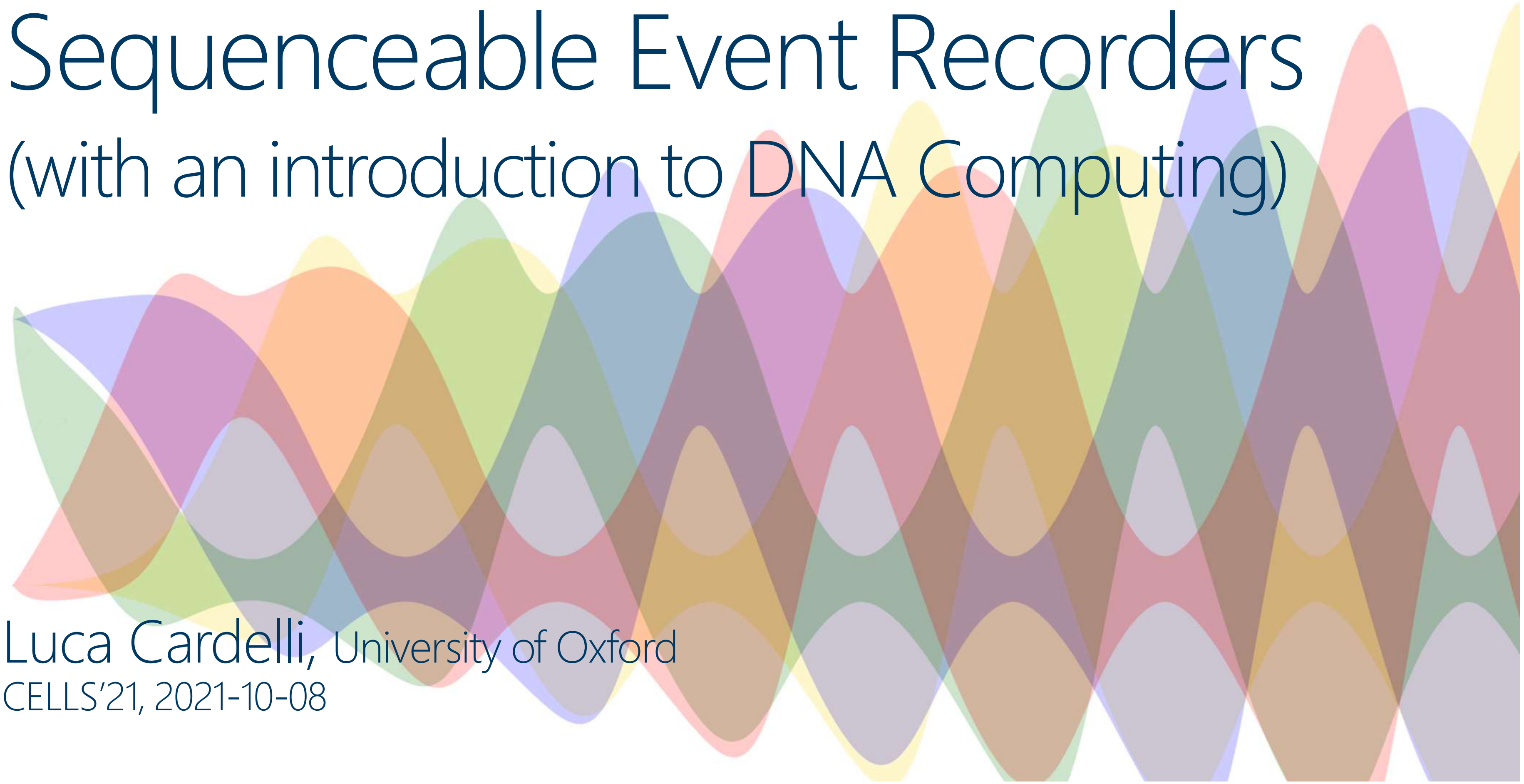
Sequenceable Event Recorders

(with an introduction to DNA Computing)

Luca Cardelli, University of Oxford

CELLS'21, 2021-10-08



Outline

1. From “any” Digital or Analog system
   to a Chemical Reaction Networks

2. From (made-up) Chemical Reaction Networks
   to (real) Molecules that implement them

3. Detecting Molecular Events

Part 1

From (almost) any algorithm and (almost) any dynamical system to a Chemical Reaction Network

Chemical Reaction Networks (CRN)

$$X + Y \ \text{->}^{r} \ Z + W$$

- A *phenomenological model* of kinetics in the natural sciences

  By *(only) observing* naturally occurring reactions

- A *programming language*, *finitely* encoded in the genome

  By which living things manage the *unbounded* processing of matter and information

- A *mathematical structure*, rediscovered in many forms

  Vector Addition Systems, Petri Nets, Bounded Context-Free Languages, Population Protocols, ...

- A description of *mechanism* ("instructions" / "interactions") rather than *behavior* ("equations" / "approximations")

  Although the two are related in precise ways

  Enabling, e.g., the study of the evolution of *mechanism* through unchanging *behavior*

Programming Examples

spec	*program*
$Y := 2X$	X -> Y + Y
$Y := \lfloor X/2 \rfloor$	X + X -> Y
$Y := X1 + X2$	X1 -> Y X2 -> Y
$Y := \min(X1, X2)$	X1 + X2 -> Y

Advanced Programming Examples

spec

program

Y := max(X1, X2)

```
X1 -> L1 + Y
X2 -> L2 + Y
L1 + L2 -> K
Y + K -> 0
```

max(X1,X2)=
(X1+X2)-min(X1,X2)

(but is not computed
"sequentially")

Approximate Majority

```
(X,Y) :=
        if X≥Y then (X+Y, 0)
        if Y≥X then (0, X+Y)
```

```
X + Y -> Y + B
Y + X -> X + B
B + X -> X + X
B + Y -> Y + Y
```

"approximately"

Programming *any* algorithm as a CRN

A CRN is a *finite* set of reactions over a *finite* set of species

CRNs are not Turing complete

Like Petri nets: reachability is decidable

But unlike Petri nets, CRNs are *approximately* Turing complete

Because reactions have also *rates*

This make it possible to approximate Turing completeness by approximating test-for-zero in a register machine.
The probability of error (in test-for-zero) can be made arbitrarily small over the entire (undecidably long) computation.

Adding polymerization to the model makes it fully Turing complete

Register Machines (almost...)

i: INC R_1; JMP j	$PC_i \rightarrow R_1 + PC_j$
i: DEC R_1; JMP j	$PC_i + R_1 \rightarrow PC_j$
i: IF $R_2>0$ {INC R_1; JMP j}	$PC_i + R_2 \rightarrow R_2 + R_1 + PC_j$
i: IF $R_2=0$ ...	??? Whatever trick we use will have some error

- Turing-complete up to an arbitrarily small error
  - The error bound is set in advance uniformly for any computation of arbitrary length (because we cannot know how long the computation will last), and the machine will progressively "slow down" to always stay below that bound.

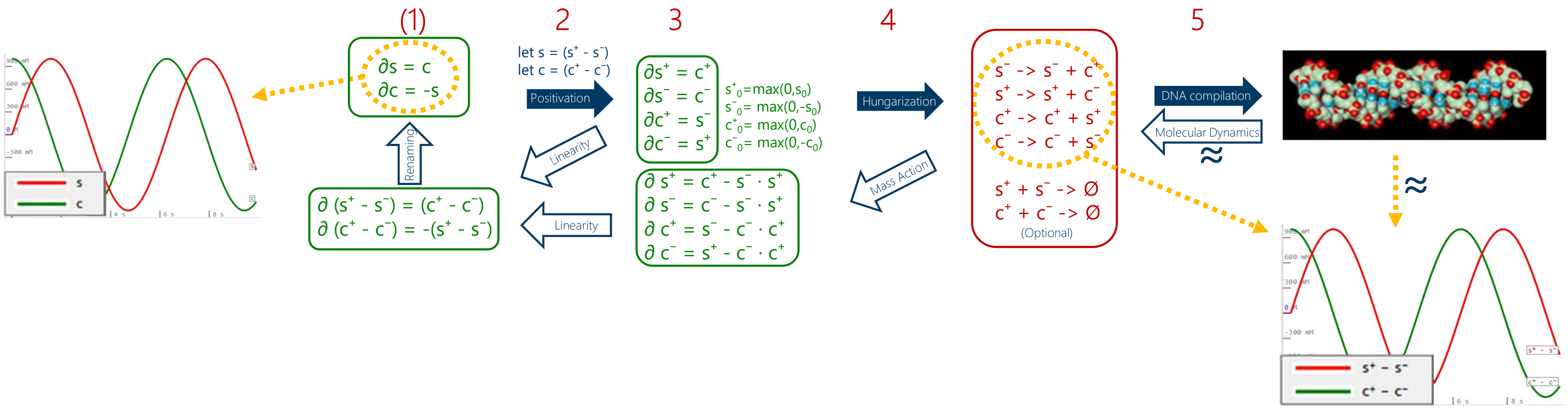
- David Soloveichik, Matt Cook, Erik Winfree, Shuki Bruck, "Computation with Finite Stochastic Chemical Reaction Networks".
[Natural Computing, (online Feb 2008), or Technical Report: CaltechPARADISE:2007.ETR085: .pdf]

Programming *any* "elementary" dynamical system as a CRN

$\partial^2\theta = -g/l \sin\theta$

Equation of motion of a simple pendulum

For example, take *the* canonical oscillator: sine/cosine



(1)

$\partial s = c$
$\partial c = -s$

Renaming

$\partial (s^+ - s^-) = (c^+ - c^-)$
$\partial (c^+ - c^-) = -(s^+ - s^-)$

2

let s = $(s^+ - s^-)$
let c = $(c^+ - c^-)$

Positivation

Linearity

Linearity

3

$\partial s^+ = c^+$
$\partial s^- = c^-$
$\partial c^+ = s^-$
$\partial c^- = s^+$

$s^+_0 = \max(0, s_0)$
$s^-_0 = \max(0, -s_0)$
$c^+_0 = \max(0, c_0)$
$c^-_0 = \max(0, -c_0)$

$\partial s^+ = c^+ - s^- \cdot s^+$
$\partial s^- = c^- - s^- \cdot s^+$
$\partial c^+ = s^- - c^- \cdot c^+$
$\partial c^- = s^+ - c^- \cdot c^+$

4

Hungarization

Mass Action

$s^- \rightarrow s^- + c^+$
$s^+ \rightarrow s^+ + c^-$
$c^+ \rightarrow c^+ + s^+$
$c^- \rightarrow c^- + s^-$

$s^+ + s^- \rightarrow \emptyset$
$c^+ + c^- \rightarrow \emptyset$
(Optional)

