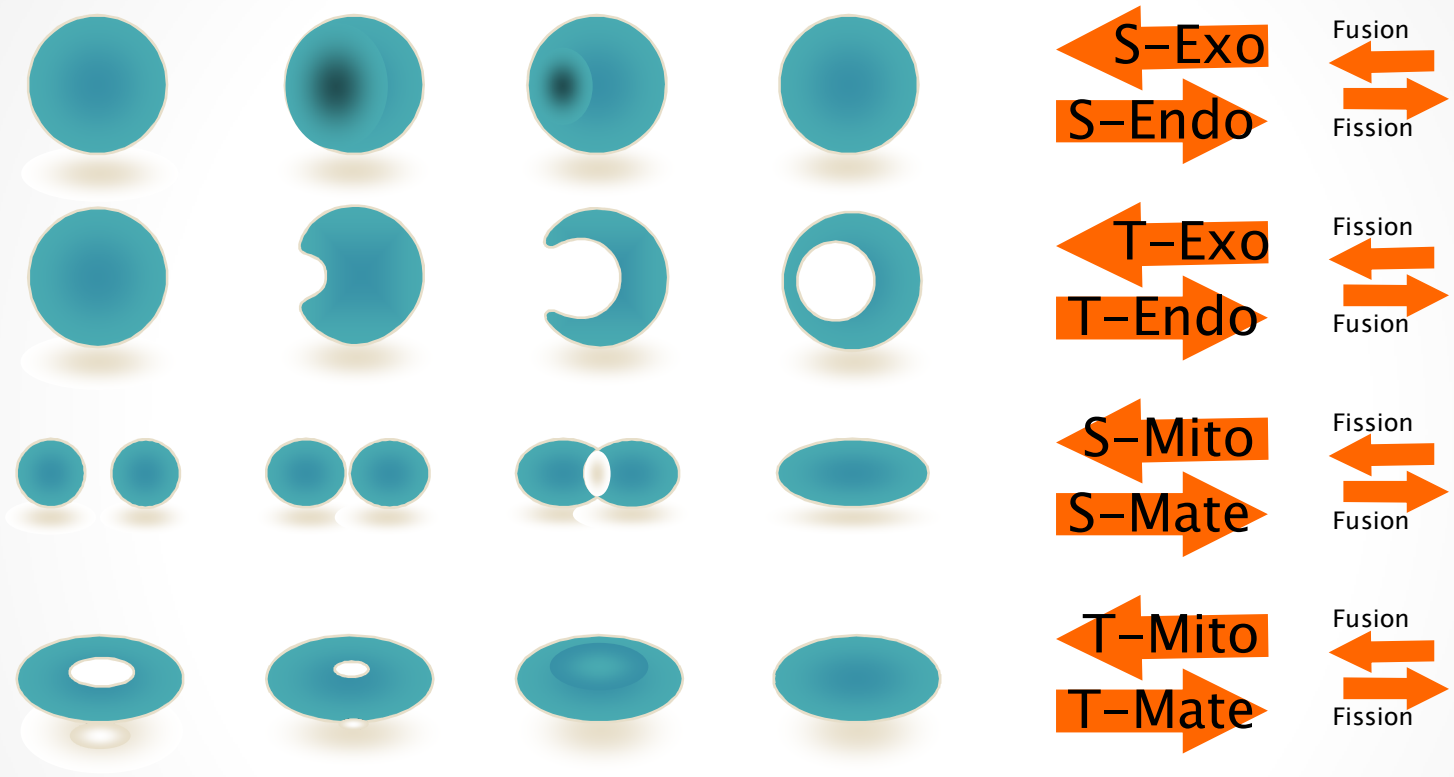
Voet, Voet & Pratt
Fundamentals of Biochemistry
Wiley 1999. Ch10 Fig 10-22.

# Bitonal Diagrams



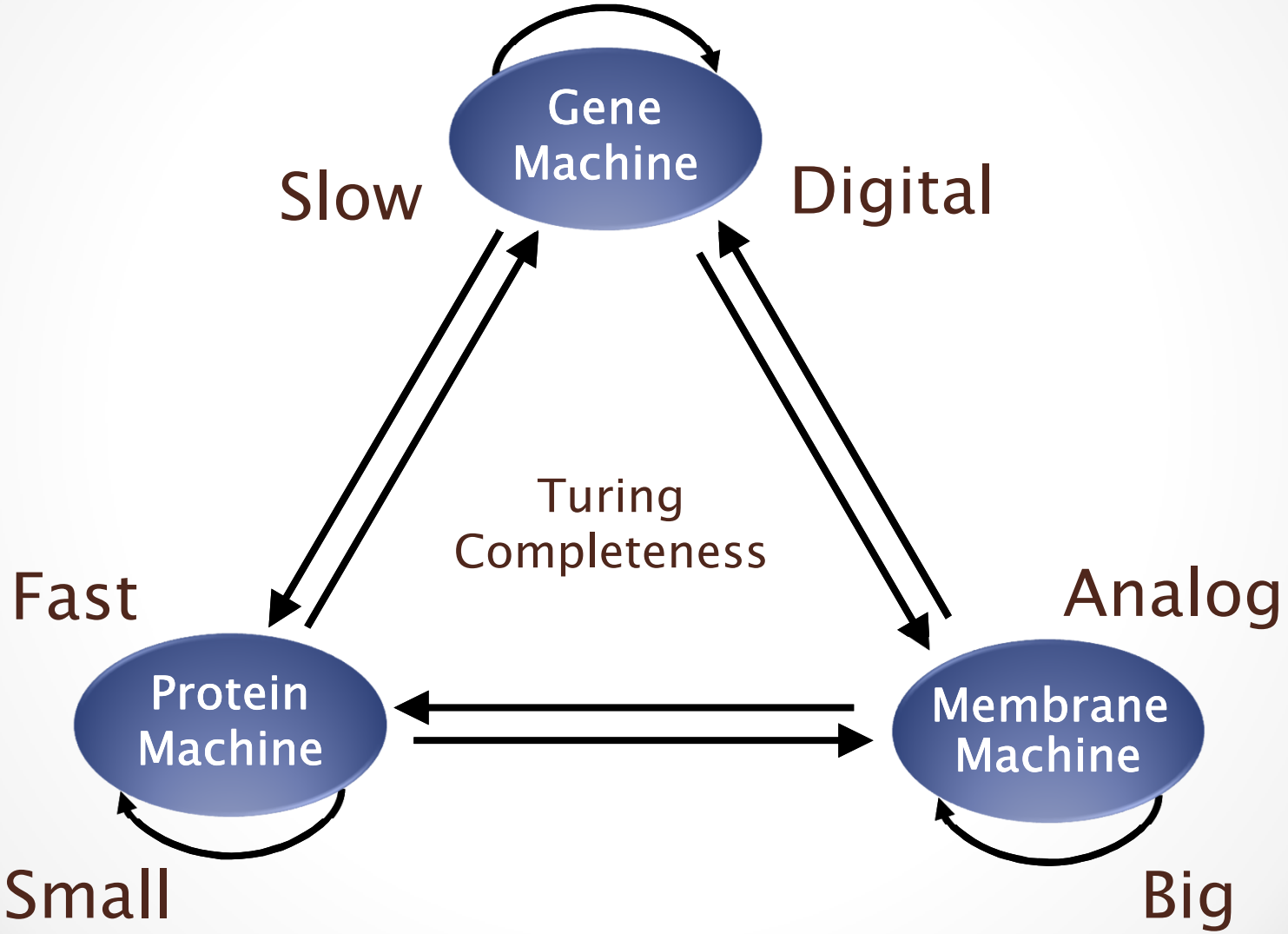



# … in 3D




Controlled by surface proteins

# Integration



# Molecular Languages

## - modeling languages -

# From Instructions to Programs

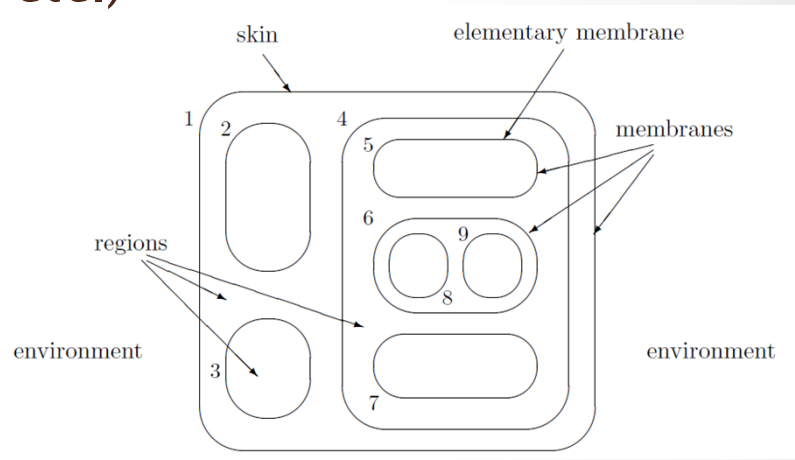
- We have seen the instruction sets:
  - Proteins – complexation, phosphorilation
  - Genes – activation, inhibition
  - Membranes – fusion, fission
- How do we combine them into programs?
  - I.e., into models (quantitative programs)
- How do we study their semantics?
  - I.e., their kinetics (quantitative semantics)

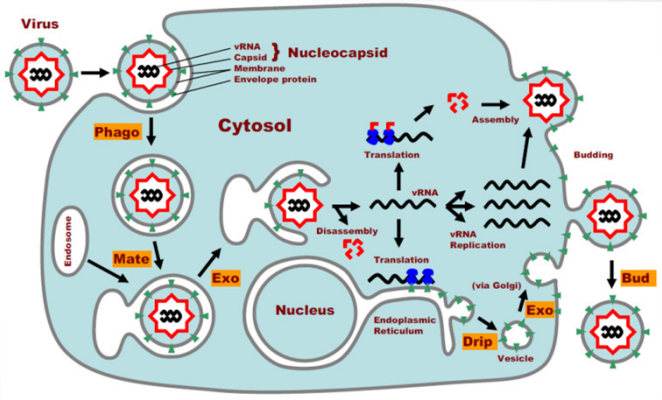
# Chemistry

- Chemical reactions
  - $A + B \rightarrow_r C + D$ (a program)
- Ordinary Differential Equations
  - $d[A]/dt = -r[A][B]$ … (a semantics)
- Rich analytical techniques based on Calculus
- But prone to combinatorial explosion
  - Due to the peculiarities of protein interactions

# High(er)-Level Languages

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



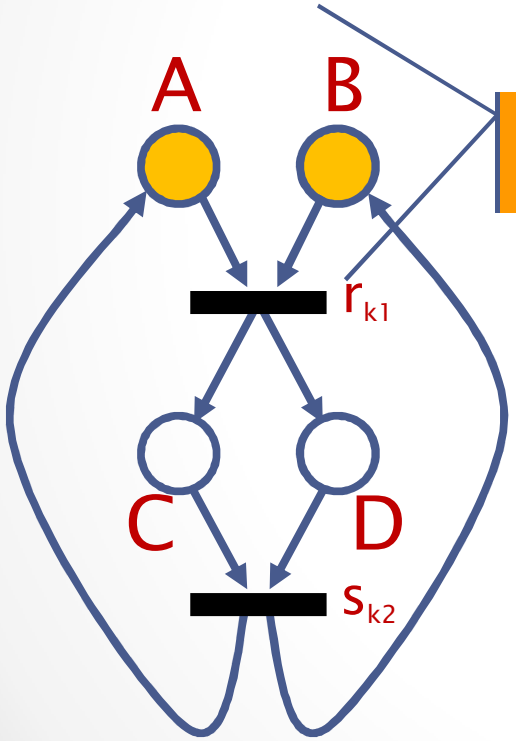
# Reactions vs. Reagents

Says what "A" *does*.

$r: A + B \rightarrow_{k1} C + D$

$s: C + D \rightarrow_{k2} A + B$

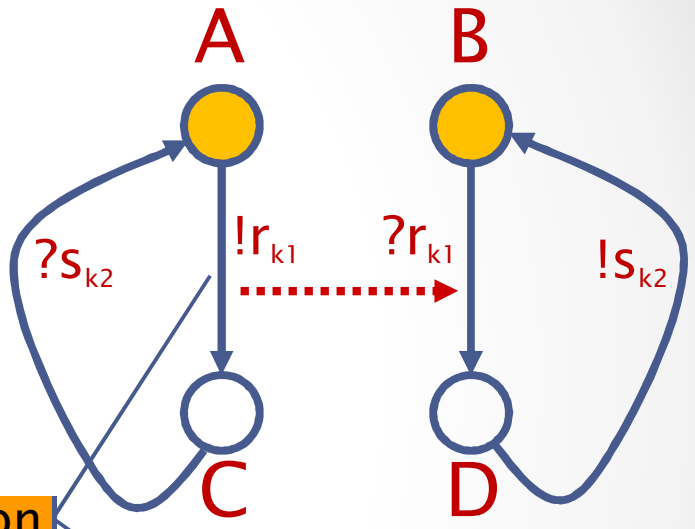
Does A become C or D?




Reaction oriented

1 line per reaction

Says what "A" *is*.




Interaction oriented

1 line per agent

$A = !r_{k1}; C$

$C = ?s_{k2}; A$

$B = ?r_{k1}; D$

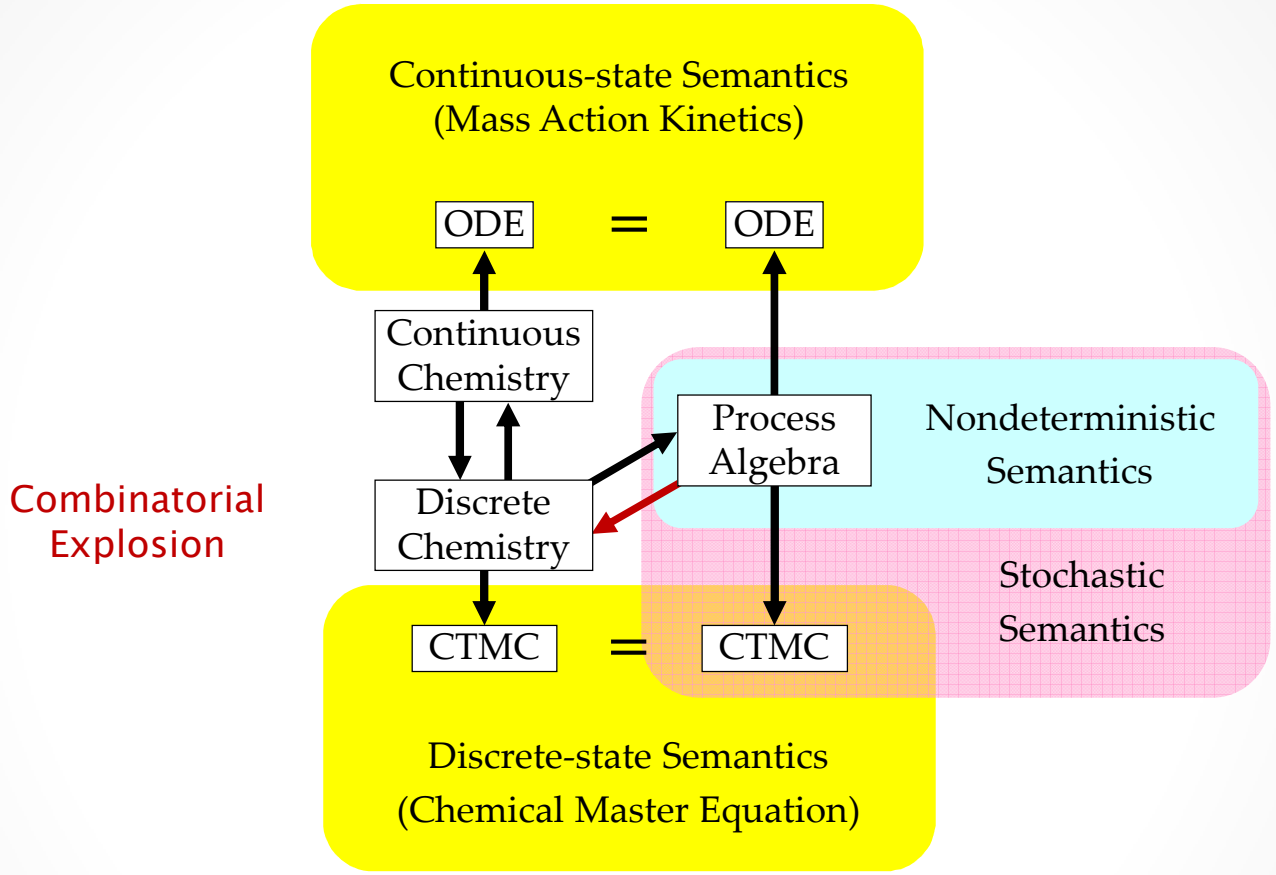
$D = !s_{k2}; B$

A becomes C not D!

The same "math model"

CTMC

# Formal Connections




These diagrams commute via appropriate maps.

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

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

# Execution?

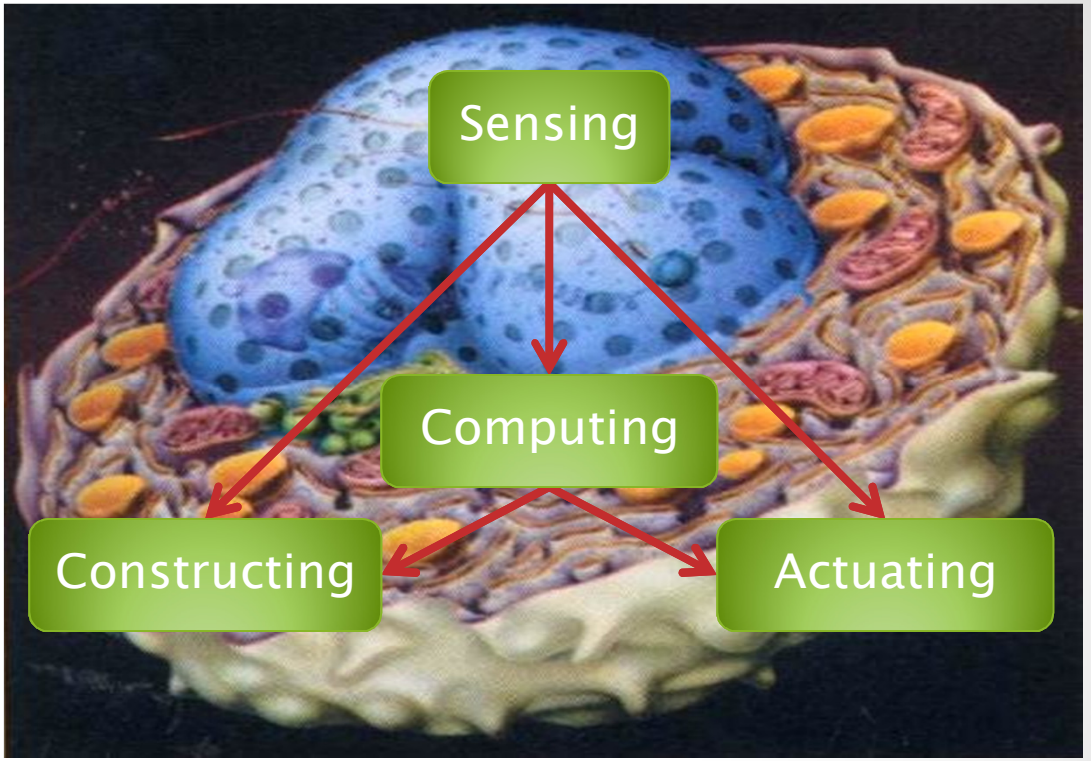
- Chemistry is not easily executable
  - Please Mr Chemist, execute me these reactions that I just made up
- Similarly, the molecular languages seen so fare are descriptive (modeling) languages
- How can we actually execute molecular languages? With real molecules?

