Computing with Molecules

Luca Cardelli
Microsoft Research

Development in Computational Models
Cambridge 2012-06-17
http://lucacardelli.name

A computational model

- Molecular Soups
  - Molecules randomly collide and can change state or composition.
    - Can we compute with that?
  - Based on the classical atomic theory of matter
    - with Brownian motion
    - nothing quantum here

- Related to:
  - "In small numbers" (macroscopic systems)
    - Process Algebra
    - Petri Nets
  - "In large numbers" (microscopic systems)
    - Population Protocols [Angluin et al.]
    - Amorphous Computing [Abelson et al.]
    - Swarm Intelligence – Ant Colonies
    - Epidemiology
    - Chemistry

A notion of algorithm

- Data as populations
  - Inputs and outputs are composed of uniform *populations* of agents that do *not* have an identity
  - Algorithms ‘emerge’ from the ‘dumb’ interactions of ‘simple’ agents
- In computing
  - Mostly explored in discrete or nondeterministic time
- In science and nature
  - Mostly explored in stochastic time
  - Stochastic because ‘interactions’ typically correspond to random collisions or chance meetings

A mathematical model

- The underlying model is Continuous-Time Markov Chains
  - Which also underlies chemistry via the Chemical Master Equation (changes of probabilities of discrete states over continuous time).
- Can be presented discretely, stochastically
  - As stochastic Petri nets, stochastic process algebras, etc.
- NOT a probabilistic model
  - Probabilities emerge from the stochastic structure (as the underlying DMC), but are not primary. We are in continuous time and we care about how long things take.
  - Non-determinism exists only in the form of 'quantitative races' among possibilities: who is faster is more likely to win, but there is no pure, timeless, random or probabilistic choice.
  - Interleaving rules: no two events (interactions) can ever happen at the same time in real time.

Basic Results

- The class of functions ‘over individuals’ that are computable
  - Turing machines can be encoded up to an arbitrarily small uniform error bound. “Approximately Turing-Complete”.
    - David Soloveichik, Matt Cook, Erik Winfree, Shuki Bruck, **Computation with Finite Stochastic Chemical Reaction Networks.** Natural Computing, 2008.
    - Luca Cardelli, Gianluigi Zavattaro. **Turing Universality of the Biochemical Ground Form.** MSCS 2010.
    - Wiedermann et al. …
- The class of predicates ‘over collectives’ that are ‘stably computable’
  - Semi-linear predicates (first-order theory of $(\mathbb{N},+,<)$).
  - Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. **The computational power of population protocols.** Distributed Computing, 2007.

Paradigm

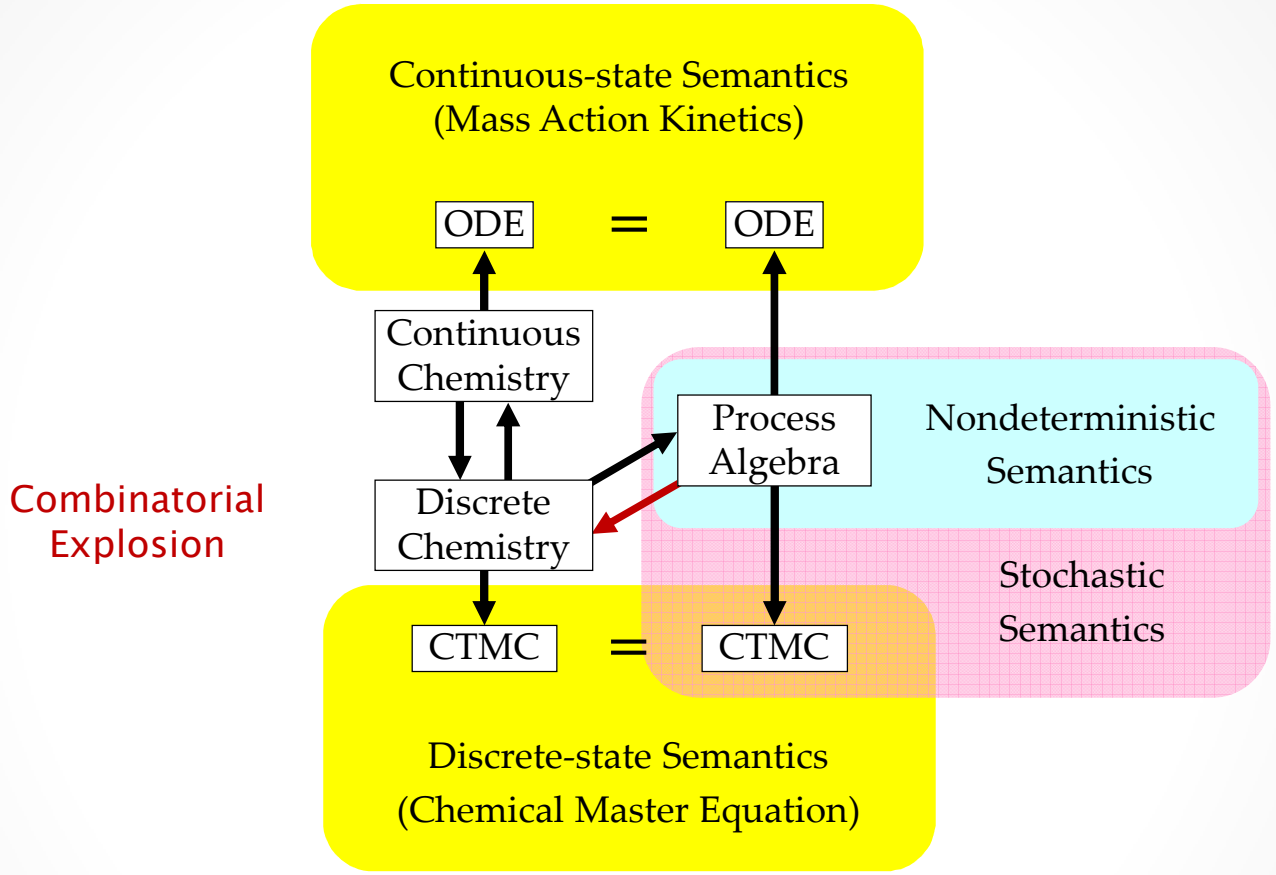
- Stochastic chemistry is the simplest paradigm for this model
  - Finite collections of chemical reactions (with real-number rates) among a finite set of species.
  - Not necessarily preserving mass or energy (assumed to be freely provided from the 'outside').
  - Usually restricted to null-, uni-, and bi-molecular reactions.

• $x \rightarrow_r y + y$	multiply x by 2
• $x + y \rightarrow_r y + b$ $y + x \rightarrow_r x + b$ $b + x \rightarrow_r x + x$ $b + y \rightarrow_r y + y$	compute the majority of populations x and y in log time [Angluin et al.]

Molecular Languages

- Reaction-Based (A + B → C + D) (Chemistry)
  - Limited to finite set of species (no polymerization)
  - Practically limited to small number of species (no run-away complexation)
- Interaction-Based (A = !c. B) (Process Algebra)
  - Reduces combinatorial complexity of models by combining independent submodels connected by interactions.
- Rule-Based (A{-}:B{p} → A{p}:B{-}) (Logic, Graph Rewriting)
  - Further reduces model complexity by describing molecular state, and by allowing one to 'ignore the context': a *rule* is a reaction in an unspecified (complexation/phosphorylatio) context.
  - Similar to informal descriptions of biochemical events ("narratives").
- Syntactic connections
  - The latter two can be translated (to each other and) to the first, but doing so may introduce an infinite, or anyway *extremely large*, number of species.

Semantic Connections



These diagrams commute via appropriate maps.

L. Cardelli: "On Process Rate Semantics" (TCS)

L. Cardelli: "A Process Algebra Master Equation" (QEST'07)

Paradigm Lost

- But chemistry *is not* a computational science!
  - Real chemical reactions just 'happen' between real molecules that exist in nature. We don't control them.
  - Chemists 'transcribe' nature and write down 'its' reactions. They do not write their own chemical programs.
    - Ok, they often design new molecules, but they do not have 'full computational control' over what those do.
- Similarly electronics *was not* computational
  - Electron exchanges just 'happen' in nature.
  - Early physicists did not have the ability to program them.
  - But now we do!

Paradigm Encoded

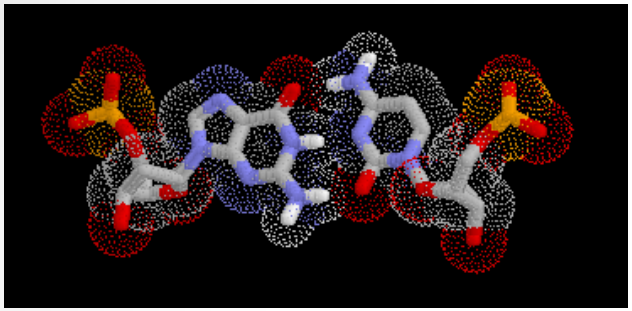
- Find some ‘universal molecules’ that can do ‘what all the other molecules can do’
  - By ‘doing something’ here we mean ‘implementing a chemical kinetics’.
  - That is: find a universal class of molecules that can emulate the kinetics of arbitrary systems of chemical reactions among real or fictitious molecules, up to some abstraction (e.g. time dilation).
- Find a way to actually execute molecular languages, with real molecules.

Computing with DNA

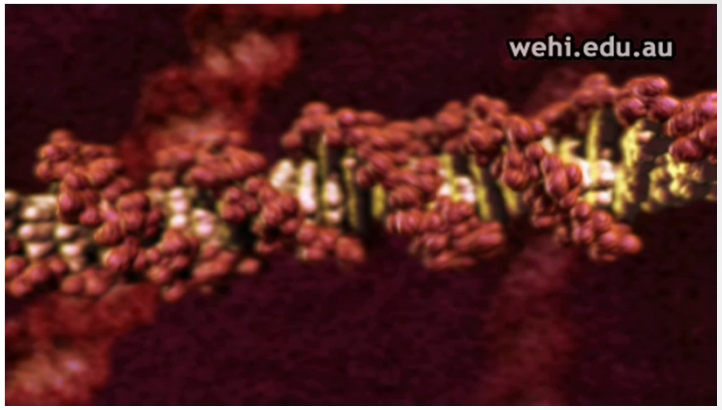
- Computing with molecules was, of course, the original idea in DNA computing
  - Early examples [Adelman] encoded specific algorithms.
- But only recently people have proposed ‘universal DNA molecules’
  - Soloveichik, D., Seelig, G., Winfree, E., DNA as a Universal Substrate for Chemical kinetics.

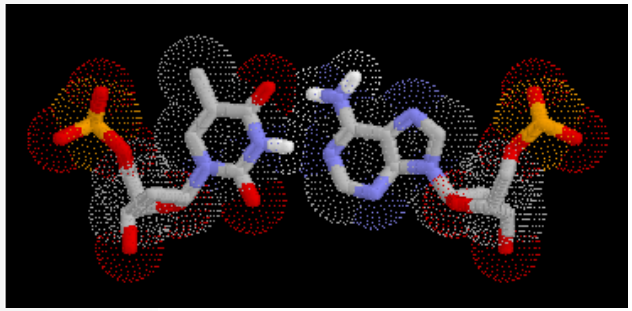
DNA

DNA



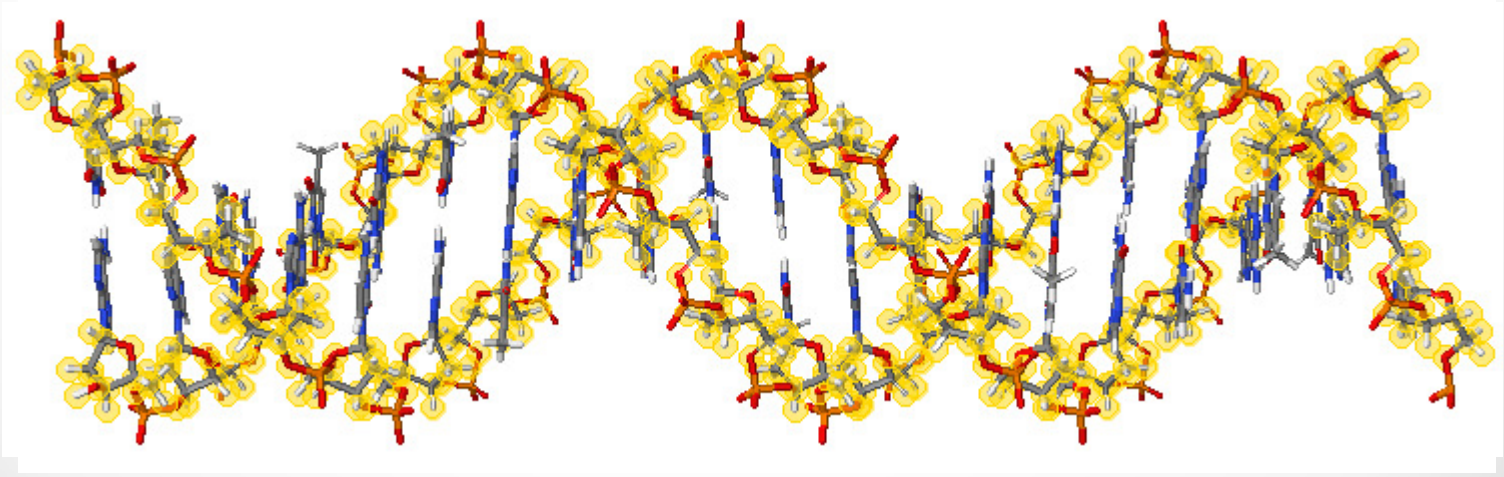
GC Base Pair
Guanine-Cytosine





TA Base Pair
Thymine-Adenine

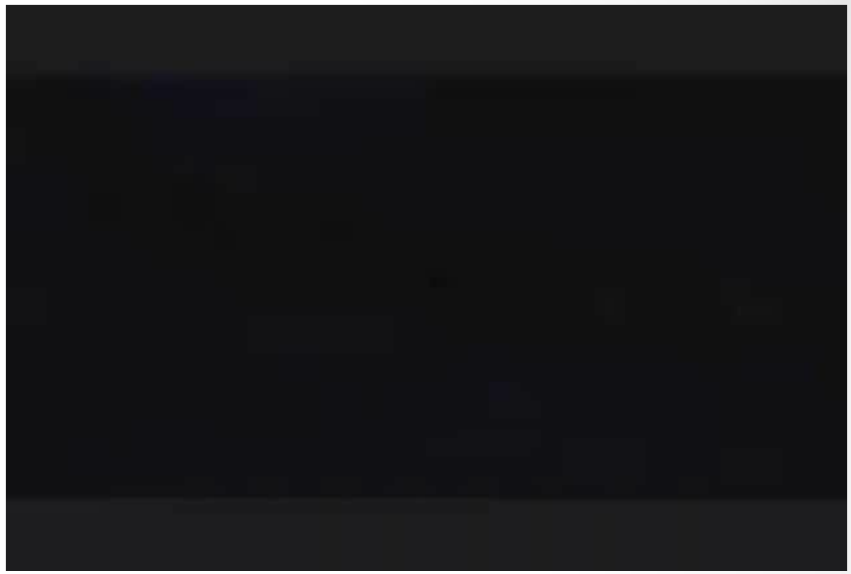
Interactive DNA Tutorial
(http://www.biosciences.bham.ac.uk/labs/minchin/tutorials/dna.html)