5

DNA compilation

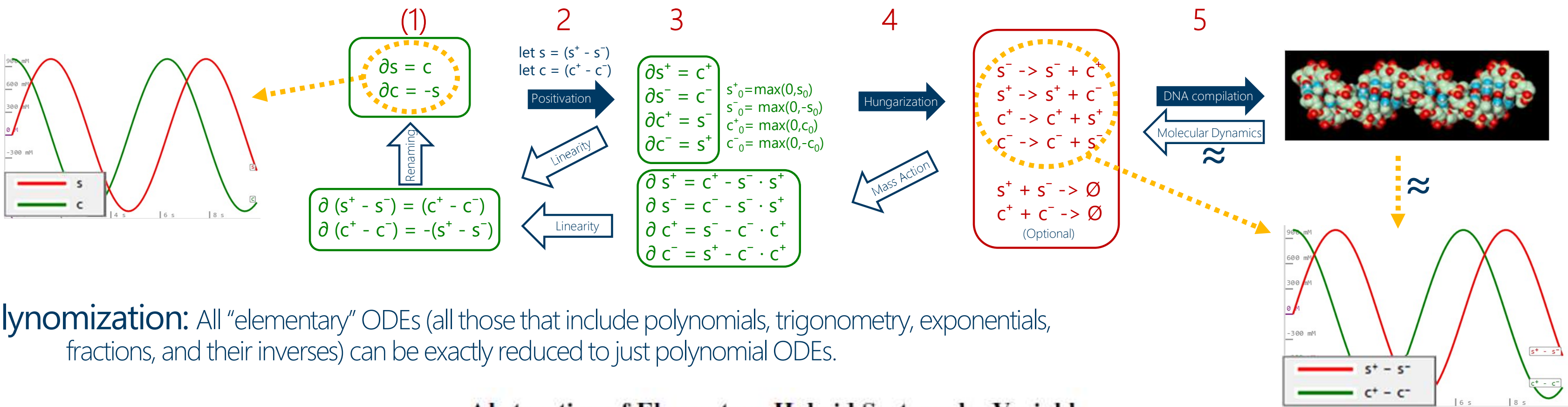
Molecular Dynamics

≈

≈

Programming *any* *"elementary"* dynamical system as a CRN

For example, take *the* canonical oscillator: sine/cosine



(1)

$\partial s = c$
$\partial c = -s$

$\partial (s^+ - s^-) = (c^+ - c^-)$
$\partial (c^+ - c^-) = -(s^+ - s^-)$

2

let $s = (s^+ - s^-)$
let $c = (c^+ - c^-)$

3

$\partial s^+ = c^+$
$\partial s^- = c^-$
$\partial c^+ = s^-$
$\partial c^- = s^+$

$s^+_0 = \max(0, s_0)$
$s^-_0 = \max(0, -s_0)$
$c^+_0 = \max(0, c_0)$
$c^-_0 = \max(0, -c_0)$

$\partial s^+ = c^+ - s^- \cdot s^+$
$\partial s^- = c^- - s^- \cdot s^+$
$\partial c^+ = s^- - c^- \cdot c^+$
$\partial c^- = s^+ - c^- \cdot c^+$

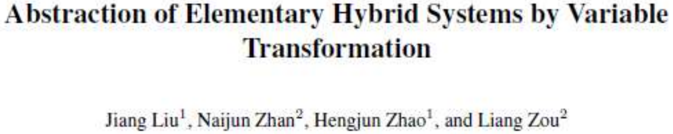
4

$s^- \to s^- + c^+$
$s^+ \to s^+ + c^-$
$c^+ \to c^+ + s^+$
$c^- \to c^- + s^-$

$s^+ + s^- \to \varnothing$
$c^+ + c^- \to \varnothing$
(Optional)

5

1. Polynomization: All "elementary" ODEs (all those that include polynomials, trigonometry, exponentials, fractions, and their inverses) can be exactly reduced to just polynomial ODEs.

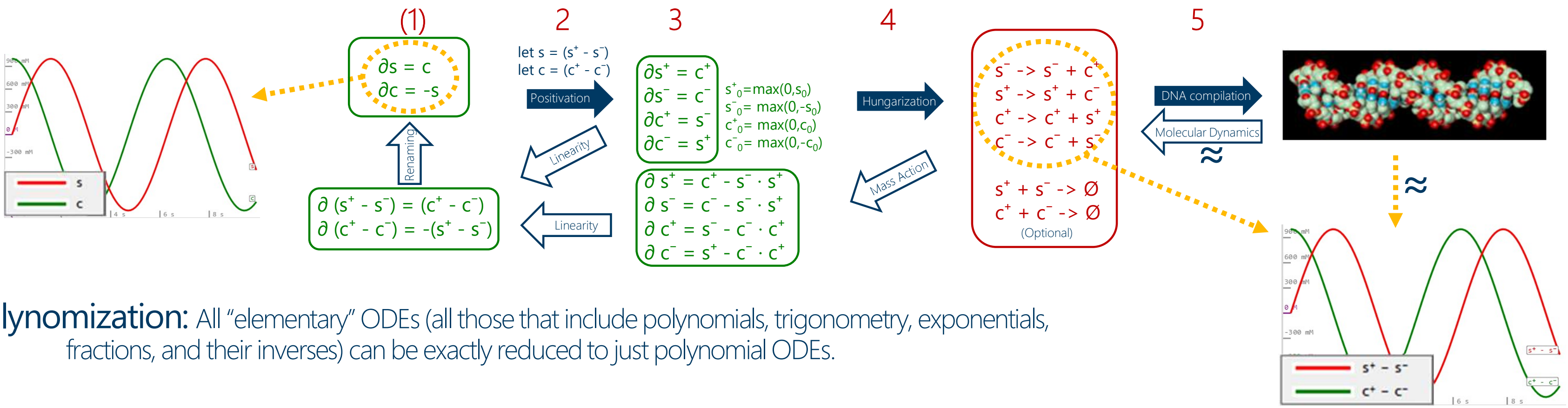


Abstraction of Elementary Hybrid Systems by Variable Transformation

Jiang Liu[1], Naijun Zhan[2], Hengjun Zhao[1], and Liang Zou[2]

Programming *any* "elementary" dynamical system as a CRN

For example, take *the* canonical oscillator: sine/cosine



(1)

$\partial s = c$

$\partial c = -s$

Renaming

$\partial (s^+ - s^-) = (c^+ - c^-)$

$\partial (c^+ - c^-) = -(s^+ - s^-)$

2

let $s = (s^+ - s^-)$

let $c = (c^+ - c^-)$

Positivation

Linearity

Linearity

3

$\partial s^+ = c^+$

$\partial s^- = c^-$

$\partial c^+ = s^-$

$\partial c^- = s^+$

$s^+_0 = \max(0, s_0)$

$s^-_0 = \max(0, -s_0)$

$c^+_0 = \max(0, c_0)$

$c^-_0 = \max(0, -c_0)$

$\partial s^+ = c^+ - s^- \cdot s^+$

$\partial s^- = c^- - s^- \cdot s^+$

$\partial c^+ = s^- - c^- \cdot c^+$

$\partial c^- = s^+ - c^- \cdot c^+$

4

Hungarization

Mass Action

$s^- \to s^- + c^+$

$s^+ \to s^+ + c^-$

$c^+ \to c^+ + s^+$

$c^- \to c^- + s^-$

$s^+ + s^- \to \varnothing$

$c^+ + c^- \to \varnothing$

(Optional)

5

DNA compilation

Molecular Dynamics

≈

≈

1. **Polynomization:** All "elementary" ODEs (all those that include polynomials, trigonometry, exponentials, fractions, and their inverses) can be exactly reduced to just polynomial ODEs.

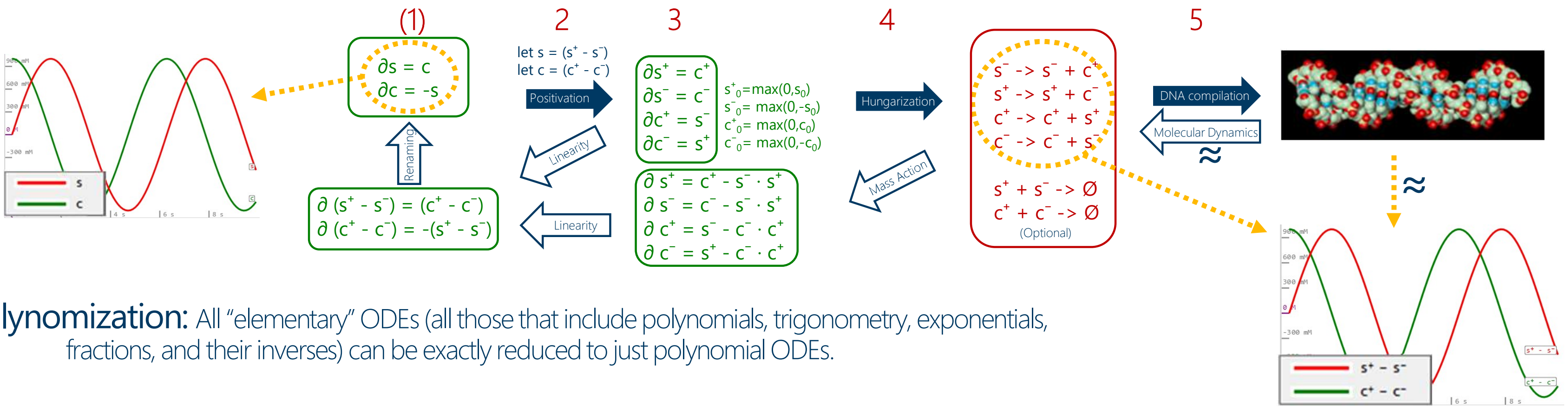
2. **Positivation:** All polynomial ODEs can be exactly reduced to polynomial ODEs in the positive quadrant (as differences).

Biomolecular implementation of linear I/O systems

K. Oishi E. Klavins

Programming *any* "elementary" dynamical system as a CRN

For example, take *the* canonical oscillator: sine/cosine



(1)

$$\partial s = c \qquad \partial c = -s$$

$$\partial (s^+ - s^-) = (c^+ - c^-) \qquad \partial (c^+ - c^-) = -(s^+ - s^-)$$

2

let $s = (s^+ - s^-)$

let $c = (c^+ - c^-)$

3

$$\partial s^+ = c^+ \qquad \partial s^- = c^- \qquad \partial c^+ = s^- \qquad \partial c^- = s^+$$

$s^+_0 = \max(0, s_0)$

$s^-_0 = \max(0, -s_0)$

$c^+_0 = \max(0, c_0)$

$c^-_0 = \max(0, -c_0)$

$$\partial s^+ = c^+ - s^- \cdot s^+ \qquad \partial s^- = c^- - s^- \cdot s^+ \qquad \partial c^+ = s^- - c^- \cdot c^+ \qquad \partial c^- = s^+ - c^- \cdot c^+$$

4

$s^- \rightarrow s^- + c^+$

$s^+ \rightarrow s^+ + c^-$

$c^+ \rightarrow c^+ + s^+$

$c^- \rightarrow c^- + s^-$

$s^+ + s^- \rightarrow Ø$

$c^+ + c^- \rightarrow Ø$

(Optional)

5

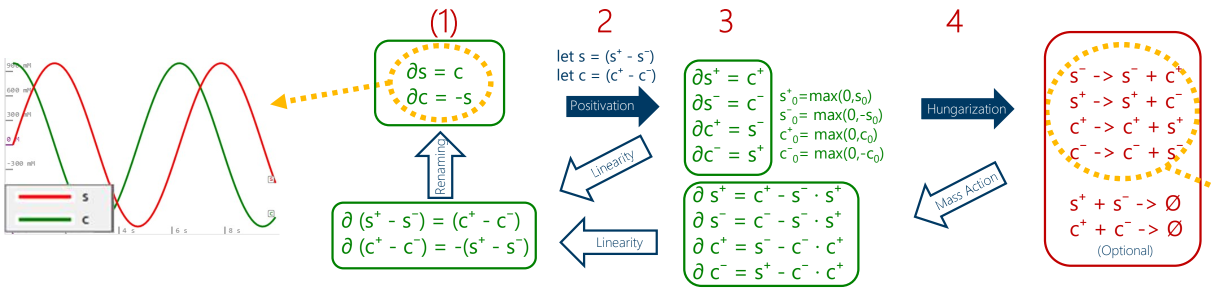
1. **Polynomization:** All "elementary" ODEs (all those that include polynomials, trigonometry, exponentials, fractions, and their inverses) can be exactly reduced to just polynomial ODEs.