# Molecular Languages

## - executable languages -

# Nanoscale Control Systems

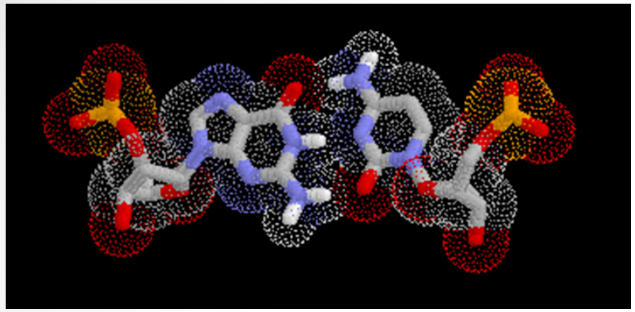
- Sensing
  - Reacting to forces
  - Binding to molecules
- Actuating
  - Releasing molecules
  - Producing forces
- Constructing
  - Chassis
  - Growth
- Computing
  - Signal Processing
  - Decision Making



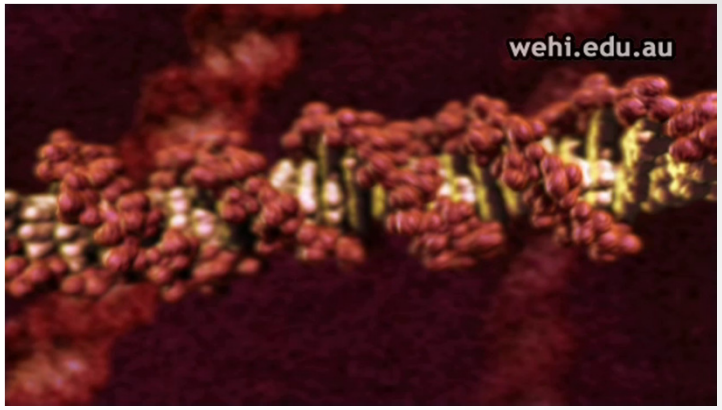

Nucleic Acids can do all this.

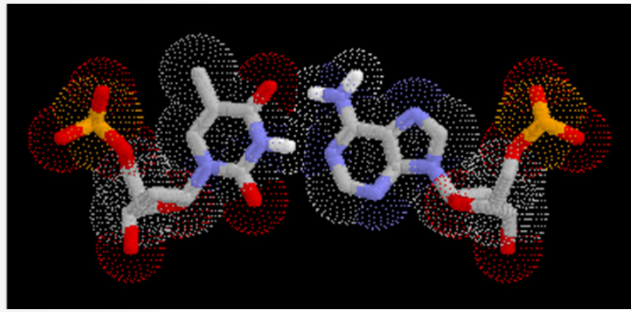
And interface to biology.

# DNA



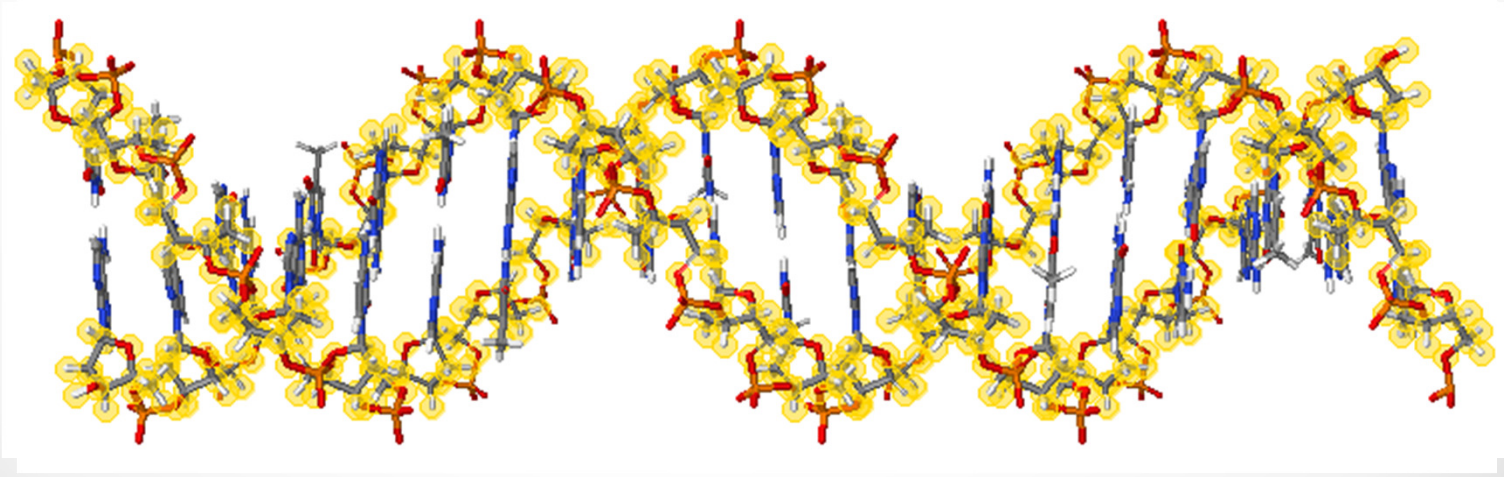
GC Base Pair
Guanine-Cytosine



TA Base Pair
Thymine-Adenine

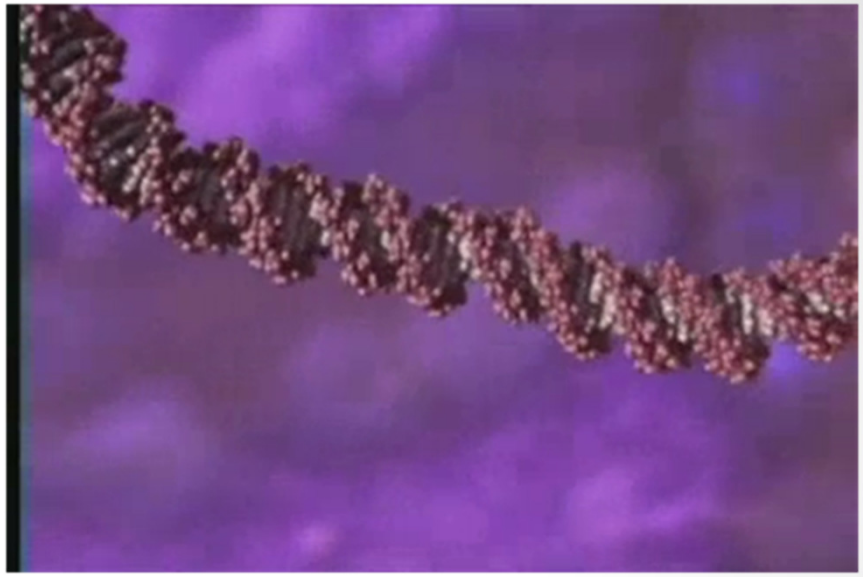
Interactive DNA Tutorial
(http://www.biosciences.bham.ac.uk/labs/minchin/tutorials/dna.html)



Sequence of Base Pairs (GACT alphabet)

# Robust, and *Long*

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



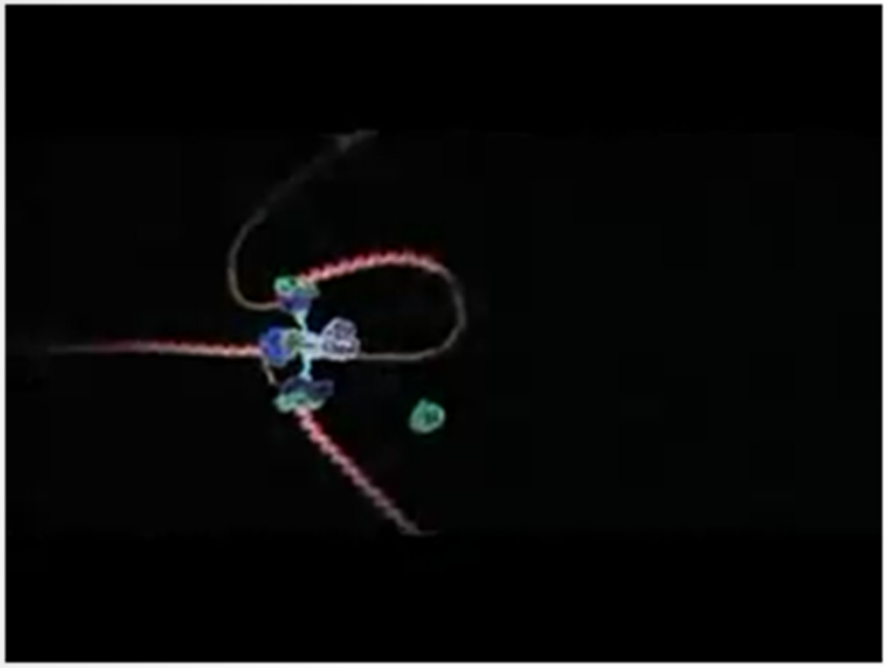
DNA wrapping into chromosomes

wehi.edu.au

Andromeda Galaxy
2.5 million light years away

# Zipping Along

- DNA can support structural and computational complexity.



DNA replication in *real time*

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

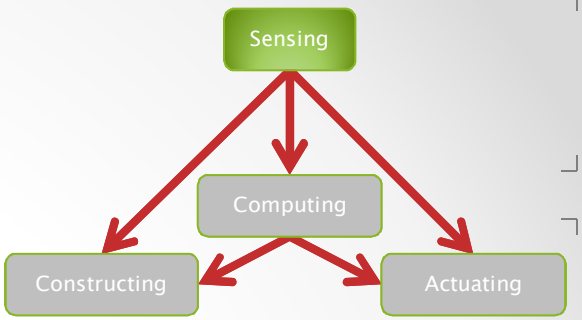
In Bacteria: 1000 nucleotides/second
(higher error rate)

DNA transcription in *real time*

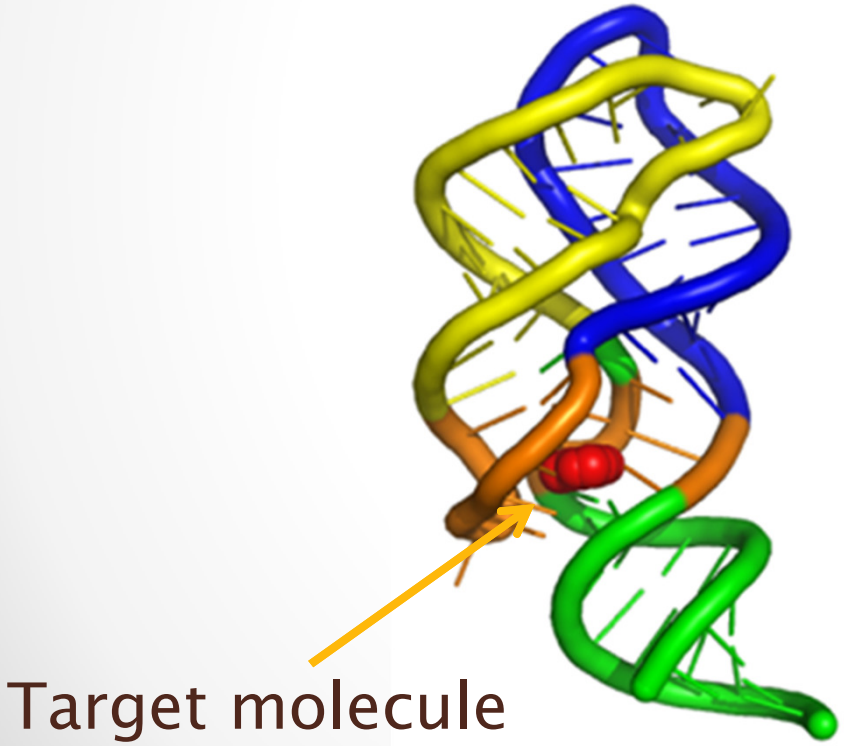
RNA polymerase II:
15-30 bases/second

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

# Sensing



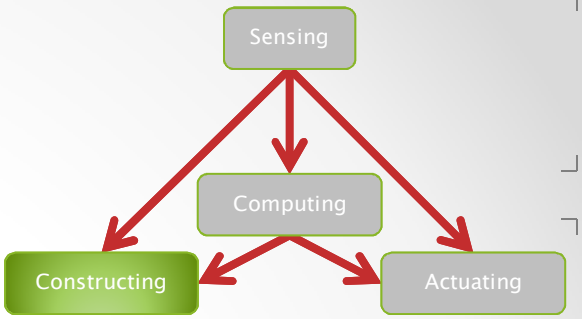

Aptamers: natural or artificially evolved DNA molecules that stick to other molecules (highly selectively).



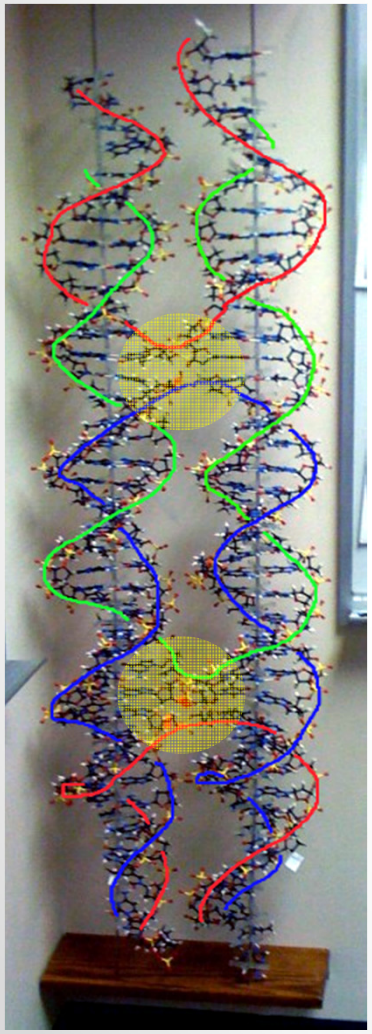

## Adenine riboswitch aptamer

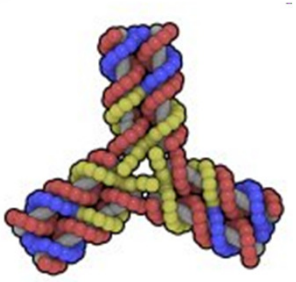
Structural basis for discriminative regulation of gene expression by adenine- and guanine-sensing mRNAs. Chem Biol. 2004 Dec;11(12):1729-41.

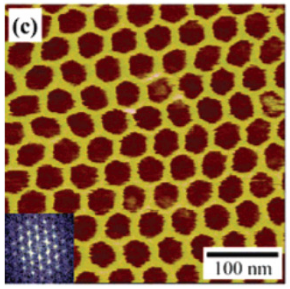
# Constructing



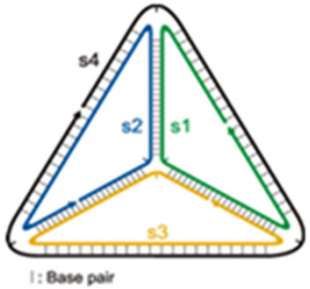

## Crosslinking

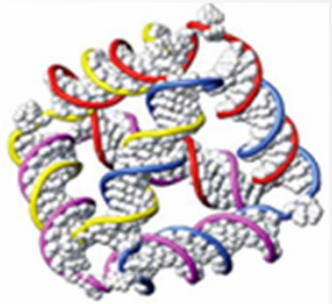







Chengde Mao, Purdue



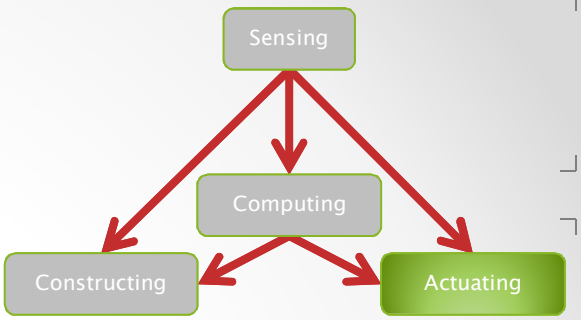
Andrew Turberfield, Oxford

**Folding DNA into Twisted and Curved Nanoscale Shapes**

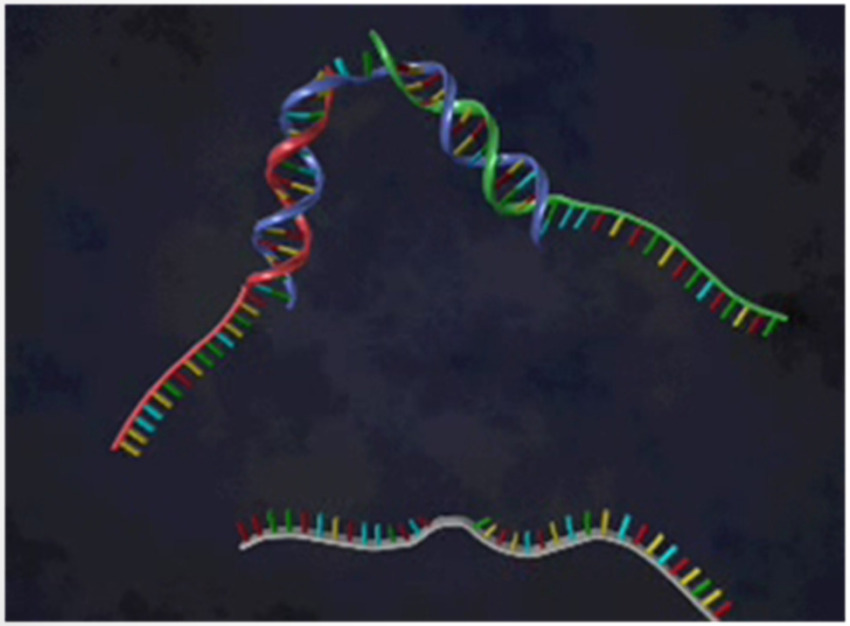
Hendrik Dietz, Shawn M. Douglas, & William M. Shih

Science, 325:725–730, 7 August 2009.

