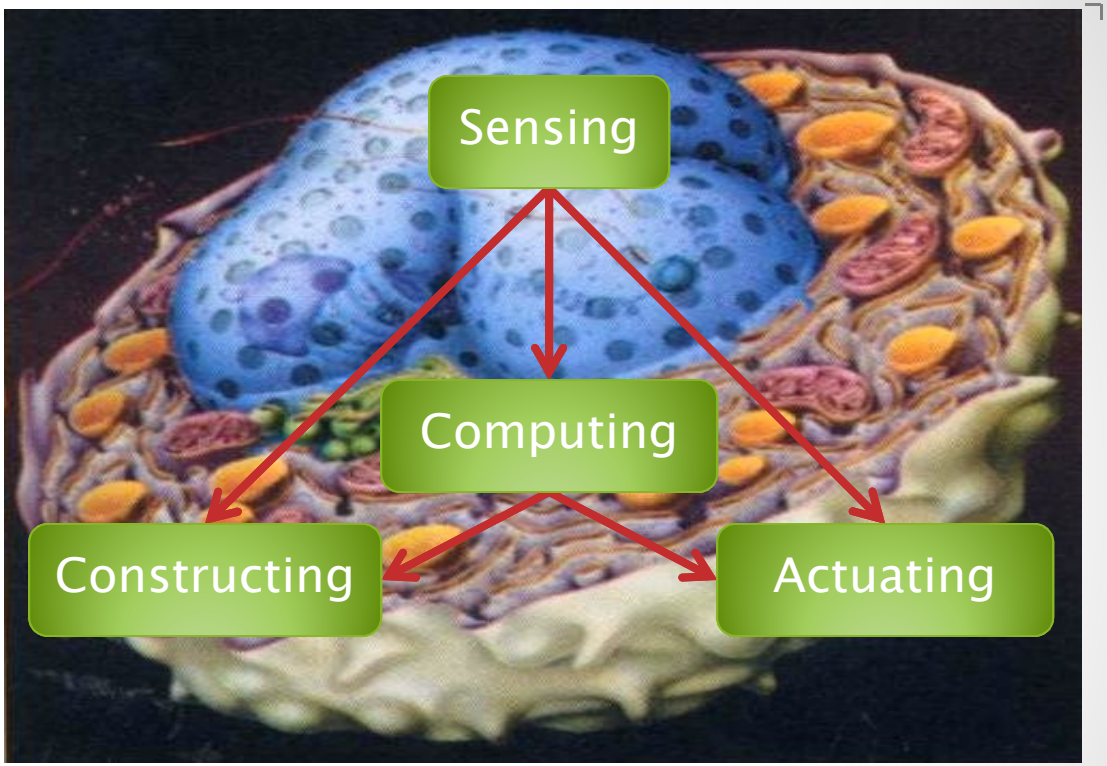
Two-Domain DNA Strand Displacement

Luca Cardelli
Microsoft Research

TAB Cambridge 2011-06-14
http://lucacardelli.name

Nanoscale Control Systems

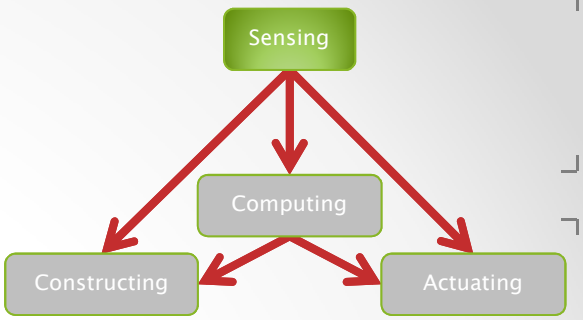
- Sensing
  - Reacting to forces
  - Binding to molecules
- Actuating
  - Releasing molecules
  - Producing forces
- Constructing
  - Chassis
  - Growth
- Computing
  - Signal Processing
  - Decision Making



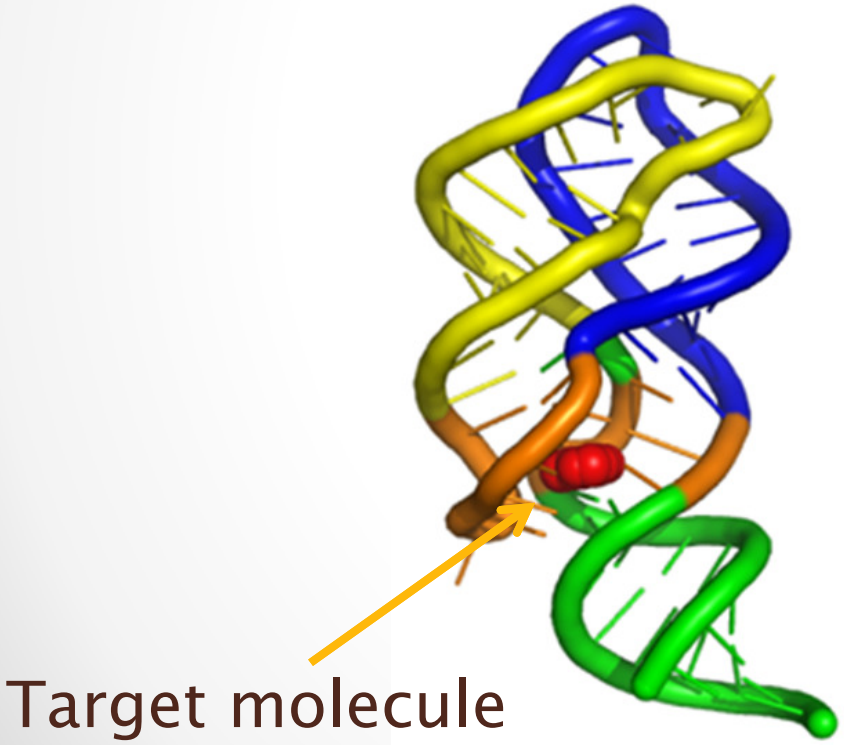
Nucleic Acids can do all this.

And interface to biology.

Sensing



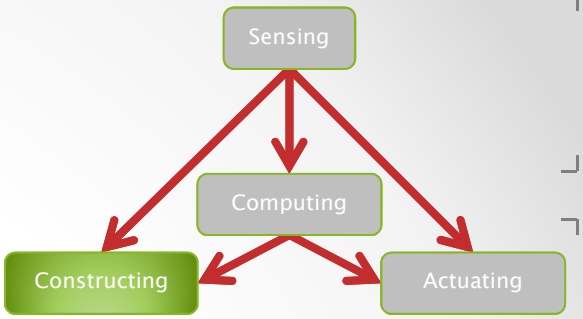
Aptamers: natural or artificially evolved DNA molecules that stick to other molecules (highly selectively).



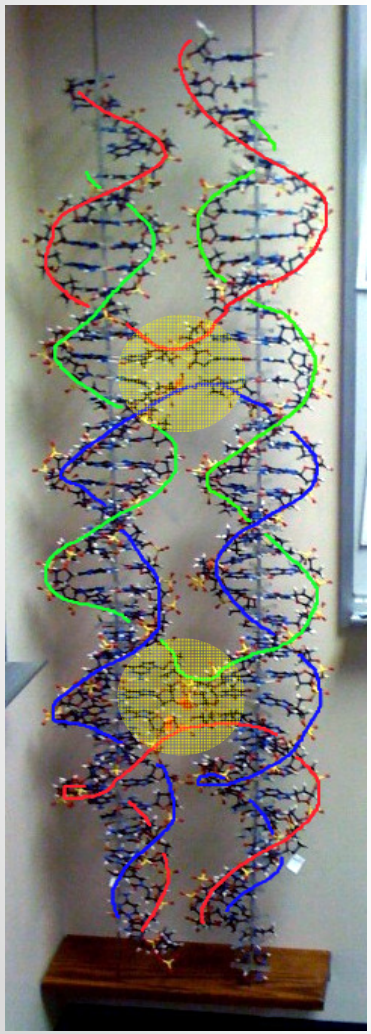
Adenine riboswitch aptamer

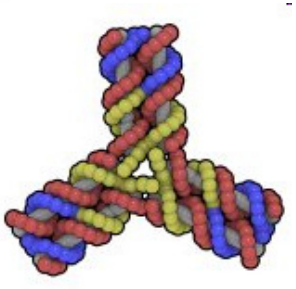
Structural basis for discriminative regulation of gene expression by adenine- and guanine-sensing mRNAs. Chem Biol. 2004 Dec;11(12):1729-41.

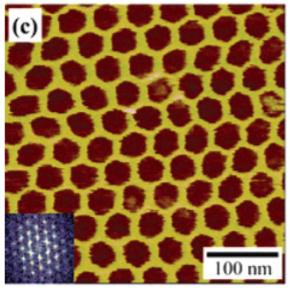
Constructing



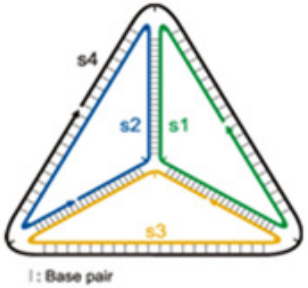
Crosslinking

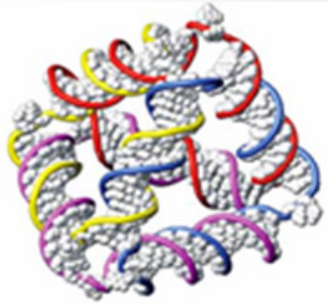






Chengde Mao, Purdue





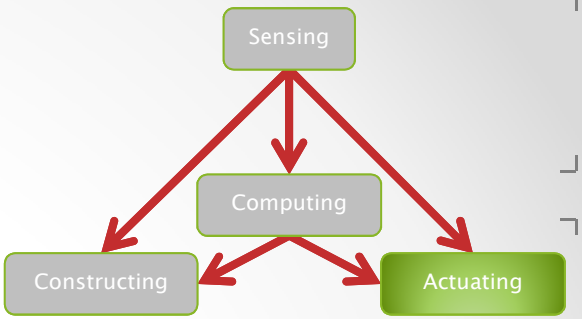
Andrew Turberfield, Oxford

Folding DNA into Twisted and Curved Nanoscale Shapes

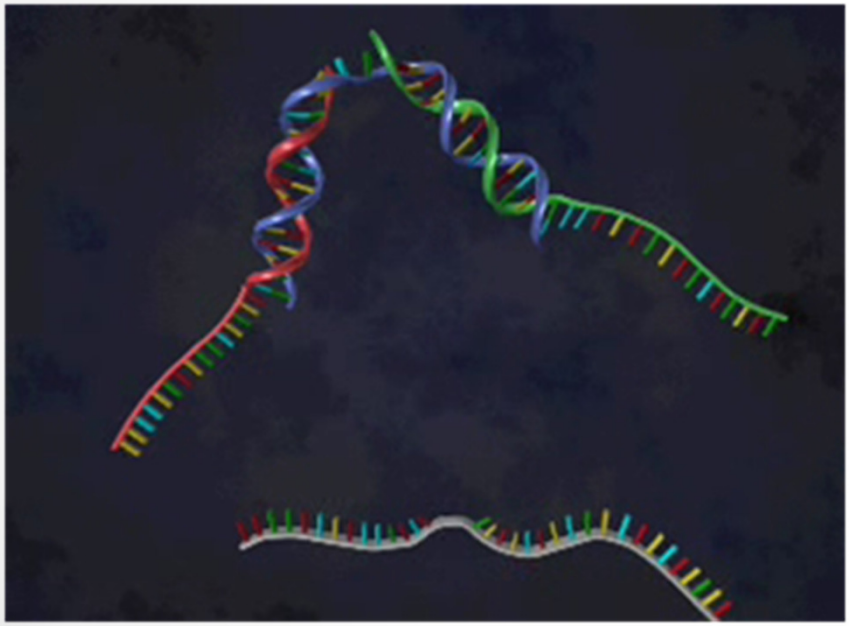
Hendrik Dietz, Shawn M. Douglas, & William M. Shih

Science, 325:725–730, 7 August 2009.