2. **Positivation:** All polynomial ODEs can be exactly reduced to polynomial ODEs in the positive quadrant (as differences).

3. **All positivized ODEs are Hungarian:** I.e., all negative monomials have their l.h.s. differential variable as a factor.

Programming *any* "elementary" dynamical system as a CRN

For example, take *the* canonical oscillator: sine/cosine



$$\partial s = c \qquad \partial c = -s$$

$$\partial(s^+ - s^-) = (c^+ - c^-) \qquad \partial(c^+ - c^-) = -(s^+ - s^-)$$

let $s = (s^+ - s^-)$
let $c = (c^+ - c^-)$

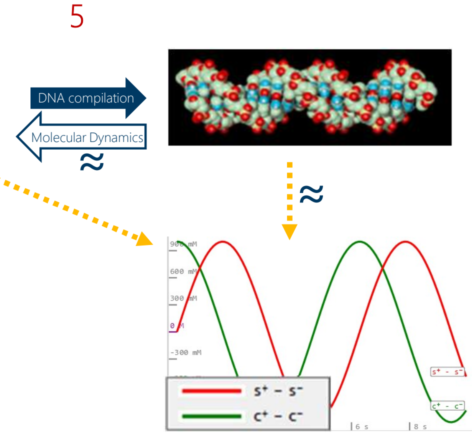
$$\partial s^+ = c^+ \qquad \partial s^- = c^- \qquad \partial c^+ = s^- \qquad \partial c^- = s^+$$

$s^+_0 = \max(0, s_0)$, $s^-_0 = \max(0, -s_0)$, $c^+_0 = \max(0, c_0)$, $c^-_0 = \max(0, -c_0)$

$$\partial s^+ = c^+ - s^- \cdot s^+ \qquad \partial s^- = c^- - s^- \cdot s^+ \qquad \partial c^+ = s^- - c^- \cdot c^+ \qquad \partial c^- = s^+ - c^- \cdot c^+$$

$s^- \to s^- + c^+$
$s^+ \to s^+ + c^-$
$c^+ \to c^+ + s^+$
$c^- \to c^- + s^-$

$s^+ + s^- \to \emptyset$
$c^+ + c^- \to \emptyset$
(Optional)



1. **Polynomization:** All "elementary" ODEs (all those that include polynomials, trigonometry, exponentials, fractions, and their inverses) can be exactly reduced to just polynomial ODEs.

2. **Positivation:** All polynomial ODEs can be exactly reduced to polynomial ODEs in the positive quadrant (as differences).

3. **All positivized ODEs are Hungarian:** I.e., all negative monomials have their l.h.s. differential variable as a factor.

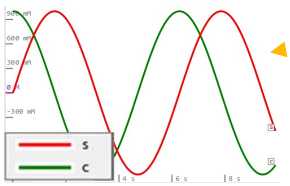
4. **Hungarization:** All Hungarian ODEs can be exactly reduced to mass action CRNs.

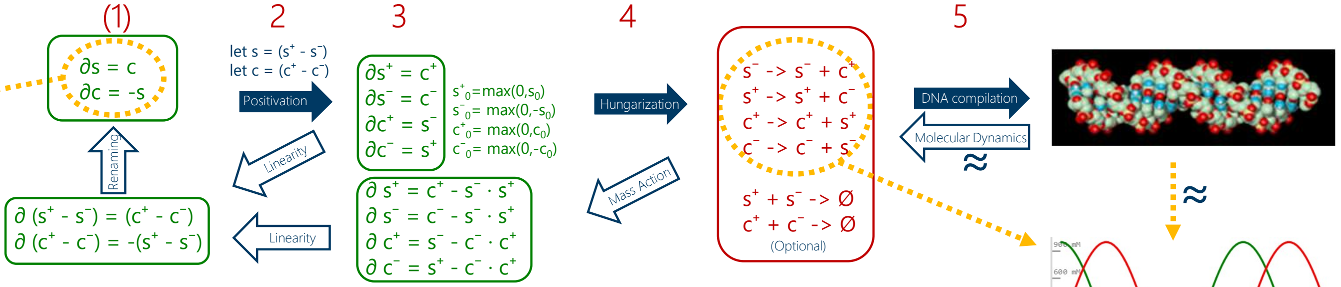
ON THE INVERSE PROBLEM OF REACTION KINETICS

V. HÁRS - J. TÓTH

Programming *any* *"elementary"* dynamical system as a CRN

For example, take *the* canonical oscillator: sine/cosine





$\partial s = c$
$\partial c = -s$

$\partial (s^+ - s^-) = (c^+ - c^-)$
$\partial (c^+ - c^-) = -(s^+ - s^-)$

let $s = (s^+ - s^-)$
let $c = (c^+ - c^-)$

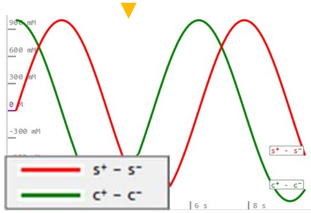
$\partial s^+ = c^+$
$\partial s^- = c^-$
$\partial c^+ = s^-$
$\partial c^- = s^+$

$s^+_0 = \max(0, s_0)$
$s^-_0 = \max(0, -s_0)$
$c^+_0 = \max(0, c_0)$
$c^-_0 = \max(0, -c_0)$

$\partial s^+ = c^+ - s^- \cdot s^+$
$\partial s^- = c^- - s^- \cdot s^+$
$\partial c^+ = s^- - c^- \cdot c^+$
$\partial c^- = s^+ - c^- \cdot c^+$

$s^- \to s^- + c^+$
$s^+ \to s^+ + c^-$
$c^+ \to c^+ + s^+$
$c^- \to c^- + s^-$

$s^+ + s^- \to \emptyset$
$c^+ + c^- \to \emptyset$
(Optional)



1. Polynomization: All "elementary" ODEs (all those that include polynomials, trigonometry, exponentials, fractions, and their inverses) can be exactly reduced to just polynomial ODEs.

2. Positivation: All polynomial ODEs can be exactly reduced to polynomial ODEs in the positive quadrant (as differences).

3. All positivized ODEs are Hungarian: I.e., all negative monomials have their l.h.s. differential variable as a factor.

4. Hungarization: All Hungarian ODEs can be exactly reduced to mass action CRNs.

5. Molecular Programming: All mass action CRNs, up to time rescaling, can be arbitrarily approximated by engineered DNA molecules.

DNA as a universal substrate for chemical kinetics

David Soloveichik, Georg Seelig, and Erik Winfree

PNAS March 23, 2010 107 (12) 5393-5398; https://doi.org/10.1073/pnas.0909380107

CRN Semantics (deterministic)

- ODE semantics of CRNs

State produced by a CRN $\mathcal{C} = (\mathcal{A}, \mathcal{R})$ (species $\mathcal{A}$, reactions $\mathcal{R}$)
with flux F (r.h.s. of its mass action ODEs) at time *t*,
from initial state (x_0, V, T) (initial concentrations x_0, volume *V*, temperature *T*):

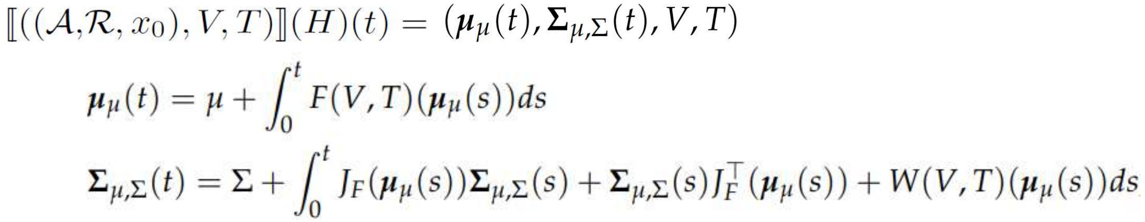
$$[\![((\mathcal{A},\mathcal{R}, x_0), V, T)]\!](H)(t) = (G(t), V, T)$$

$$let\, G : [0...H) \rightarrow \mathbb{R}^{|\mathcal{A}|} \; be \; the \; solution \; of \, G(t') = x_0 + \int_0^{t'} F(V,T)(G(s))ds$$

CRN Semantics (stochastic)

- CME semantics of CRNs (Chemical Master Equation)
  - Kolmogorov forward equation of the Markov Chain produced by the CRN
  - Unfeasible to solve or even simulate (to compute the distribution of outcomes)
  - The Gillespie algorithm produces individual samples (traces) of the CME distribution
- LNA semantics of CRNs (Linear Noise Approximation)

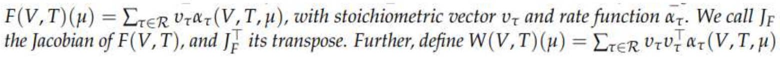
Gaussian state (mean & variance) produced by a CRN $\mathcal{C} = (\mathcal{A}, \mathcal{R})$ (species $\mathcal{A}$, reactions $\mathcal{R}$) with flux F (r.h.s. of its mass action ODEs) at time *t*, *with* $\boldsymbol{\mu}_\mu(0) = \mu$ *and* $\boldsymbol{\Sigma}_{\mu,\Sigma}(0) = \Sigma$.



$$[\![((\mathcal{A},\mathcal{R},x_0),V,T)]\!](H)(t) = (\boldsymbol{\mu}_\mu(t), \boldsymbol{\Sigma}_{\mu,\Sigma}(t), V, T)$$

$$\boldsymbol{\mu}_\mu(t) = \mu + \int_0^t F(V,T)(\boldsymbol{\mu}_\mu(s))ds$$

$$\boldsymbol{\Sigma}_{\mu,\Sigma}(t) = \Sigma + \int_0^t J_F(\boldsymbol{\mu}_\mu(s))\boldsymbol{\Sigma}_{\mu,\Sigma}(s) + \boldsymbol{\Sigma}_{\mu,\Sigma}(s)J_F^\top(\boldsymbol{\mu}_\mu(s)) + W(V,T)(\boldsymbol{\mu}_\mu(s))ds$$



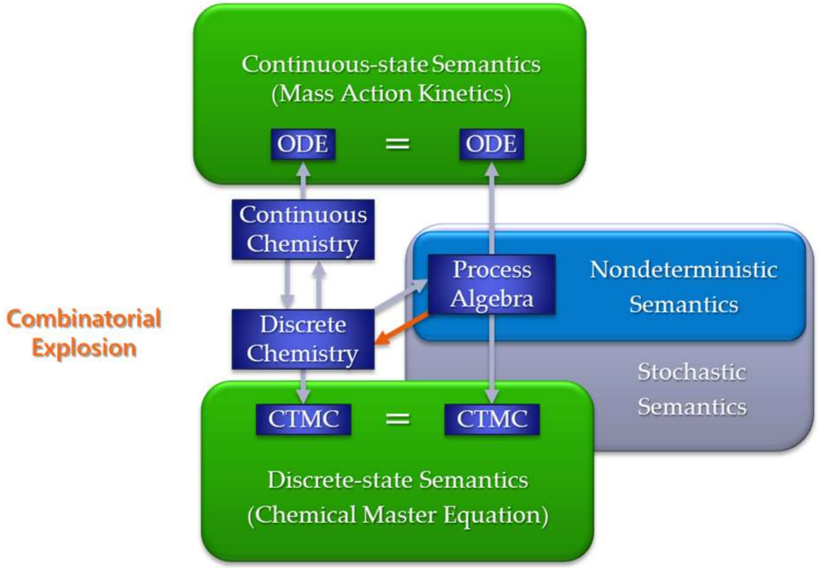
$F(V,T)(\mu) = \sum_{\tau\in\mathcal{R}} v_\tau\alpha_\tau(V,T,\mu)$, *with stoichiometric vector* v_τ *and rate function* $\bar{\alpha}_\tau$. *We call* J_F *the Jacobian of* $F(V,T)$, *and* $J_F^\top$ *its transpose. Further, define* $W(V,T)(\mu) = \sum_{\tau\in\mathcal{R}} v_\tau v_\tau^\top \alpha_\tau(V,T,\mu)$

A Language for Modeling And Optimizing Experimental Biological Protocols

Luca Cardelli [1], Marta Kwiatkowska [1] and Luca Laurenti [1,†]

Chemistry as a Concurrent Language

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



Finally, Some *Bad* Programs

X -> Y

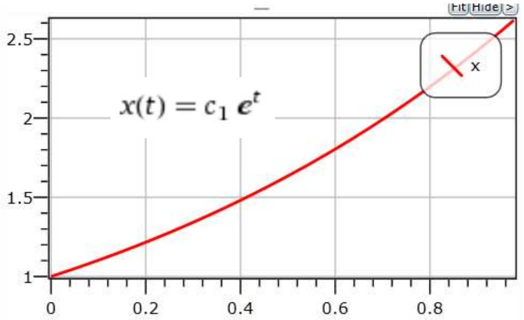
Violates thermodynamics.
(No biggie, assume there is a tiny reverse reaction.)