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

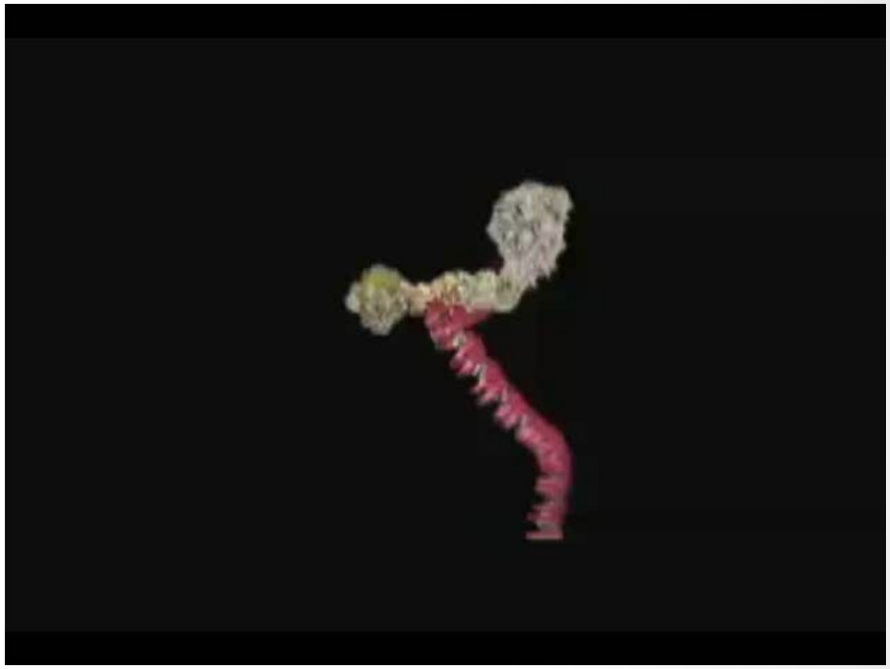
Natural DNA Operation

- 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*

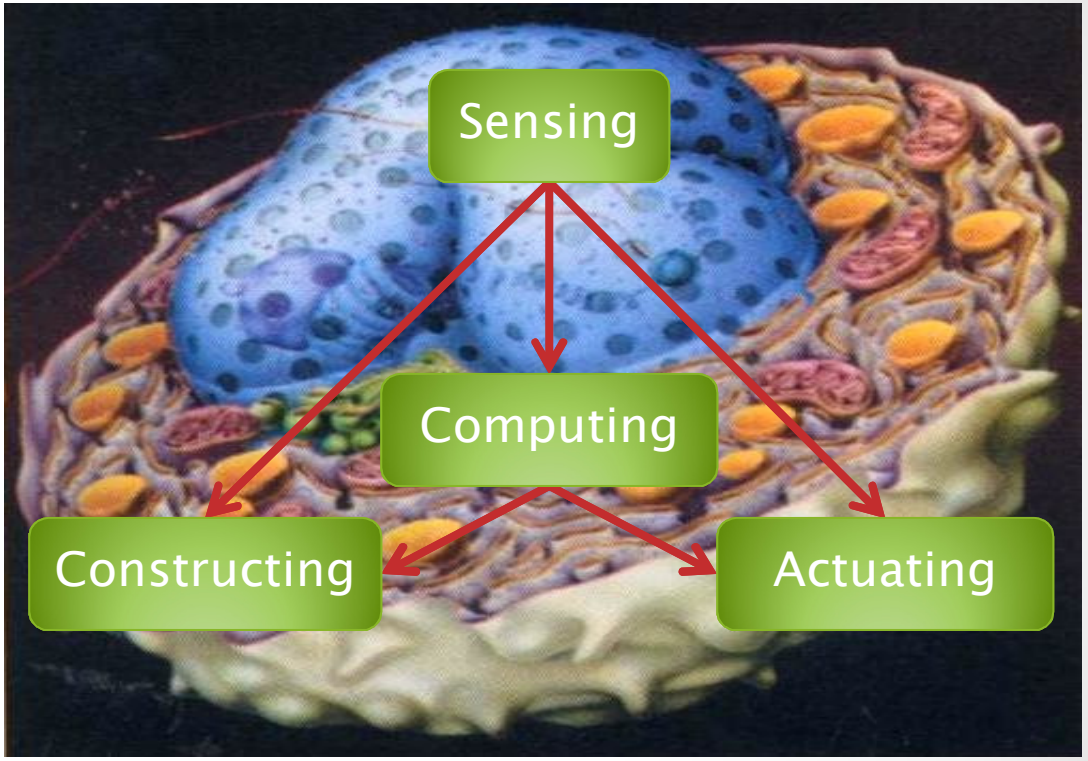
RNA polymerase II:
15-30 bases/second

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

Unnatural DNA Operation

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

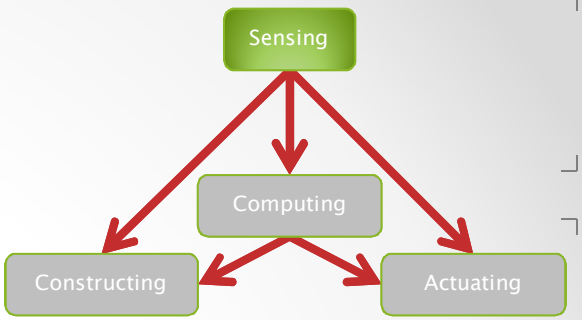
Nanoscale Control Systems



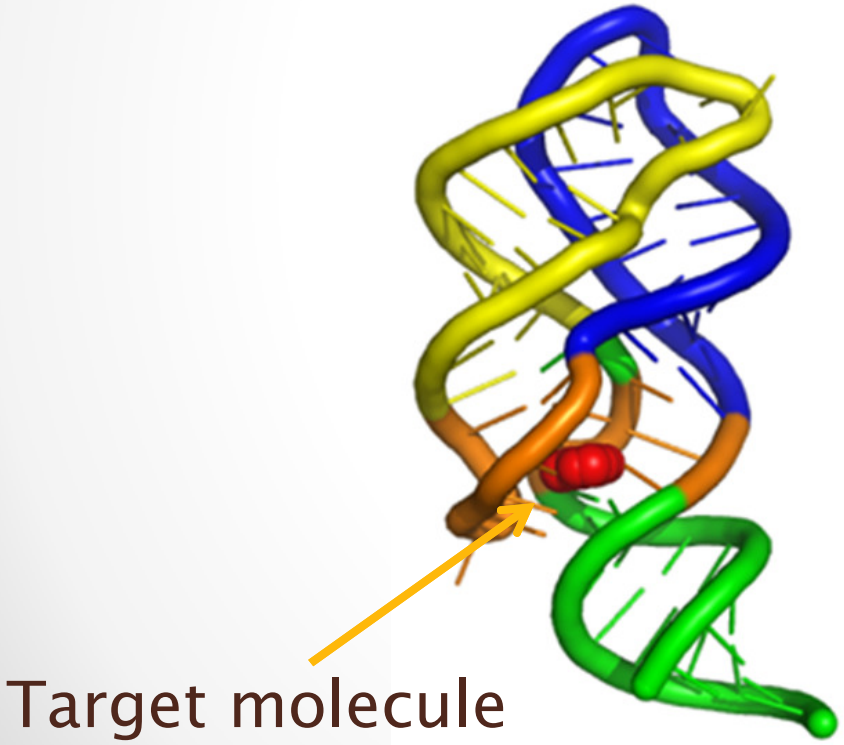
Nucleic Acids can do all this.

And interface to biology.

Sensing



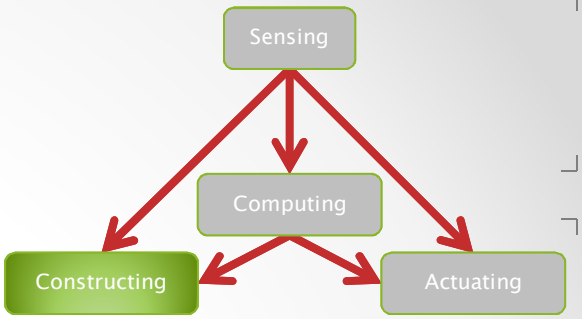
Aptamers: natural or artificially evolved DNA molecules that stick to other molecules (highly selectively).



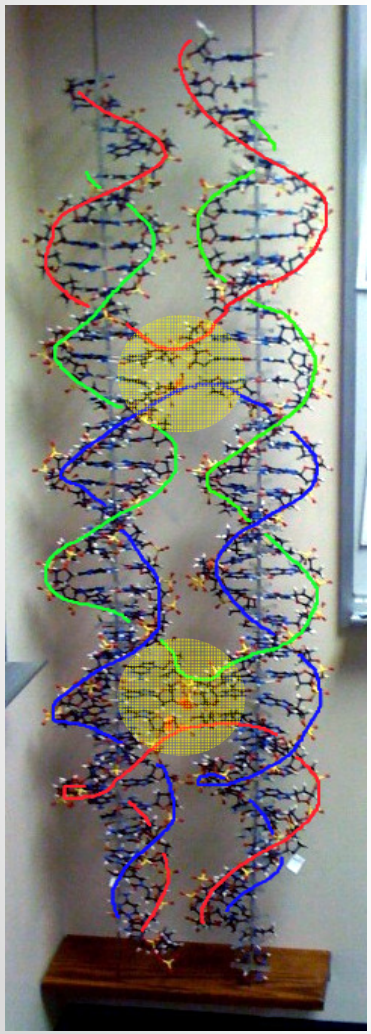
Adenine riboswitch aptamer

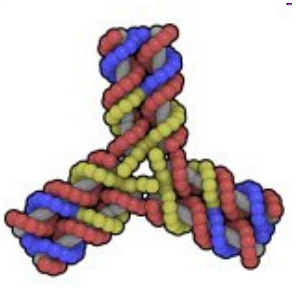
Structural basis for discriminative regulation of gene expression by adenine- and guanine-sensing mRNAs. Chem Biol. 2004 Dec;11(12):1729-41.

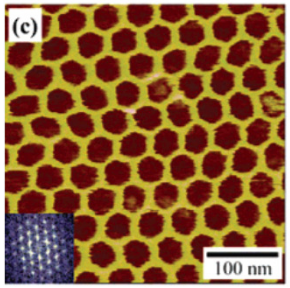
Constructing



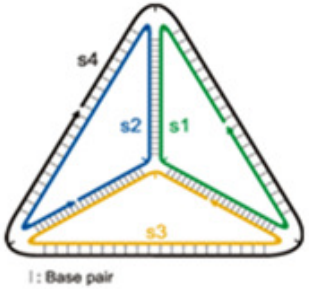
Crosslinking

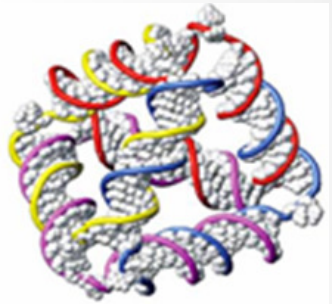






Chengde Mao, Purdue





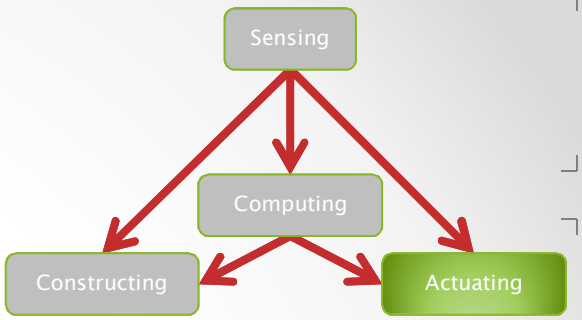
Andrew Turberfield, Oxford

Folding DNA into Twisted and Curved Nanoscale Shapes

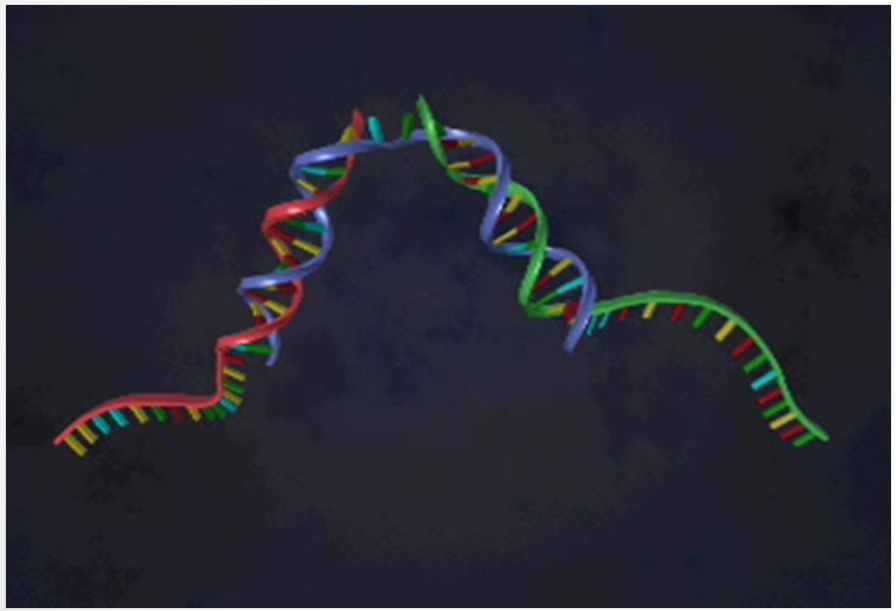
Hendrik Dietz, Shawn M. Douglas, & William M. Shih

Science, 325:725–730, 7 August 2009.

Actuating

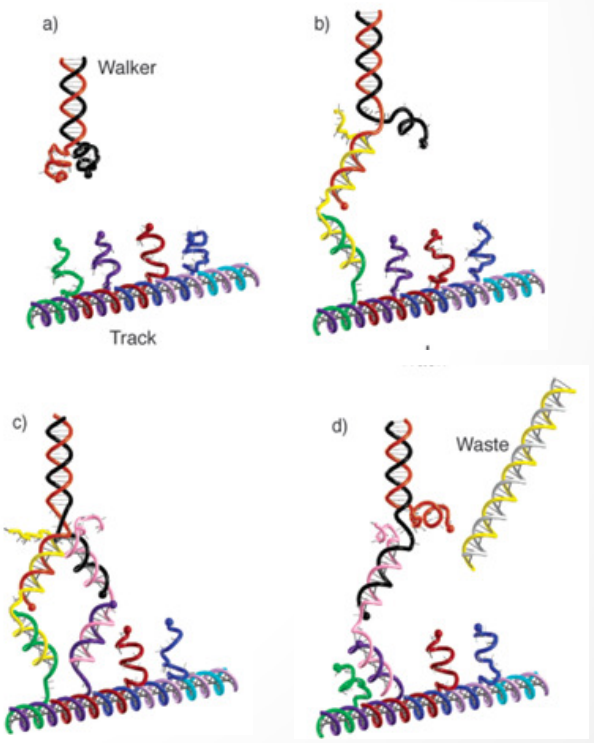


DNA tweezers

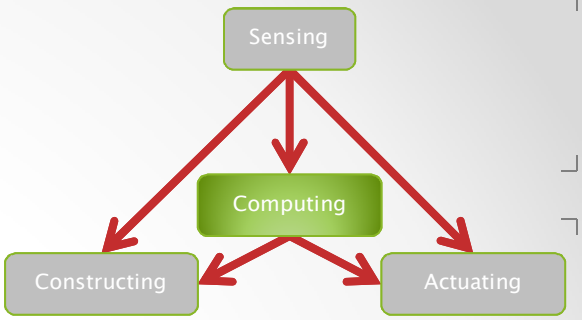


Bernard Yurke, Boise State

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’

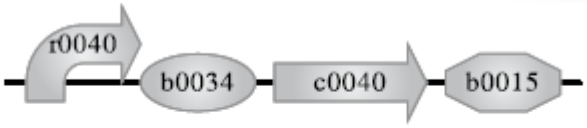
“Embedded” Computing

(Synthetic Biology)

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

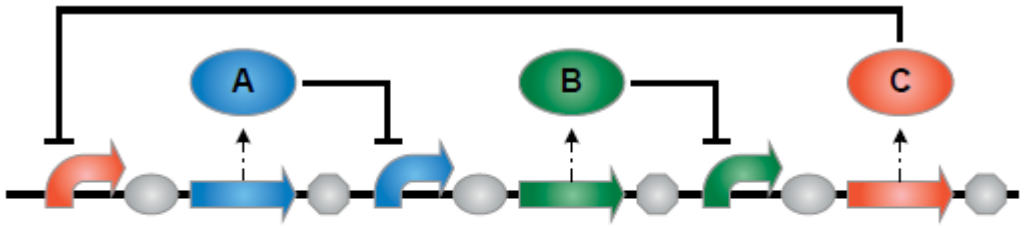
- MIT Registry of Standard Biological Parts

- GenoCAD
  - Meaningful sequences [Cai et al.]