Actuating

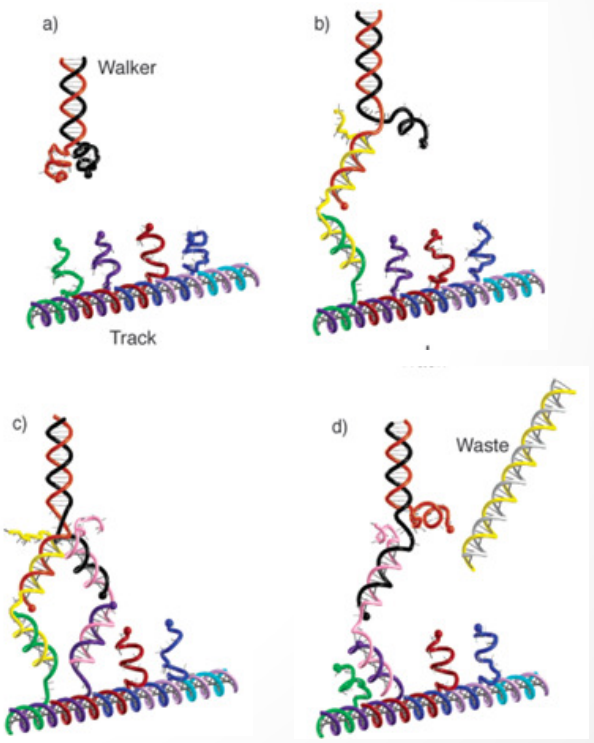


DNA tweezers

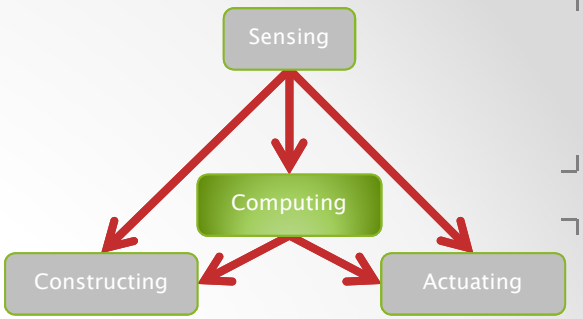


Bernard Yurke, Boise State

DNA walkers



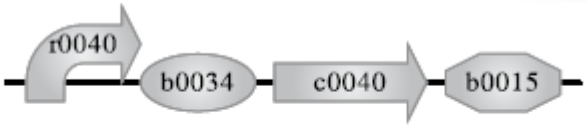
Computing



- 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’

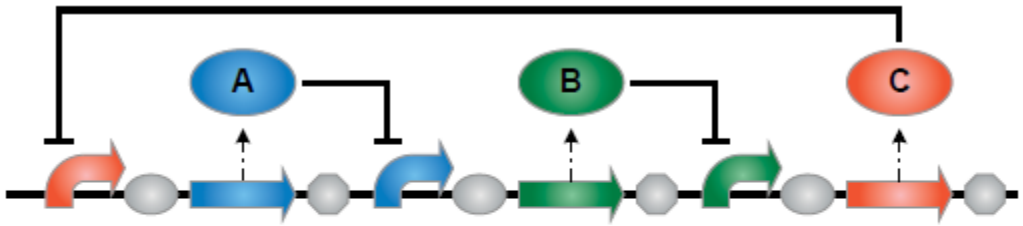
“Embedded” Computing

- 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]



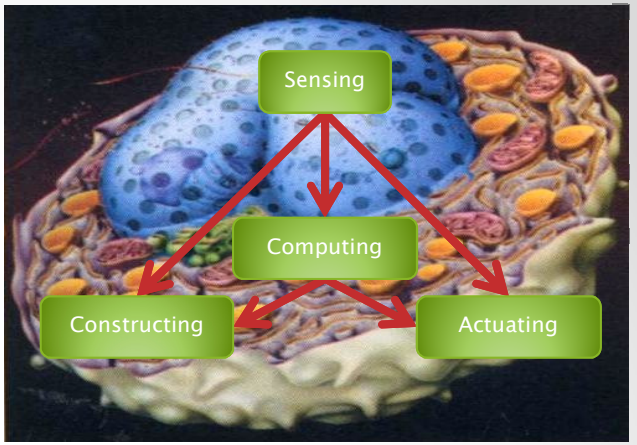

```
prom<neg(C)>; rbs; pcr<codes(A)>; ter;
prom<neg(A)>; rbs; pcr<codes(B)>; ter;
prom<neg(B)>; rbs; pcr<codes(C)>; ter
```

“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



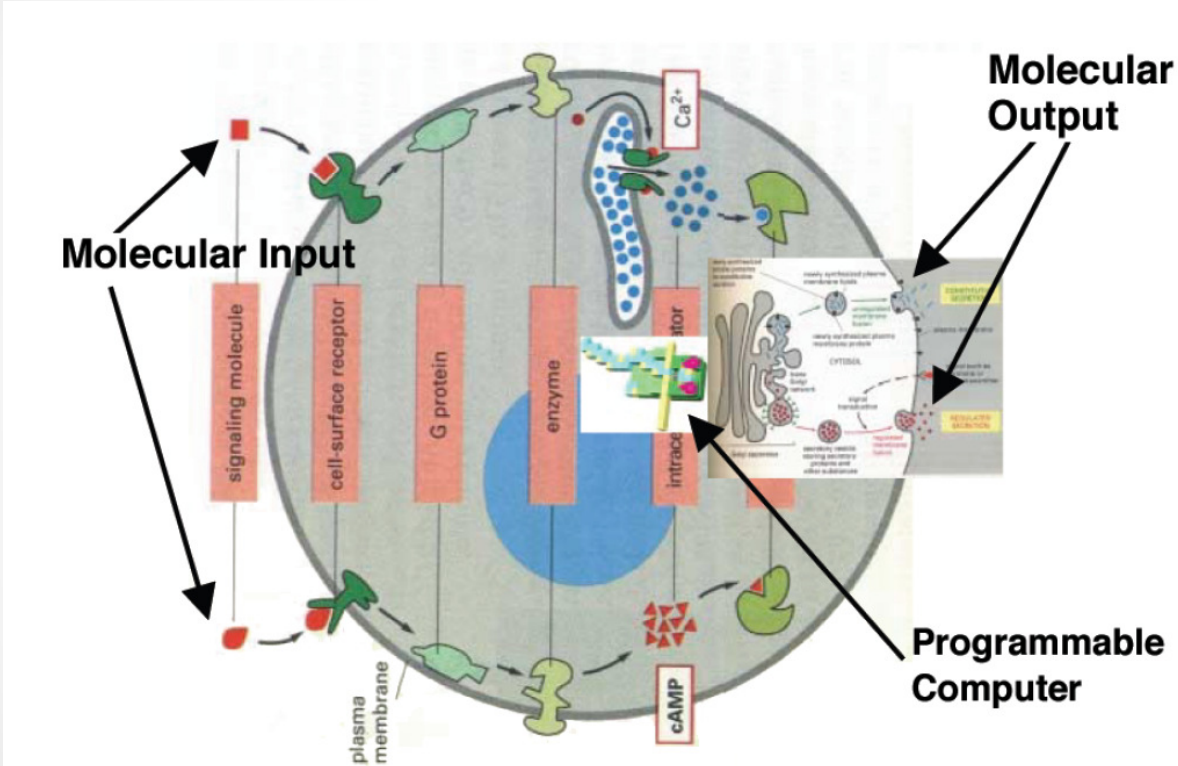


Fig. 1 *Medicine in 2050: "Doctor in a Cell"*

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.

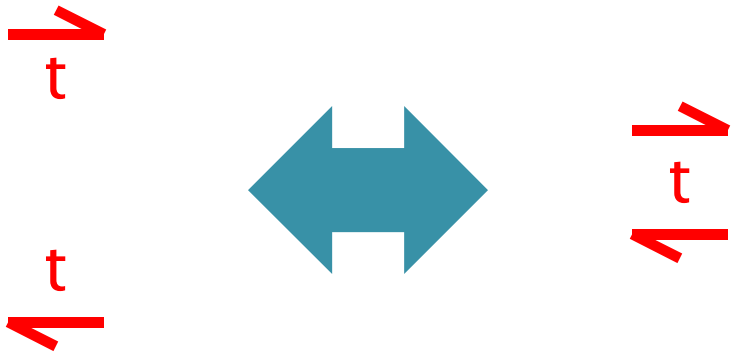
Domains

- 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.

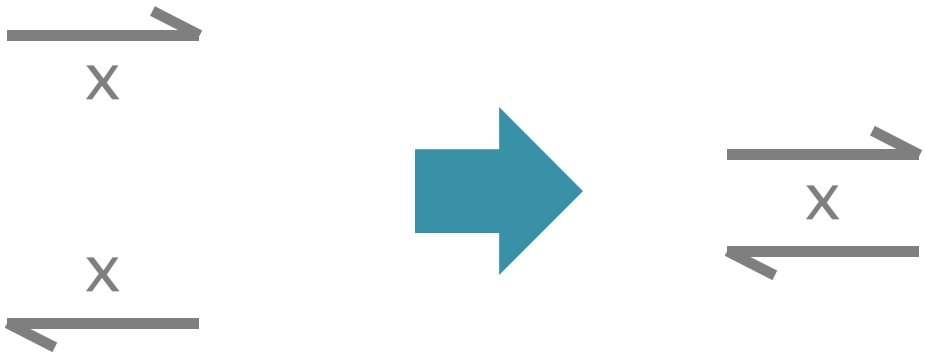
- Choosing domains (subsequences) that are suitably independent is a tricky issue that is still somewhat of an open problem (with a vast literature). But it can work in practice.

Short Domains



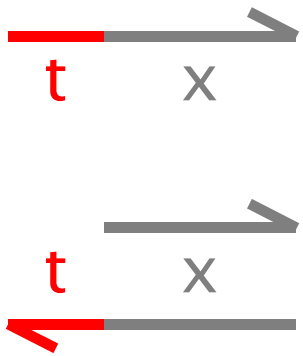
Reversible Hybridization

Long Domains



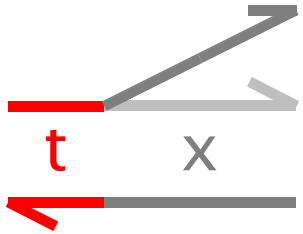
Irreversible Hybridization

Strand Displacement



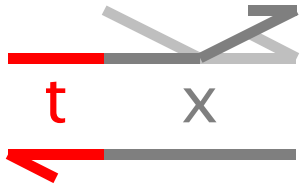
"Toehold Mediated"