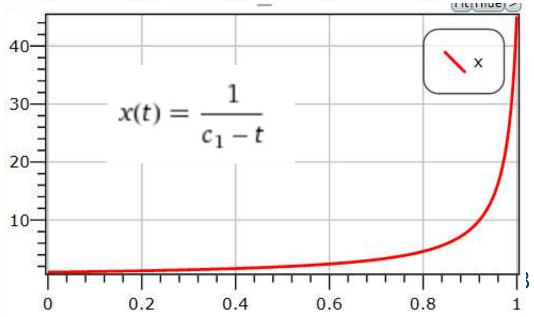
X -> X + X

Violates conservation of mass.
(No biggie, assume there is inflow/outflow.)

X + X -> X + X + X

Violates finite density.
(This is *really* bad.)





Chemistry is (also) a formal language that we can use to implement *any* algorithm and *any* dynamical system with *real* (DNA) molecules

- Turing complete and "Shannon complete"
- ANY collection of abstract chemical reactions can be implemented with specially designed DNA molecules, with accurate kinetics (up to time scaling).
- Approaching a situation where we can "systematically compile" (synthesize) a model to DNA molecules, run an (automated) protocol, and observe (sequence) the results in a closed loop.

Summarizing

- Our models are (chemical) programs
- We can compute their behavior (their final state)
- We can (virtually) run them by integration of the ODEs
- We can (physically) run them by DNA nanotech

Part 2

From a Chemical Reaction Network to a set of DNA molecules that do “the same thing”

How do we “run” Chemistry?

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

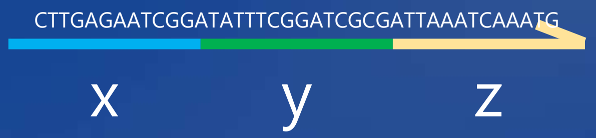
DNA Strand Displacement

An "unnatural" use of DNA for emulating
any system of chemical reactions
with real molecules

Domains

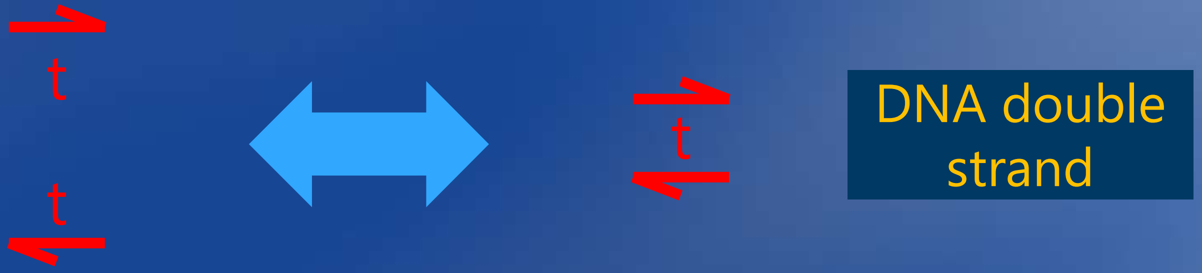
- Subsequences on a DNA strand are called domains
  - *provided* they are “independent” of each other

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



oriented DNA single strand

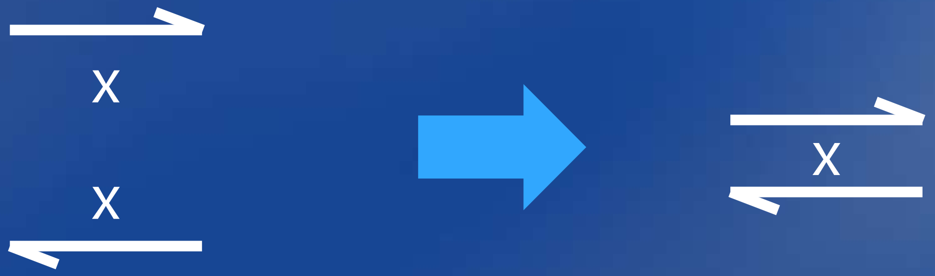
Short Domains



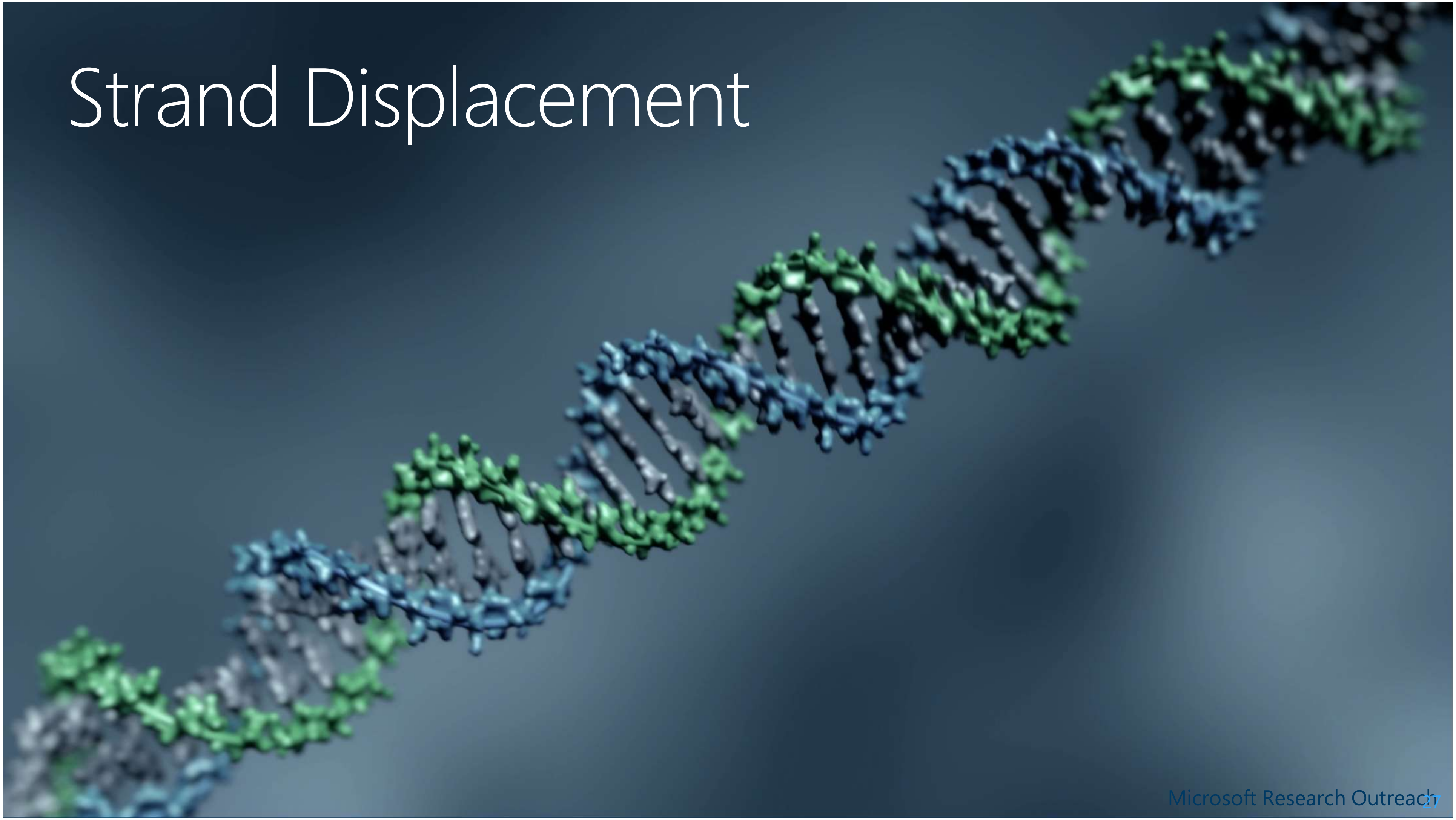
DNA double strand

Reversible Hybridization

Long Domains

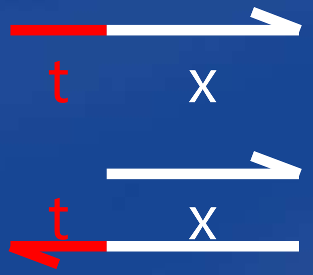


Irreversible Hybridization



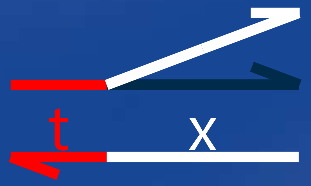
Strand Displacement

Strand Displacement



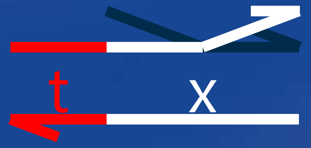
“Toehold Mediated”