# Actuating

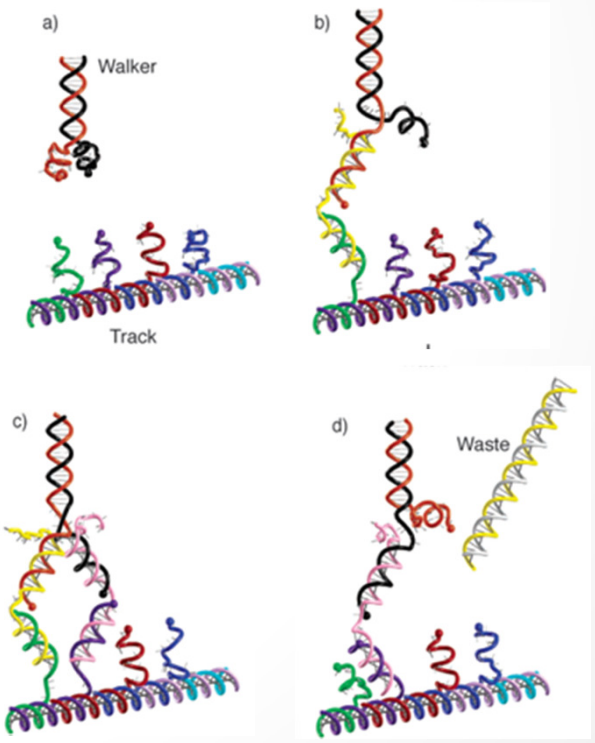



## DNA tweezers

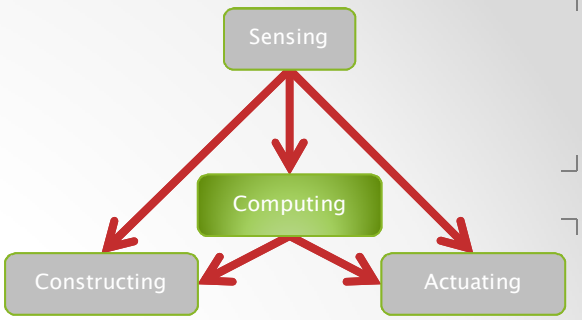


Bernard Yurke, Boise State

## DNA walkers



# Computing



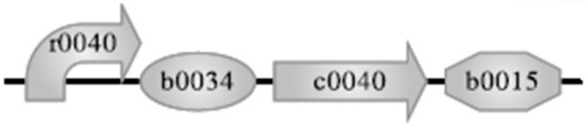

- Sensors and Actuators at the 'edge' of the system
  - They can use disparate technologies and phenomena
- Computation in the 'kernel' of the system
- Compositionality in the kernel
  - The components should use uniform inputs and outputs
  - The components should be ‘computationally complete’

# “Embedded” Computing

- Using bacterial machinery (e.g.) as the hardware.
  Using embedded gene networks as the software.

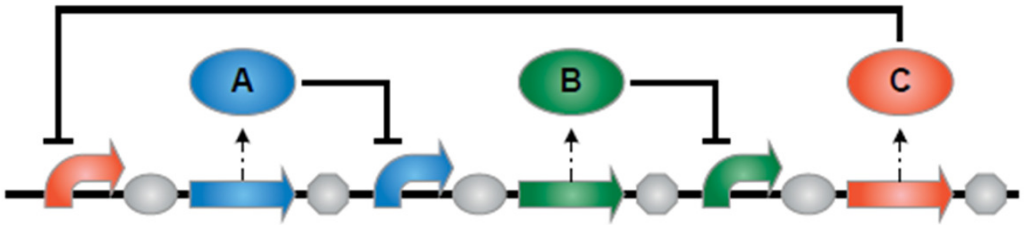
- MIT Registry of Standard Biological Parts

- GenoCAD
  - Meaningful sequences [Cai et al.]




```
r0040:prom; b0034:rbs; c0040:pcr; b0015:ter
```

- GEC
  - [Pedersen & Phillips]

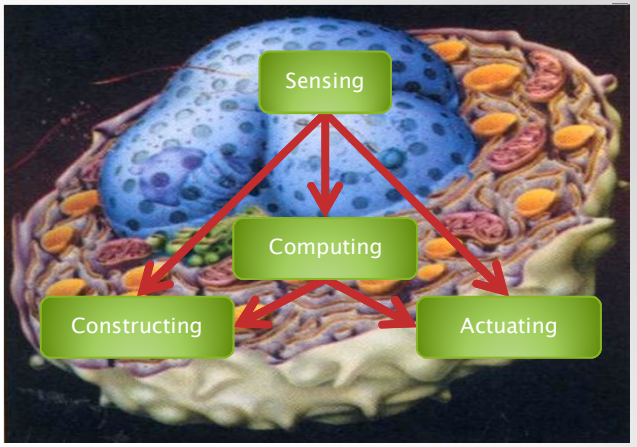



```
prom<neg(C)>; rbs; pcr<codes(A)>; ter;
prom<neg(A)>; rbs; pcr<codes(B)>; ter;
prom<neg(B)>; rbs; pcr<codes(C)>; ter
```

# “Autonomous” Computing

- Mix & go
  - All (or most) parts are synthesized
  - No manual cycling (cf. early DNA computing)
  - In some cases, all parts are made of DNA (no enzyme/proteins)
- Self-assembled and self-powered
  - Can run on its own (e.g. environmental sensing)
  - Or be embedded into organisms, but running ‘separately’

# Curing




## A doctor in each cell

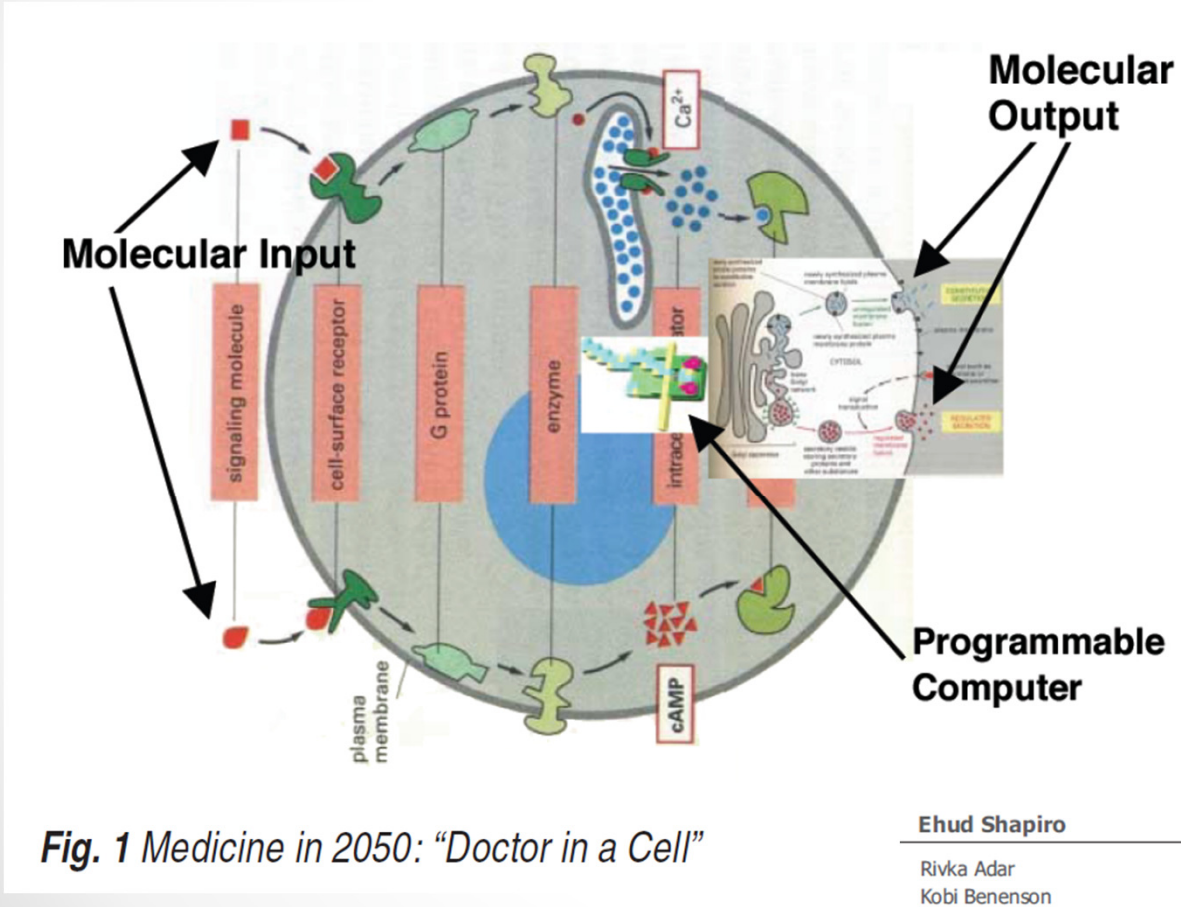



***Fig. 1*** *Medicine in 2050: "Doctor in a Cell"*

**Ehud Shapiro**

Rivka Adar
Kobi Benenson
Gregory Linshitz
Aviv Regev
William Silverman

**Molecules and computation**

# Autonomous DNA Computing

# Why Compute with DNA?

- Non-goals
  - Not to solve NP-complete problems.
  - Not to replace electronics.
  - Not necessarily using genes or producing proteins.

- For general 'molecular programming'
  - To precisely control the organization and dynamics of matter and information at the molecular level.
  - To interact algorithmically with biological entities.
  - The use of DNA is "accidental": no genes involved.
  - In fact, no material of biological origin.

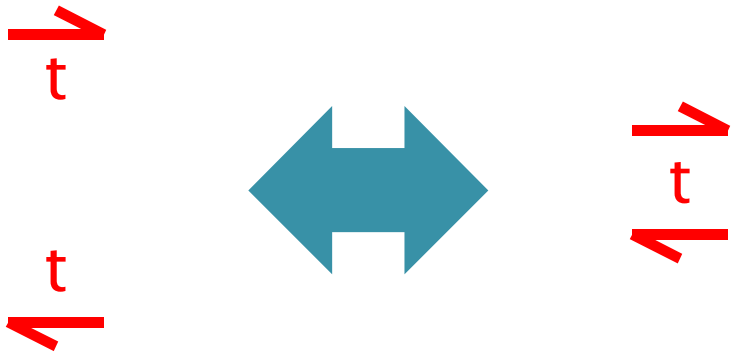
# Domains

- Subsequences on a DNA strand are called domains. *PROVIDED* they are "independent" of each other.


- I.e., 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.

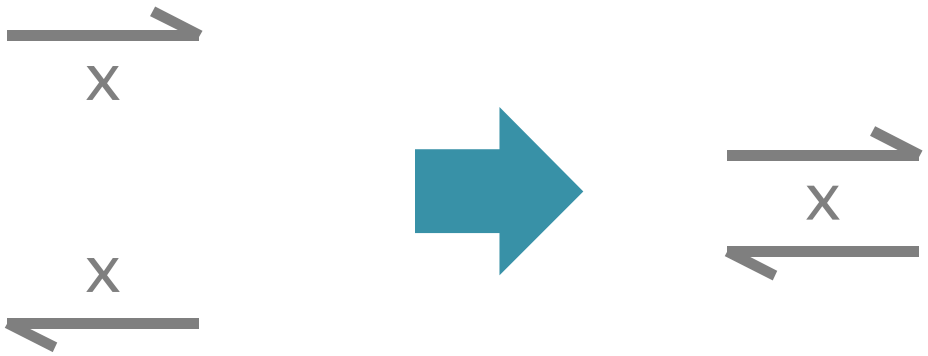
- Choosing domains (subsequences) that are suitably independent is a tricky issue that is still somewhat of an open problem (with a vast literature). But it can work in practice.

# Short Domains



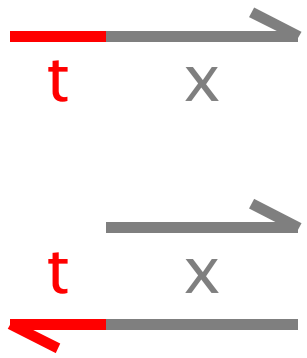

Reversible Hybridization

# Long Domains



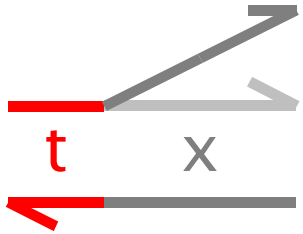

Irreversible Hybridization

# Strand Displacement



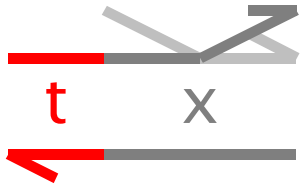

"Toehold Mediated"

# Strand Displacement



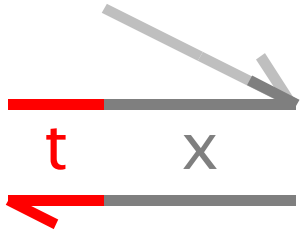

Toehold Binding

# Strand Displacement



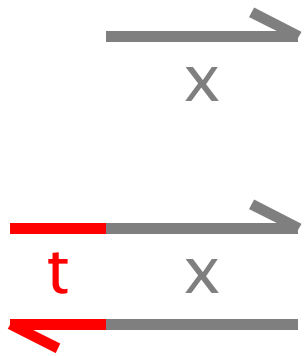

Branch Migration

# Strand Displacement



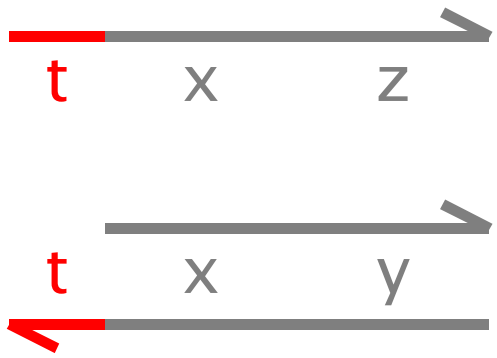

Displacement

# Strand Displacement

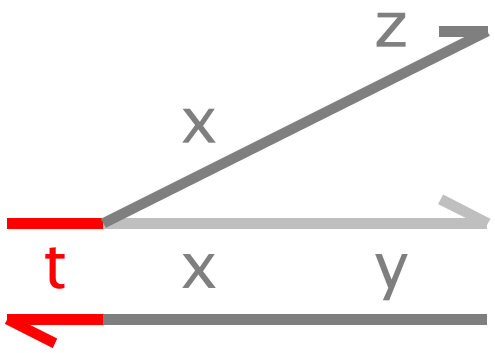



Irreversible release

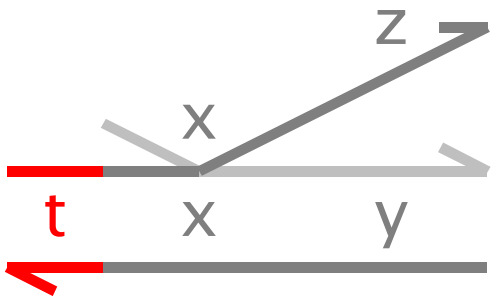
# Bad Match



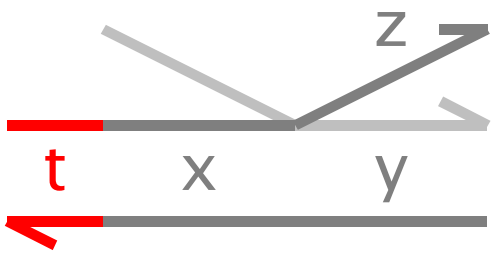
# Bad Match



# Bad Match



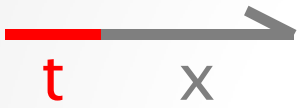
# Bad Match




Cannot proceed
Hence will undo

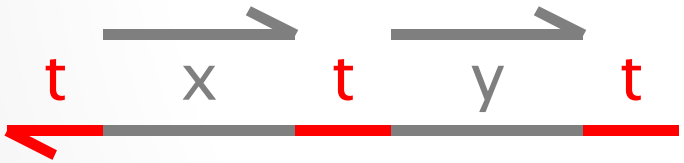
# Two-Domain Architecture

- Signals: 1 toehold + 1 recognition region




Garbage collection
"built into" the gates

- Gates: "top-nicked double strands"
(or equivalently double strands with open toeholds)