Strand Displacement



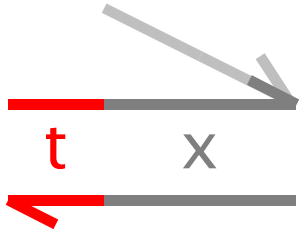
Toehold Binding

Strand Displacement



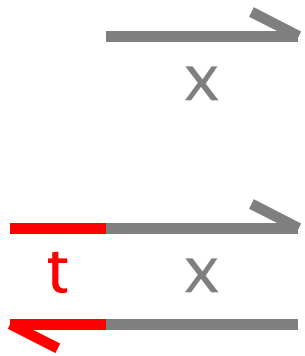
Branch Migration

Strand Displacement



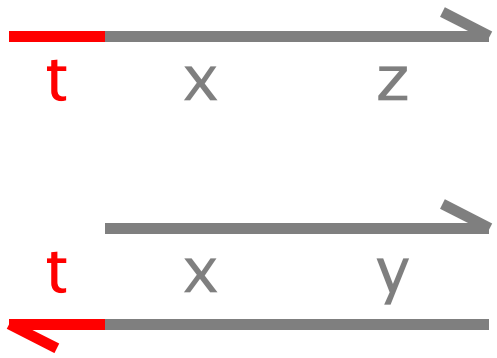
Displacement

Strand Displacement

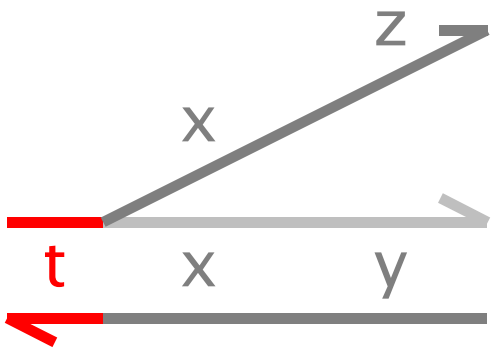


Irreversible release

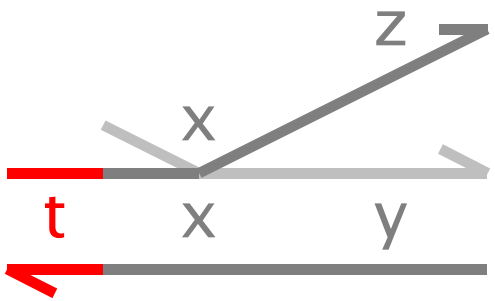
Bad Match



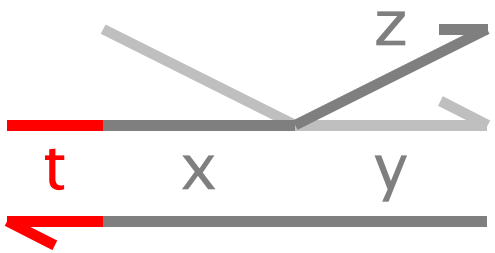
Bad Match



Bad Match



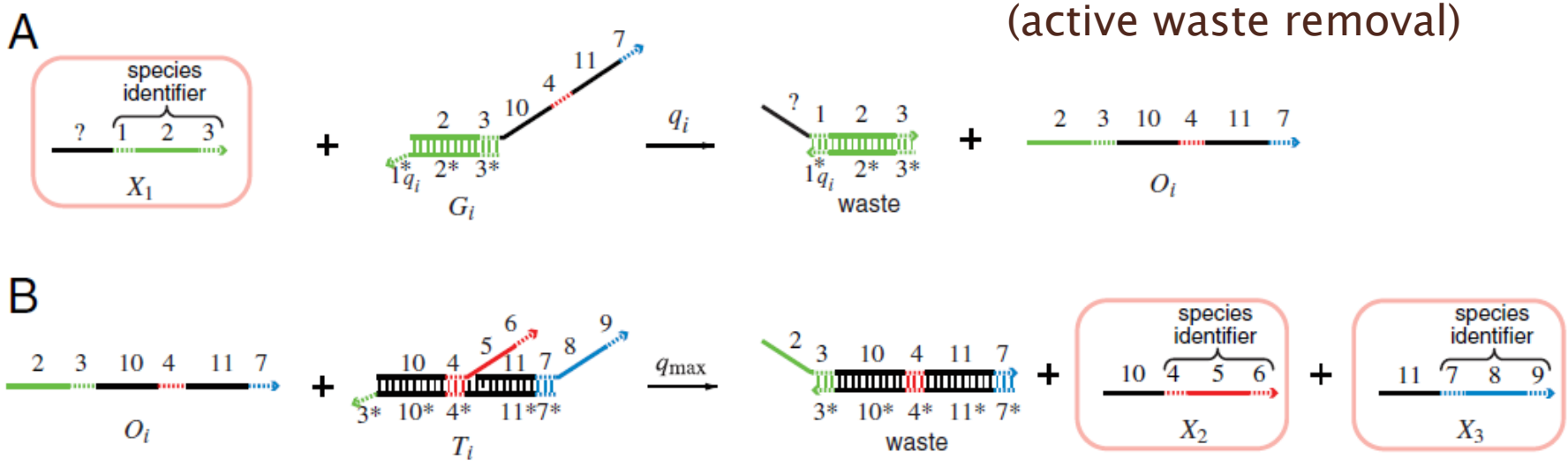
Bad Match



Cannot proceed
Hence will undo

Four-Domain Architecture

No “garbage collection” (active waste removal)



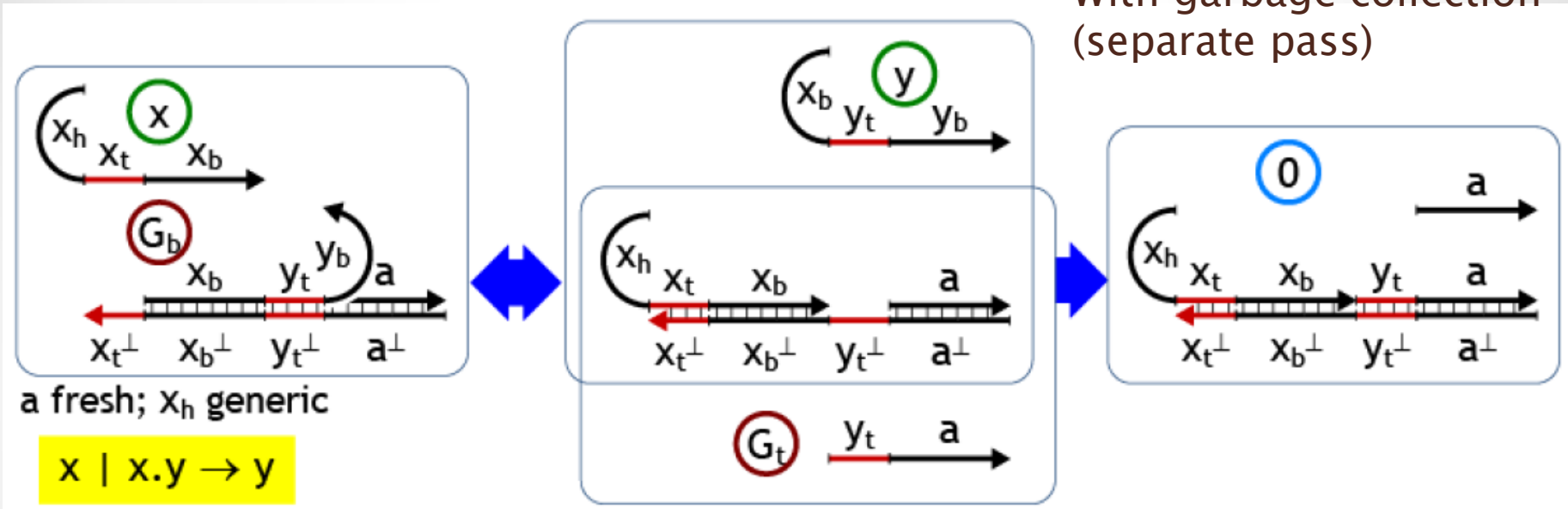
DNA as a universal substrate for chemical kinetics

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

PNAS | March 23, 2010 | vol. 107 | no. 12 | 5393–5398

Three-Domain Architecture

With garbage collection (separate pass)

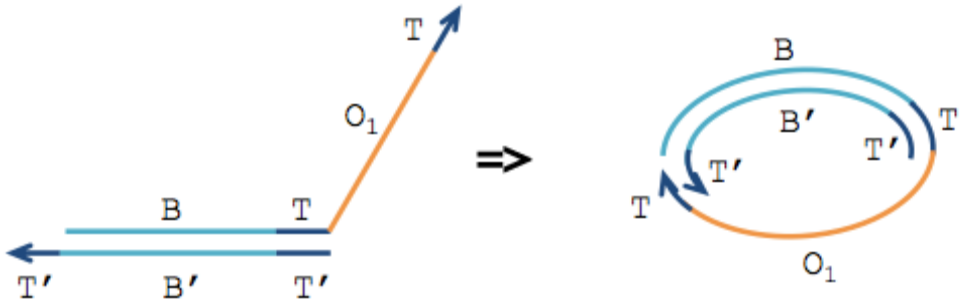


Strand Algebras for DNA Computing

Luca Cardelli

DNA Computing and Molecular Programming.
15th International Conference, DNA 15,
LNCS 5877, Springer 2009, pp 12–24.

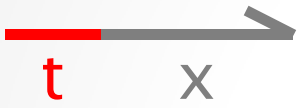
“Lulu’s Trouble”



(from D.Soloveichik)

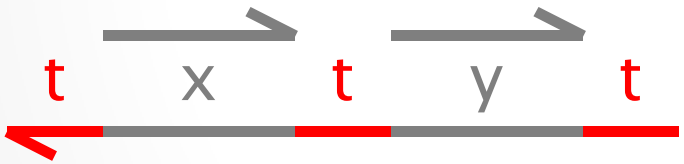
Two-Domain Architecture

- Signals: 1 toehold + 1 recognition region



Garbage collection
"built into" the gates

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

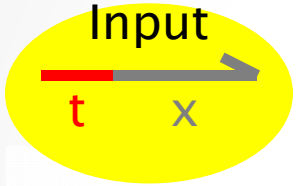


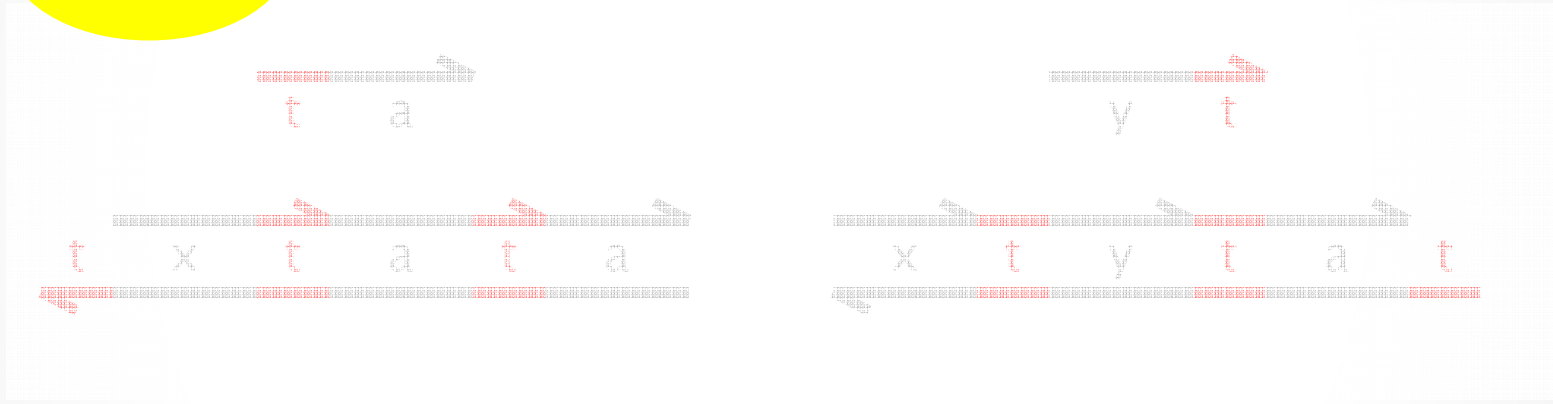
Two-Domain DNA Strand Displacement

Luca Cardelli

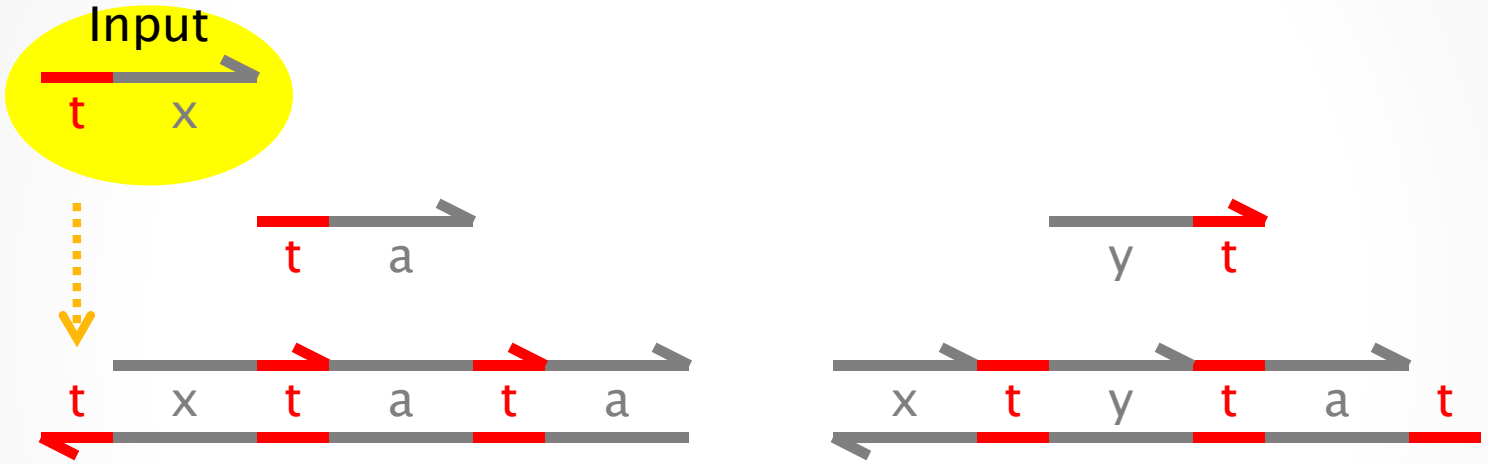
In S. B. Cooper, E. Kashefi, P. Panangaden (Eds.):
Developments in Computational Models (DCM 2010).
EPTCS 25, 2010, pp. 33-47. May 2010.