Strand Displacement



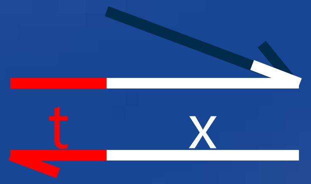
Toehold Binding

Strand Displacement



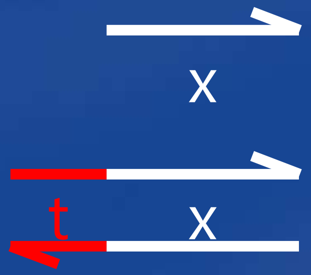
Branch Migration

Strand Displacement



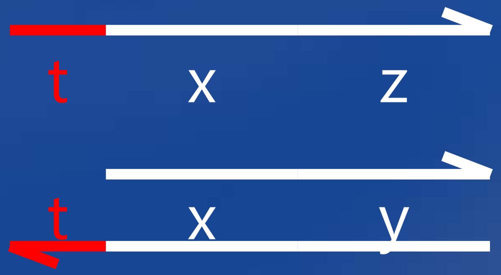
Displacement

Strand Displacement

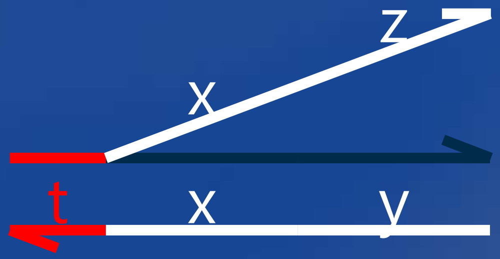


Irreversible release

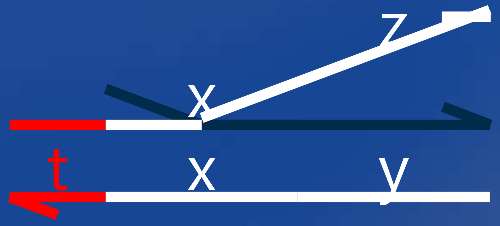
Bad Match



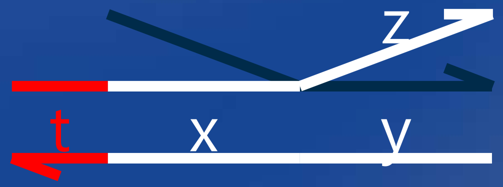
Bad Match



Bad Match



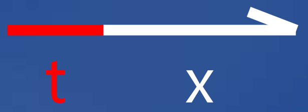
Bad Match



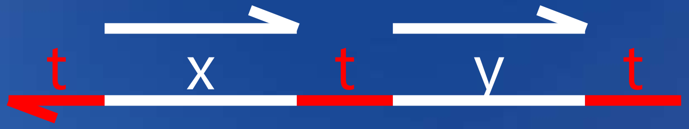
Cannot proceed
Hence will undo

Two-Domain Architecture

- Signals: 1 toehold + 1 recognition region



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



Garbage collection “built into” the gate operation

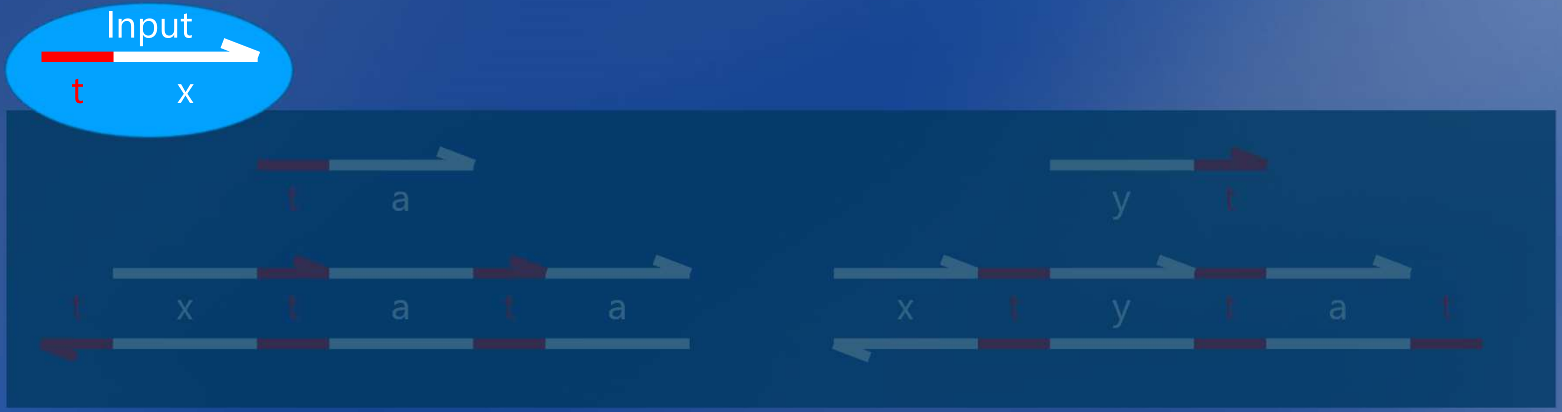
Two-Domain DNA Strand Displacement

Luca Cardelli

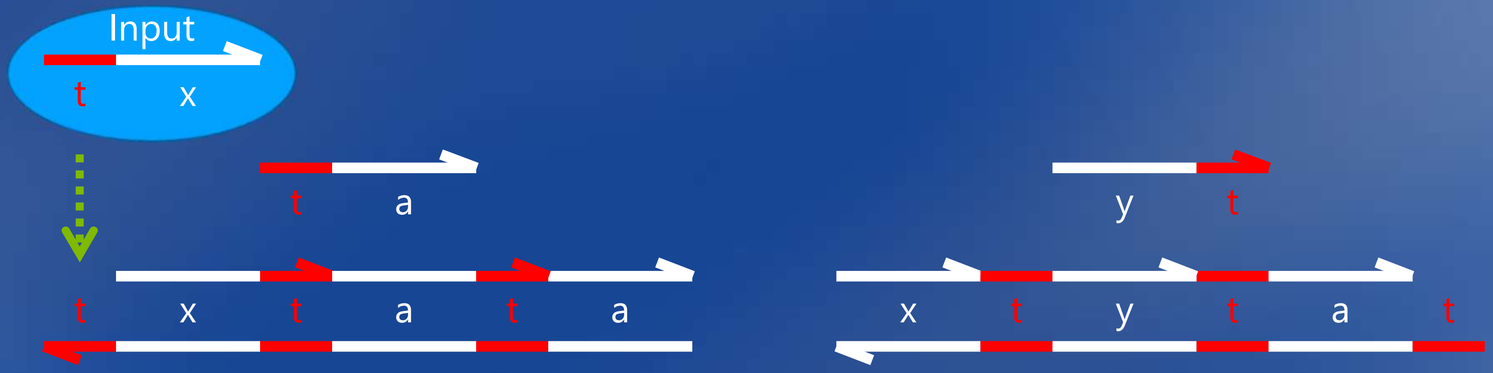
In S. B. Cooper, E. Kashefi, P. Panangaden (Eds.):
Developments in Computational Models (DCM 2010).
EPTCS 25, 2010, pp. 33-47. May 2010.

Transducer

Transducer x→y



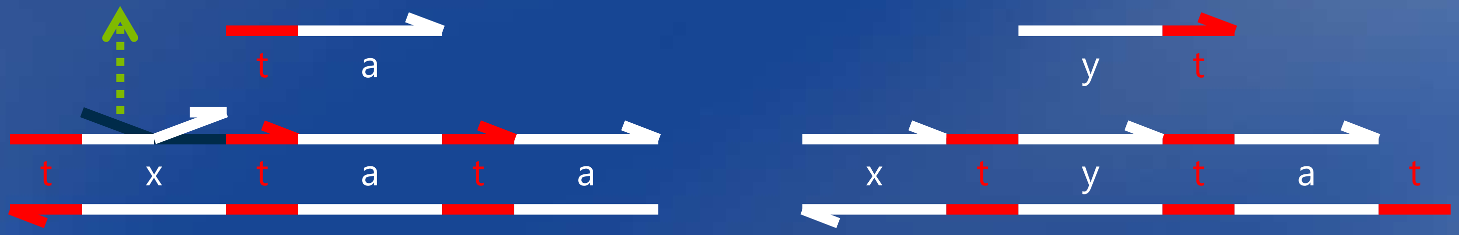
Transducer x→y



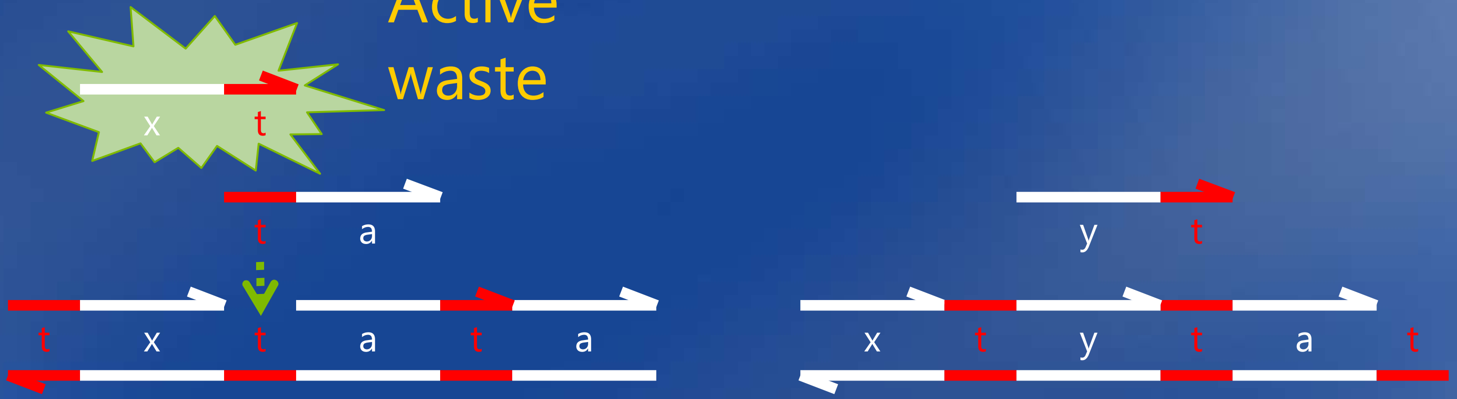
Built by self-assembly!