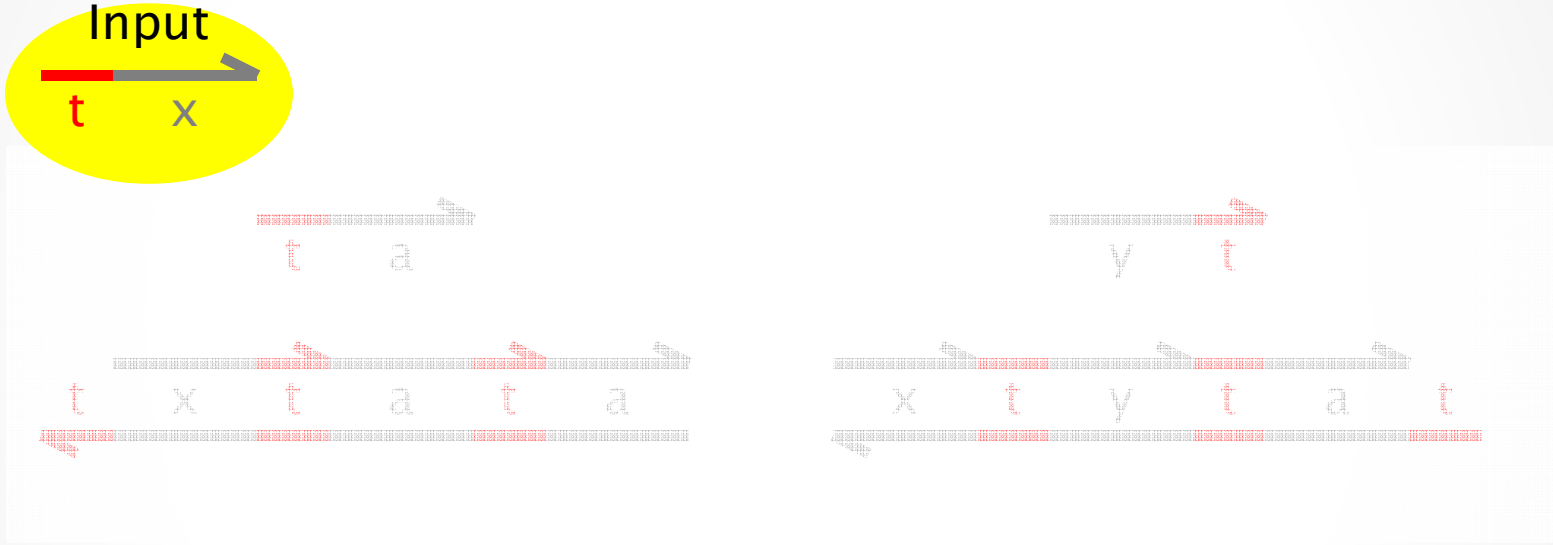



Two-Domain DNA Strand Displacement

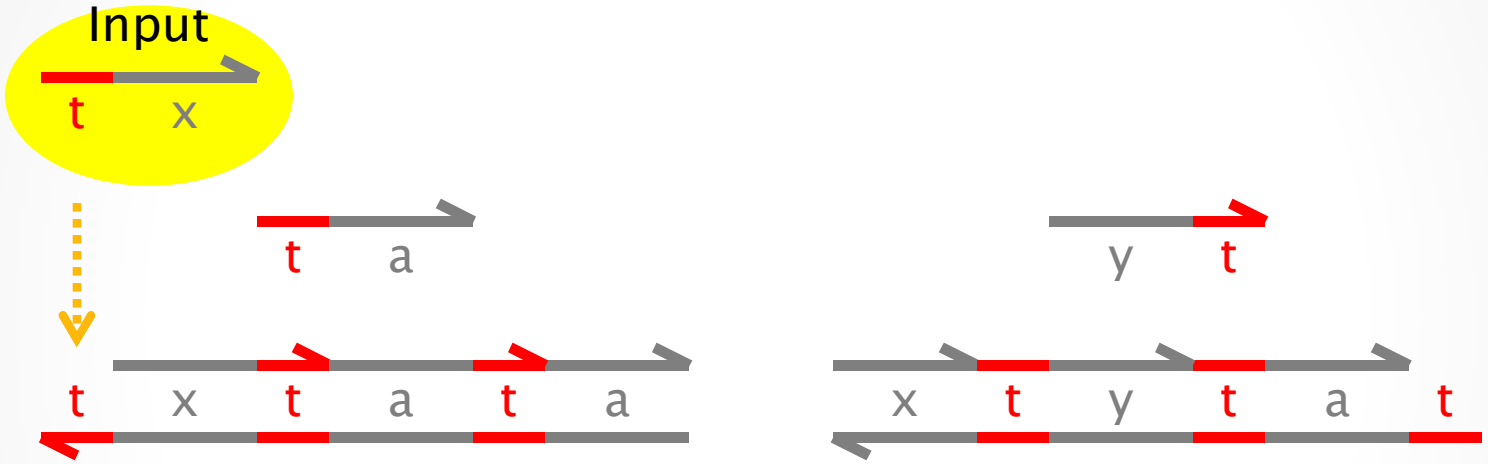
*Luca Cardelli*

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



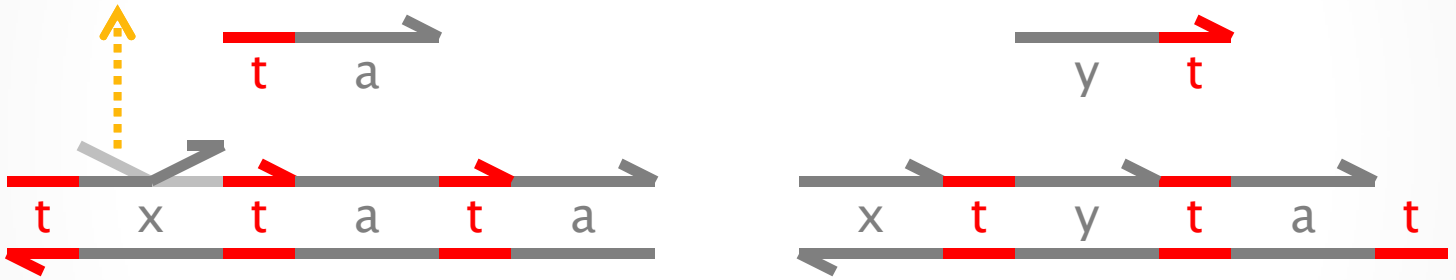
# Transducer x→y



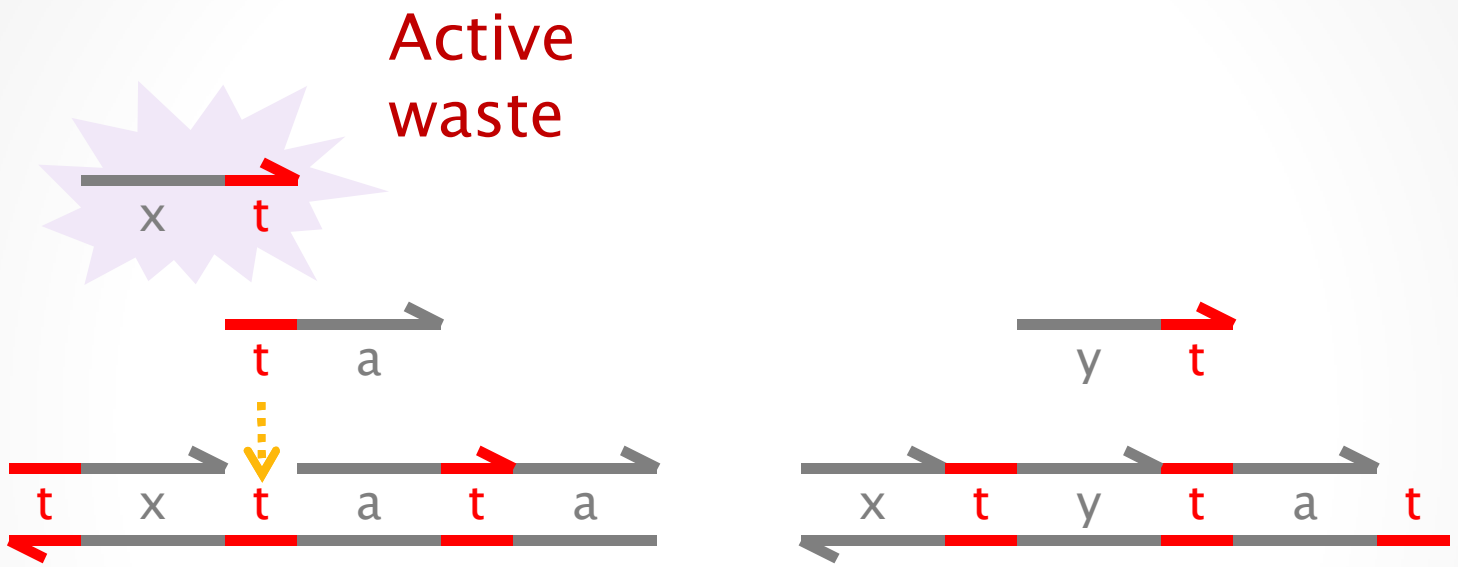

**Built by self-assembly!**

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

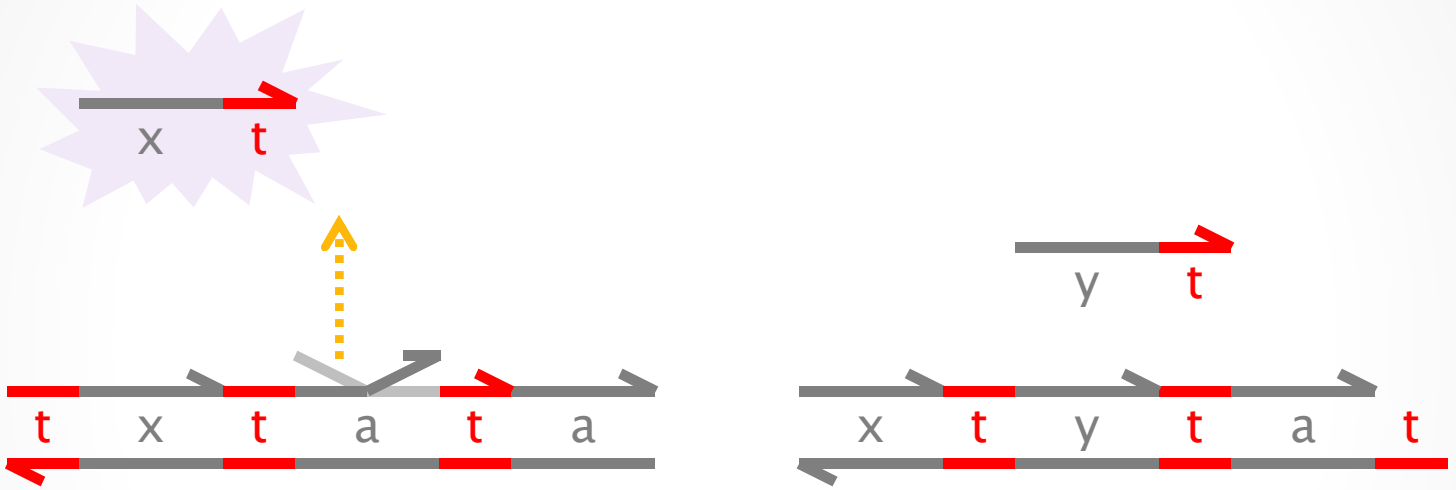
# Transducer x→y



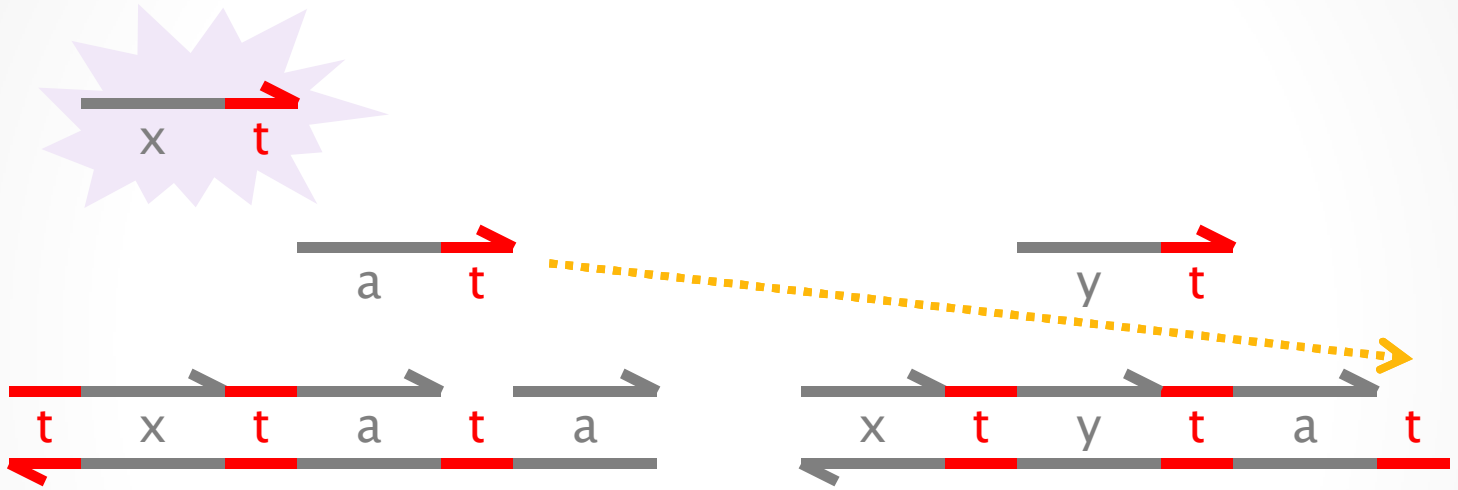
# Transducer x→y



# Transducer x→y

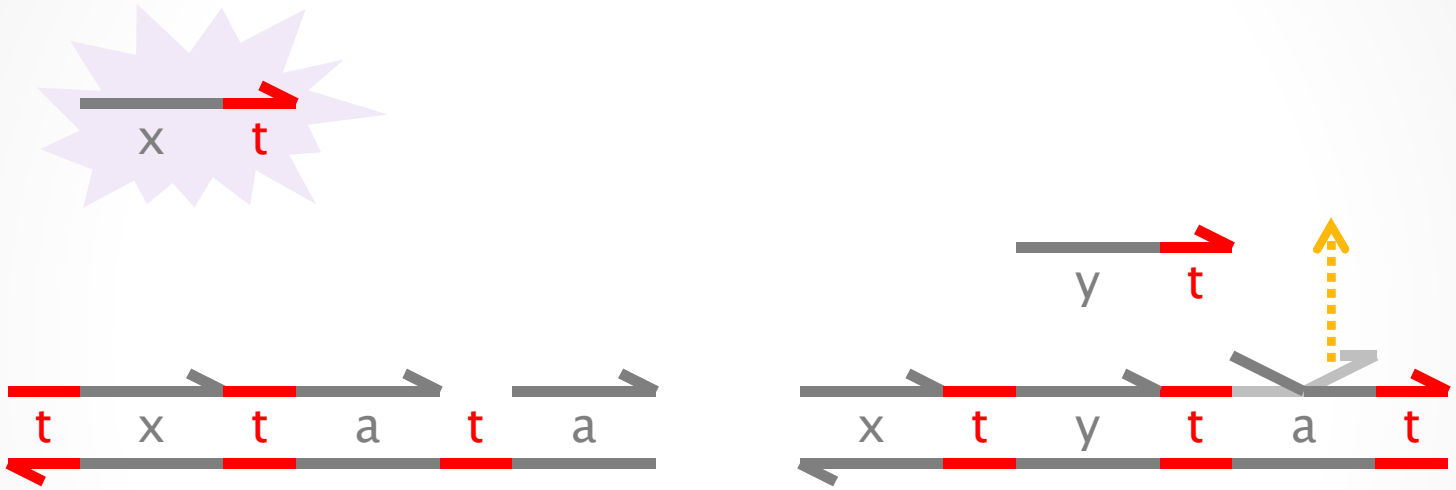


# Transducer x→y

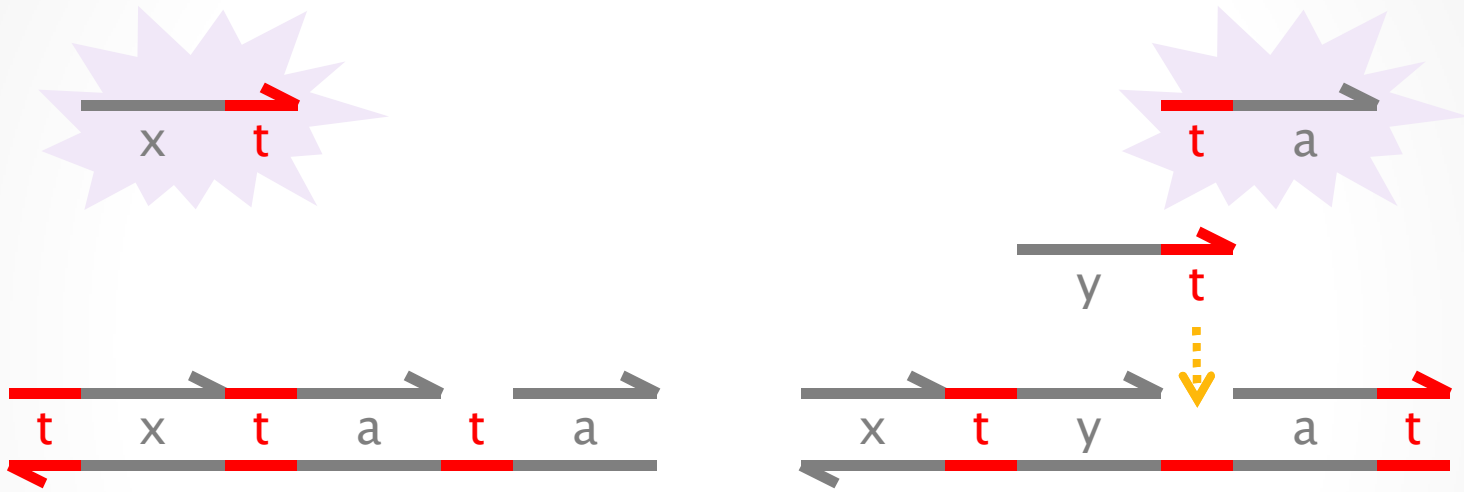



So far, a tx *signal* has produced an at *cosignal.*
But we want signals as output, not cosignals.