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

- GEC
  - [Pedersen & Phillips]



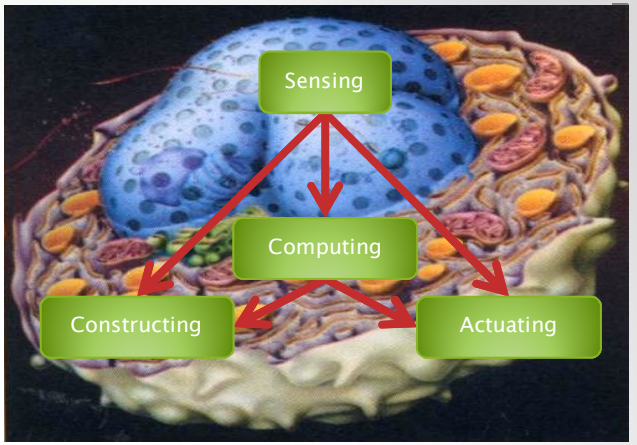

```
prom<neg(C)>; rbs; pcr<codes(A)>; ter;
prom<neg(A)>; rbs; pcr<codes(B)>; ter;
prom<neg(B)>; rbs; pcr<codes(C)>; ter
```

“Autonomous” Computing

(Nano-engineering)

- 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

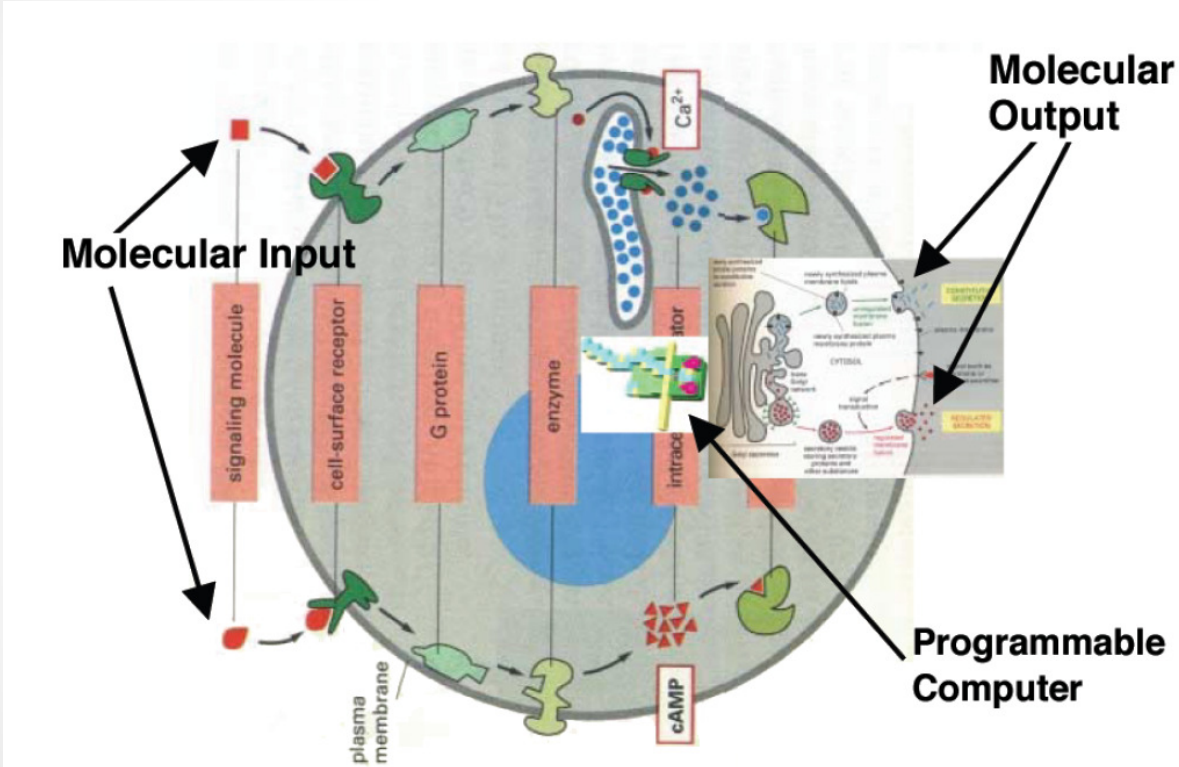


Fig. 1 *Medicine in 2050: "Doctor in a Cell"*

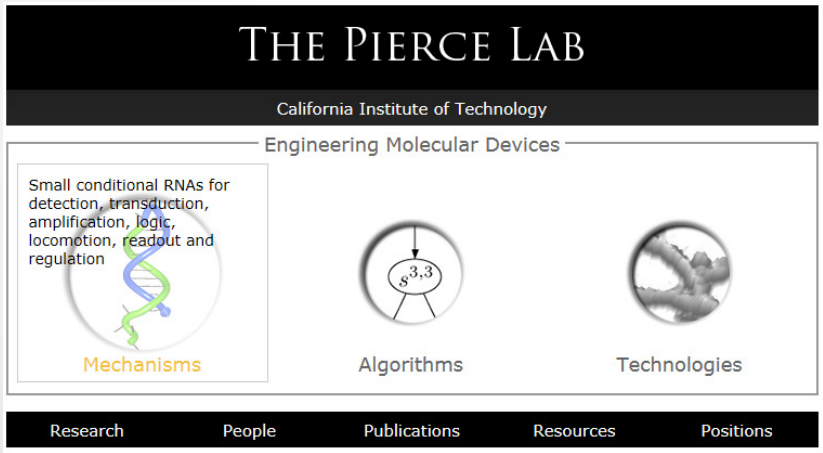
Ehud Shapiro

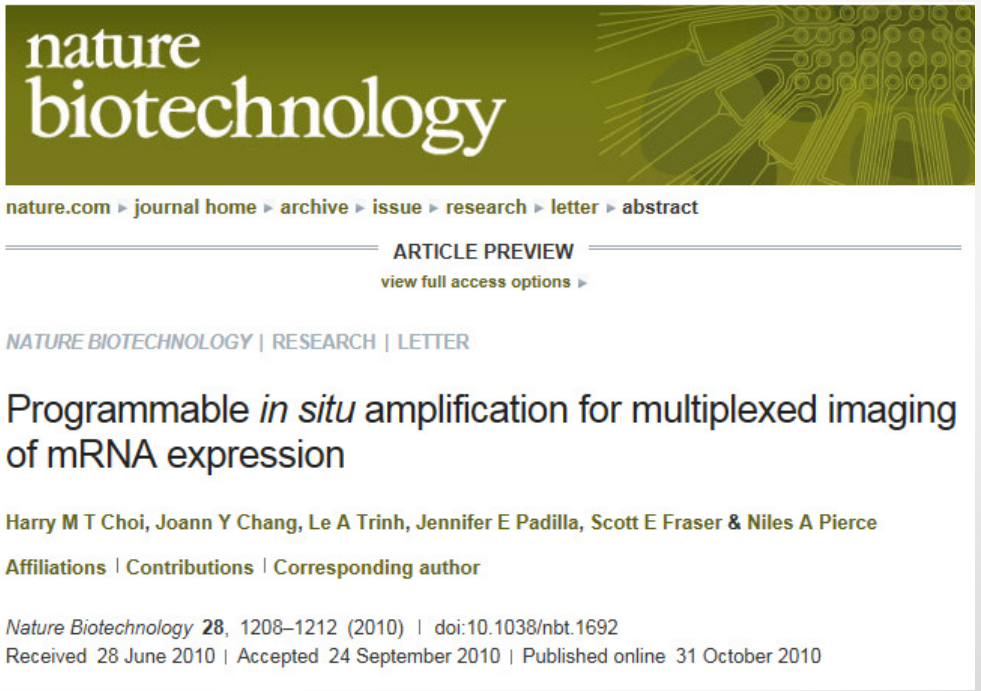
Rivka Adar
Kobi Benenson
Gregory Linshitz
Aviv Regev
William Silverman

Molecules and computation

RNA computation in dead cells

- Using RNA Hybridization Chain Reaction for imaging of mRNA expression.
  - The programmability of orthogonal RNA reactions enables spatial imaging with 5 simultaneous targets.

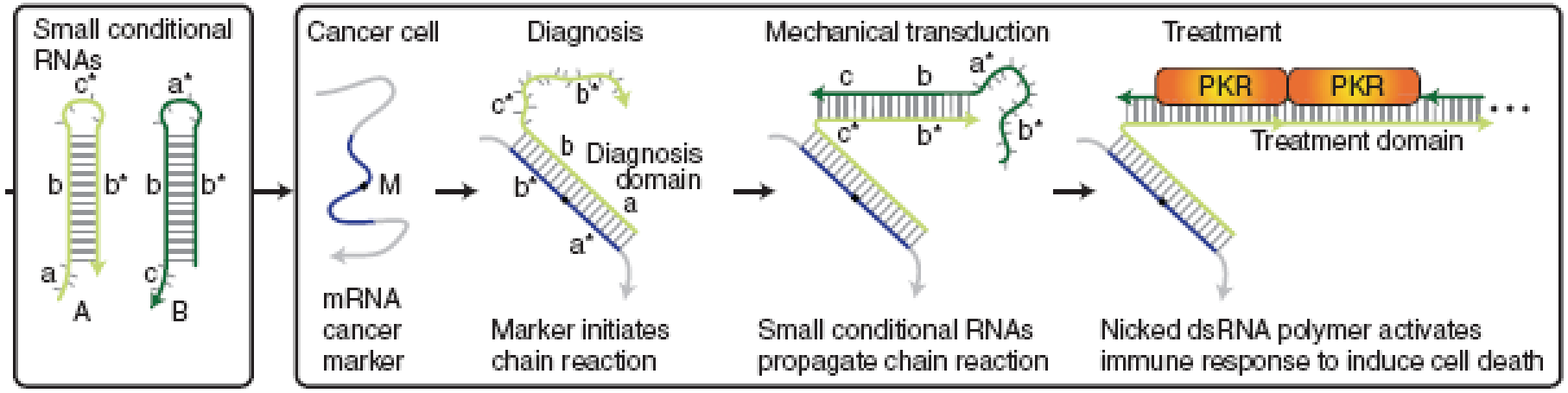




RNA computation in live cells

Selective cell death mediated by small conditional RNAs

Suvir Venkataraman[a], Robert M. Dirks[a,b], Christine T. Ueda[b], and Niles A. Pierce[a,c,1]



PNAS | September 28, 2010

Computing with DNA Strand Displacement

DNA Computing

- 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.

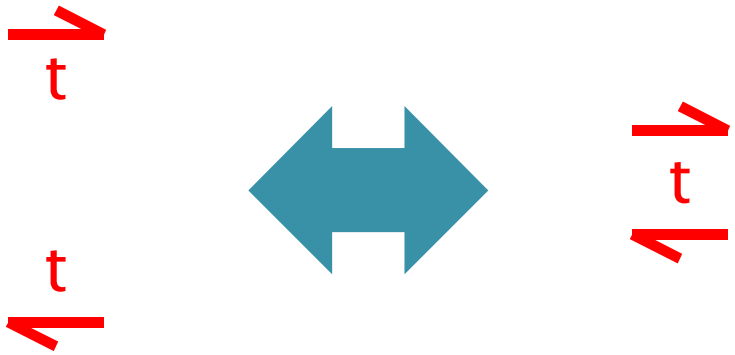
Domains

- 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.

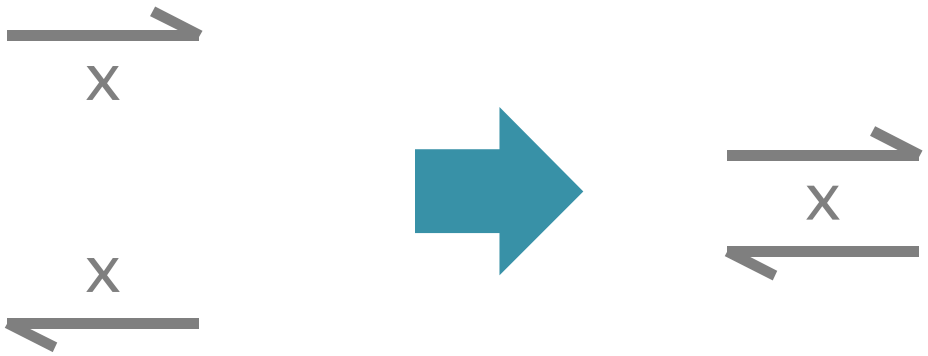
- Choosing domains (subsequences) that are suitably independent is a tricky issue that is still somewhat of an open problem (with a vast literature). But it can work in practice.

Short Domains



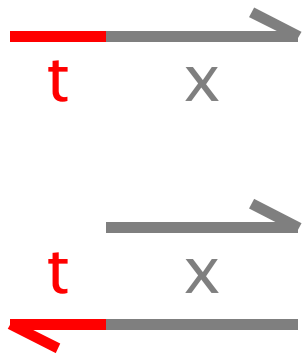
Reversible Hybridization

Long Domains



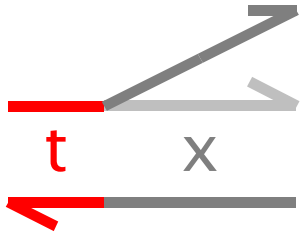
Irreversible Hybridization

Strand Displacement



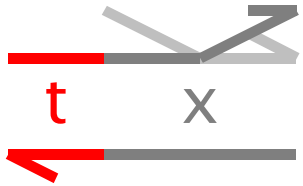
"Toehold Mediated"

Strand Displacement



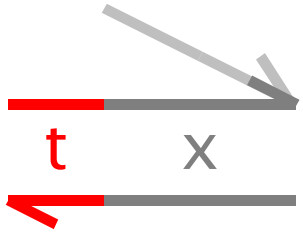
Toehold Binding

Strand Displacement



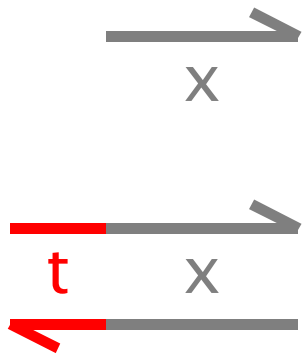
Branch Migration

Strand Displacement



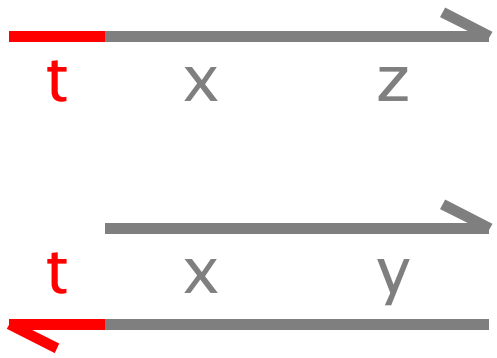
Displacement

Strand Displacement

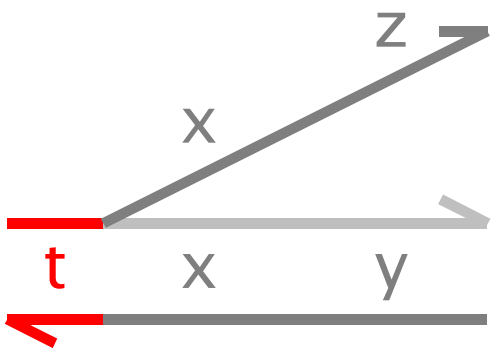


Irreversible release

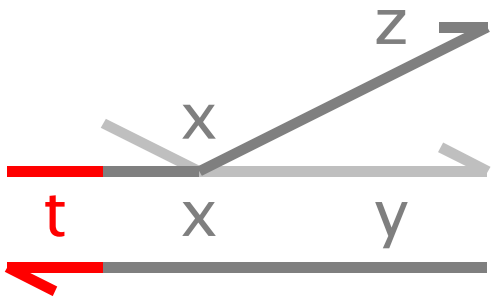
Bad Match



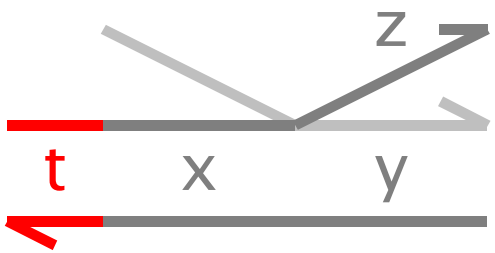
Bad Match



Bad Match



Bad Match

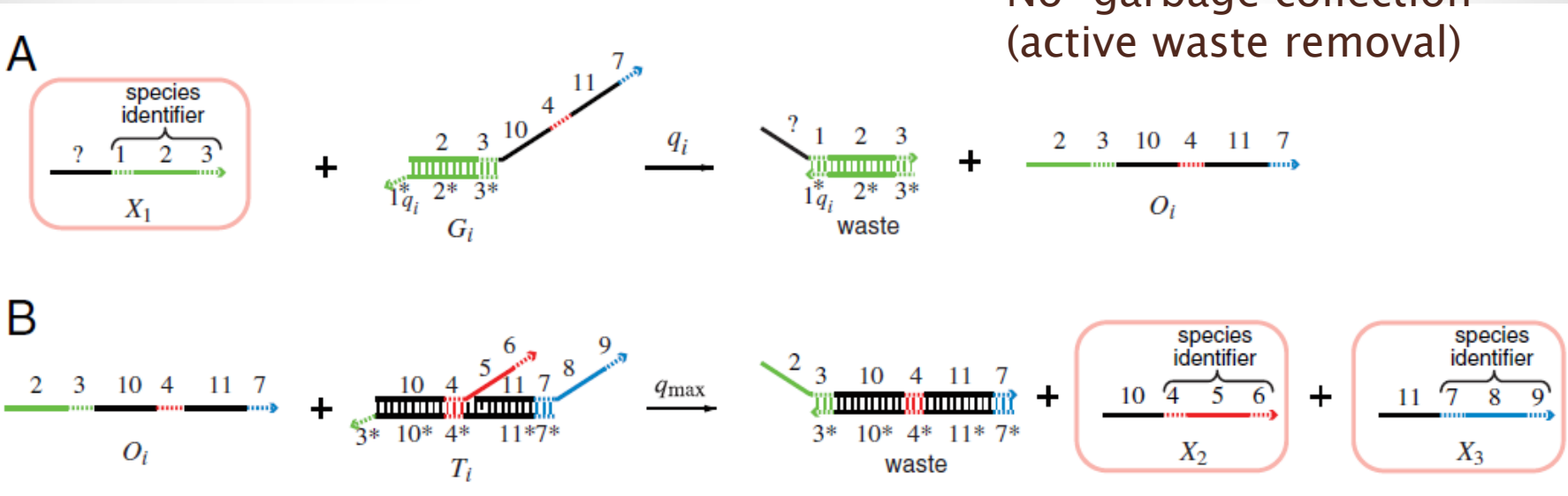


Cannot proceed
Hence will undo

Computation by DNA Strand Displacement

Four-Domain Architecture

No “garbage collection” (active waste removal)