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

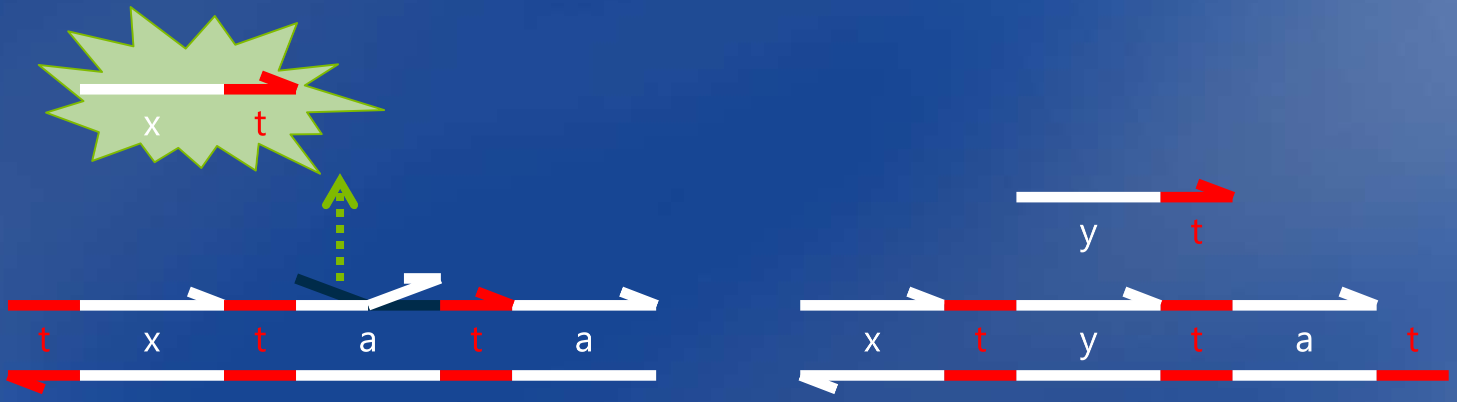
Transducer x→y



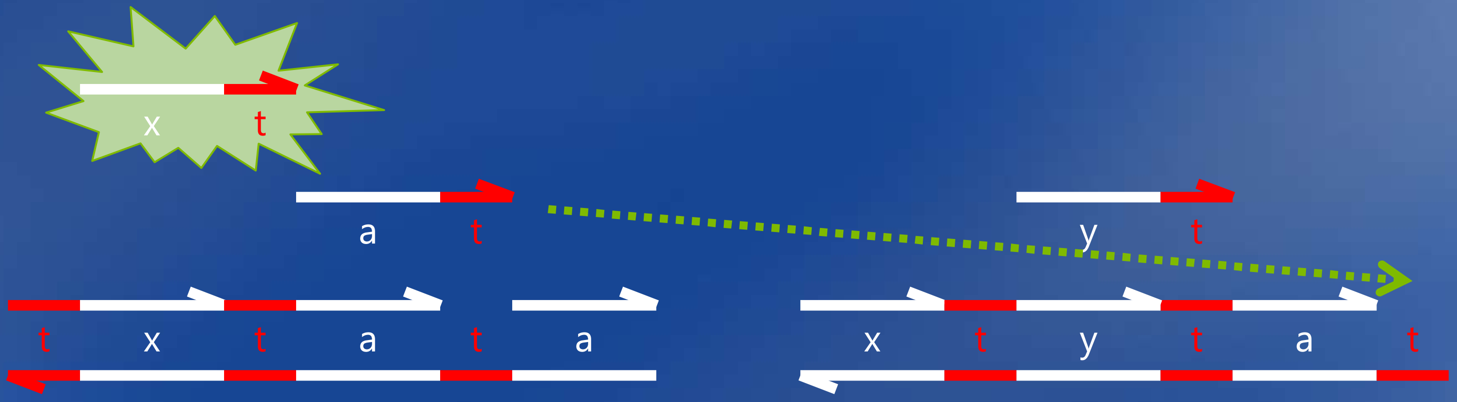
Transducer x→y



Transducer x→y

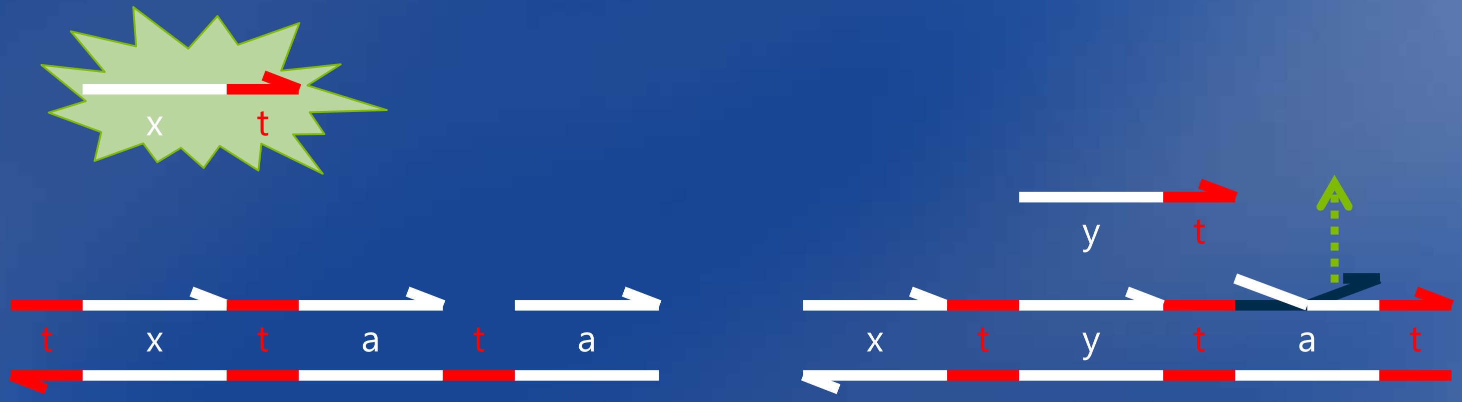


Transducer x→y

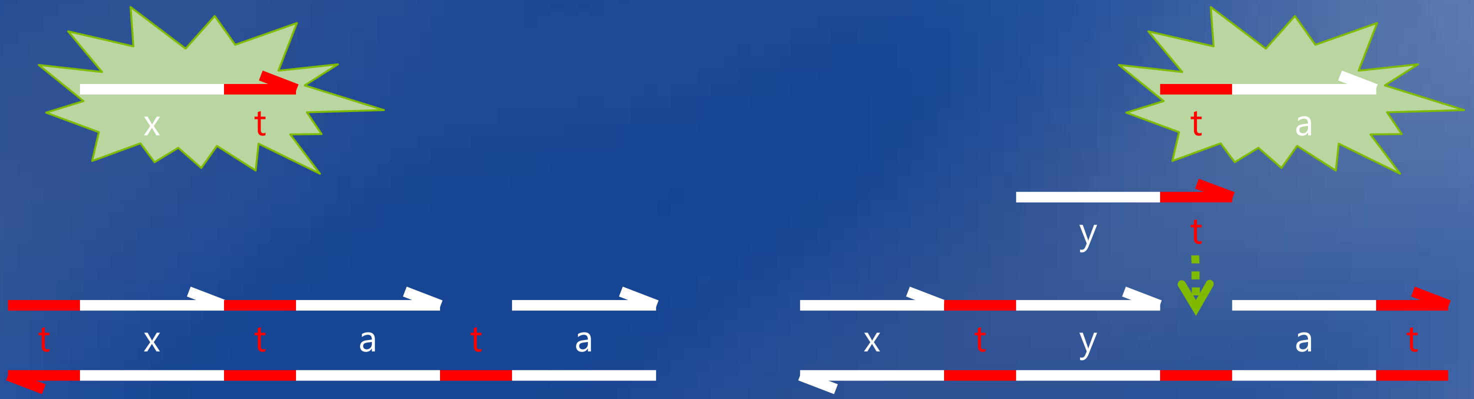


So far, a **tx** *signal* has produced an **at** *cosignal*.
But we want signals as output, not cosignals.

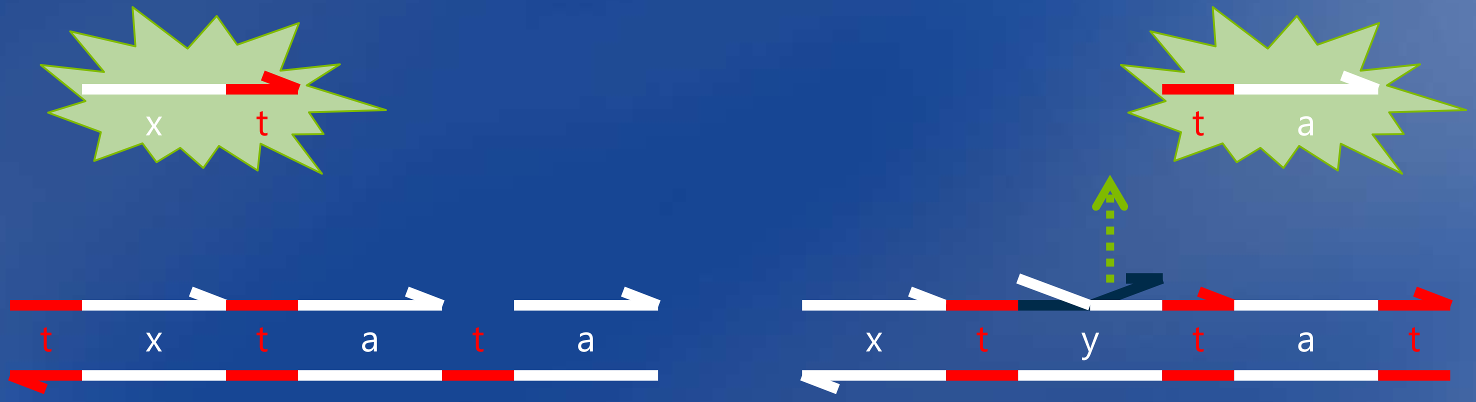
Transducer x→y



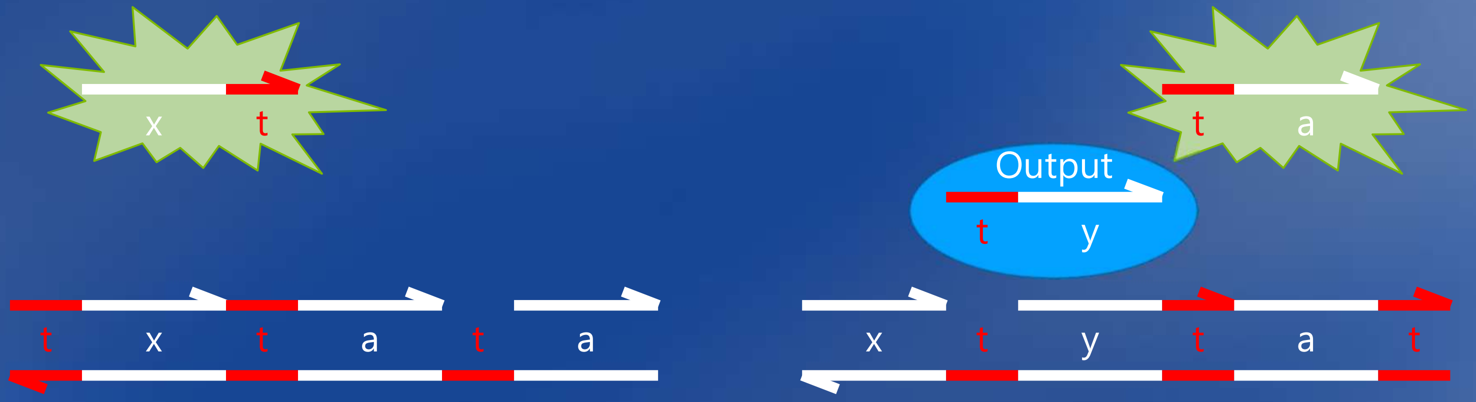
Transducer x→y



Transducer x→y



Transducer x→y



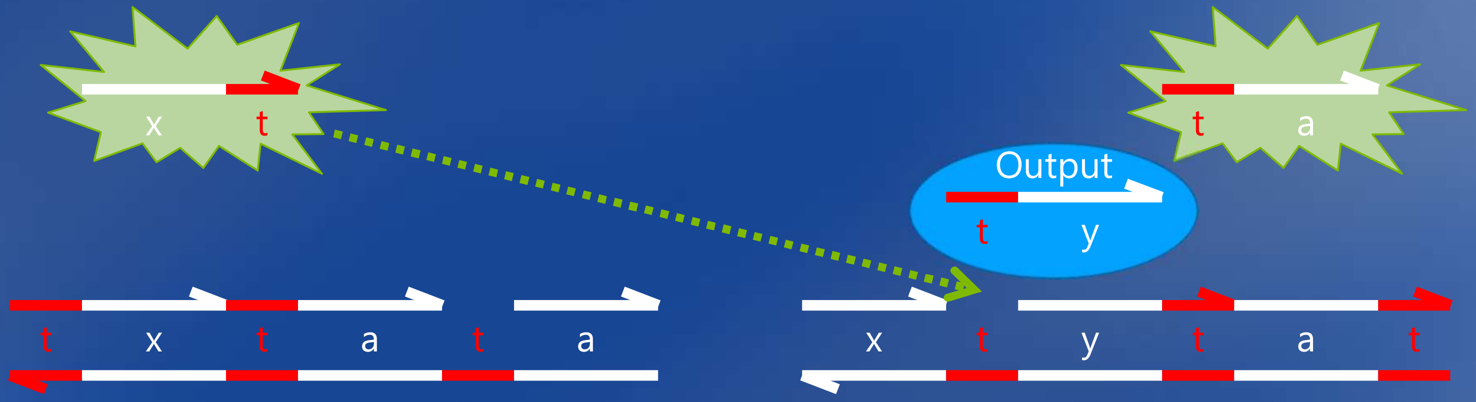
Here is our output **ty** *signal*.