Transducer x→y





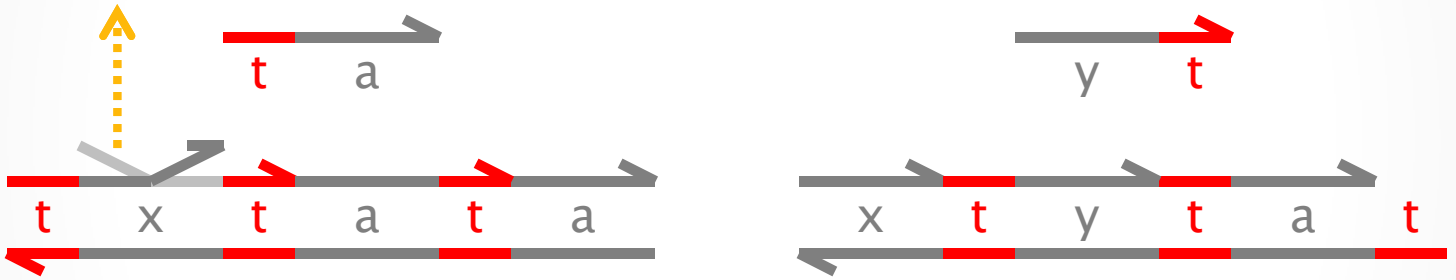
Transducer x→y



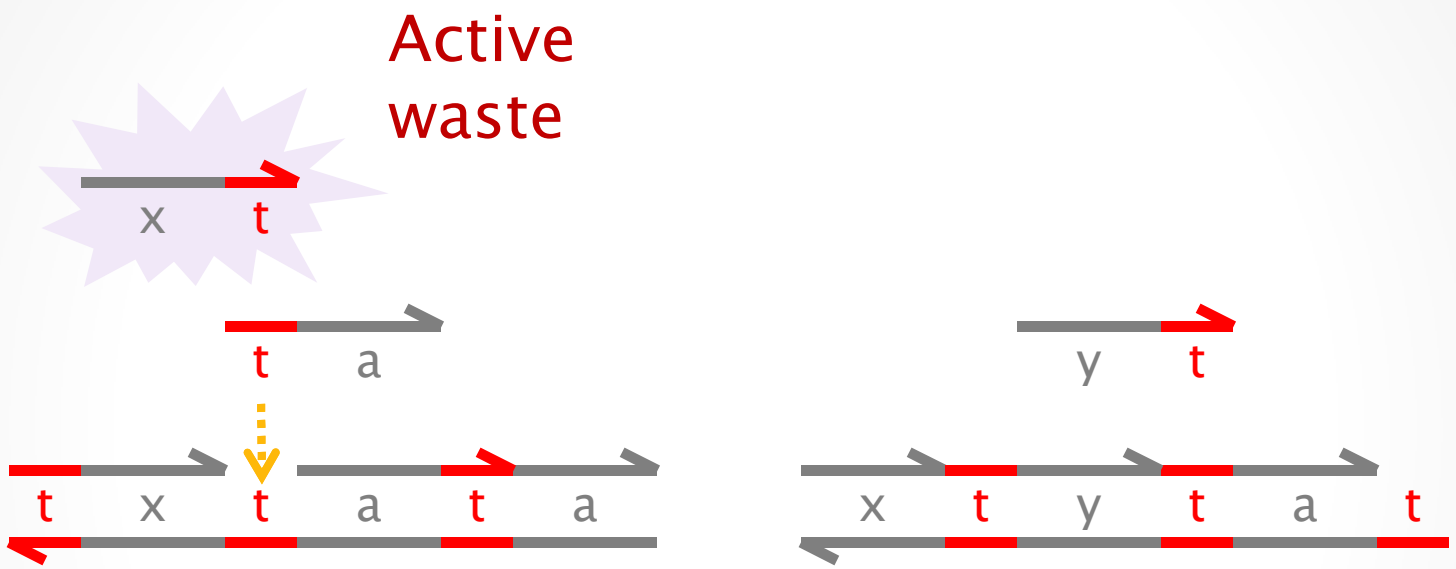
Built by self-assembly!