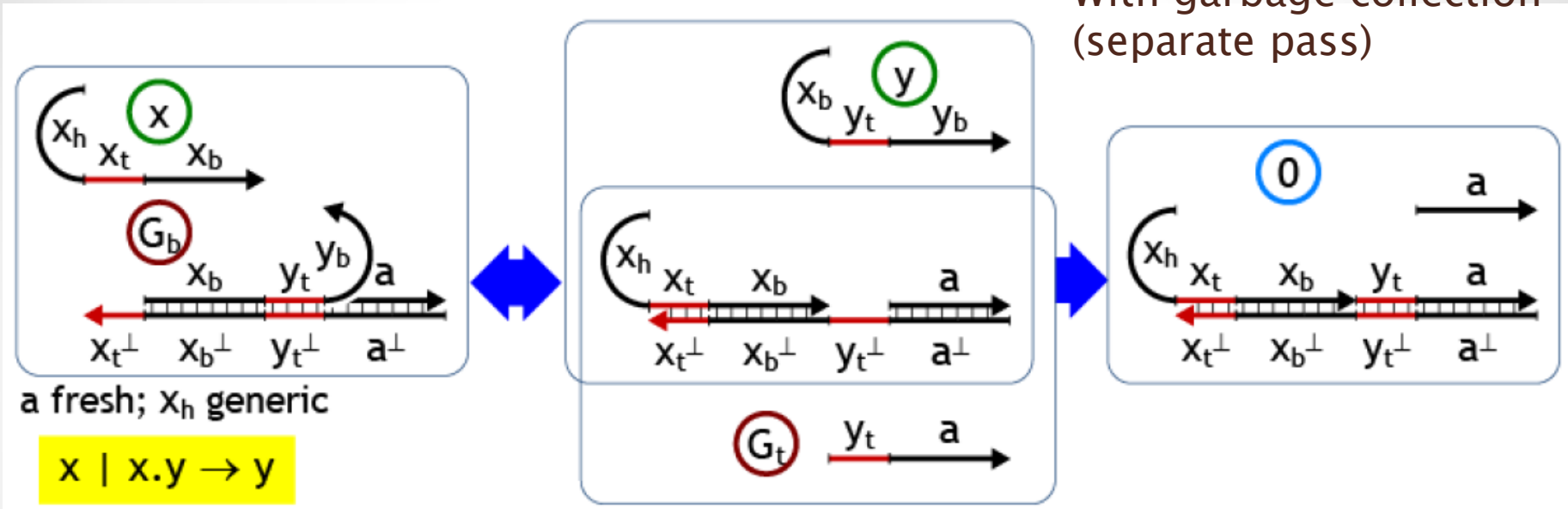
DNA as a universal substrate for chemical kinetics

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)

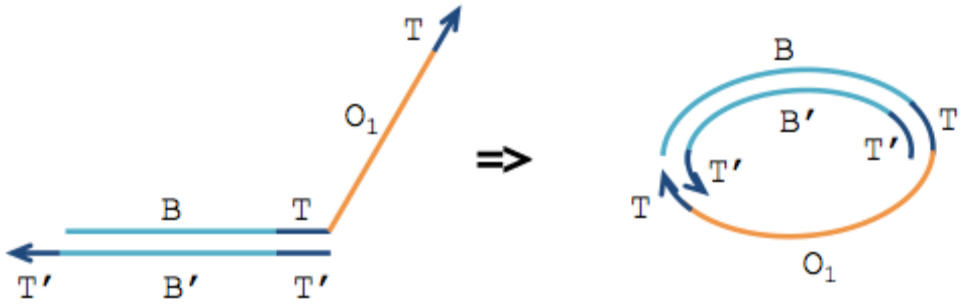


Strand Algebras for DNA Computing

Luca Cardelli

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

“Lulu’s Trouble”



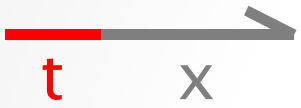
(from D.Soloveichik)

DCM 2010

- Looking for a *simple* process *algebra* for strand displacement
  - For manual or automated analysis or correctness of strand displacement ‘programs’.
  - Had to be *simple* (or you could not analyze it). Hence looking for a simpler strand displacement scheme.
  - Had to be an *algebra*, hence computation could not leave garbage around, or nothing would commute.
- The technology was to be constrained by the theory

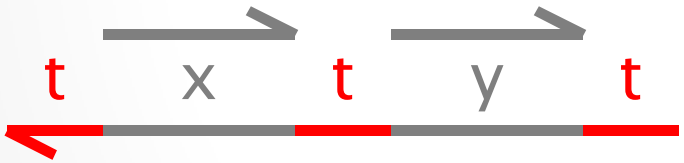
Two-Domain Architecture

- Signals: 1 toehold + 1 recognition region



Garbage collection
"built into" the gates

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

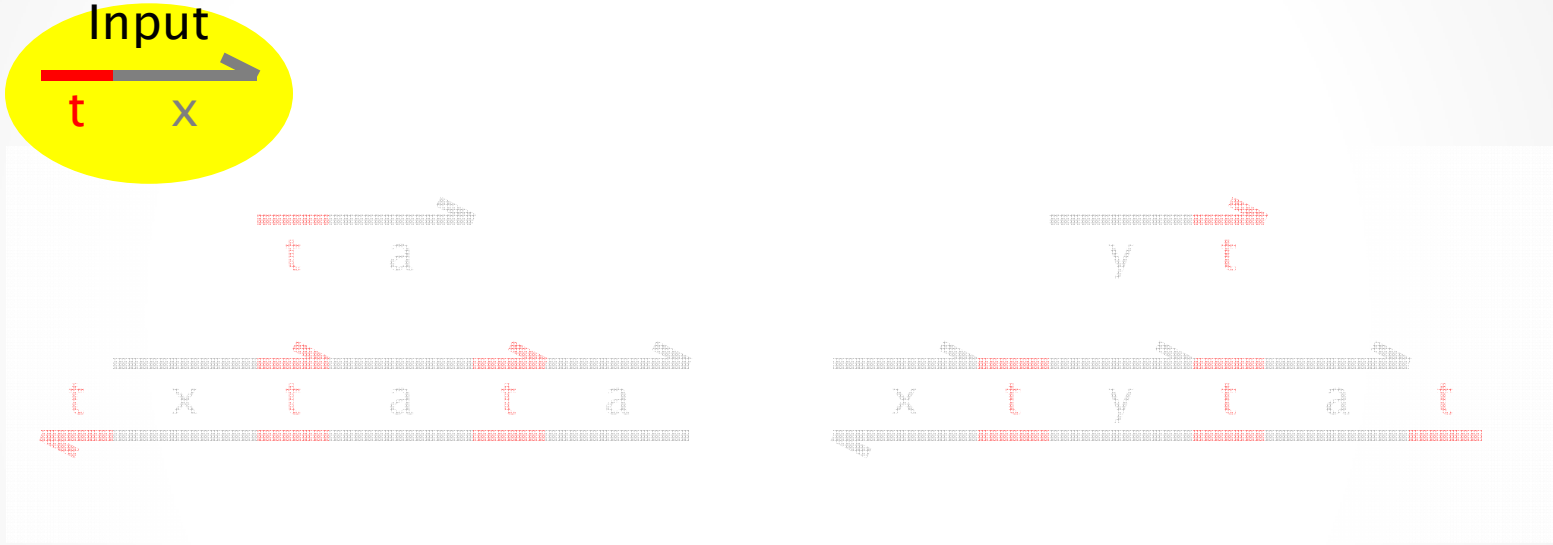


Two-Domain DNA Strand Displacement

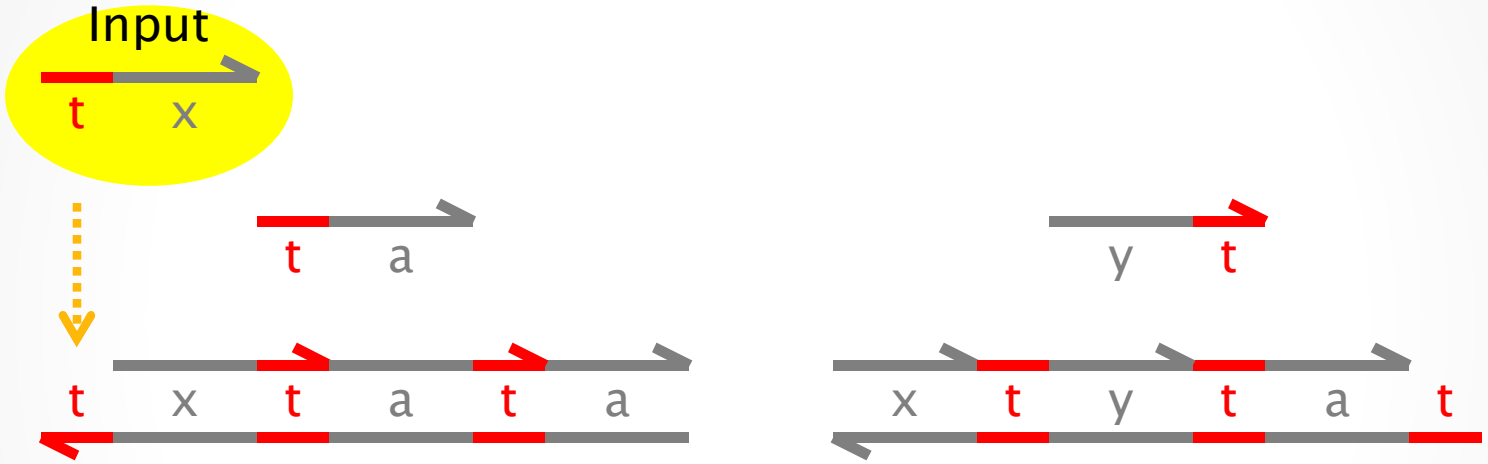
Luca Cardelli

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



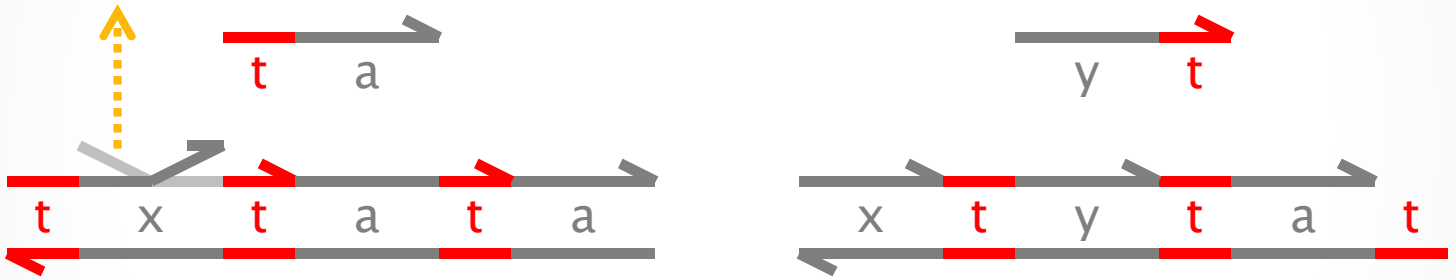
Transducer x→y



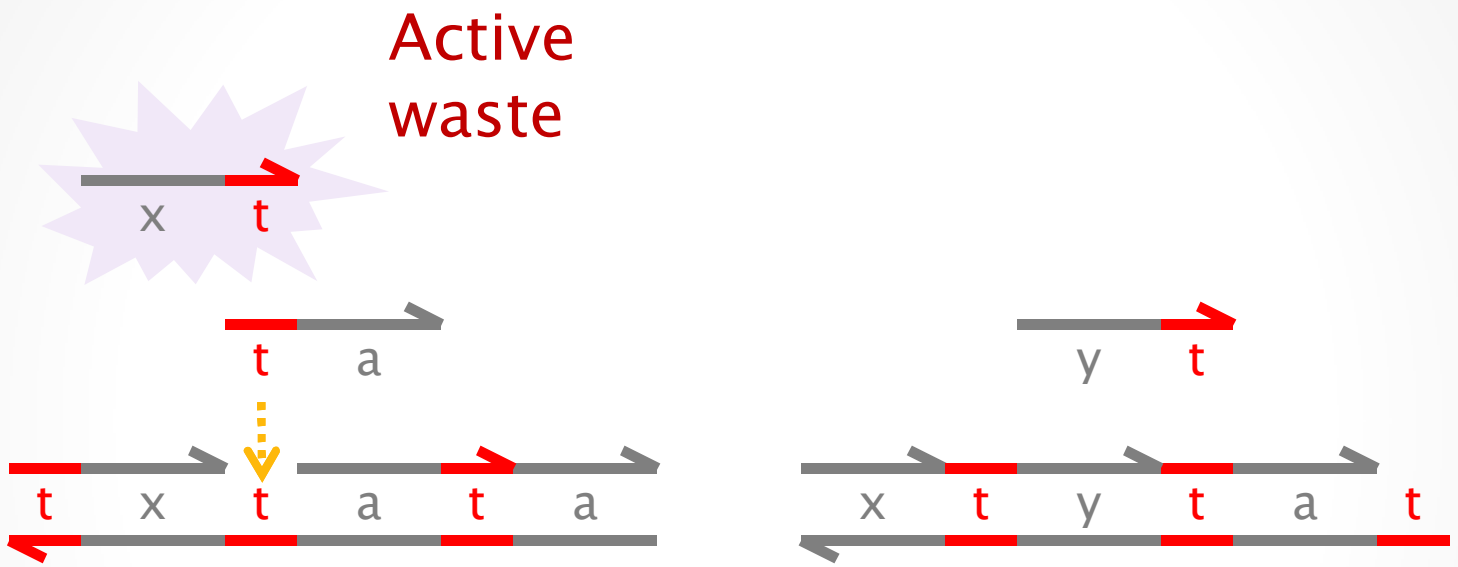
Built by self-assembly!