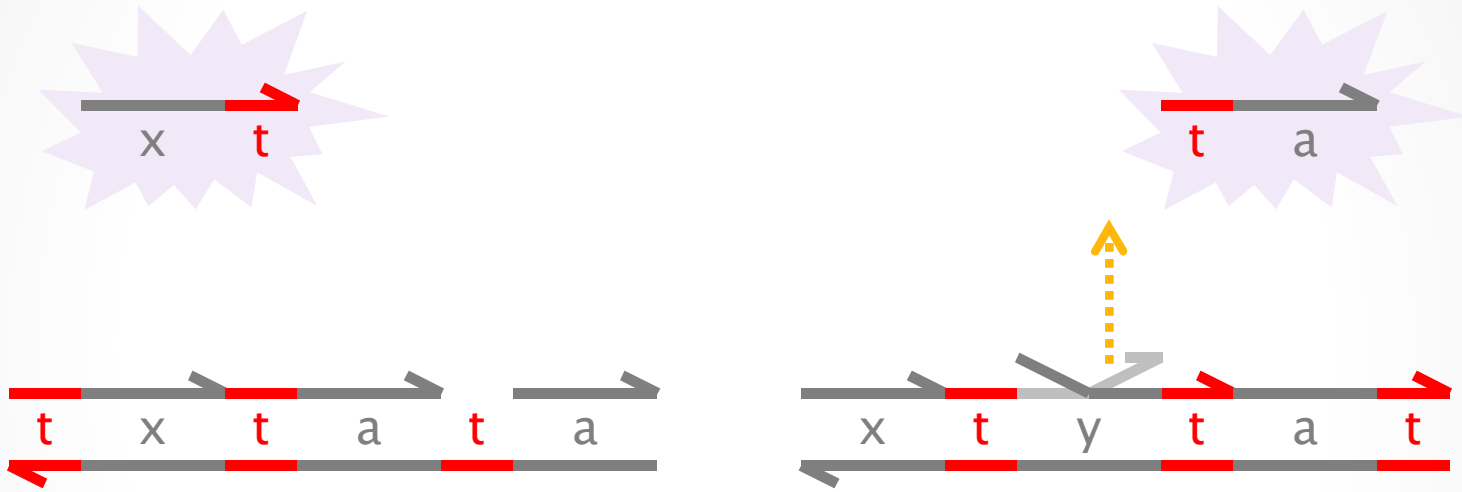
# Transducer x→y



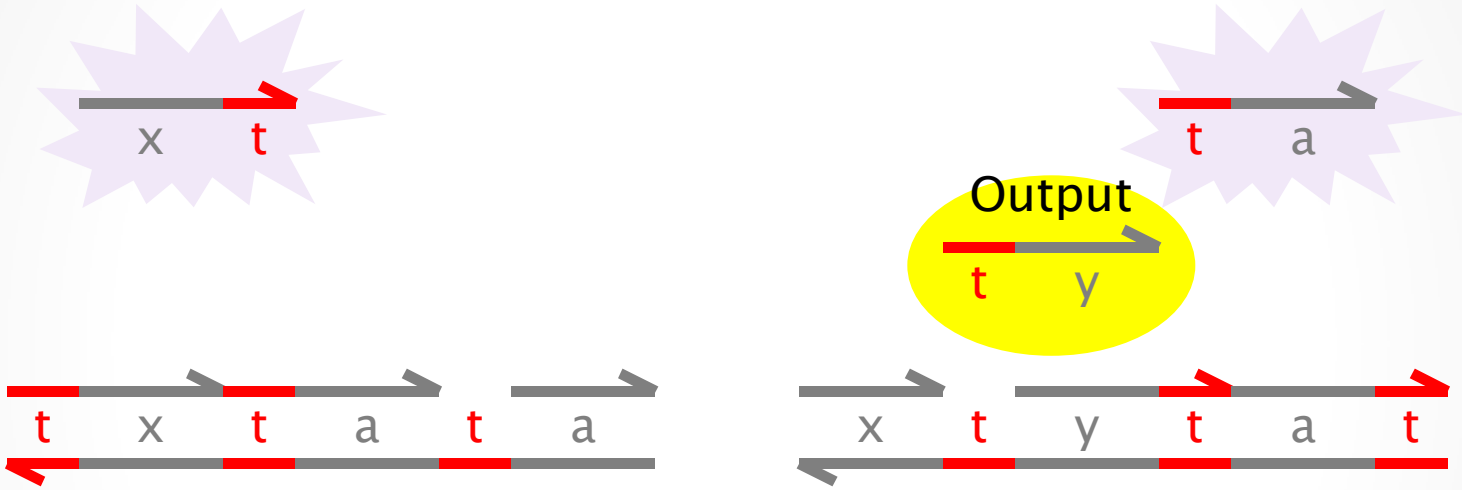
# Transducer x→y



# Transducer x→y



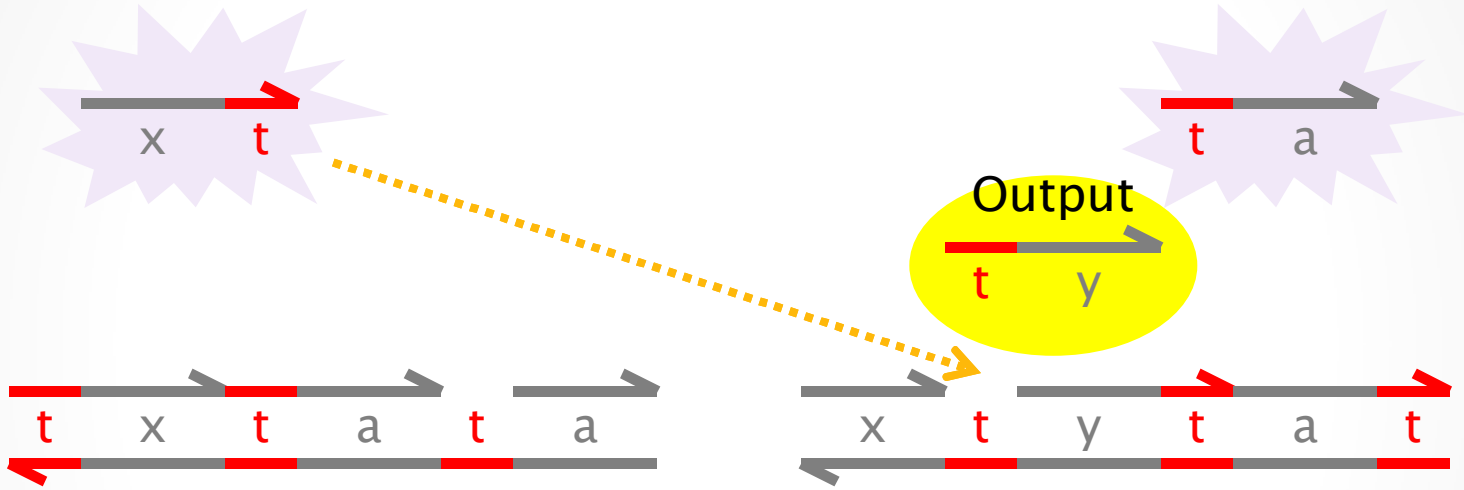
# Transducer x→y



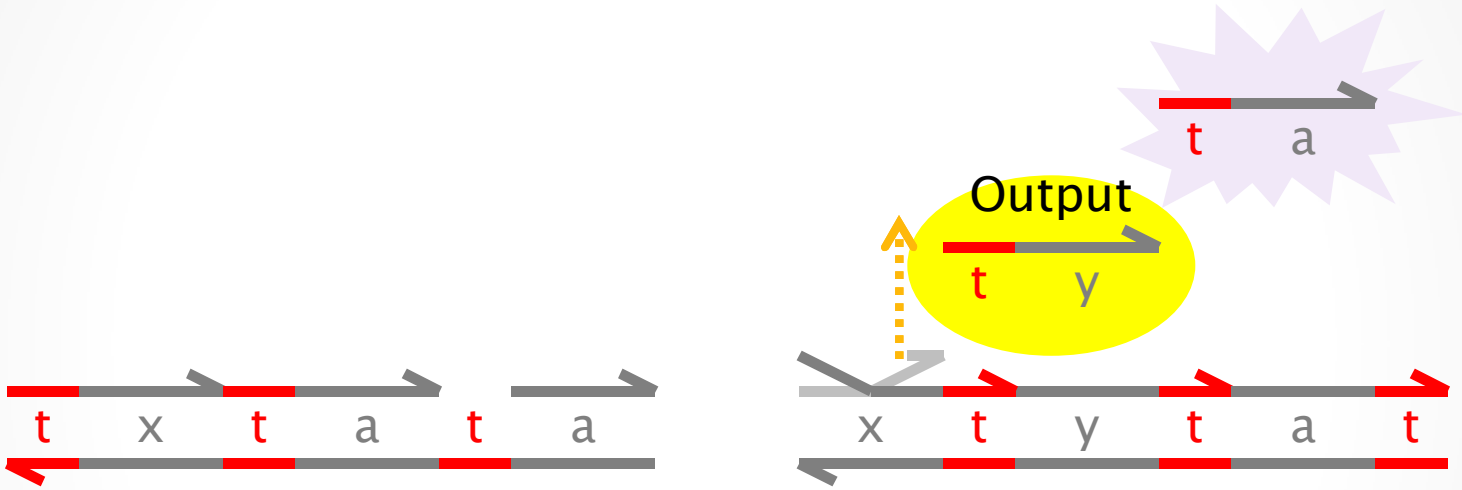

Here is our output ty *signal*.

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).

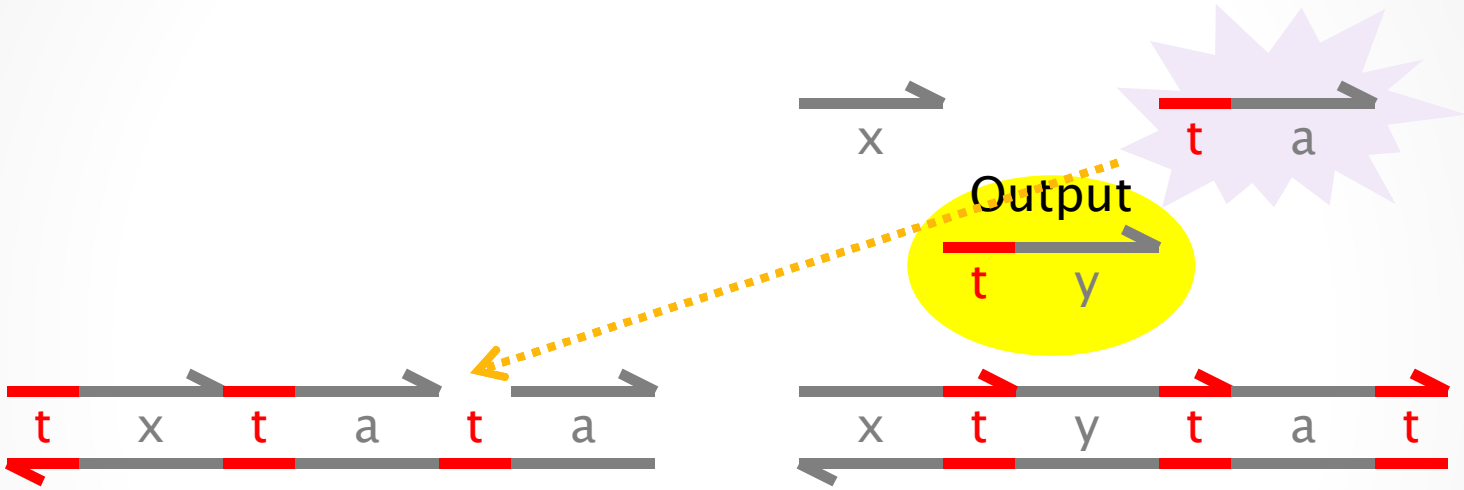
# Transducer x→y



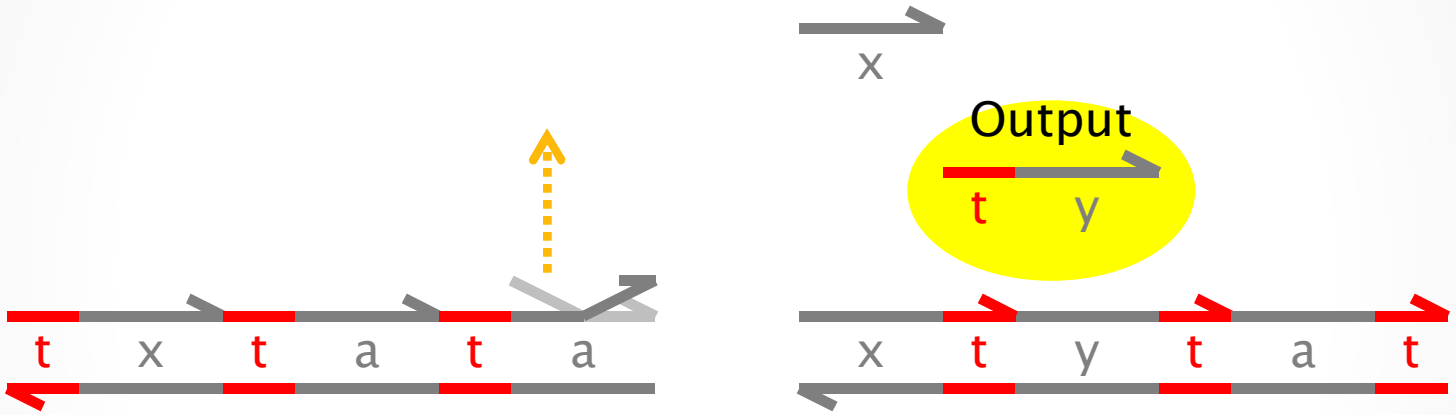
# Transducer x→y



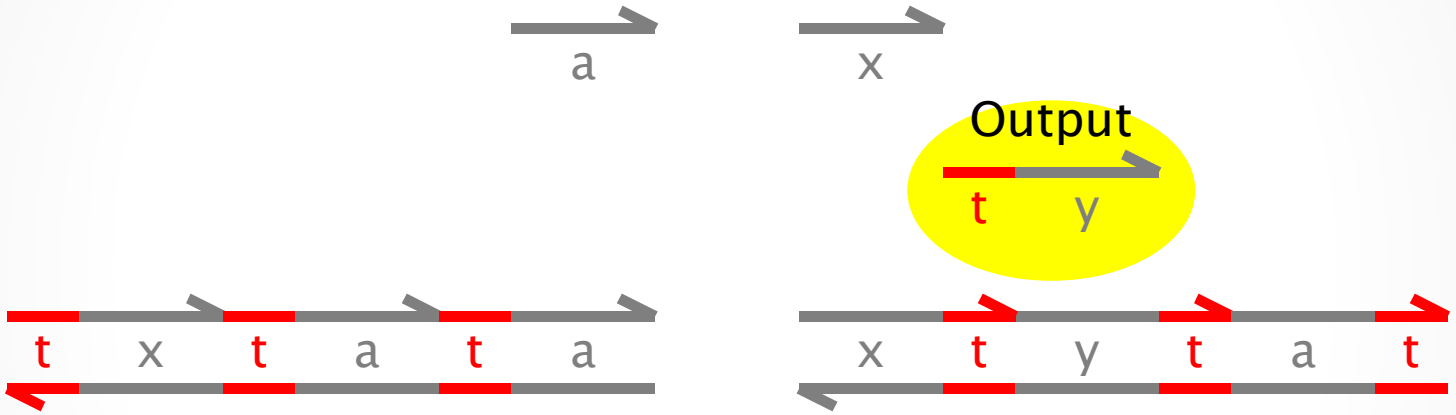
# Transducer x→y



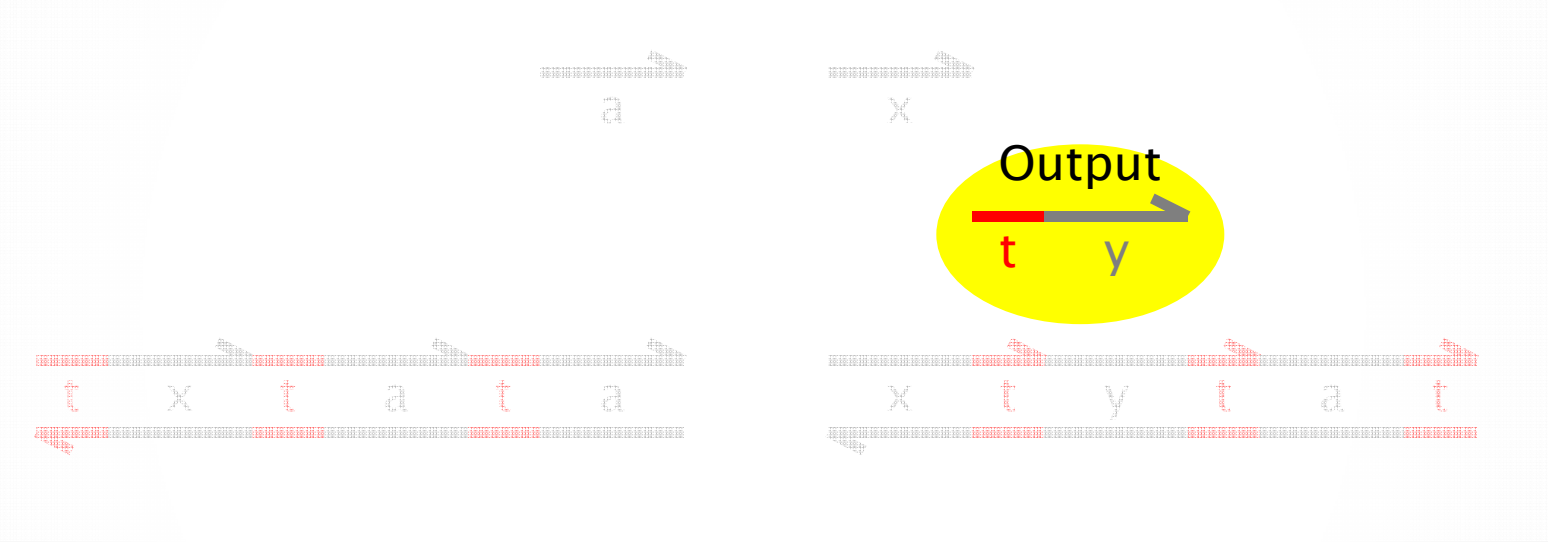
# Transducer x→y



# Transducer x→y



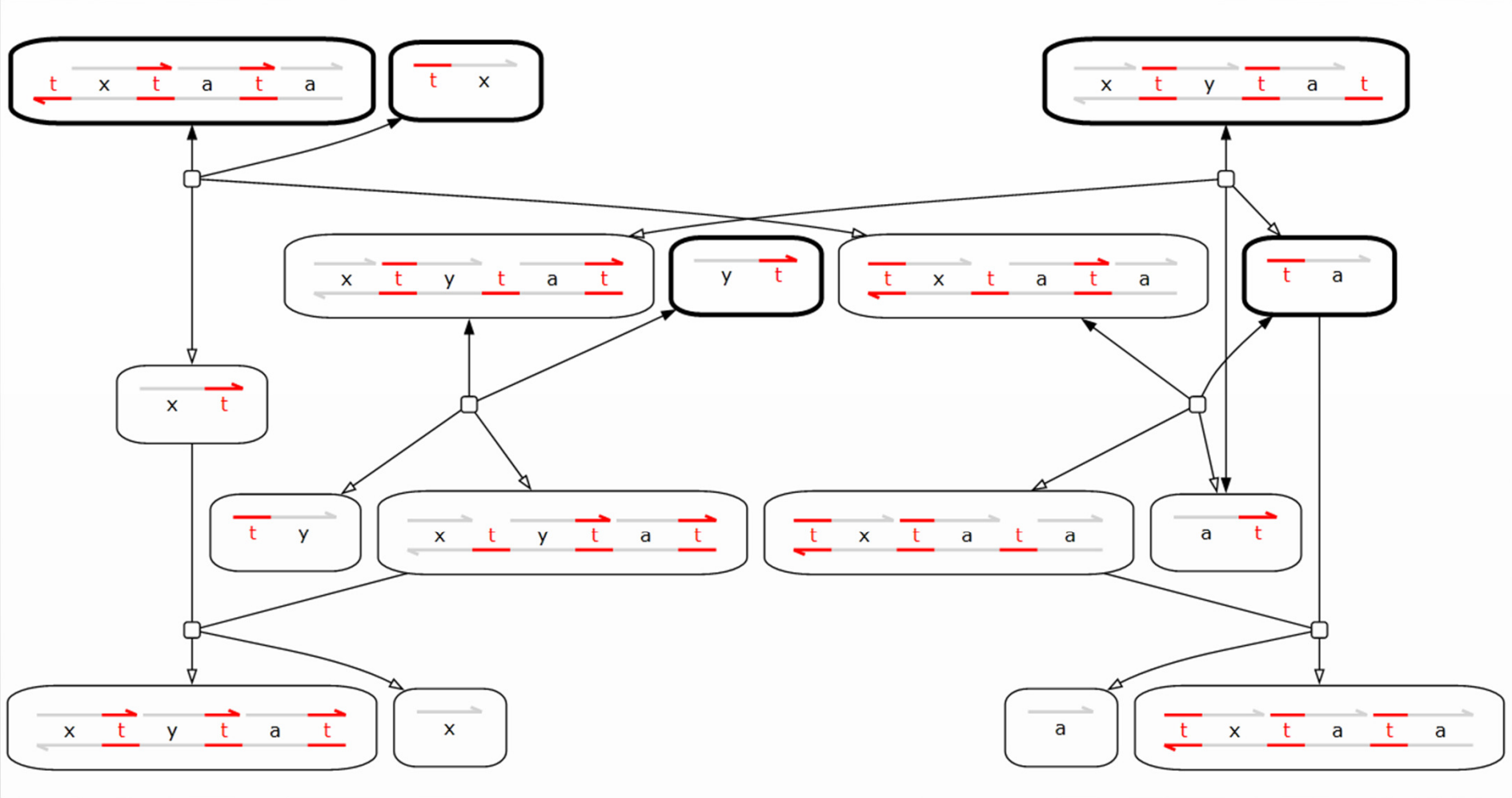
# Transducer x→y




Done.