But we are not done yet:

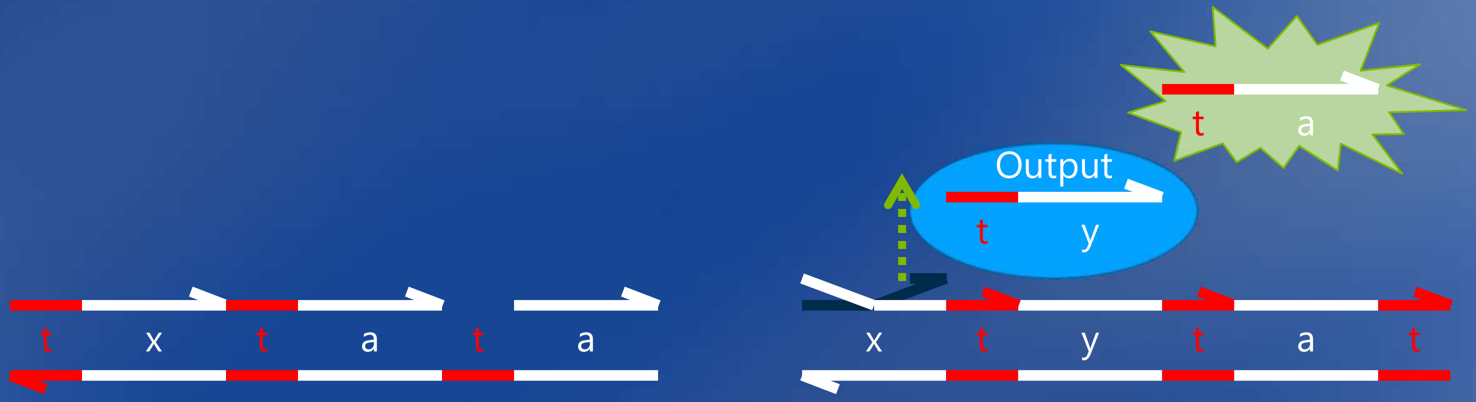
1) We need to make the output irreversible.
2) We need to remove the garbage.

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

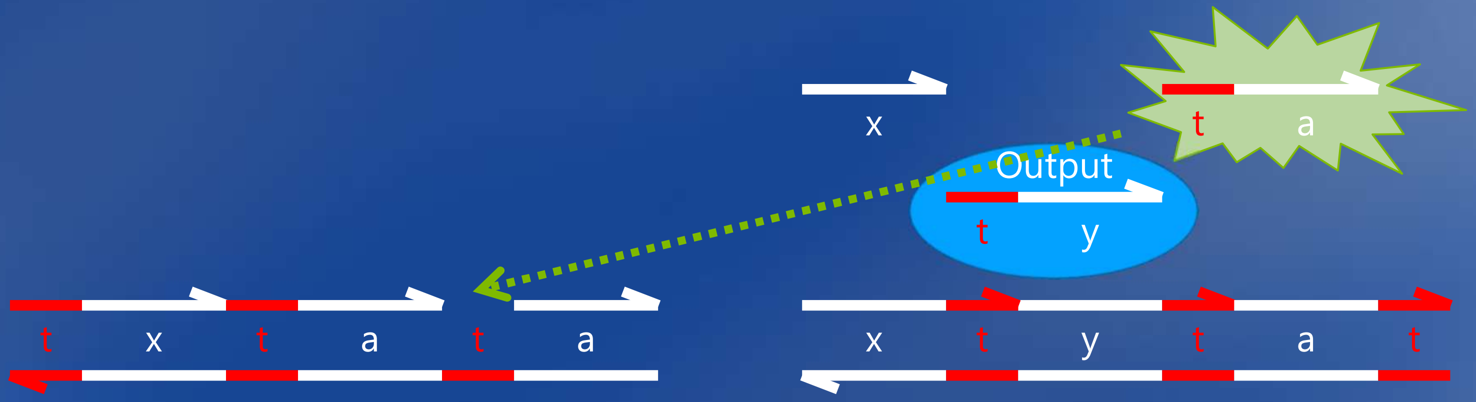
Transducer x→y



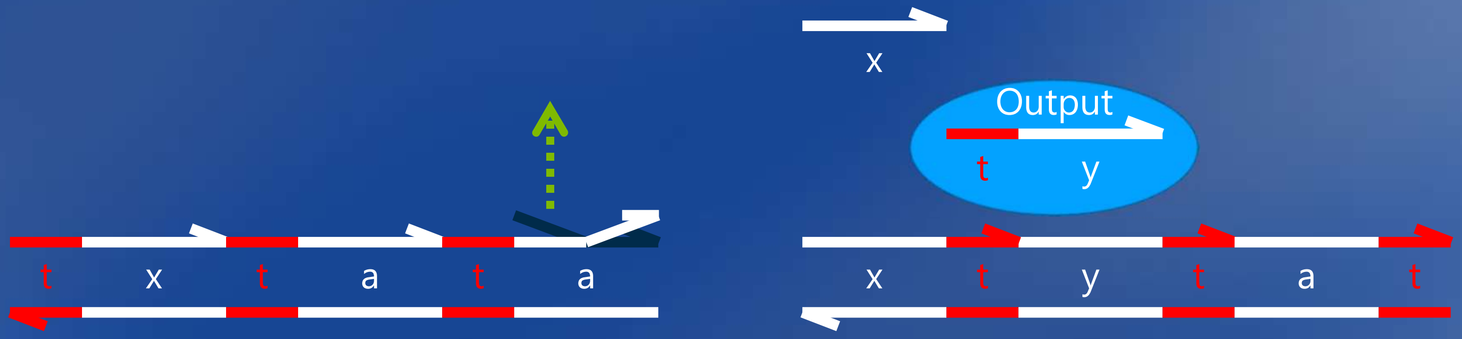
Transducer x→y



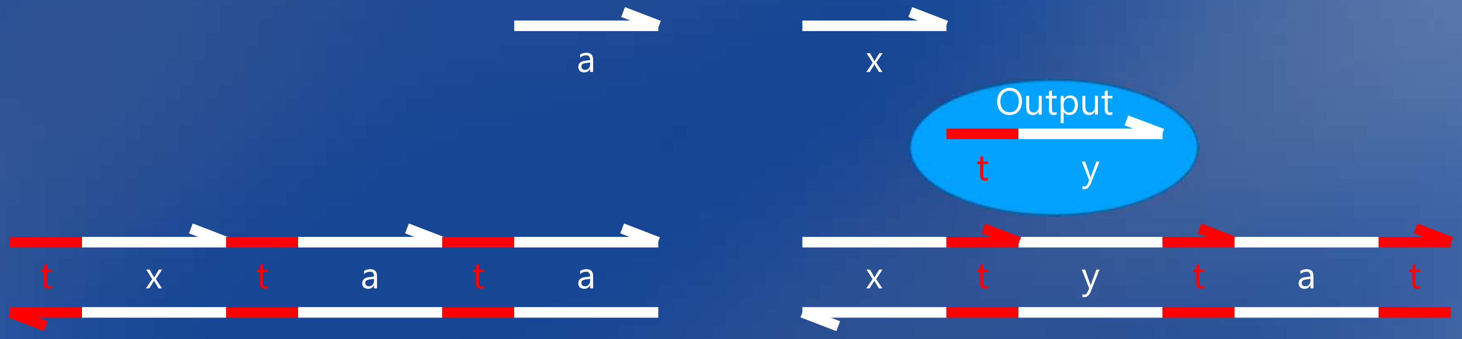
Transducer x→y



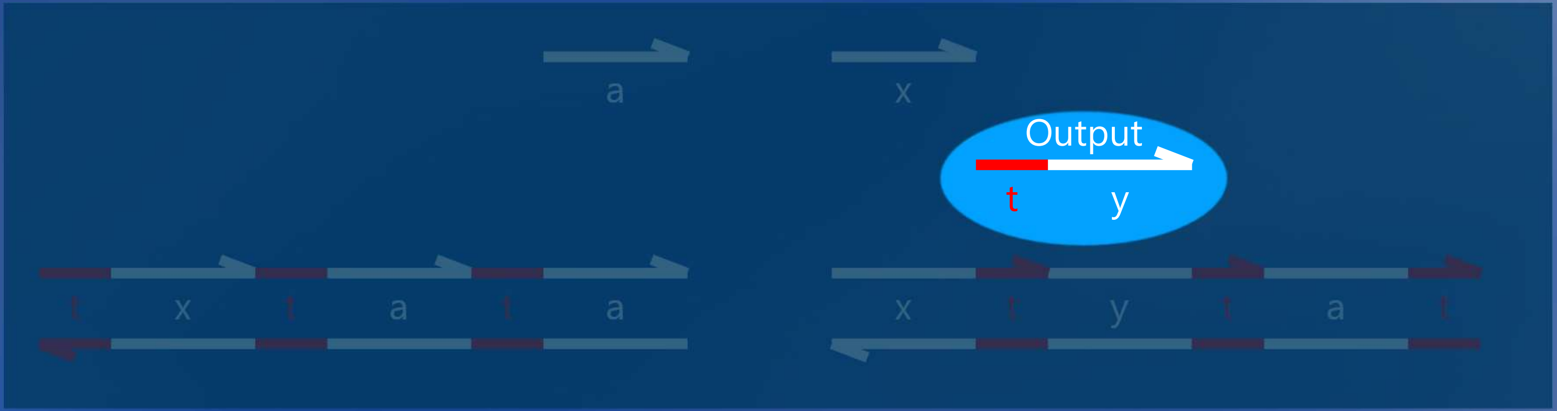
Transducer x→y



Transducer x→y



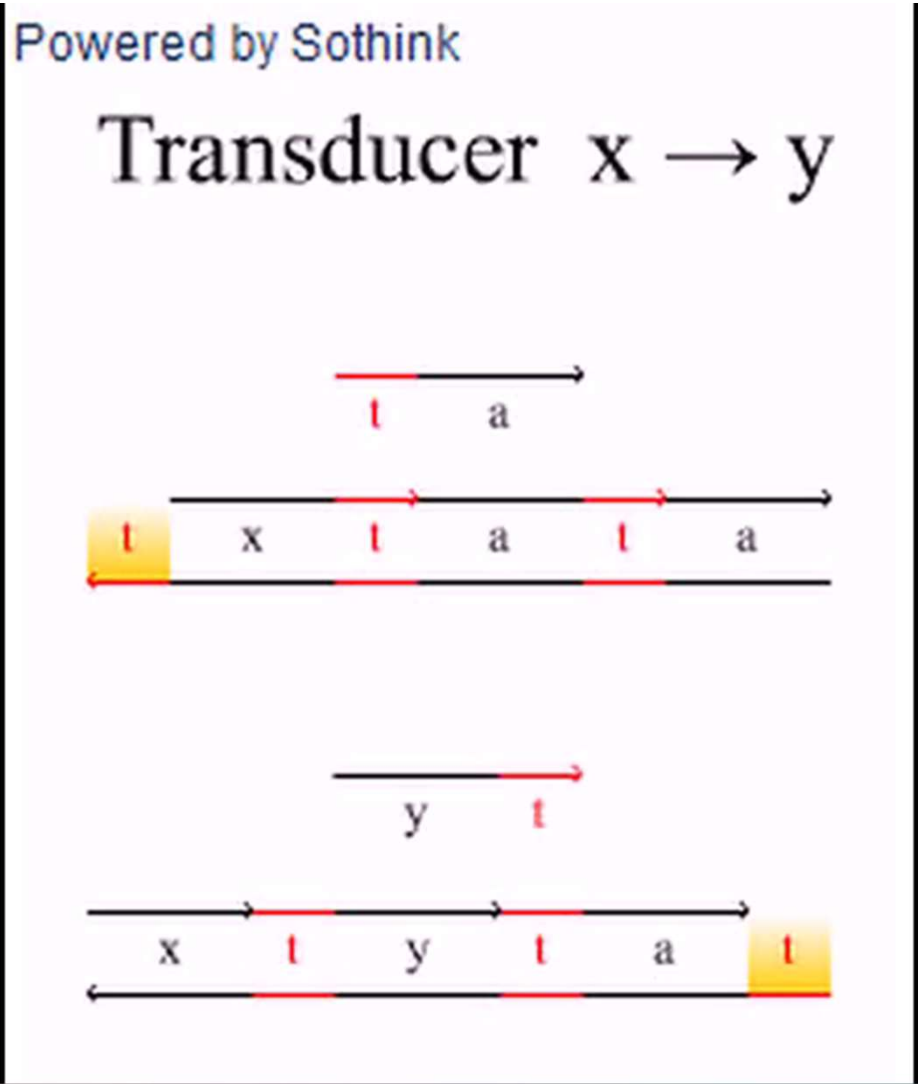
Transducer x→y



Done.