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

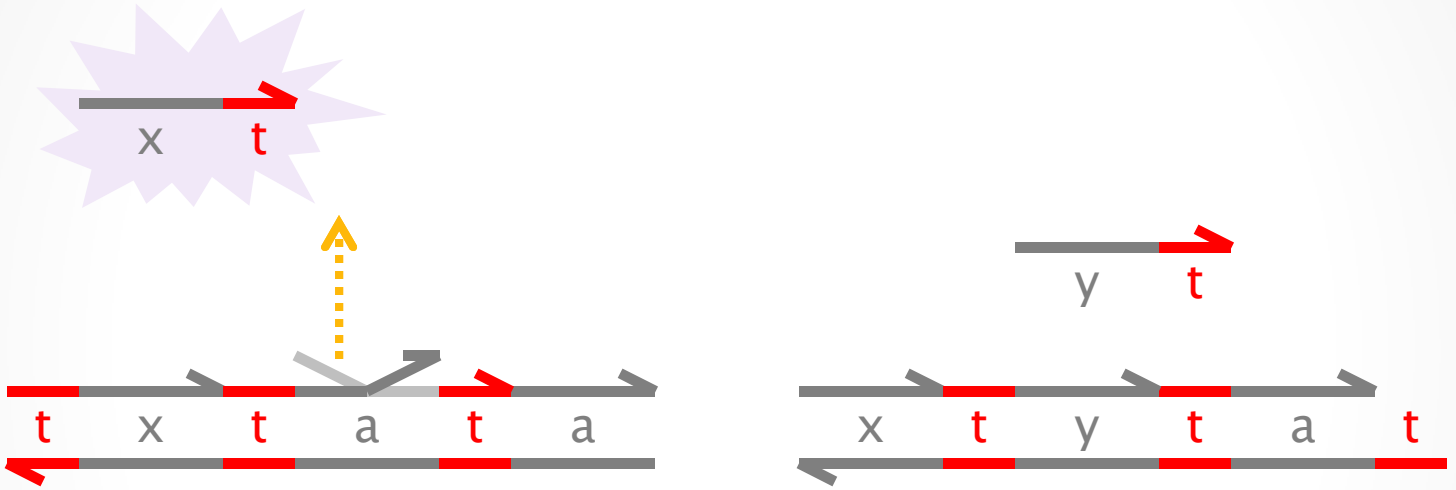
Transducer x→y



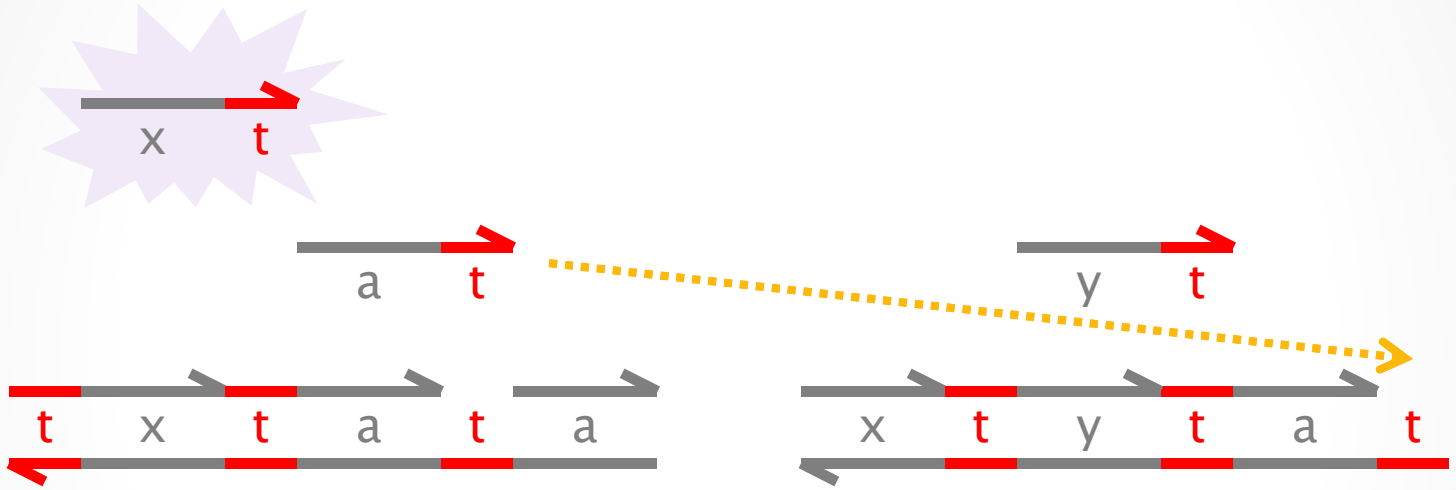
Transducer x→y



Transducer x→y

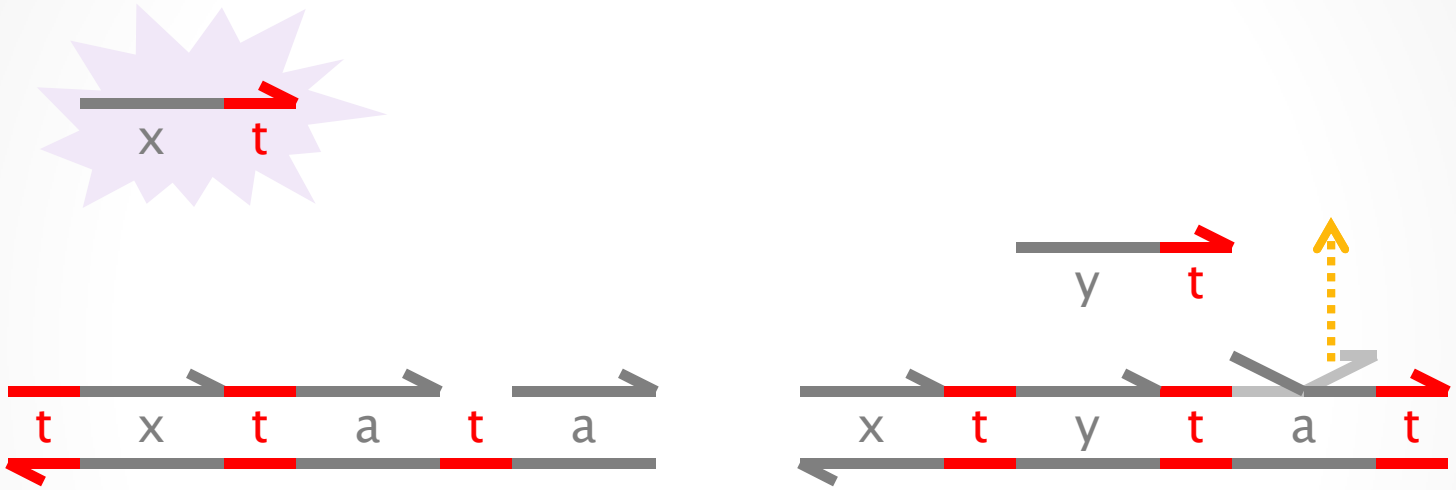


Transducer x→y

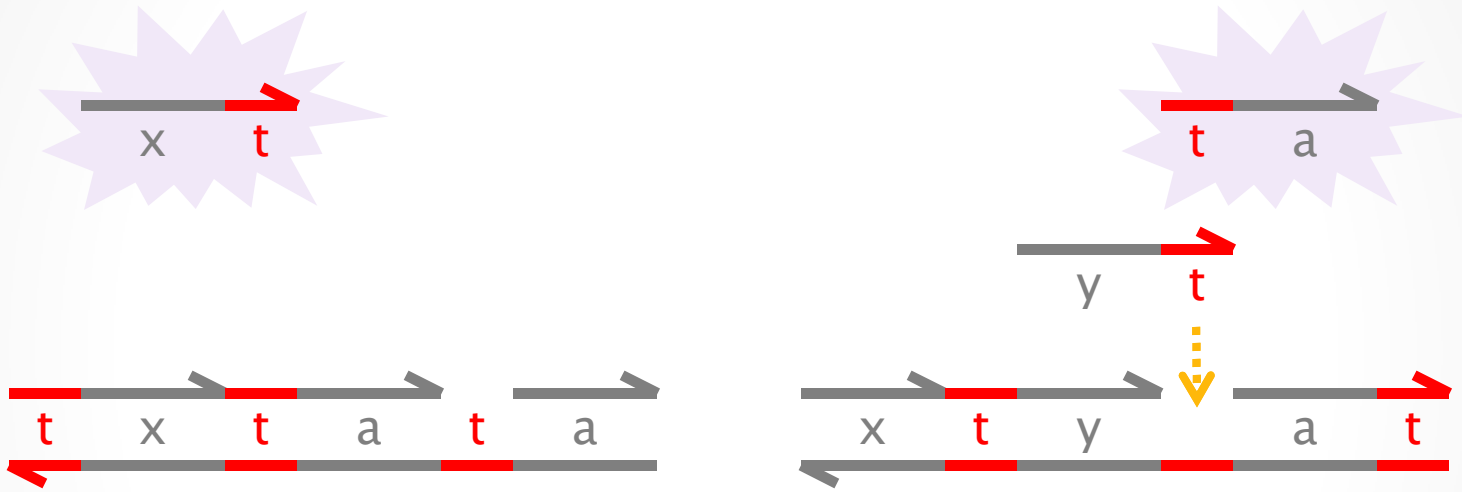


So far, a tx *signal* has produced an at *cosignal.*
But we want signals as output, not cosignals.

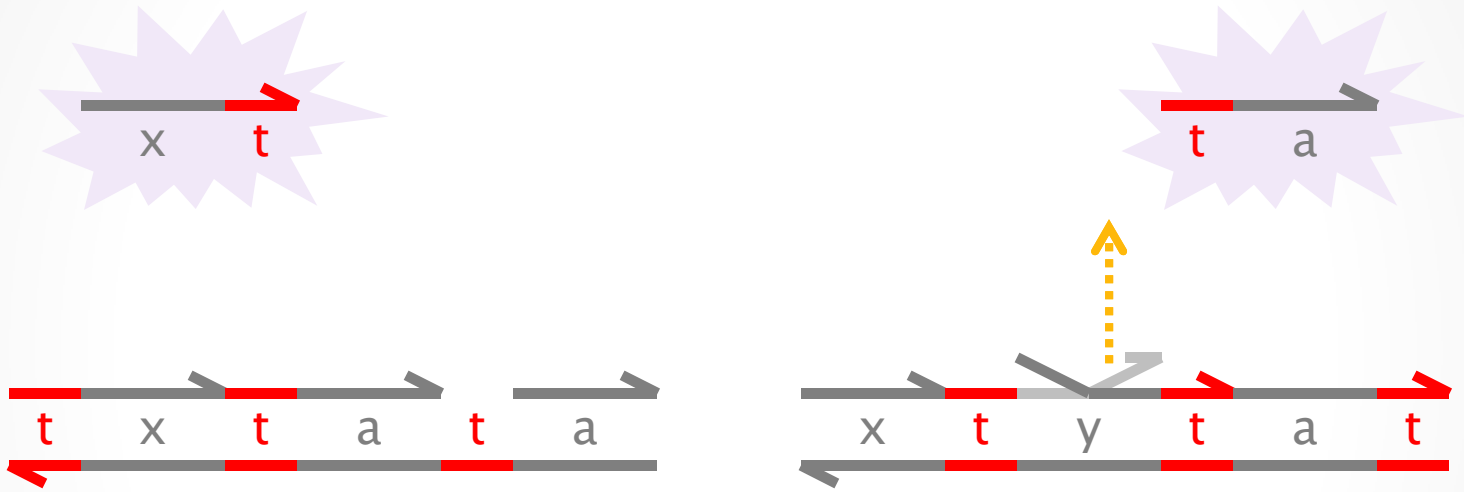
Transducer x→y



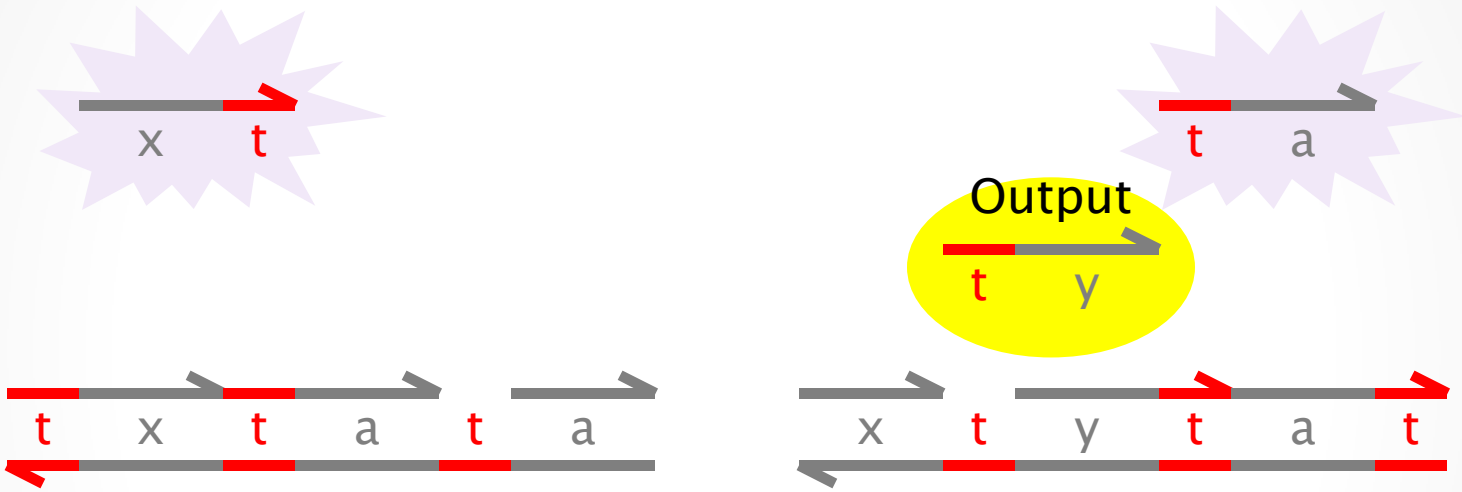
Transducer x→y



Transducer x→y



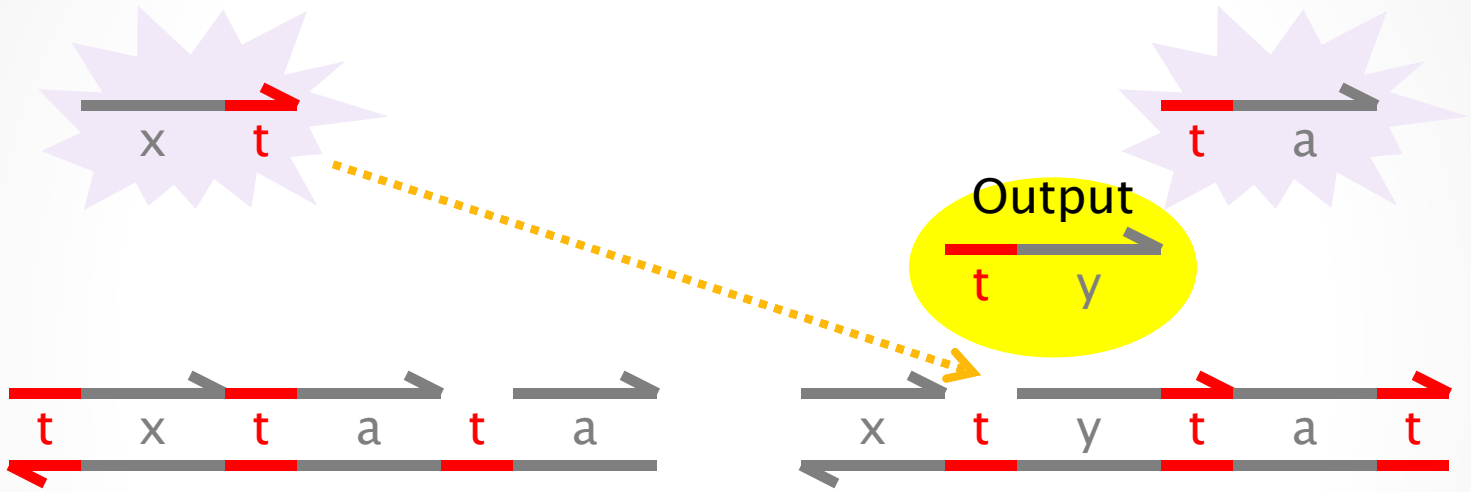
Transducer x→y



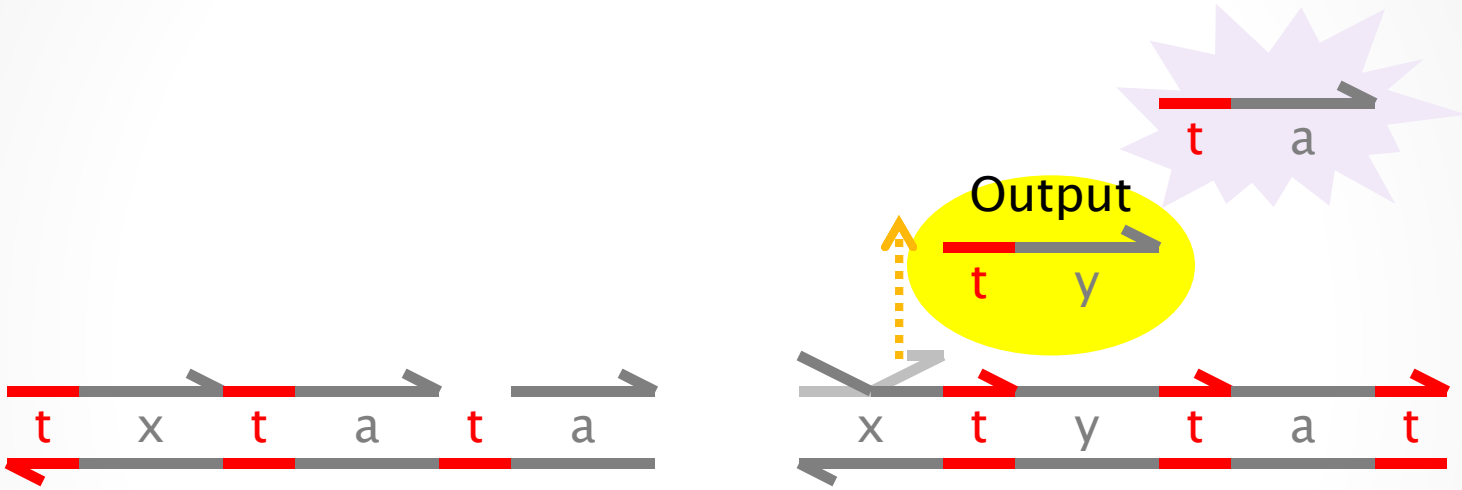
Here is our output ty *signal*.