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

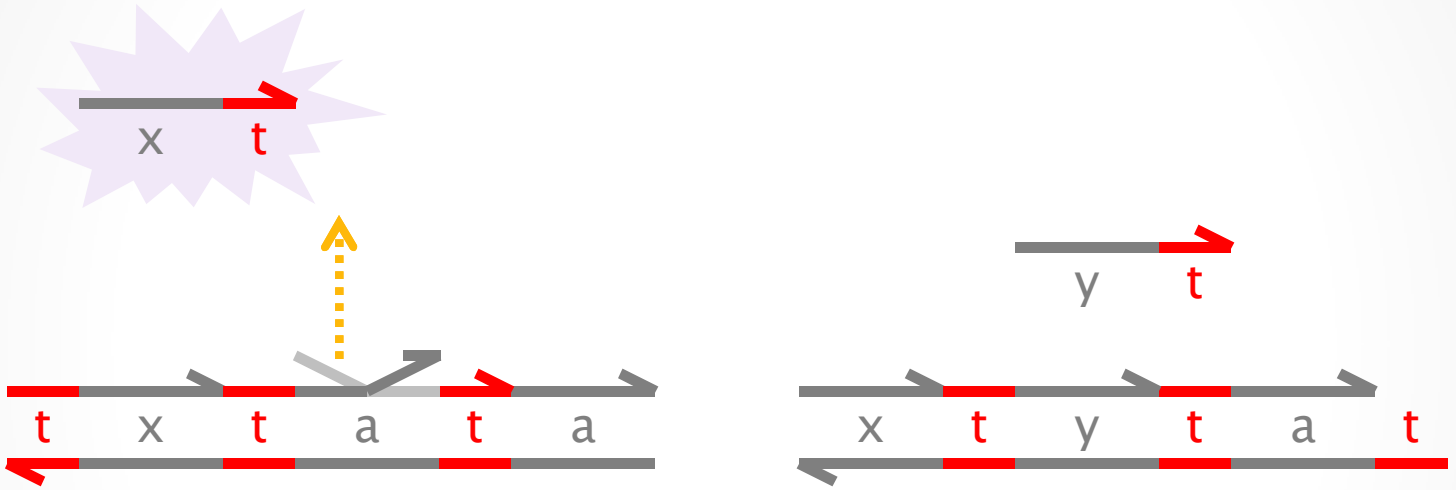
Transducer x→y



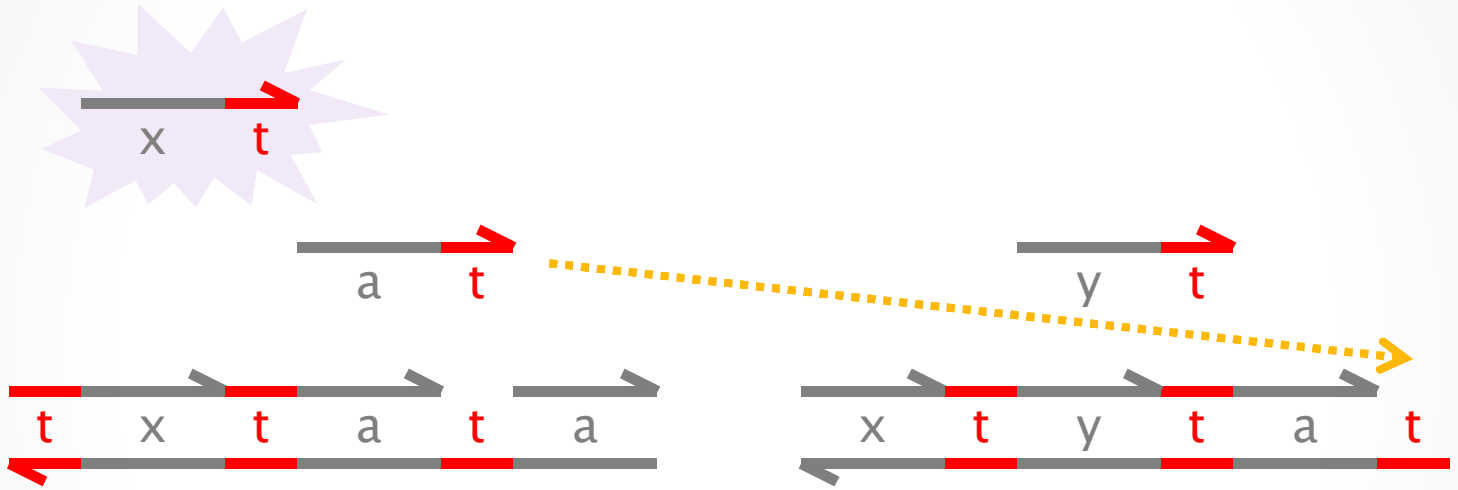
Transducer x→y



Transducer x→y

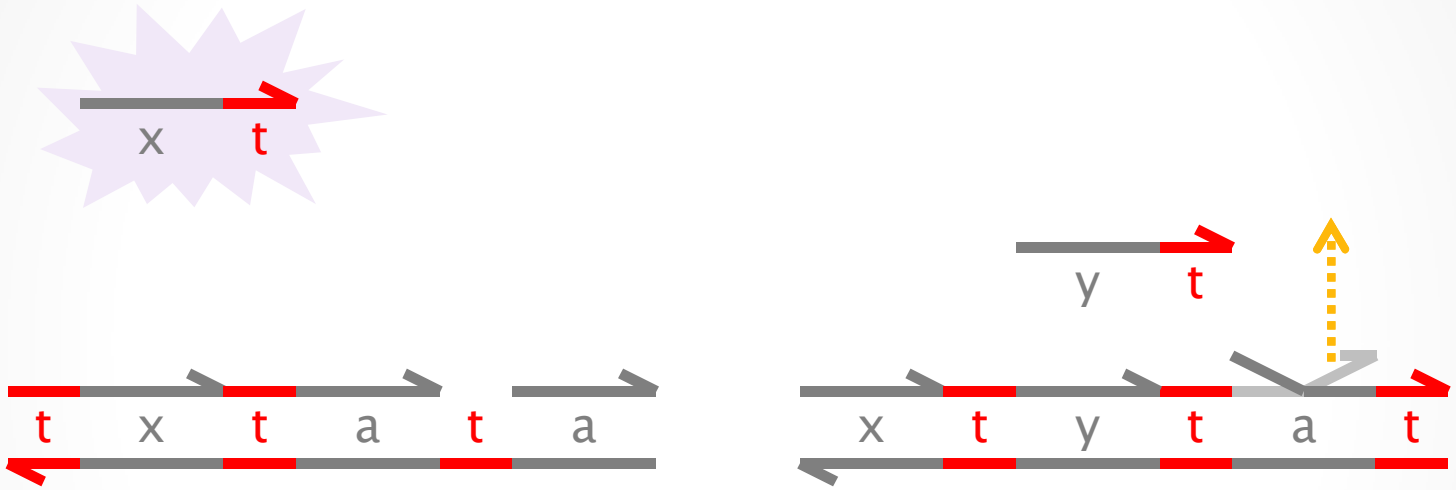


Transducer x→y

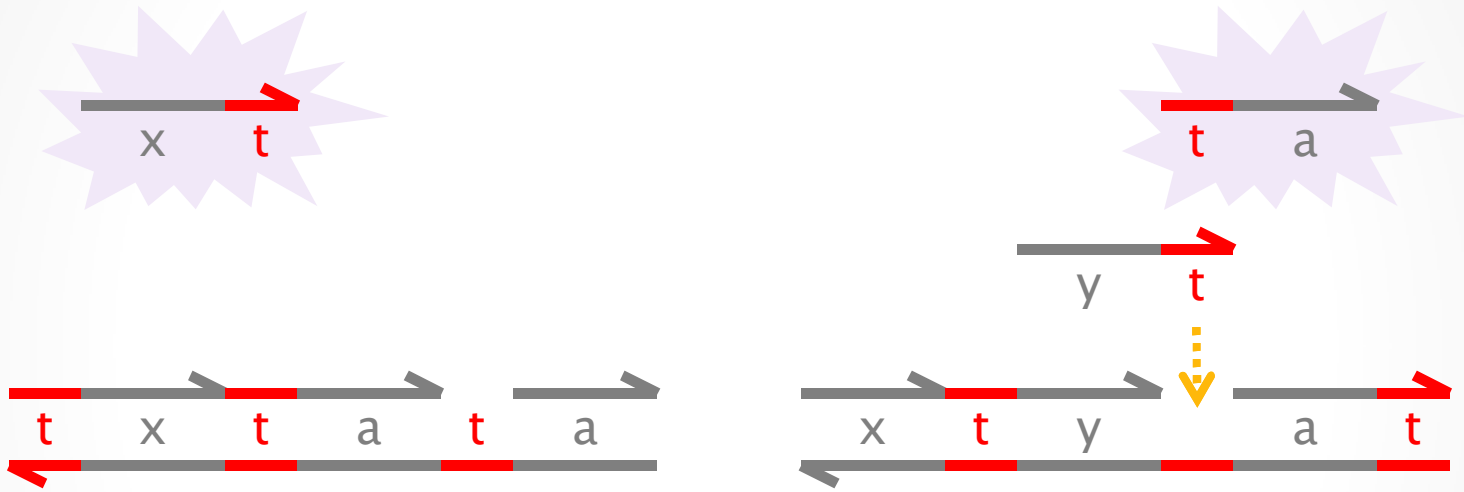


So far, a tx *signal* has produced an at *cosignal.*
But we want signals as output, not cosignals.

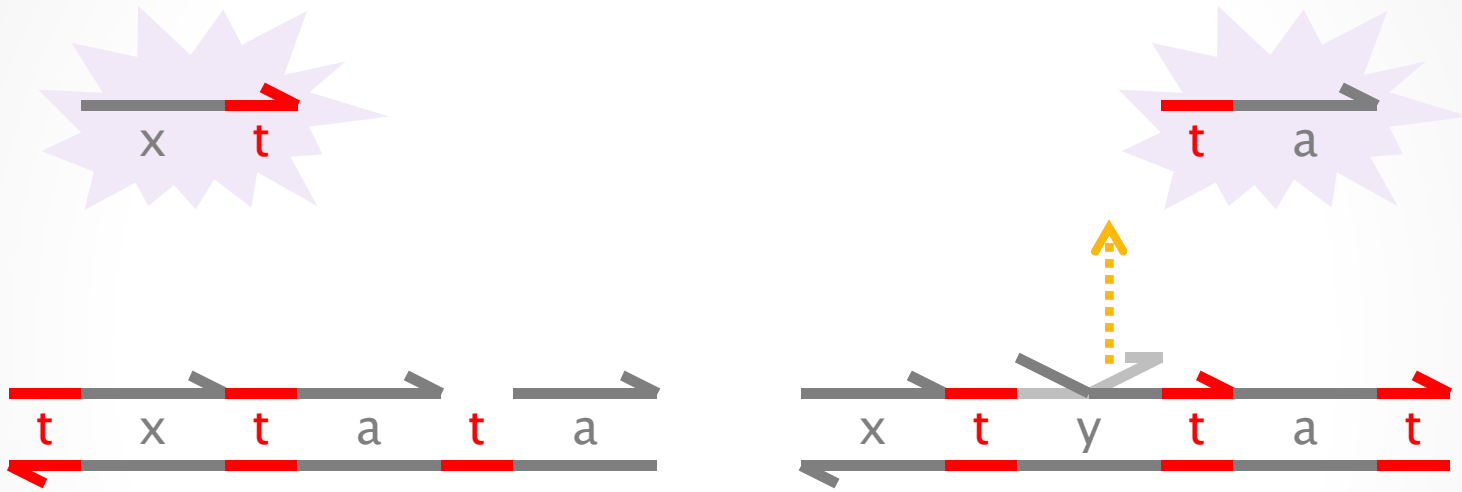
Transducer x→y



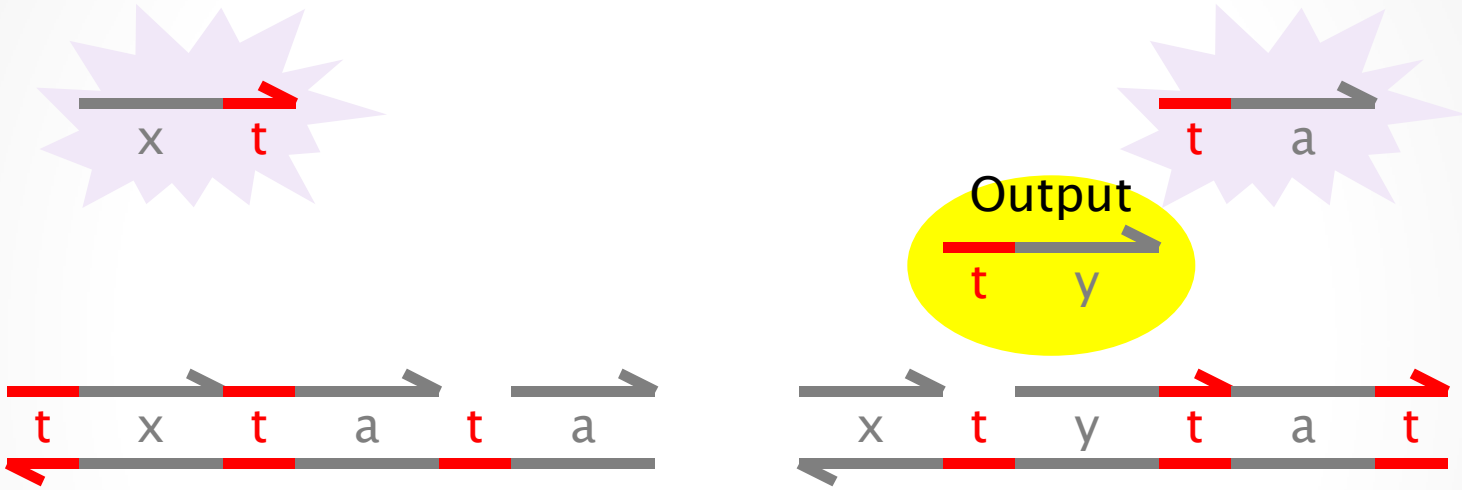
Transducer x→y



Transducer x→y



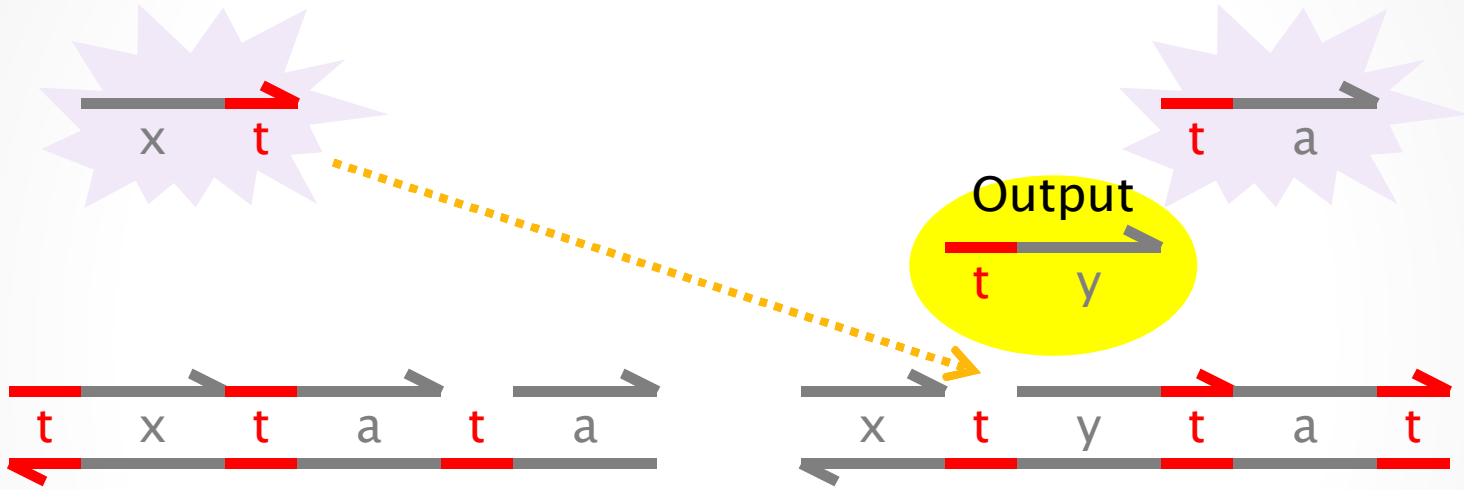
Transducer x→y



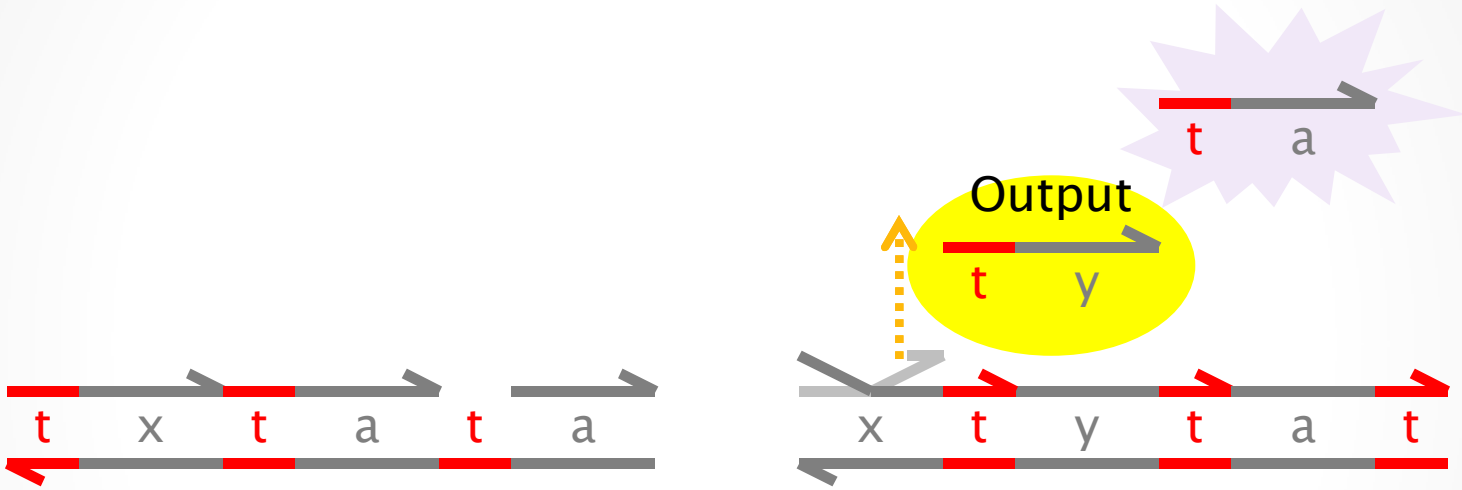
Here is our output ty *signal*.

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).