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

# Reaction Graph for x→y



# General n×m Join-Fork

- Easily generalized to 2+ inputs (with 1+ collectors).
- Easily generalized to 2+ outputs.

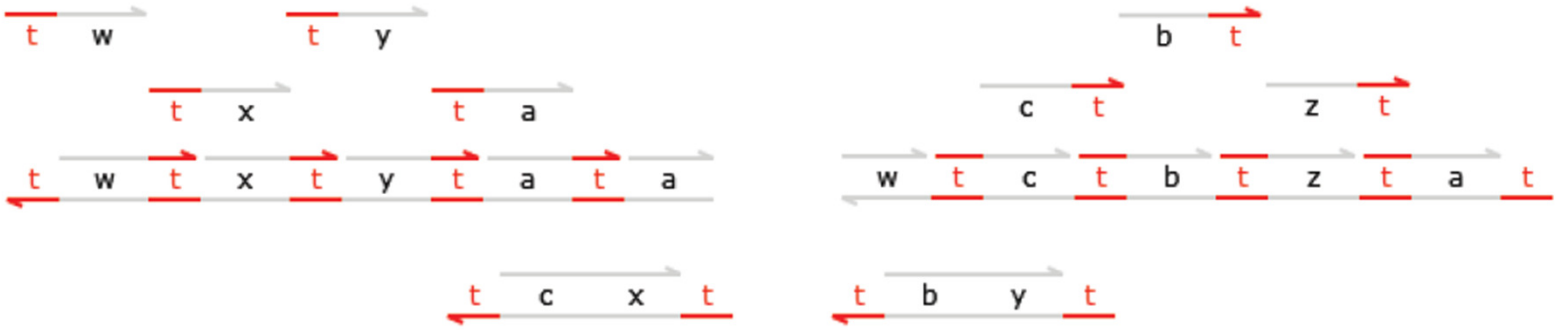
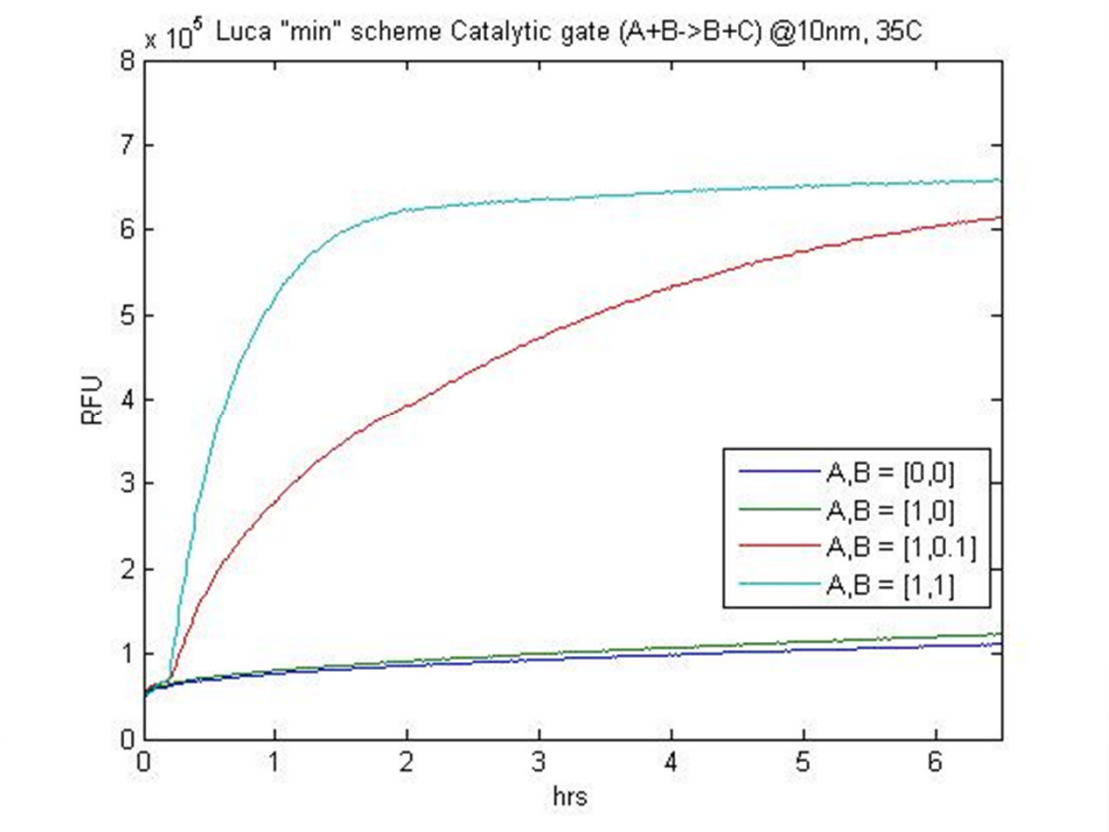



Figure 9: 3-Join $J_{wxyz} \mid tw \mid tx \mid ty \rightarrow tz$: initial state plus inputs $tw$, $tx$, $ty$.

# Experiments

Two-domain gate for $A+B \rightarrow B+C$




Matt Olson and Georg Seelig, U.Washingon.

# Strand Algebra

- An intermediate language for molecular computing
  - Signals: x
  - Gates: $[x_1,..,x_n].[y_1,..,y_m]$
  - Parallel composition: |
  - Populations: (…)*

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

Input Signals (consumed)

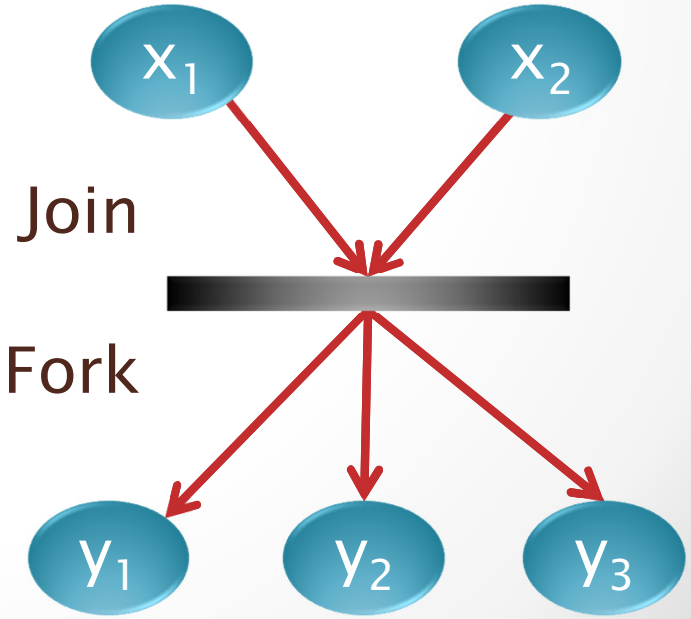
Gate (consumed)

Output Signals (produced)

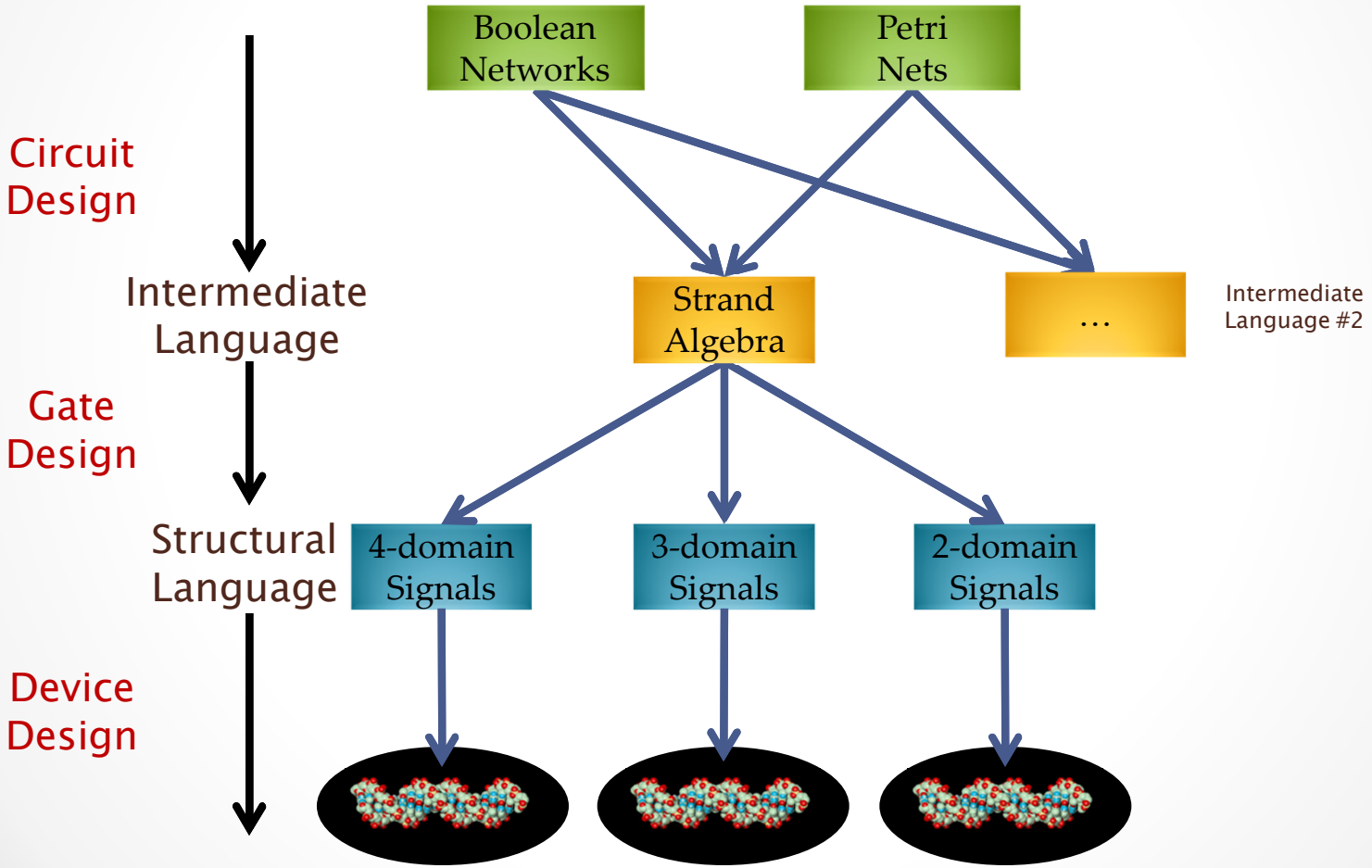
# Petri Net Transitions

- Computing power equivalent to Petri Nets
  - Not Turing complete, but as good as chemistry itself.
  - The correspondence is not completely trivial: gates are consumed by activation, hence a persistent Petri net transition requires a stable population of gates.

- Hence, many other mechanisms are expressible
  - E.g. Boolean networks



# Compilation Issues



# Optimization Issues

- Reduce number of species
- Optimize kinetics
- Etc.

# Verification Issues

- Environment
  - The nano-environment is messy (stochastic noise, failures, etc.)
  - But we should al least ensure our designs are *logically correct*
- Verifying Components
  - Reversible reactions (infinite traces)
  - Interferences (deadlocks etc.) between copies of the same gate
  - Interferences (deadlocks etc.) between copies of different gates
  - Removal of active byproducts (garbage collection) is tricky
- Verifying Populations
  - Gates come in (large) populations
  - Each population *shares private domains* (technologically unavoidable)
  - Correctness of populations means proofs with large state spaces

# Correctness

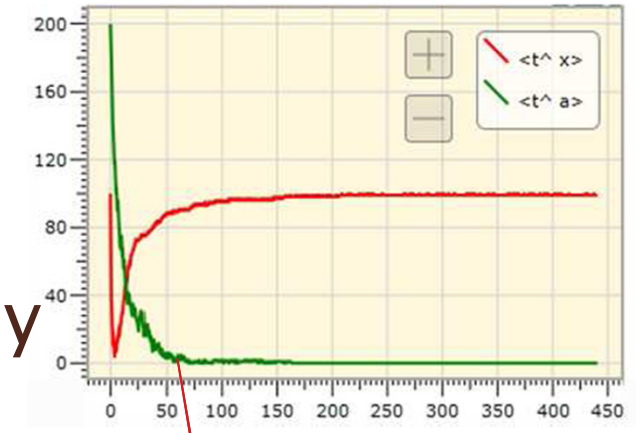
- The spec of a transducer:

$$x.y \mid x \rightarrow y$$

  - Is it true at all?
  - Is it true *possibly, necessarily,* or *probabilistically ?*
  - Is it true in the context of a *population of identical transducers?*
  - Is it true *in all possible contexts?*
  - If false, does it become true for *infinite populations*?

# Interfering Transducers

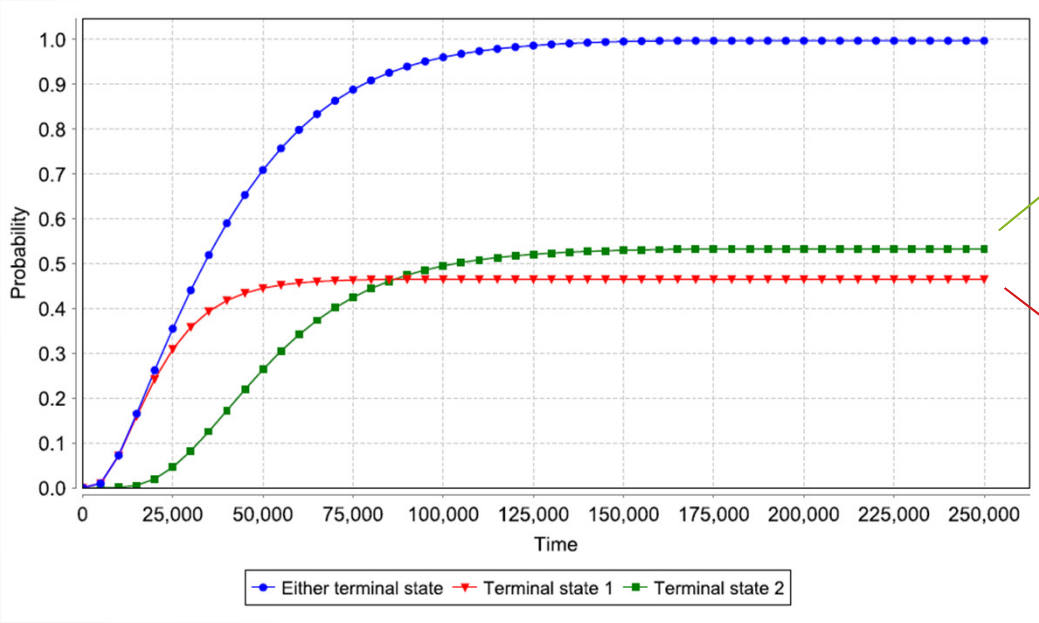
- Let a be the private transducer domain, but let's share it between x.y and y.x
- Interference: $x._a y \mid y._a x \mid x \not\rightarrow^{\forall} x$
- But still: $x._a y \mid y._a x \mid x \mid y \rightarrow^{\forall} x \mid y$
- A large population of such gates in practice does not deadlock easily.
- The wisdom of crowds: individuals can be wrong, but the population is all right.




Stuck gates in a population of 200

# Modelchecking DNA Systems

- Using the PRISM stochastic modelchecker
  - Termination probability of interfering transducers $x \mid x._a y \mid y._a z$




Correct Termination

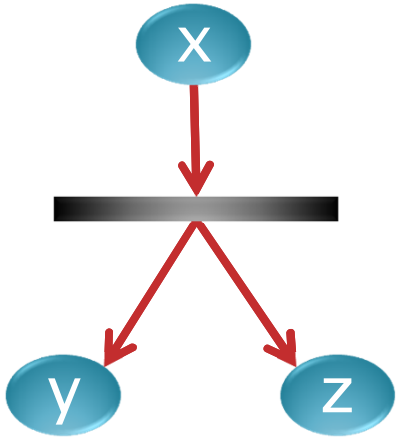
Incorrect Termination

L. Cardelli, M. Kwiatkowska, M. Lakin, D. Parker and A. Phillips.
Design and Analysis of DNA Circuits using Probabilistic Model Checking.
http://qav.comlab.ox.ac.uk/papers/dna-pmc.pdf. September 2010

# Molecular Programming Workflow

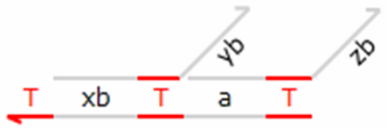
# Circuits to Signals and Gates

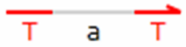
- E.g., a simple Petri Net fork transition
- In Strand Algebra: x | ([x].[y,z])*



# Signals and Gates to Structures

- Visual DSD [Andrew Phillips]




[x].[y,z] 3-domain gate

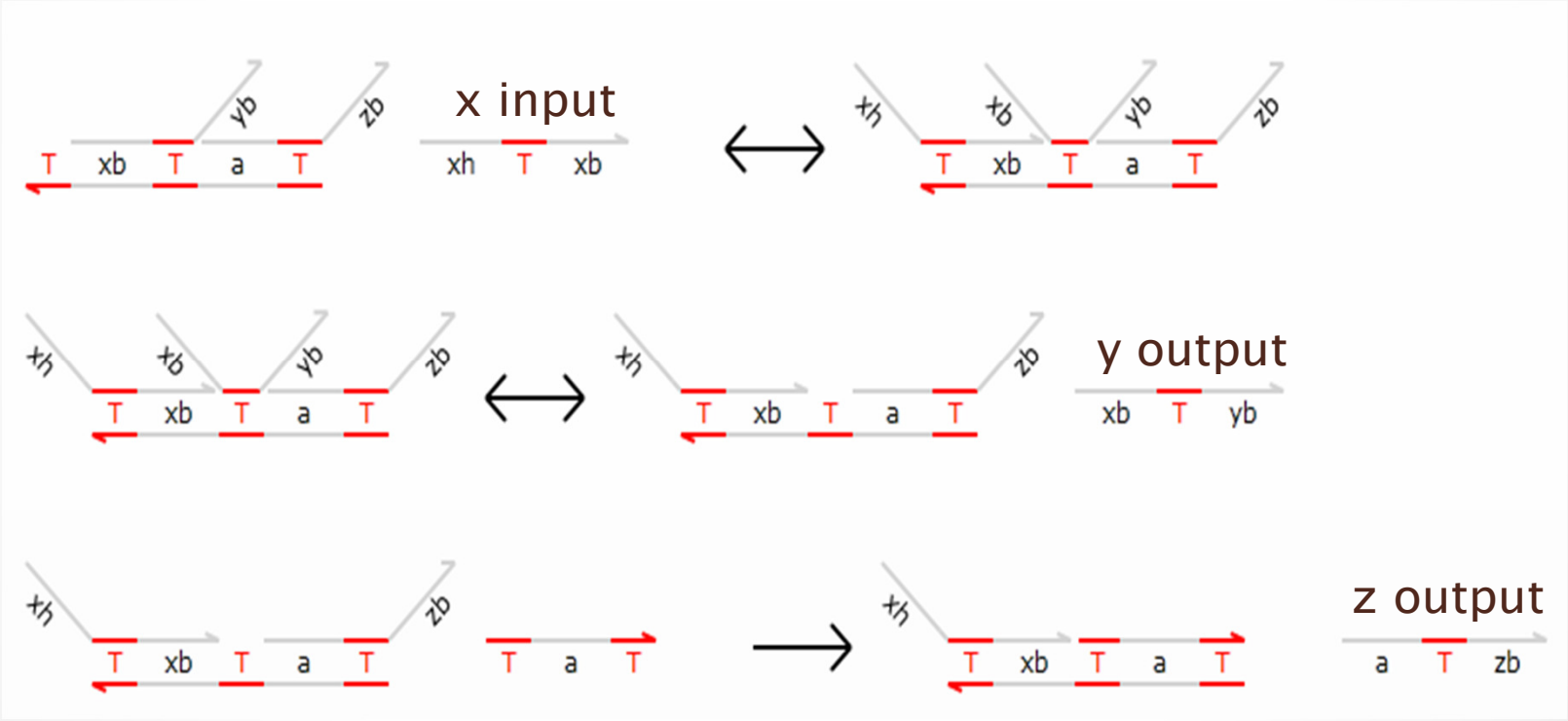

x 3-domain input

```
directive sample 5000.0 1000
directive plot sum(<_ T^ xb>); sum(<_ T^ yb>); sum(<_ T^ zb>)
def scaling = 1000
def bind = 0.0003/(float_of_int scaling) (* /nM/s *) (* =3*10^5 /M/s *)
def unbind =  0.1126 (* /s *)
new T@bind,unbind

def F1x2(N,Xb,Yb,Zb) =
new a
( N * T^:[Xb T^]<Yb>:[a T^]<Zb>
| N * <T^ a T^>
)

(   F1x2(10*scaling,xb,yb,zb)
|  (1*scaling)* <xh T^ xb>
)
```