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).

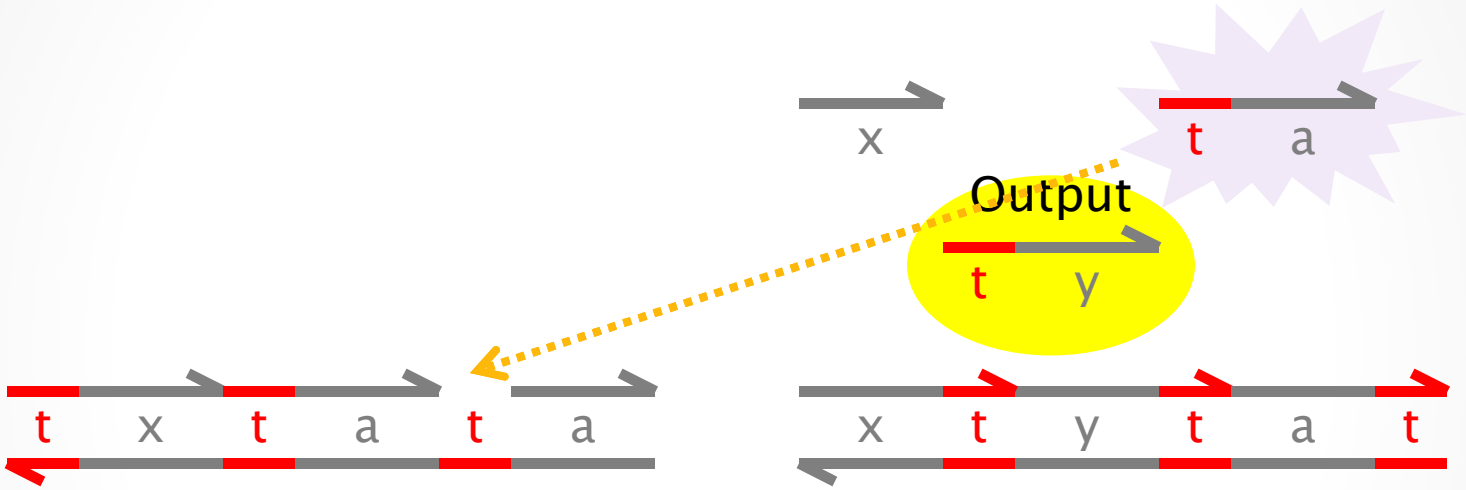
Transducer x→y



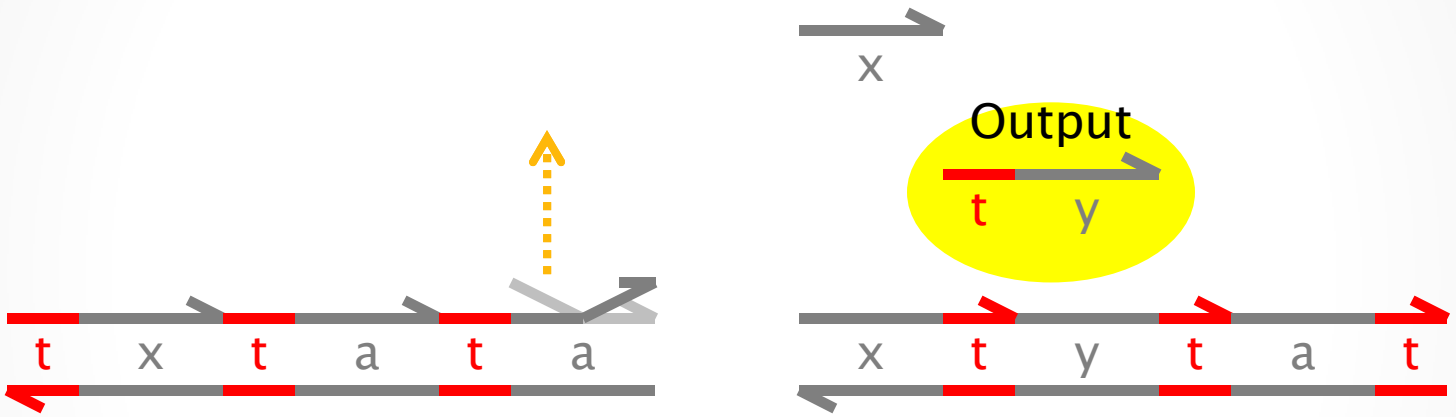
Transducer x→y



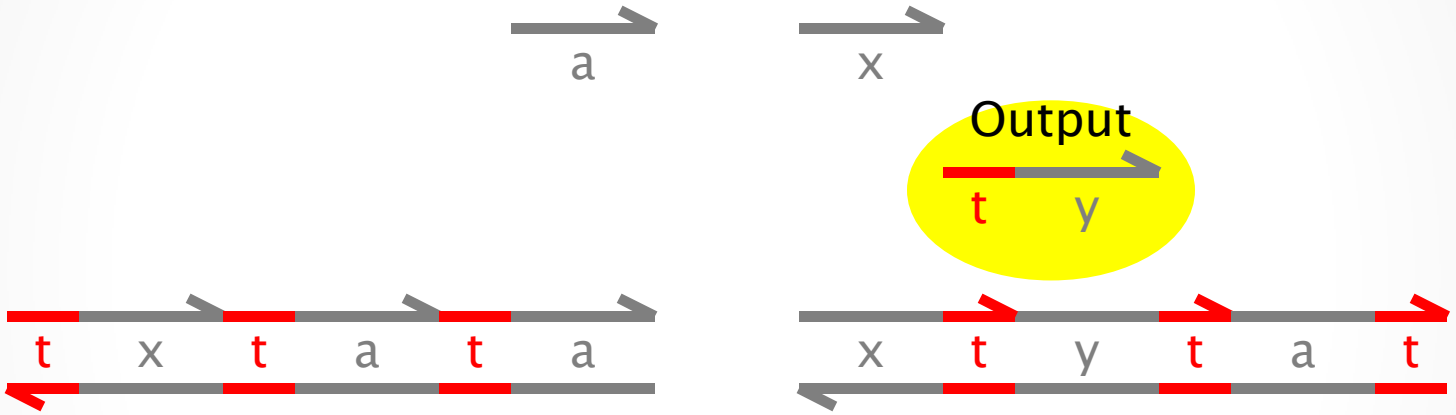
Transducer x→y



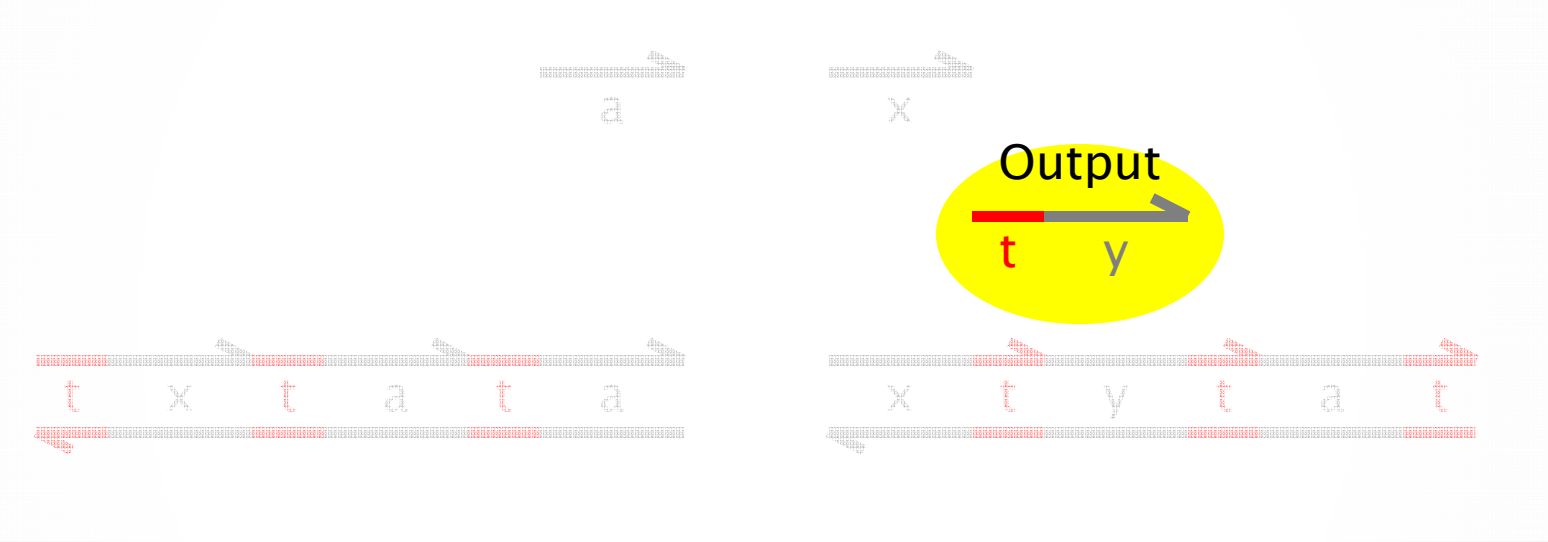
Transducer x→y



Transducer x→y



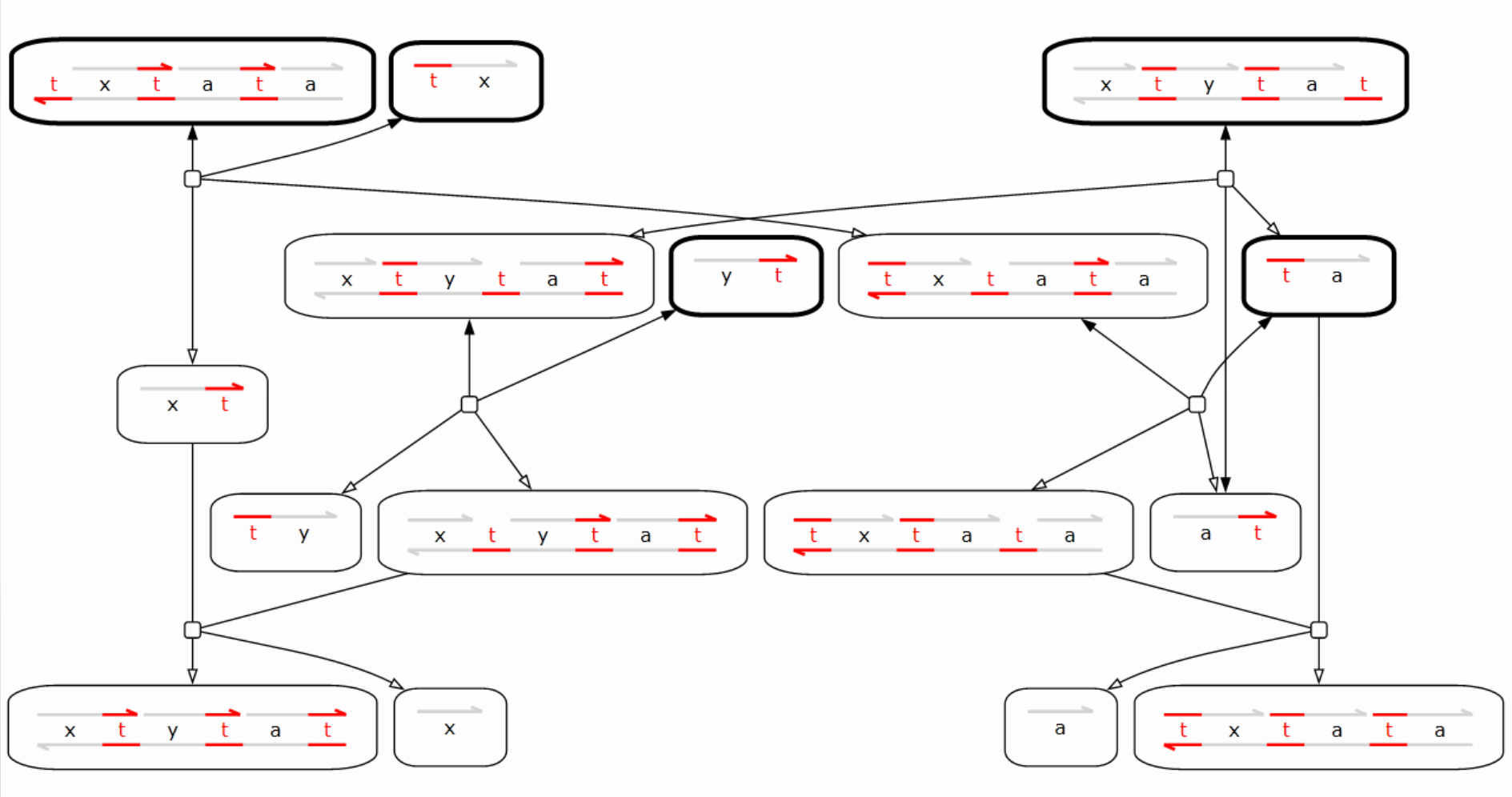
Transducer x→y



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.