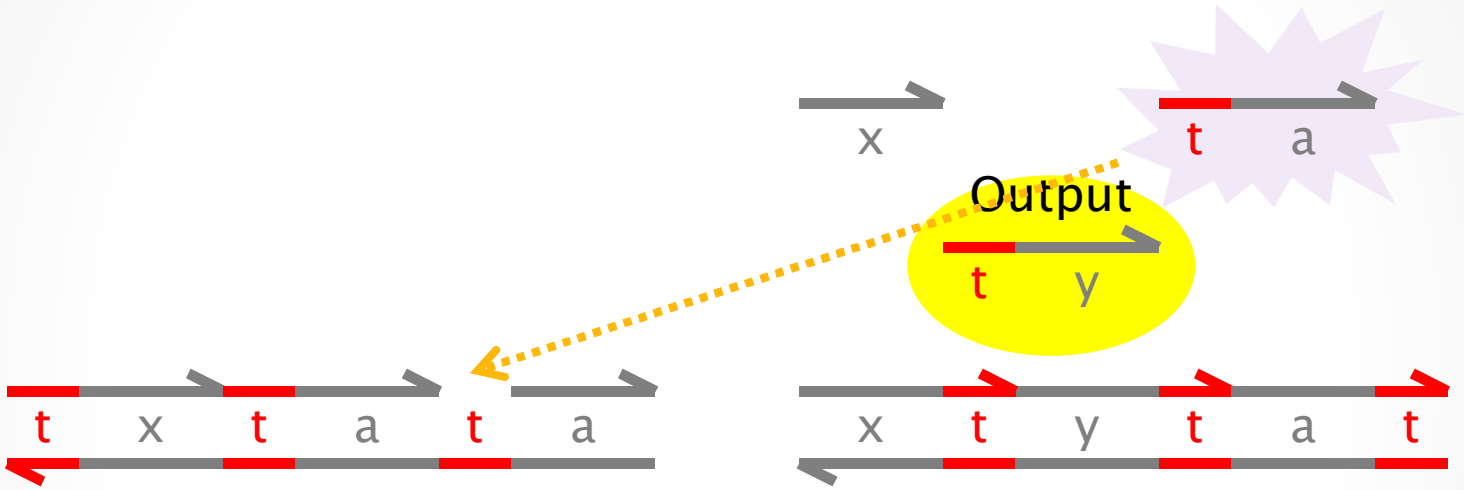
Transducer x→y



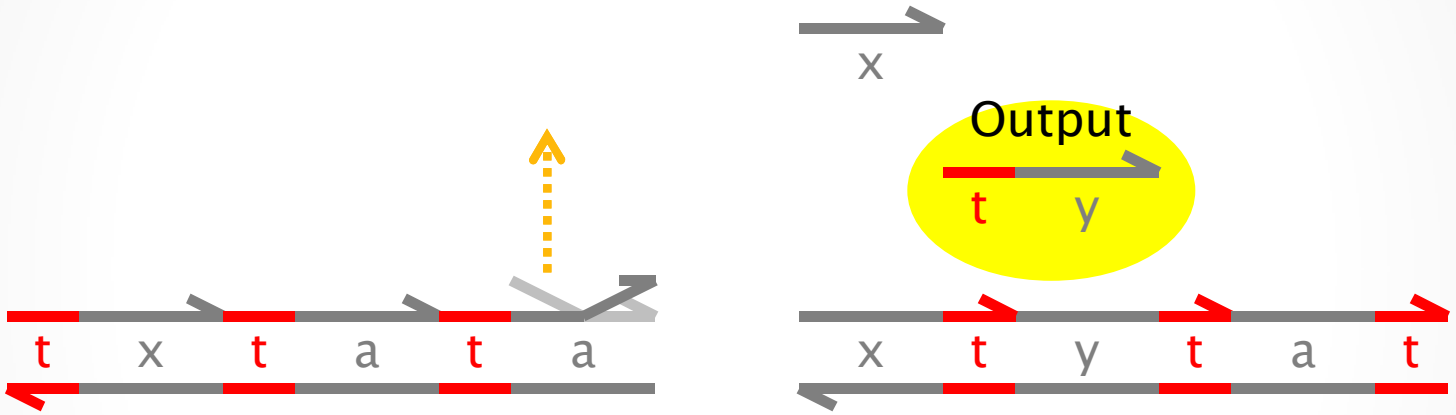
Transducer x→y



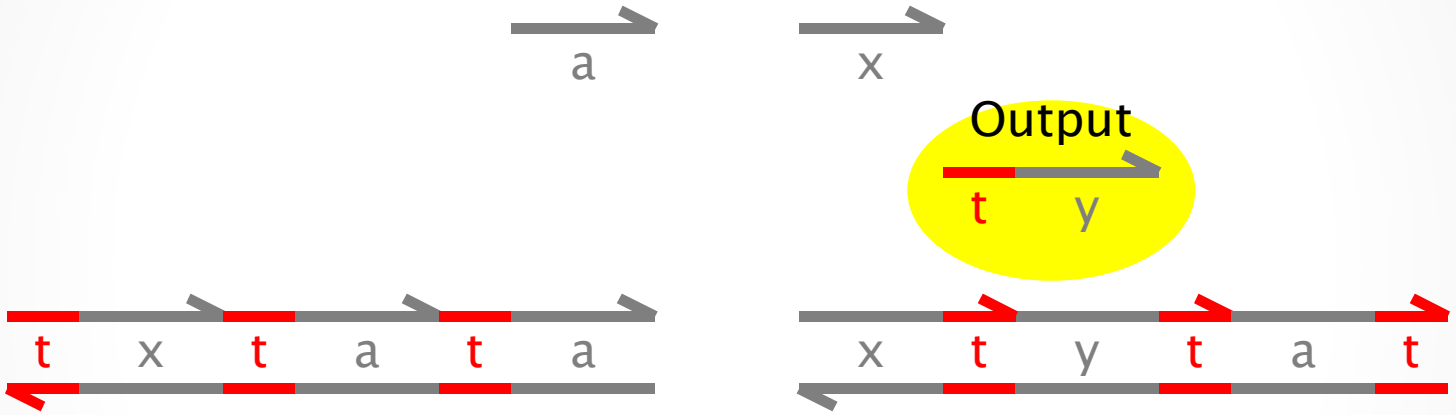
Transducer x→y



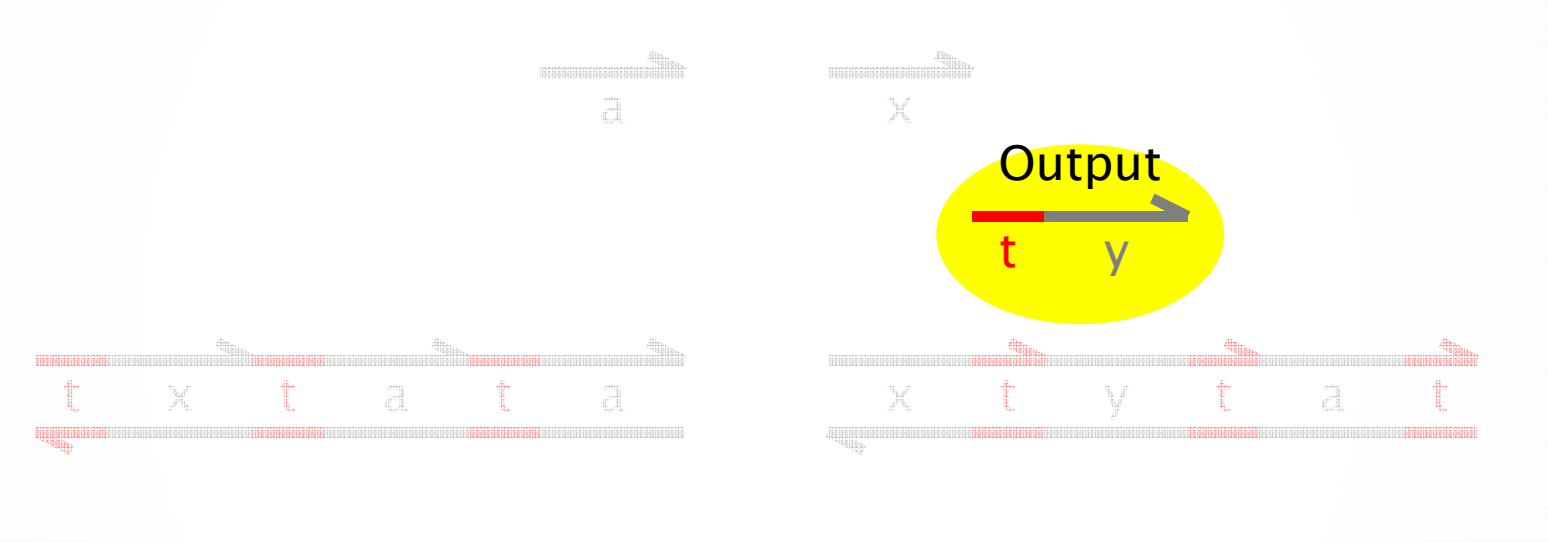
Transducer x→y



Transducer x→y

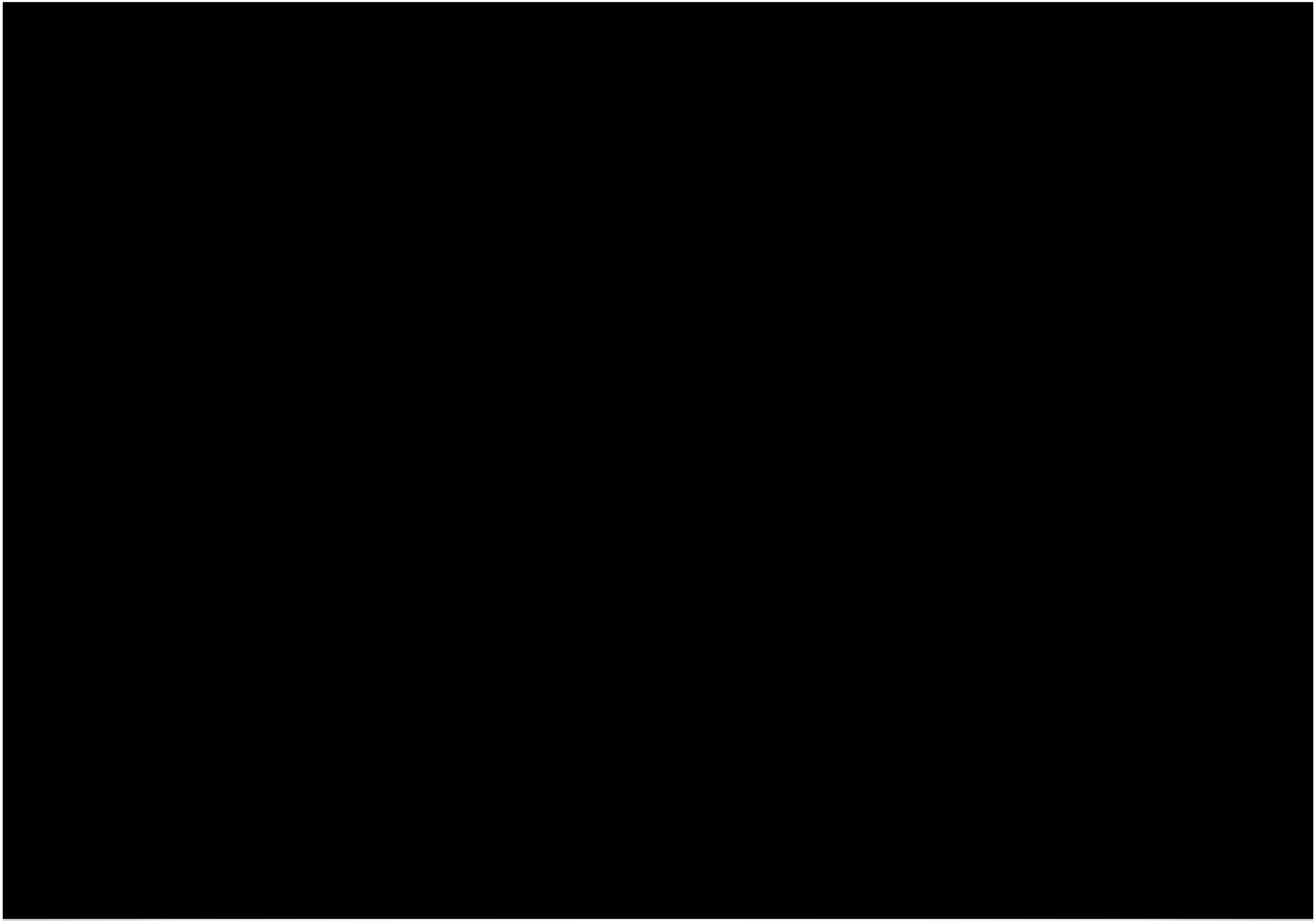


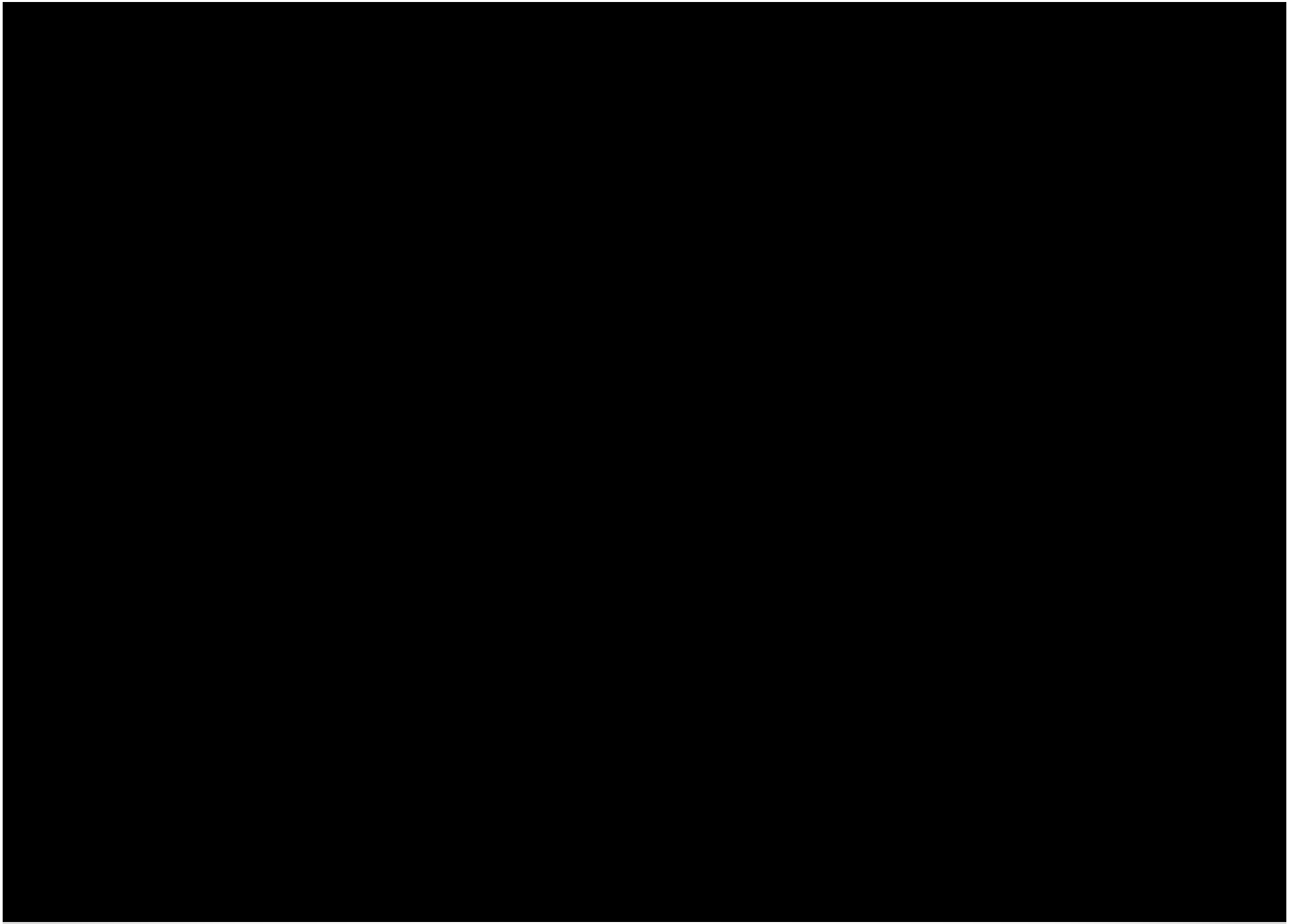
Transducer x→y



Done.

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





General n×m Join-Fork

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

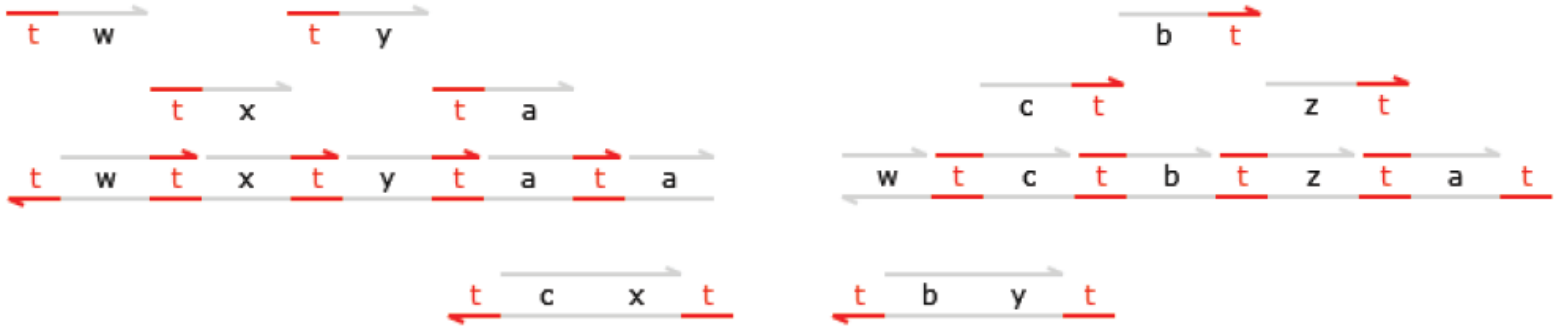
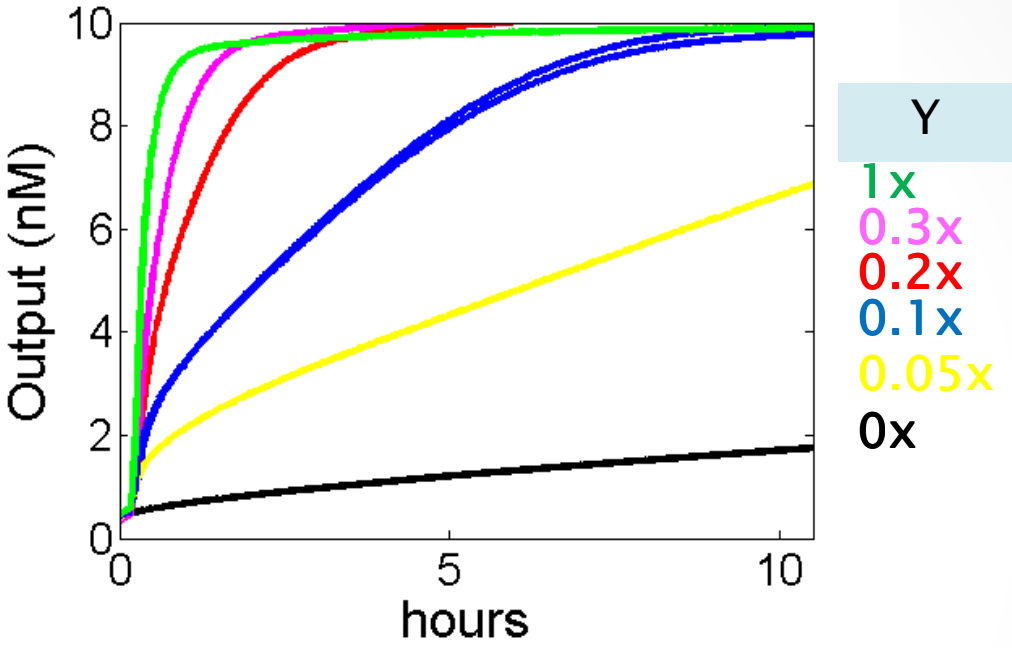


Figure 9: 3-Join $J_{wxyz} \mid tw \mid tx \mid ty \longrightarrow tz$: initial state plus inputs tw, tx, ty.