Actual script

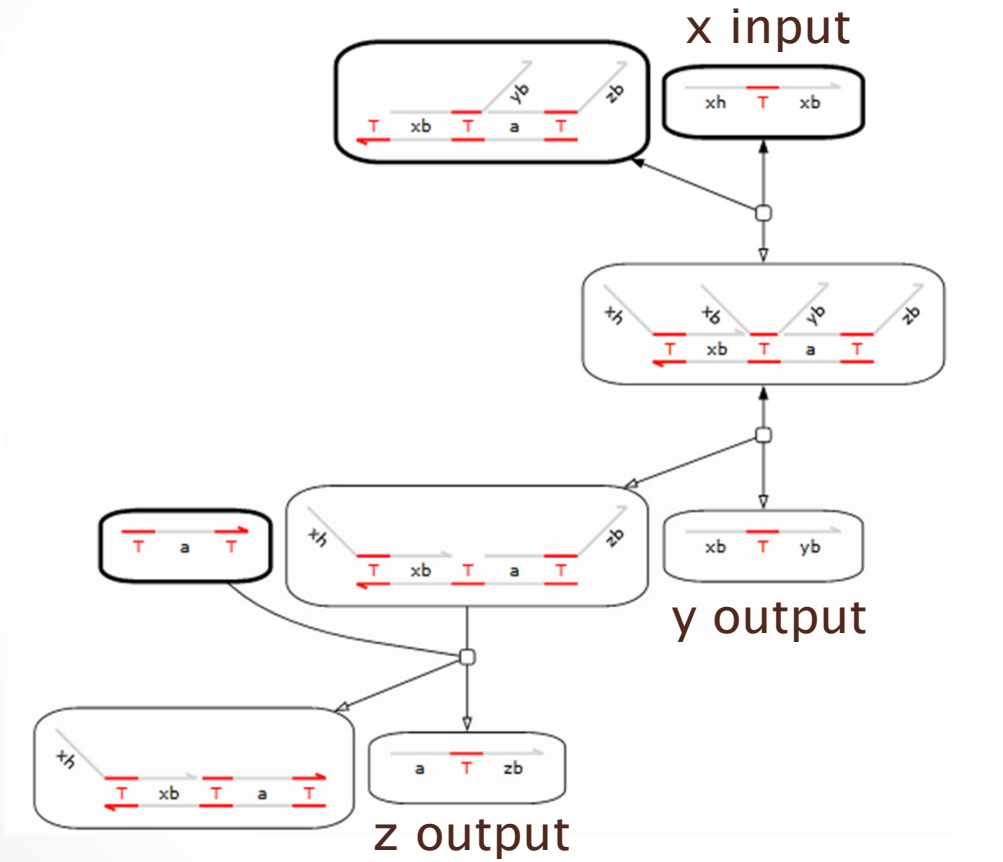
# Signals and Gates to Structures

- Fork gate: the reactions



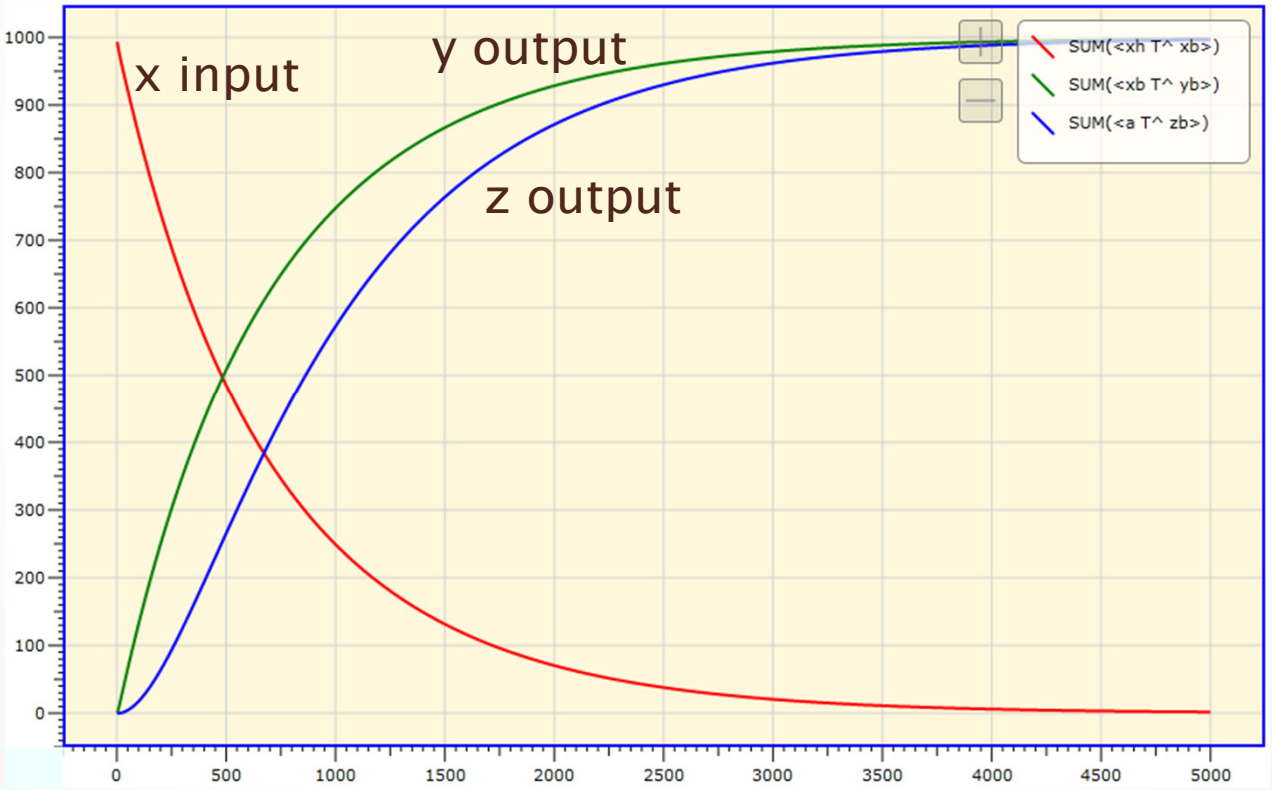
# Signals and Gates to Structures

- Fork gate: the reaction graph



# Signals and Gates to Structures

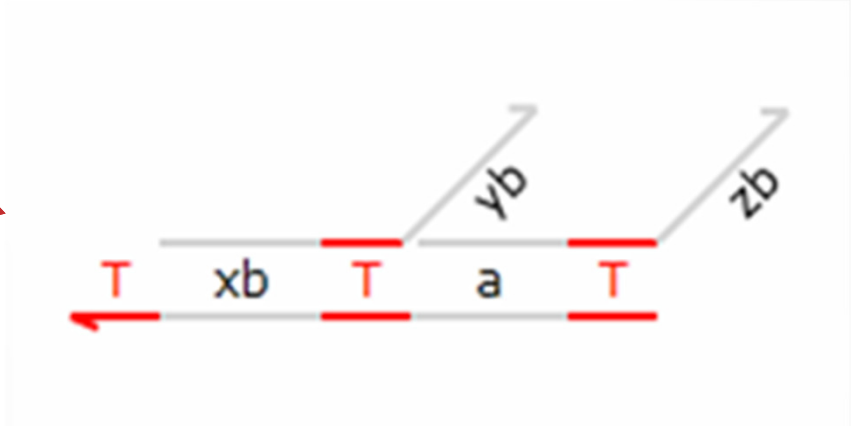
- Fork gate: the behavior



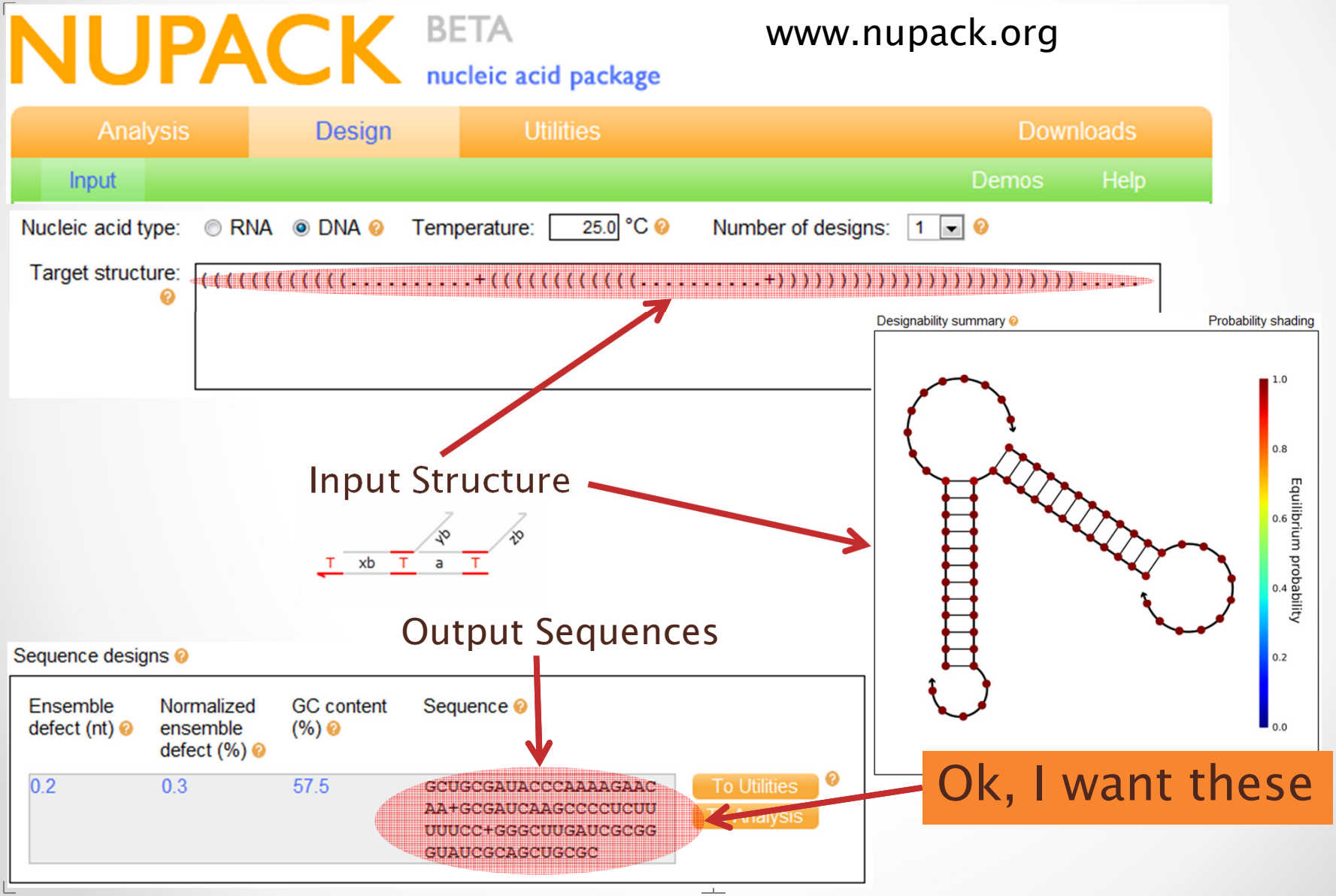
# Signals and Gates to Structures

- Fork gate: check

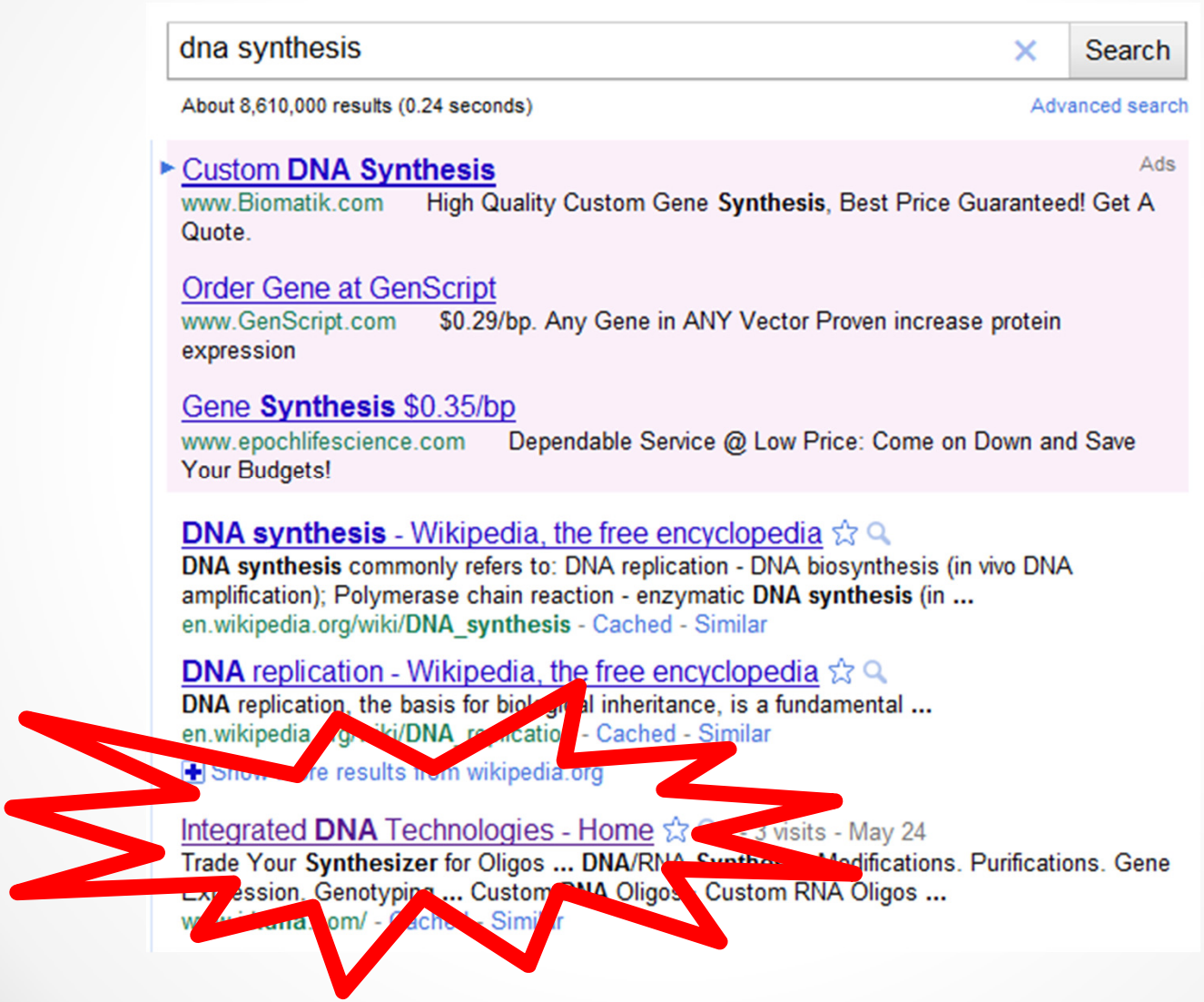
Ok, I want this



# Structures to Sequences

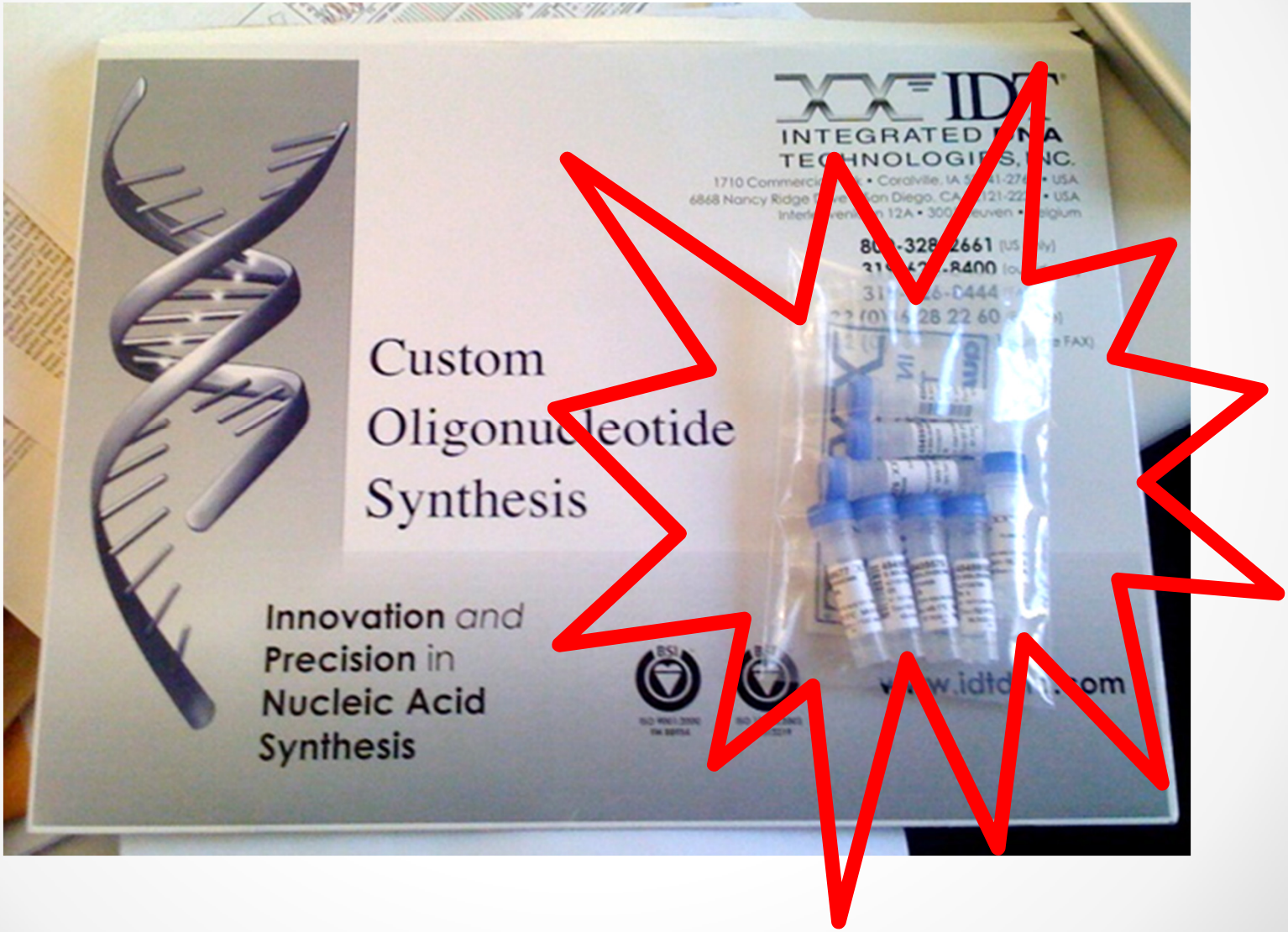


# DNA Synthesis

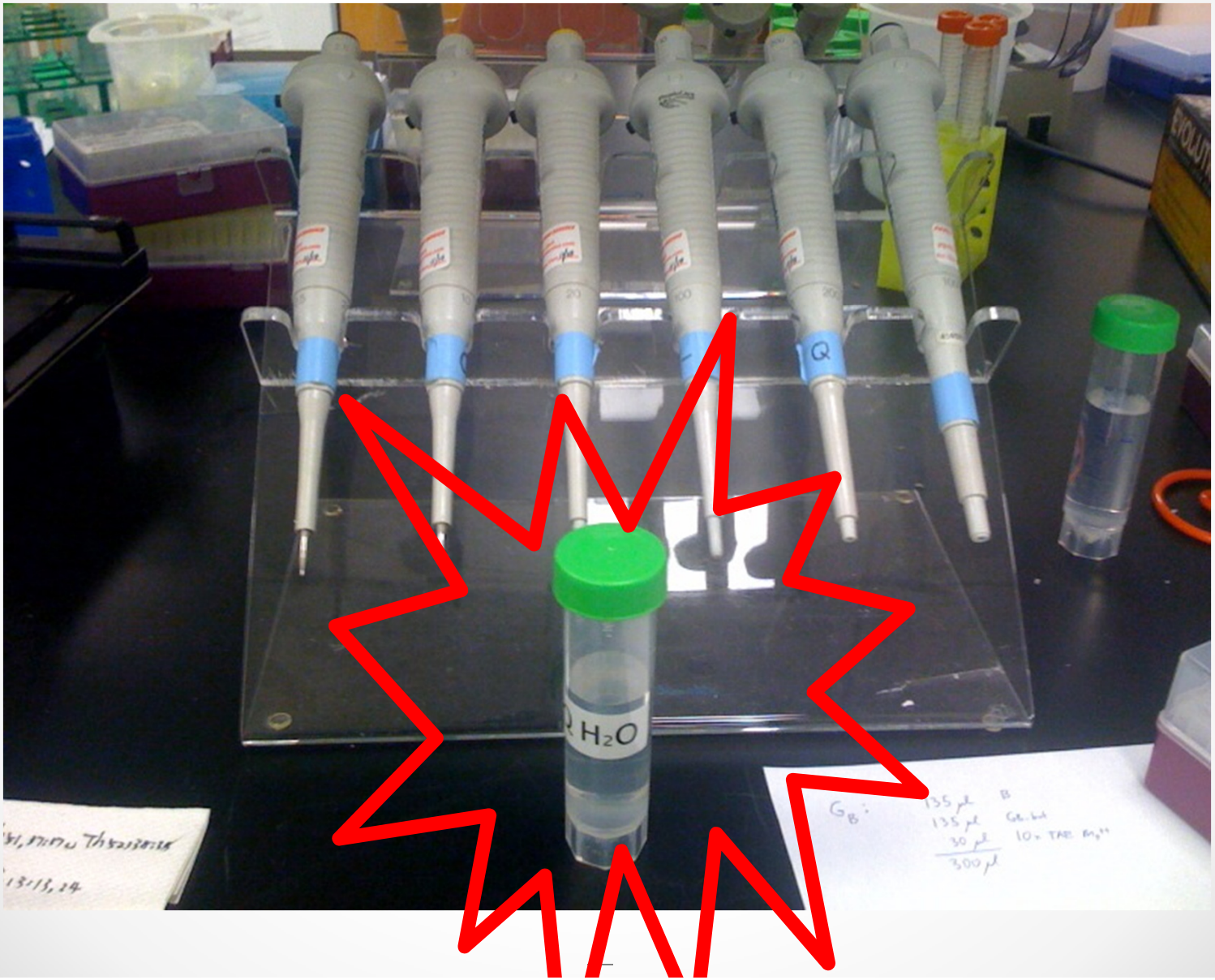


# Sequences to Molecules

# Molecules by Mail

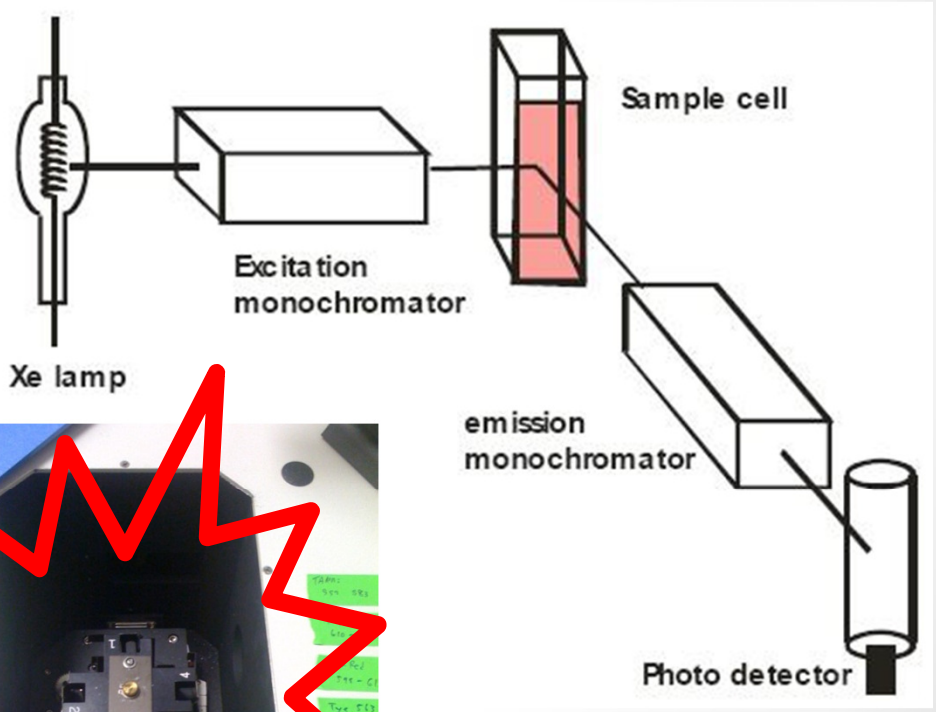


# Add Water

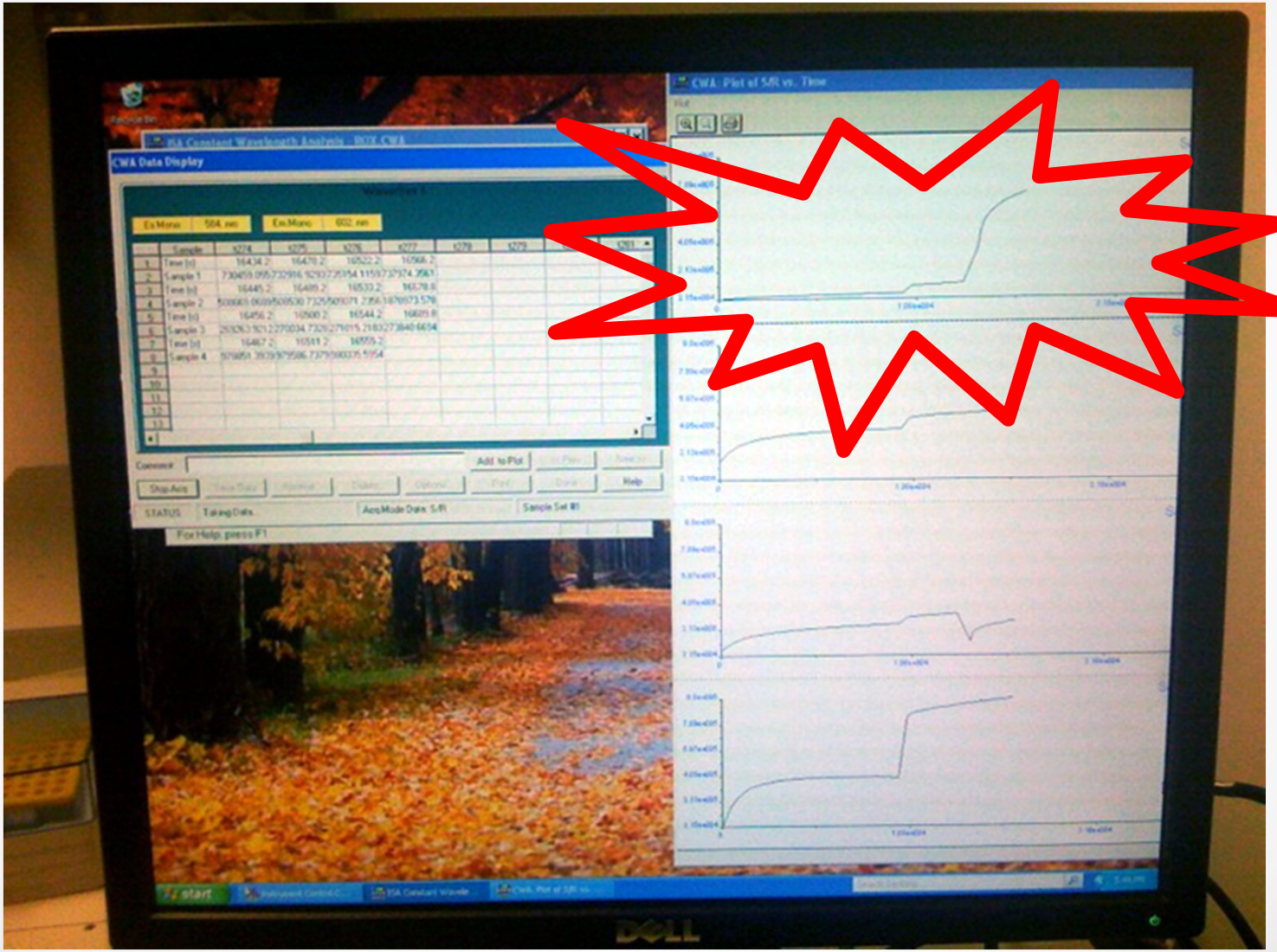