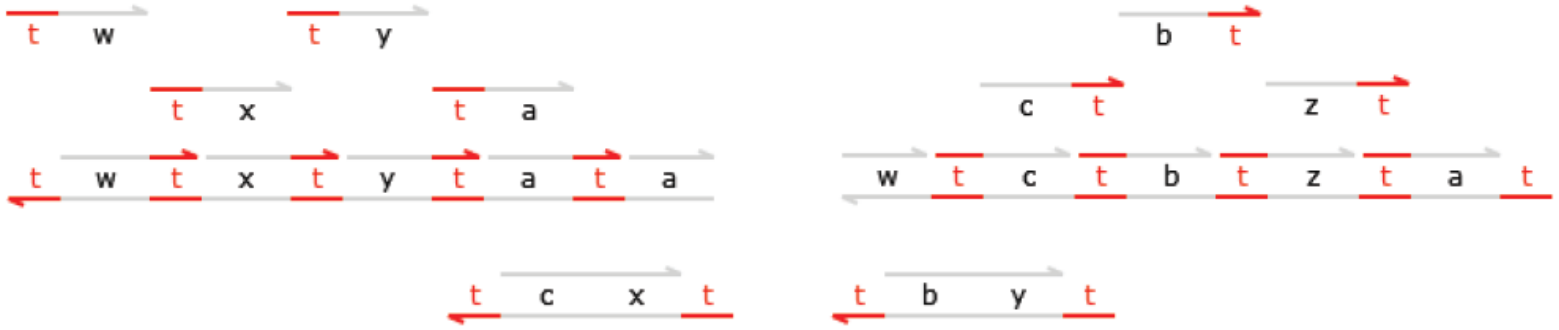


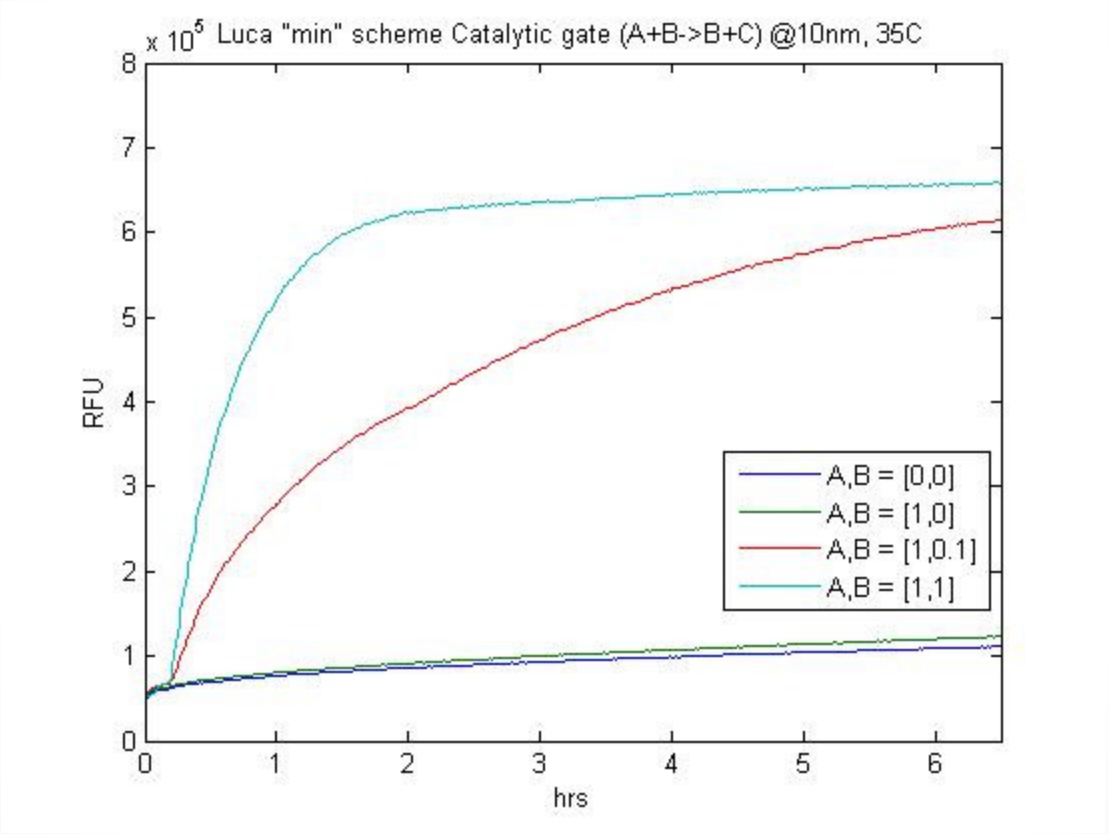
Figure 9: 3-Join $J_{wxyz} \mid tw \mid tx \mid ty \longrightarrow tz$: initial state plus inputs tw, tx, ty.

Animations

- Animations

Experiments

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



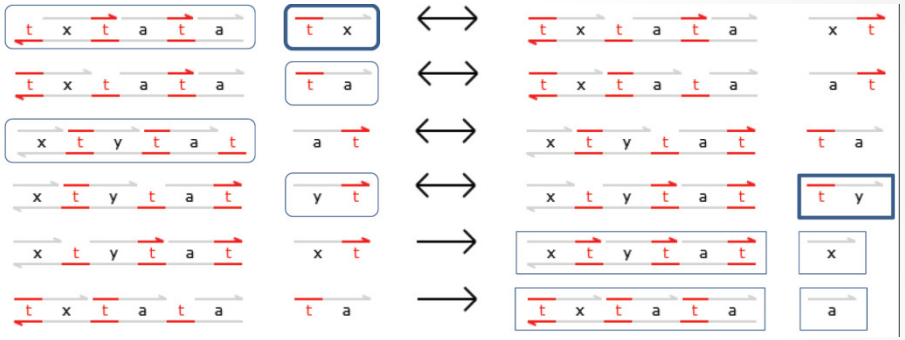
Matt Olson and Georg Seelig, U.Washingon.

Experimental Challenges

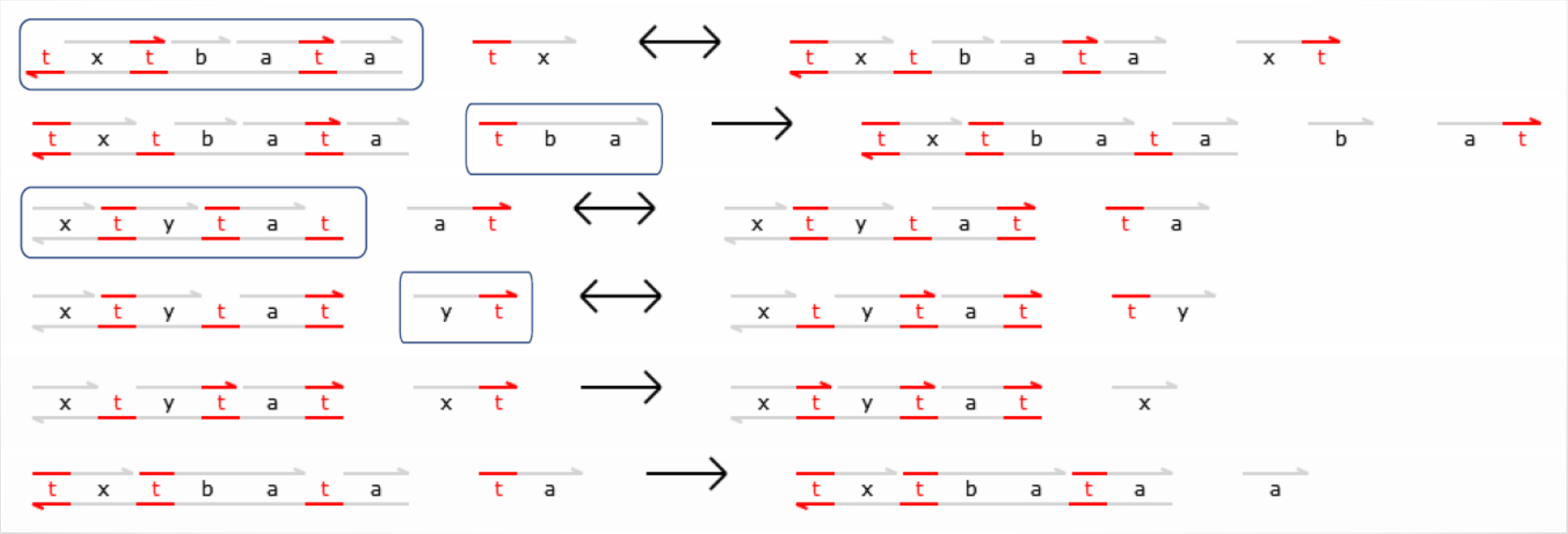
- Quality of synthetic DNA
  - Chemical synthesis is limited in length and quality.
  - Two-domain scheme enables bacterial synthesis
    - Followed by enzyme digestion to introduce 'nicks'
    - Or nicking by a photosensitive artificial backbone
- Circuit optimization
  - Coming up with *simpler* systems for simplified experiments (shorter DNA sequences and smaller number of species)
  - Coming up with *faster* systems (more irreversible operations, and garbage collection).