Experiments

Two-domain gate for $X+Y \rightarrow Y+B$

X+Y→Y+B
35C
1x = 50nM

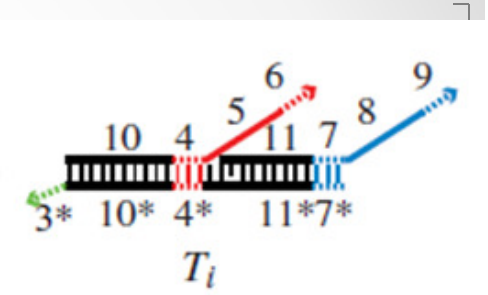


Yuan-Jyue Chen and Georg Seelig
U.Washingon.

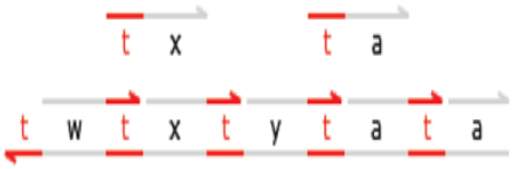
X+Y→Y+B		Concentration
LG1		1.5x
LG2		1.5x
input		1x
Catalyst		0x, 0.05x,0.1x,0.2x,0.3x,1x
~B		2x
R1		2x
B readout		3x

An Accident of Simplicity

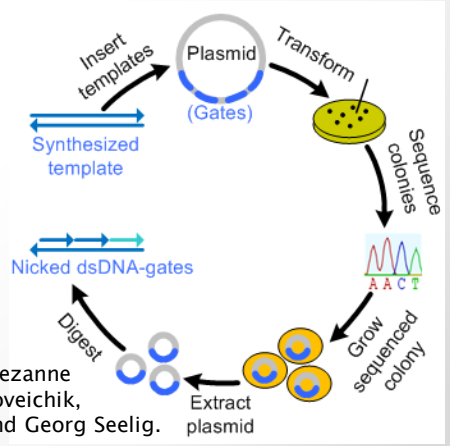
- Earlier architectures had ‘secondary structure’, which is ‘unnatural’:
  - It requires *synthetic* single-stranded DNA that is then assembled to form the desired structures.
  - Synthetic DNA has maximum length and quality problems (a fixed probability of synthesis error at each position, limiting size to about 200 bases).



- The two-domain architecture is (almost) ordinary biological DNA
  - Just double-stranded (with nicks), hence it can be produced *biologically*.
  - Biological DNA has much better quality and practically no length restriction: bacteria are so much better than we are at making it.



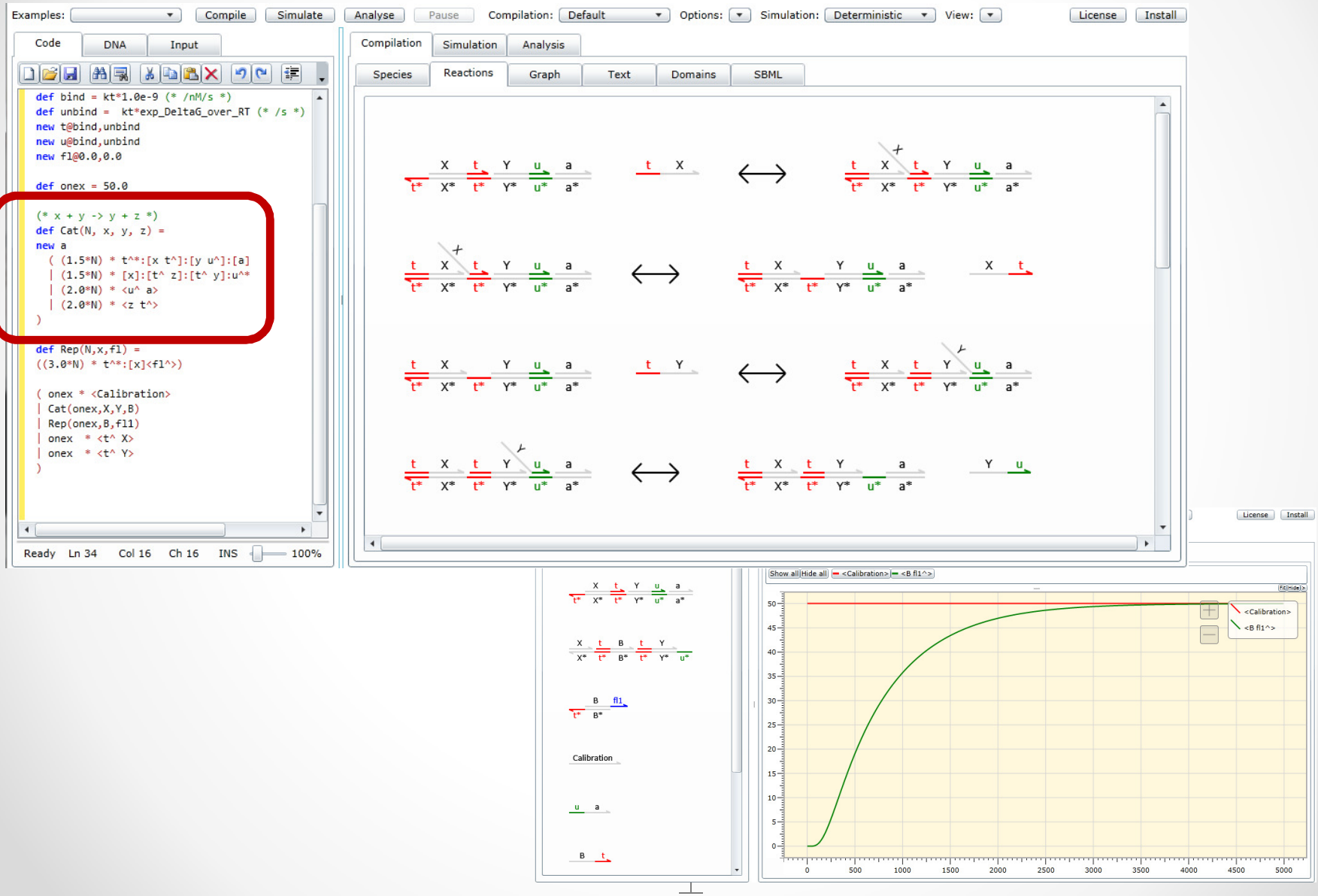
- Makes a new manufacturing technology possible
  - Gate-laden plasmids (circular DNA) are inserted into bacteria, who kindly produce large quantities of them overnight.
  - We then chop them up into gates and introduce the nicks via enzymes.



Yuan-Jyue Chen, Neil Dalchau, Cezanne Camacho, Matt Olson, David Soloveichik, Andrew Phillips, Luca Cardelli, and Georg Seelig.

DNA Programming

Strand Displacement Language



Formal Syntax and Semantics

D	syntax	description
M	N	Long Domain
	N~	Short domain
S	M	Domain
	M*	Complement Domain
	S1 S2	Concatenation of S1 and S2
L,R	_	Empty Concatenation
	S	Domain Concatenation

	syntax	description
A	<S> S	Upper strand with domain concatenation S
	{S} S	Lower strand with domain concatenation S
G	{L'}<L>[S]<R>{R'}	Double stranded complex [S] with overhanging single strands <L>, <R> and {L'}, {R'}
	G1:G2	Gates joined along a lower strand
	G1::G2	Gates joined along an upper strand
D	A	Strand A
	G	Gate G
	D1 | D2	Parallel systems D1, D2
	new N D	System D with private domain N
	X(ñ)	Module X with parameters ñ

before	rule	after
{L' N~* R'} | <L N~ R>	$RB,N+$	{L'}<L>[N~]<R>{R'}
{L'}<L>[N~]<R>{R'}	$RU,N-$	{L' N~* R'} | <L N~ R>
{L'}<L>[S]<N~ R>{N~* R'}	$RC,N^{\sim}$	{L'}<L>[S N~]<R>{R'}
{L'}<L>[S1]<S R2>:<L1>[S S2]<R>{R'}	$RM,S^{\sim}$	{L'}<L>[S1 S]<R2>:<L1 S>[S2]<R>{R'}
{L'}<L>[S1]<S R>:<L2>[S]<R2>{R'}	$RD,S^{\sim}$	<L2 S R2> | {L'}<L>[S1 S]<R>{R'}

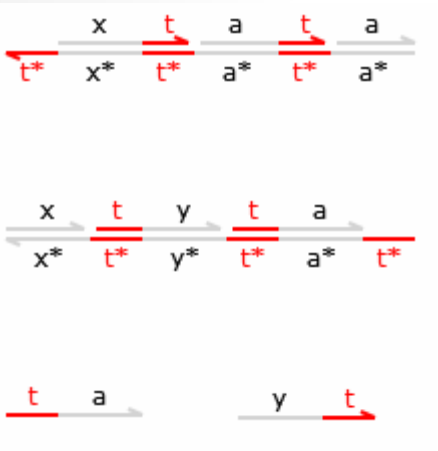
rule	condition	before	reduce	after
RGA1	{S1} | <S2> $\xrightarrow{R,r}$ G	<L>{S1}[S]{R'}<R> | <S2>	$\xrightarrow{R,r}$	G:<L>[S]{R'}<R>
RGA2	G $\xrightarrow{R,r}$ {S1} | <S2>	G:<L>[S]{R'}<R>	$\xrightarrow{R,r}$	<L>{S1}[S]{R'}<R> | <S2>
RGB	G | A $\xrightarrow{R,r}$ G'	U1:G:U2 | A	$\xrightarrow{R,r}$	U1:G':U2
RGU	G $\xrightarrow{R,r}$ G' | A	U1:G:U2	$\xrightarrow{R,r}$	U1:G':U2 | A
RGL	G | A $\xrightarrow{R,r}$ G' | A'	U1:G:U2 | A	$\xrightarrow{R,r}$	U1:G':U2 | A'
RG	G $\xrightarrow{R,r}$ G'	U1:G:U2	$\xrightarrow{R,r}$	U1:G':U2
RV	D $\xrightarrow{R,r}$ D'	rev(D)	$\xrightarrow{R,r}$	rev(D')
RC	D $\xrightarrow{R,r}$ D'	com(D)	$\xrightarrow{R,r}$	com(D')
RE	D1$\equiv_\sigma$D2$\xrightarrow{R,r}$D2'$\equiv_\sigma$D1'	D1	$\xrightarrow{R,r}$	D1'

rule	condition	before	equal	after
EC		D1 | D2	$\equiv_\sigma$	D2 | D1
EA		D1 | (D2 | D3)	$\equiv_\sigma$	(D1 | D2) | D3
ED	X(m) = D	X(n)	$\equiv_\sigma$	D{m:=n}
ENP	N $\notin$ fn(D2)	(new N D1) | D2	$\equiv_\sigma$	new N (D1 | D2)
ENN		new N1 new N2 D	$\equiv_\sigma$	new N2 new N1 D
END	N $\notin$ fn(D)	new N D	$\equiv_\sigma$	D
EP	D1 $\equiv_\sigma$ D1'	D1 | D2	$\equiv_\sigma$	D1' | D2
EN	D $\equiv_\sigma$ D'	new N D	$\equiv_\sigma$	new N D'
EL	G $\equiv_\sigma$ G'	G1:G	$\equiv_\sigma$	G1:G'
ER	G $\equiv_\sigma$ G'	G:G2	$\equiv_\sigma$	G':G2
EROTG		G	$\equiv_\sigma$	rotate(G)
EROTA		A	$\equiv_\sigma$	rotate(A)
ESL		{L1'}<L1>[S1]<R1>{R1' S}	$\equiv_\sigma$	{L1'}<L1>[S1]<R1>{R1'}
		:{L2'}<L2>[S2]<R2>{R2'}		:{S L2'}<L2>[S2]<R2>{R2'}
ESU		{L1'}<L1>[S1]<R1 S>{R1'}	$\equiv_\sigma$	{L1'}<L1>[S1]<R1>{R1'}
		::{L2'}<L2>[S2]<R2>{R2'}		::{L2'}<S L2>[S2]<R2>{R2'}

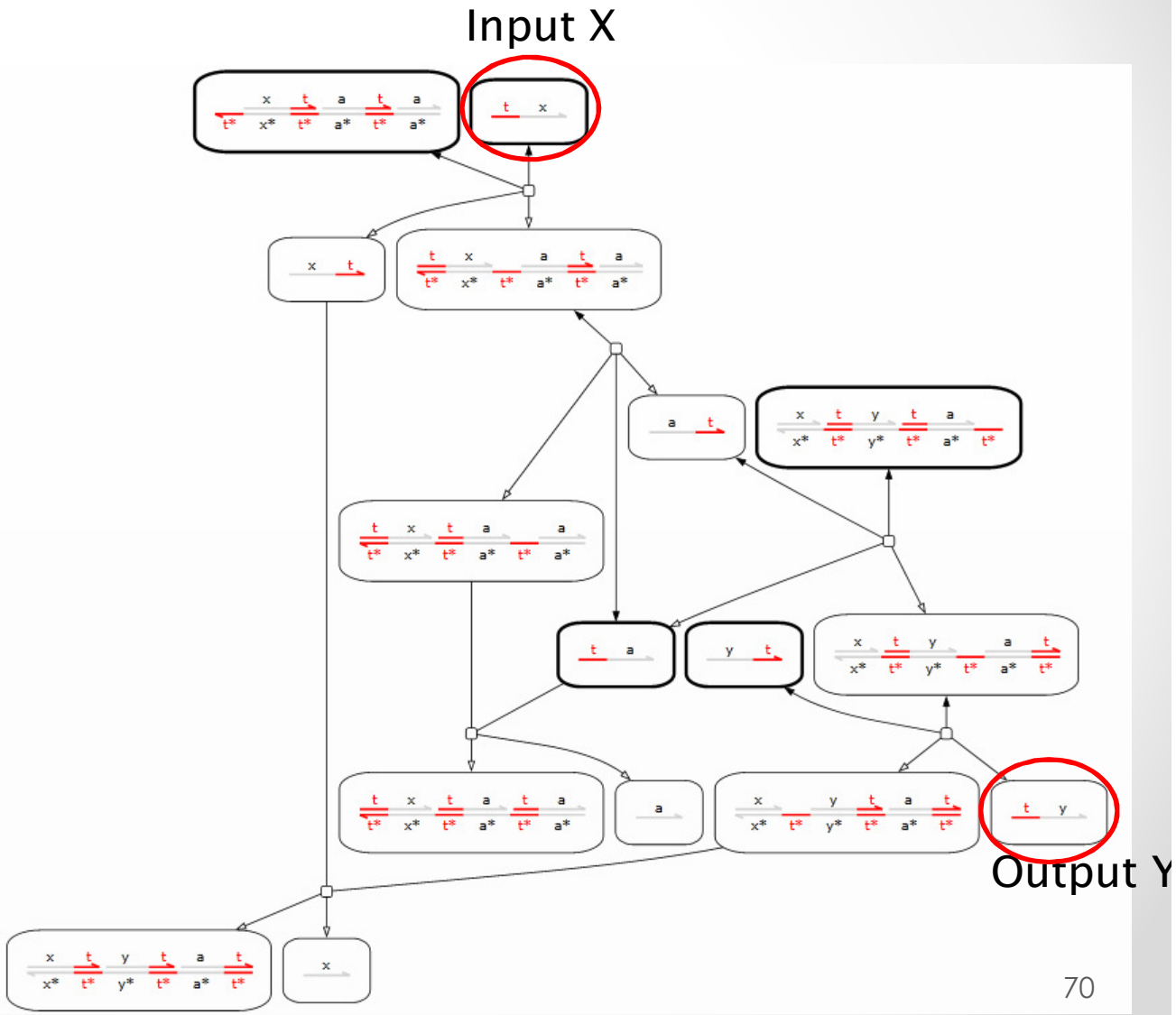
Lakin et al. Royal Society Interface, 2011

Compiling Chemistry to DNA (X→Y)

```
def R1x1(N,x,y) =
  new a
  ( N* <t^ a>
  | N* <y t^>
  | N* t^*:[x t^]:[a t^]:[a]
  | N* [x]:[t^ y]:[t^ a]:t^*
  )
```