# Execution

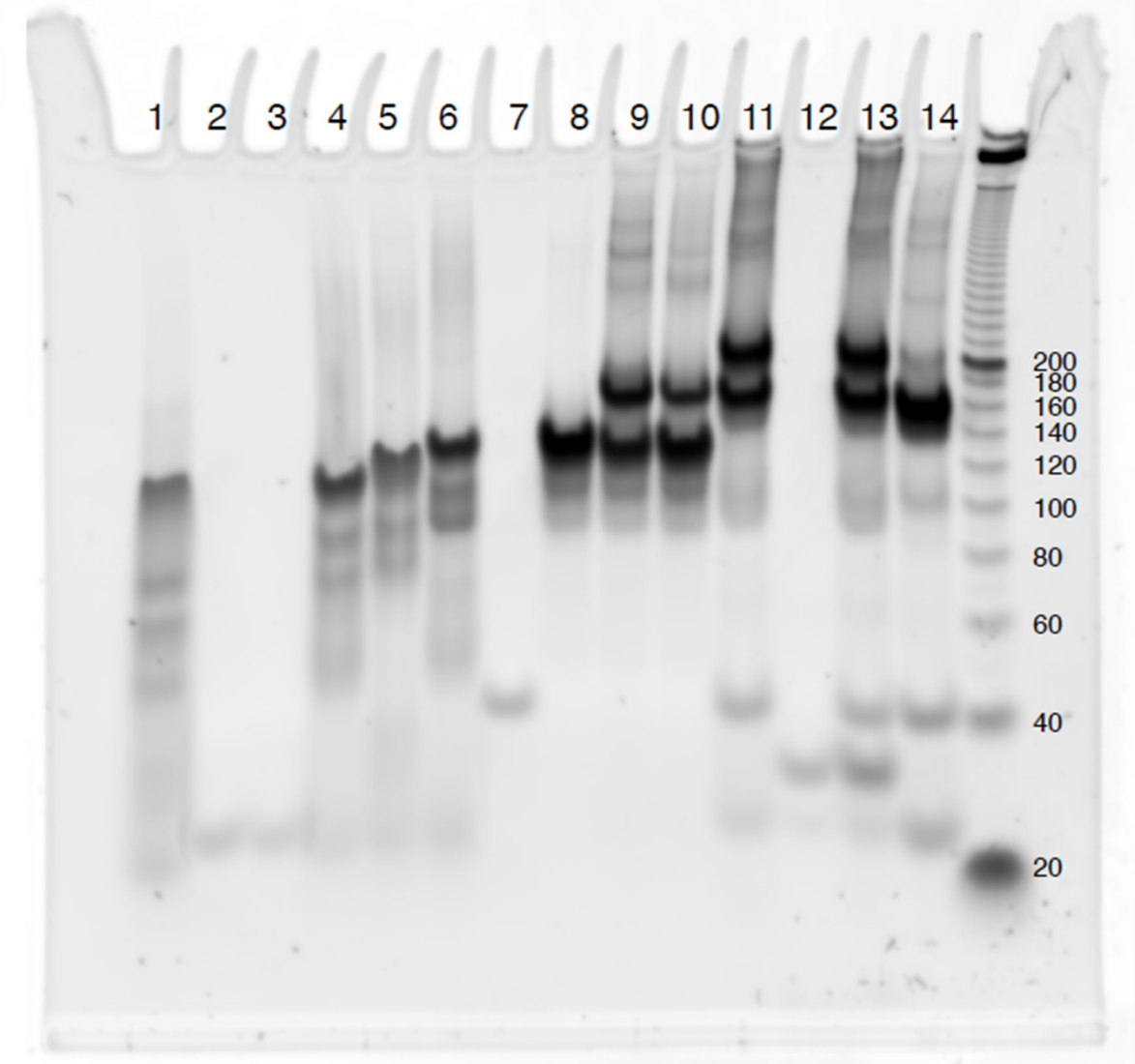
- Fluorescence is your ‘print’ statement

# Output

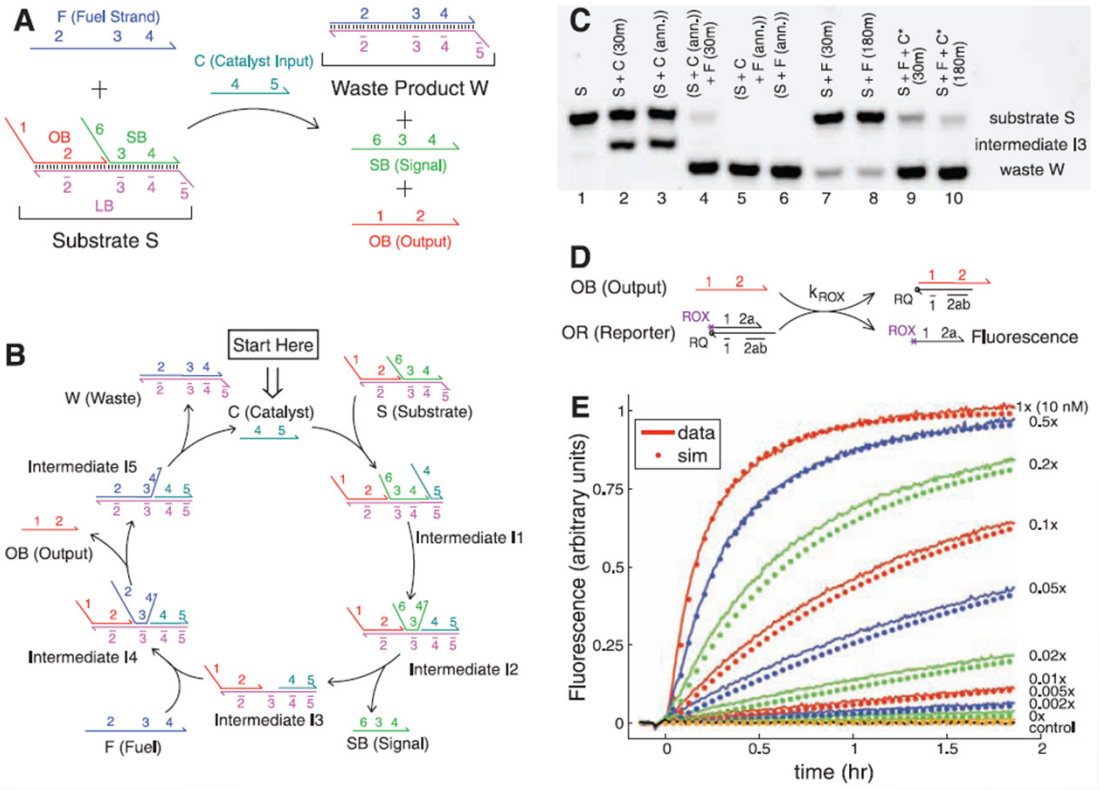

CWA Data Display
CWA: Plot of S/R vs. Time
Add to Plot
Stop Acq
STATUS
Taking Data
For Help, press F1
start
DELL

# Debugging



# Publishing!

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



# Conclusions

# Summary

- Molecular Structures
  - Hard to build… but they can build themselves!
- Molecular Languages
  - Natural and unnatural
  - Concurrent, quantitative
- Molecular Compilation
  - Molecular architectures, verification, optimization
- Molecular Programming
  - In silico, in vitro, in vivo…

# Acknowledgments


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