Ex. Irreversible Output

Standard Transducer



Irreversible-output Transducer



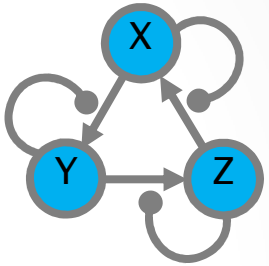
Ex. Oscillator

- Three autocatalytic reactions

  X + Y -> Y + Y

  Y + Z -> Z + Z

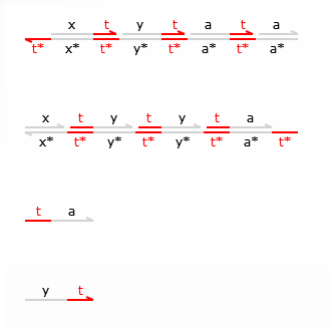
  Z + X -> X + X

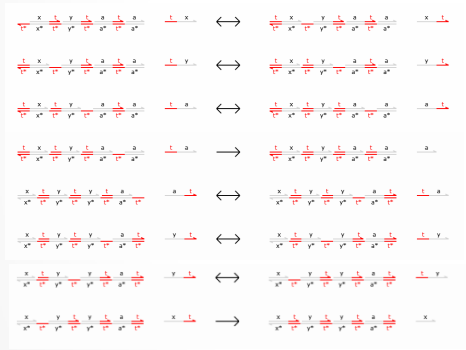


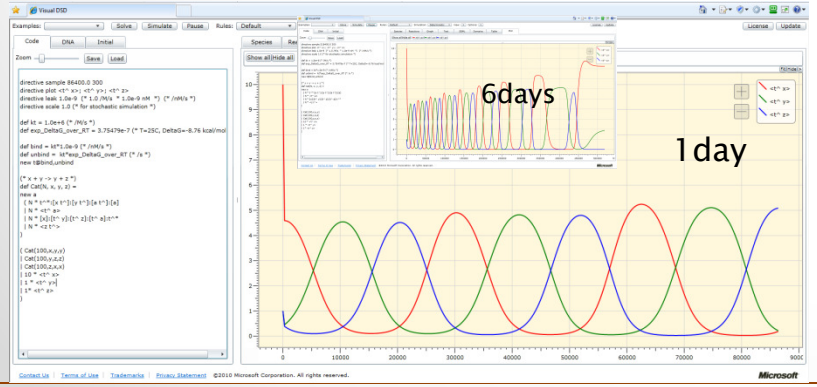
- This is a closed system
  - (Or perhaps a performance-critical subsystem)
  - Idea: use an optimized "internal" protocol that preserves the "public" interface of the system
    - Use extra toeholds (instead of private domains) to connect the two halves of each gate (saving a domain).
    - Trick:1extra toehold (instead of 3) is sufficient: there is a unique variable (x,y,z) connecting the two halves of gates.

Optimized Oscillator

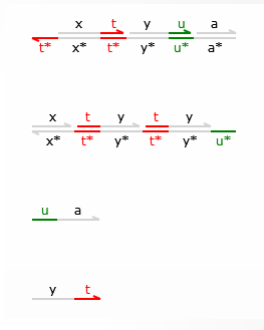
Original

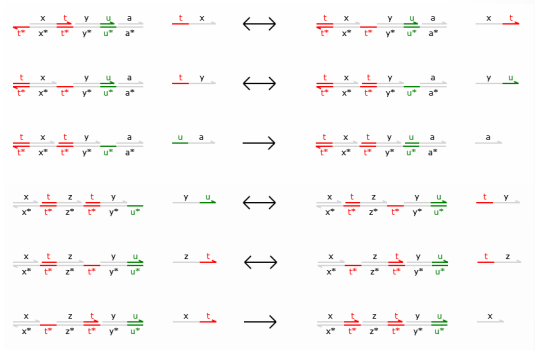


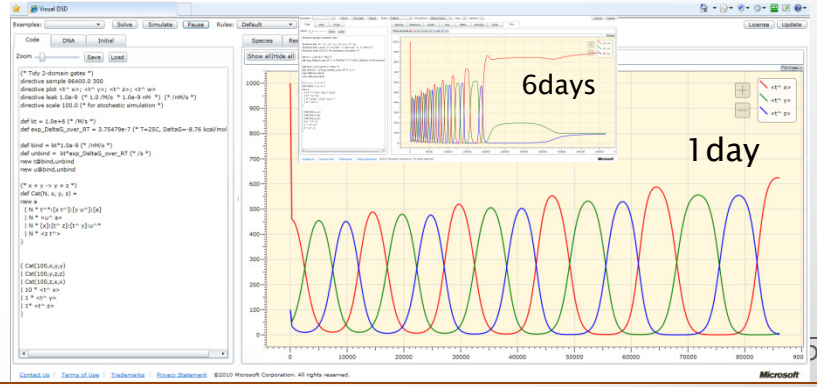




Optimized







Ex. Approximate Majority

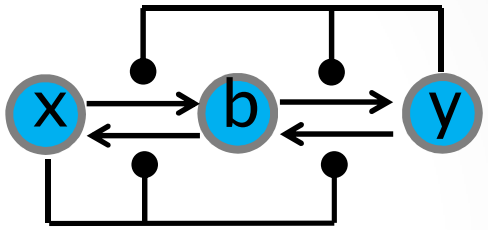
- Four catalytic/autocatalytic reactions

  x + y -> y + b

  y + x -> x + b

  b + x -> x + x

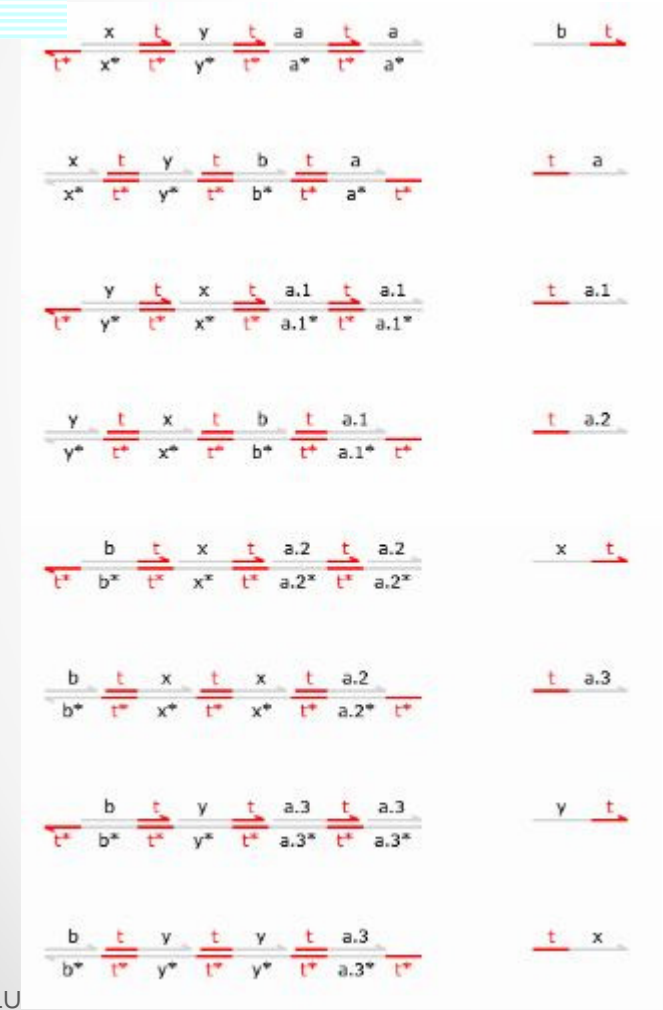
  b + y -> y + y



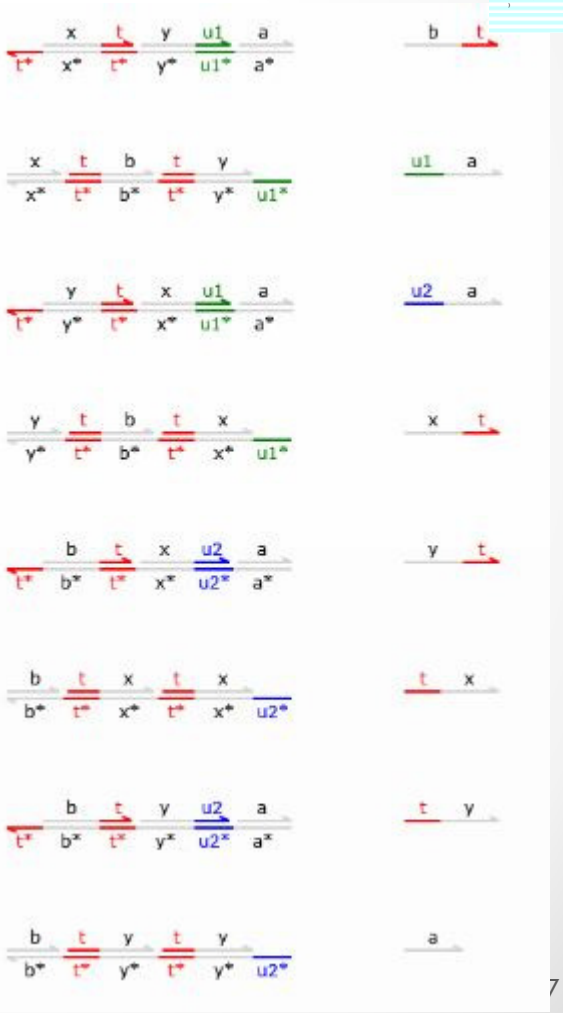
- This is a closed system
  - (Or perhaps a performance-critical subsystem)
  - Same idea.
    - But now 1 extra toehold is not sufficient (x and y catalyze two reactions). However 2 (instead of 4) toeholds are sufficient.

Optimized AM (Yuan-Jyue Chen)

Original



Optimized



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

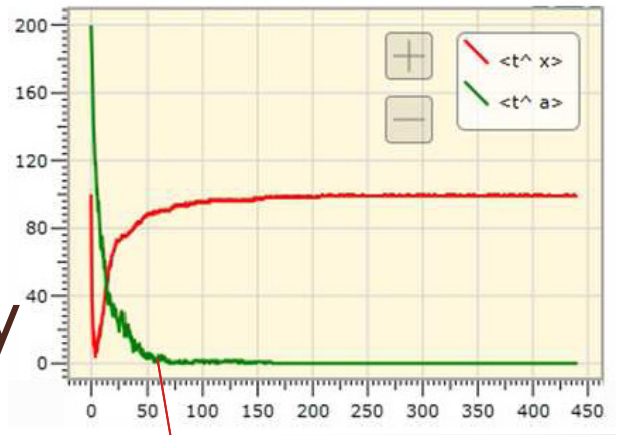
- The spec of a transducer:

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

  - 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

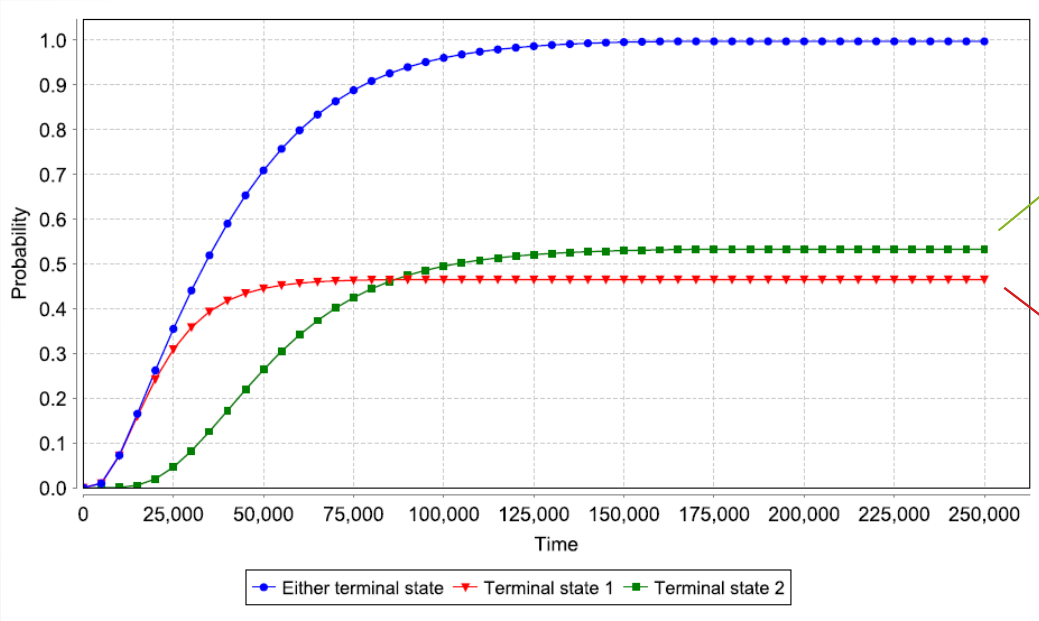
- Let a be the private transducer domain, but let's share it between x.y and y.x
- Interference: $x._a y \mid y._a x \mid x \not\rightarrow^{\forall} x$
- But still: $x._a y \mid y._a x \mid x \mid y \rightarrow^{\forall} x \mid y$
- A large population of such gates in practice does not deadlock easily.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

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