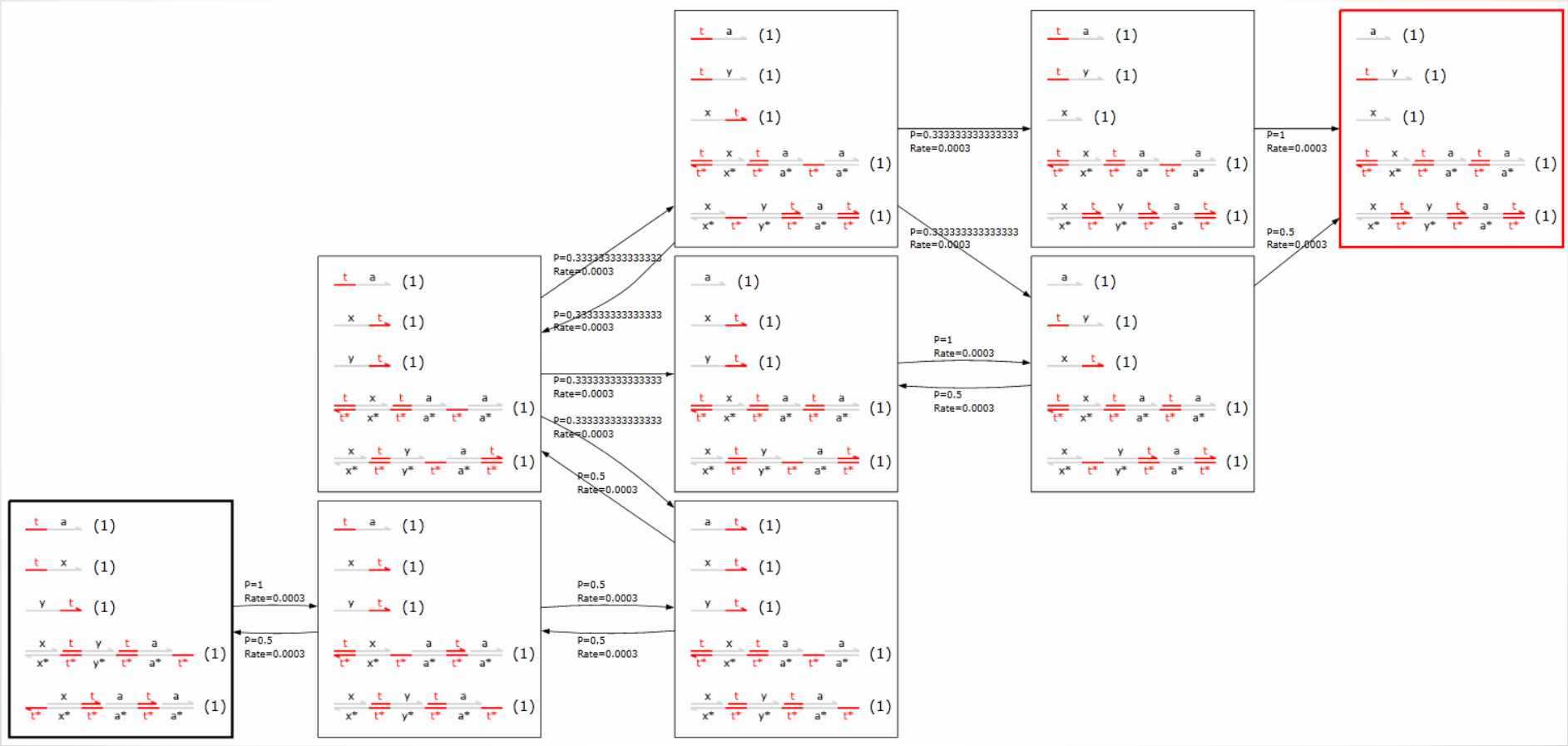



```
def Species(N,x) =
  N*<t^ x>
```



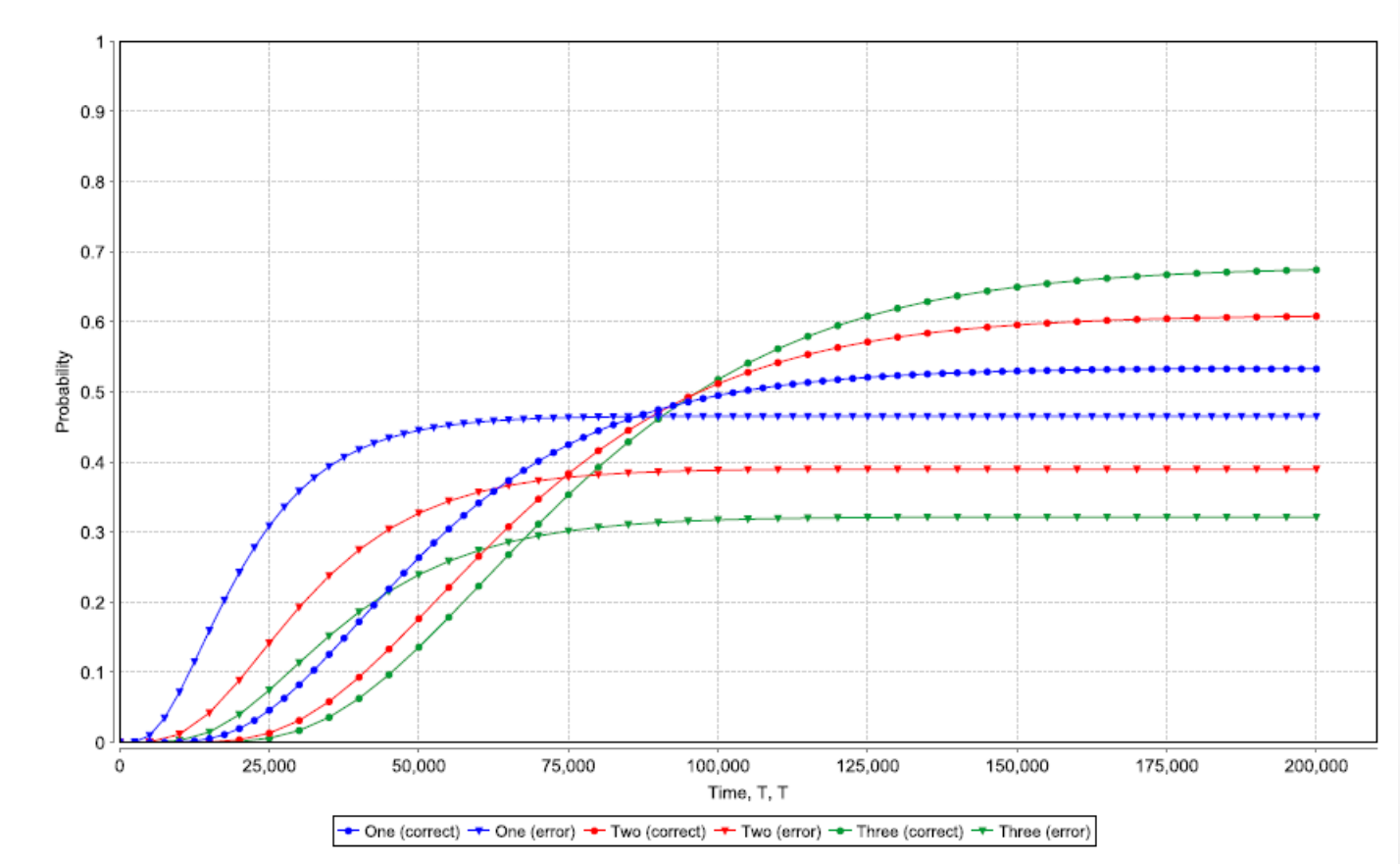
Model-Checking Compilation (X→Y)

Transducer State Space (Species(1,x) | R1x1(1,x,y))



Stochastic Model Checking

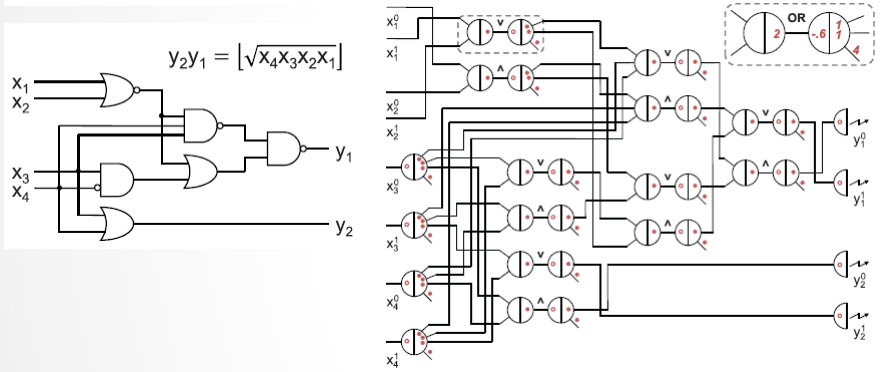
PRISM results for sequential transducers

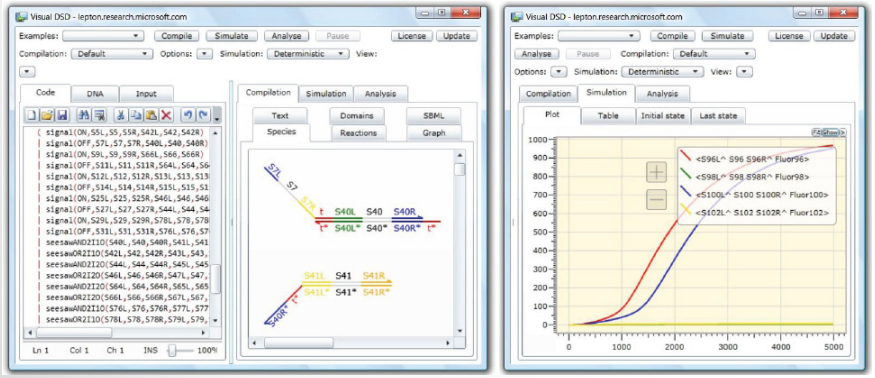


Scaling Strand Displacement Circuits

Scaling Up Digital Circuit Computation with DNA Strand Displacement Cascades

Lulu Qian[1] and Erik Winfree[1,2,3]*





3 JUNE 2011 VOL 332 SCIENCE

Scaling Up DNA Computation

John H. Reif

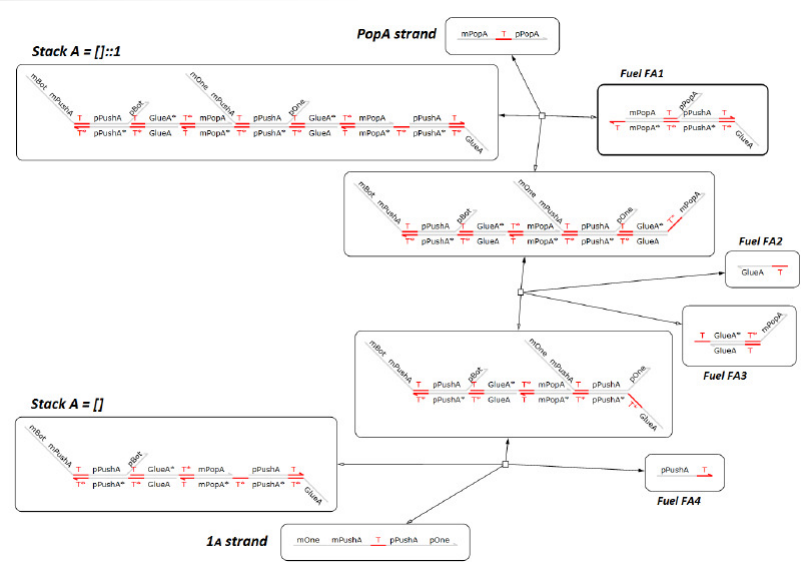
"In addition to biochemistry laboratory techniques, computer science techniques were essential."

"Computer simulations of seesaw gate circuitry optimized
the design and correlated experimental data."

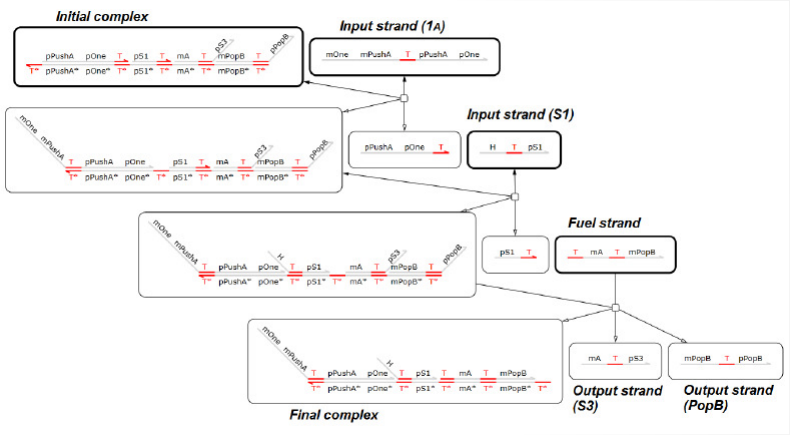
3 JUNE 2011 VOL 332 SCIENCE

Turing-Powerful DNA Computers

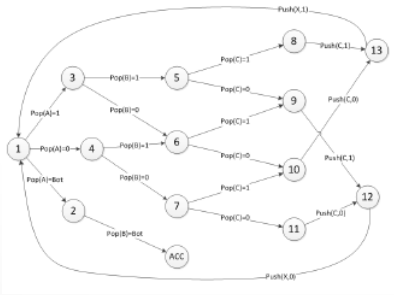
Encoding a Stack



Encoding state transitions



Model-Checking a DNA Ripple Carry Adder



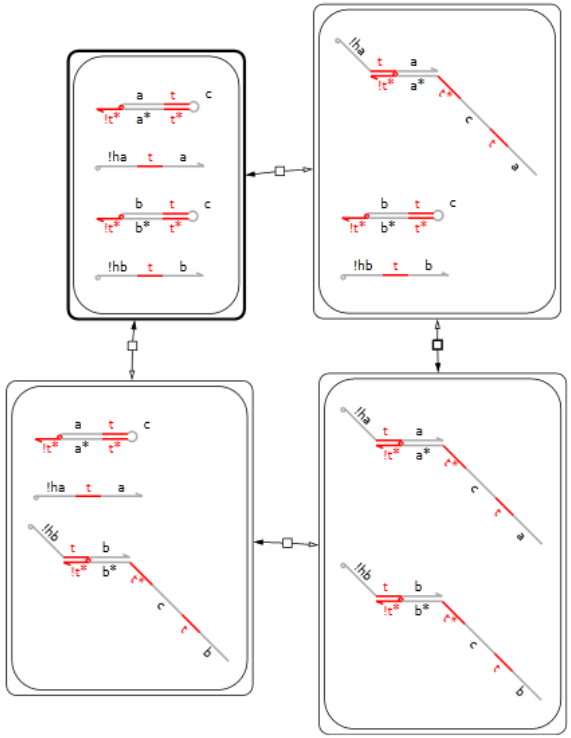
Input A MSB	Input A LSB	Input A Value	Input B MSB	Input B LSB	Input B Value	Output X LSB	Output X MSB	Output C Value	Result Value
0	0	0	0	0	0	**0**	**0**	**0**	**0**
0	0	0	0	1	1	**1**	**0**	**0**	**1**
0	0	0	1	0	2	**0**	**1**	**0**	**2**
0	0	0	1	1	3	**1**	**1**	**0**	**3**
0	1	1	0	0	0	**1**	**0**	**0**	**1**
0	1	1	0	1	1	**0**	**1**	**0**	**2**
0	1	1	1	0	2	**1**	**1**	**0**	**3**
0	1	1	1	1	3	**0**	**0**	**1**	**4**
1	0	2	0	0	0	**0**	**1**	**0**	**2**
1	0	2	0	1	1	**1**	**1**	**0**	**3**
1	0	2	1	0	2	**0**	**0**	**1**	**4**
1	0	2	1	1	3	**1**	**0**	**1**	**5**
1	1	3	0	0	0	**1**	**1**	**0**	**3**
1	1	3	0	1	1	**0**	**0**	**1**	**4**
1	1	3	1	0	2	**1**	**0**	**1**	**5**
1	1	3	1	1	3	**0**	**1**	**1**	**6**

Lakin & Phillips, DNA17 2011

Localised circuits

Hairpins tethered to origami

- Increased speed
- Reduced interference

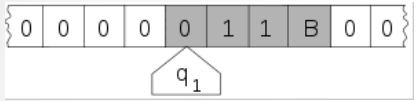


Chandran,Gopalkrishnan,Phillips,Reif. DNA Computing, 2011

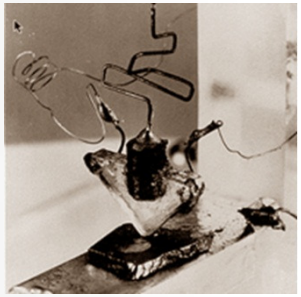
Conclusions

A Brief History of DNA

Turing Machine, 1936

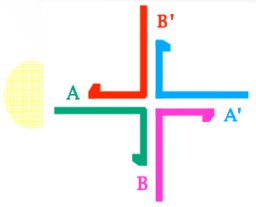


Transistor, 1947

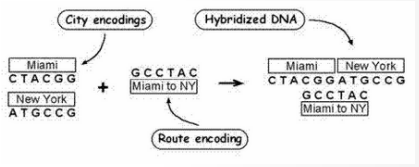


DNA, ~3,800,000,000

Structural DNA, 1982



DNA Algorithm, 1994



Digital ~~Computers~~

Computer programming

Software

systematic manipulation of information

20th century

Matterware??

systematic manipulation of matter

21th century

DNA ~~Computers~~

Molecular programming

Acknowledgments

- Microsoft Research
  - Andrew Phillips
    - Languages and tools for DNA strand displacement.
- Bologna
  - Pierluigi Zavattaro
    - Computational power of ‘chemical’ process algebras.
  - Cosimo Laneve
    - Reversibility in population models.
- Aalborg
  - Radu Mardare
    - Stochastic process algebra and logic.
- Caltech
  - Erik Winfree & Winfree Lab
    - DNA strand displacement as a computational method and technology.
  - David Soloveichik
    - The Programming Language of Chemical Kinetics.
- U.Washington
  - Georg Seelig, Yuan-Jyue Chen
    - Manufacturing two-domain gates.