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

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

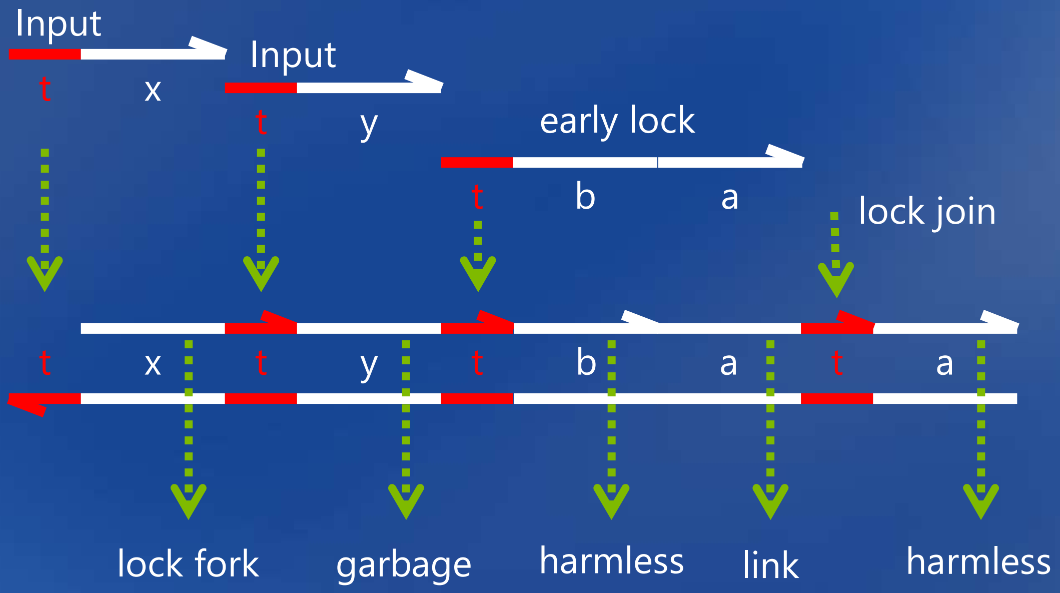


Powered by Sothink

Transducer $x \rightarrow y$

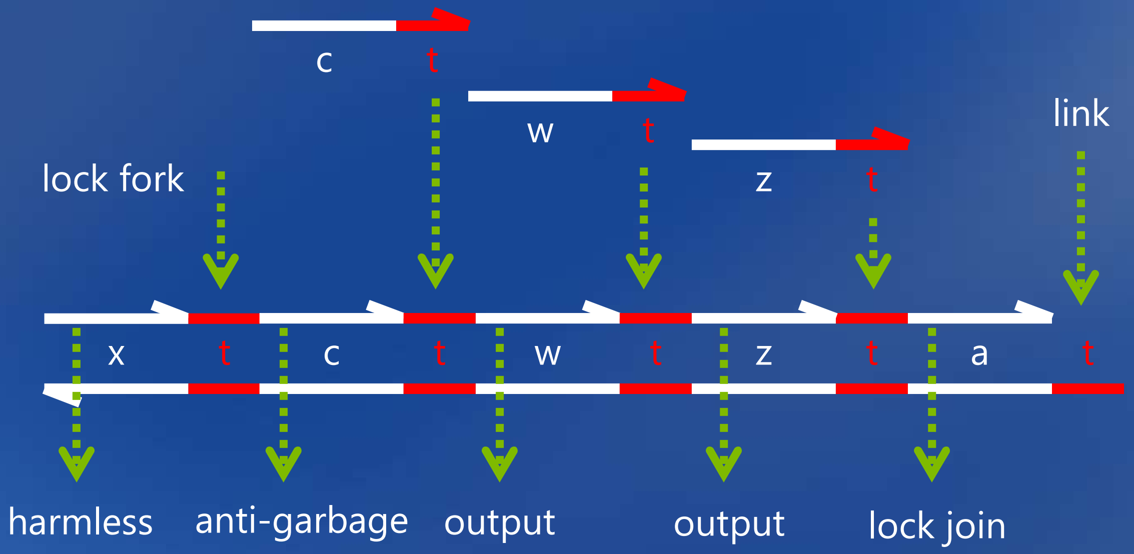
Reaction $x + y \rightarrow z + w$

join
half



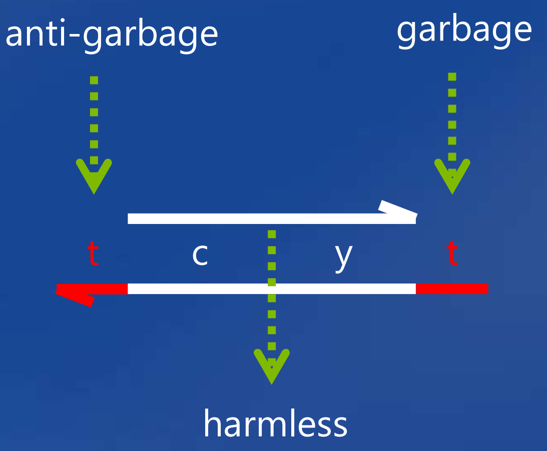
Reaction $x + y \rightarrow z + w$

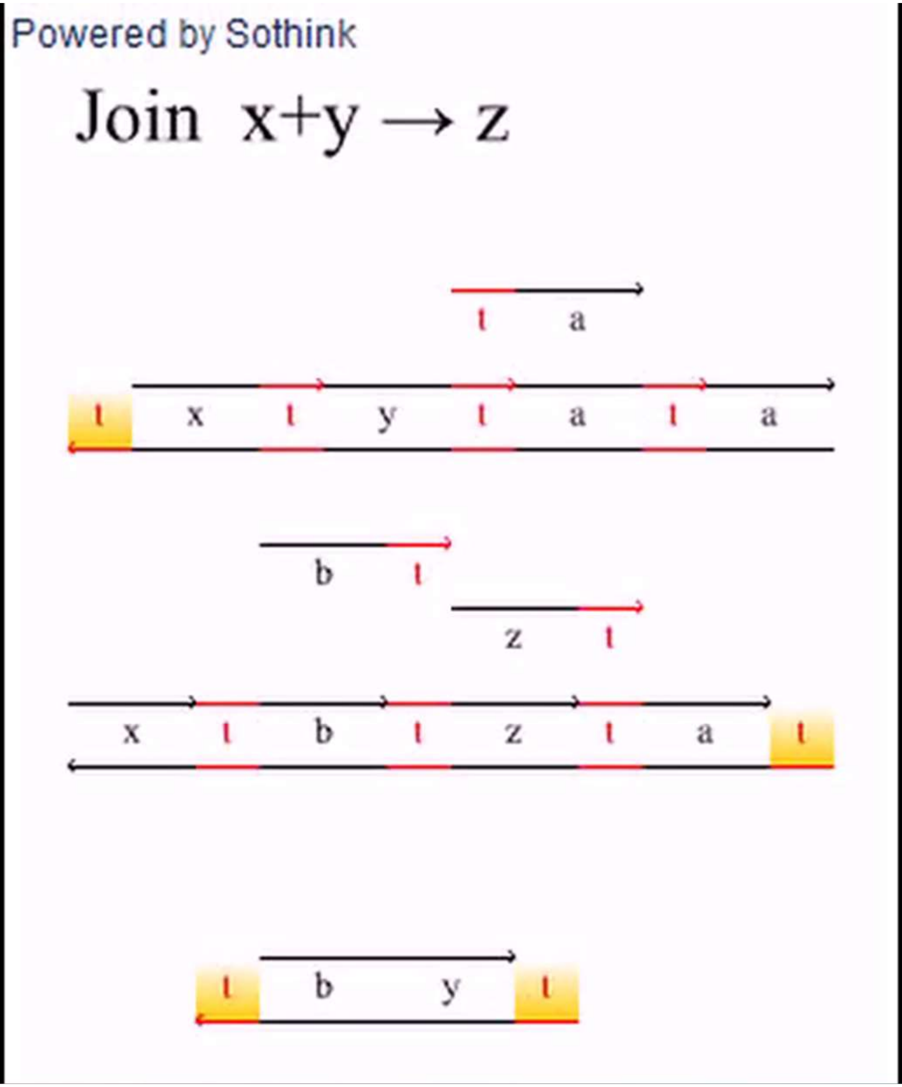
fork
half



Reaction $x + y \rightarrow z + w$

garbage collection





Powered by Sothink

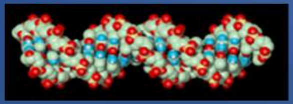
Join x+y → z

Approximate Majority Algorithm

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

- 3 rules of agent (or molecule) interaction
  - $X + Y \rightarrow B + B$
  - $B + X \rightarrow X + X$
  - $B + Y \rightarrow Y + Y$

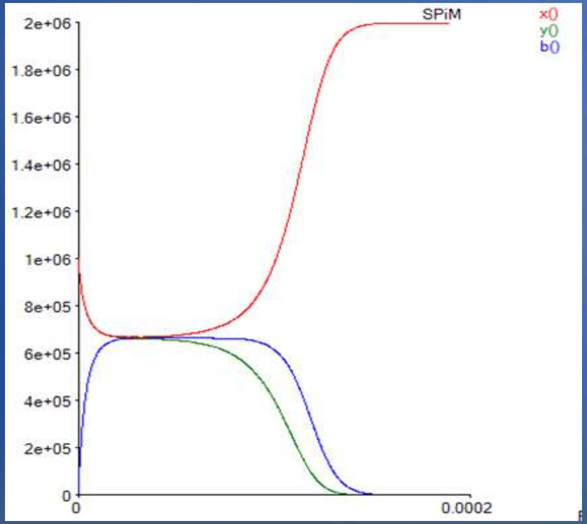
"our program" →



Optimal Consensus Algorithm

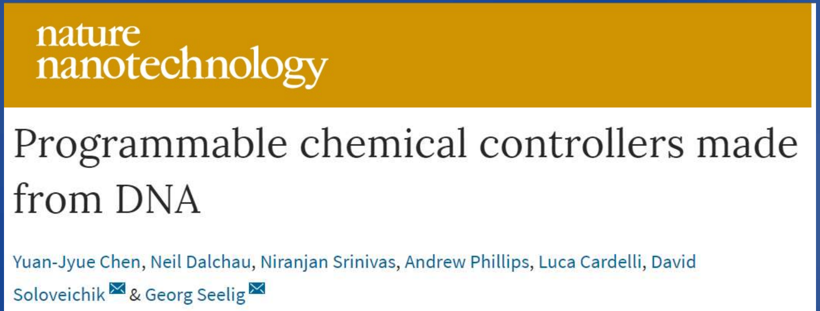
- Fast: reaches agreement in O(log n) time w.h.p.
  - O(n log n) communications/collisions
  - Even when initially #X = #Y! (stochastic symmetry breaking)

- Robust: true majority wins w.h.p.
  - If initial majority exceeds minority by ω(√n log n)
  - Hence the agreement state is stable



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

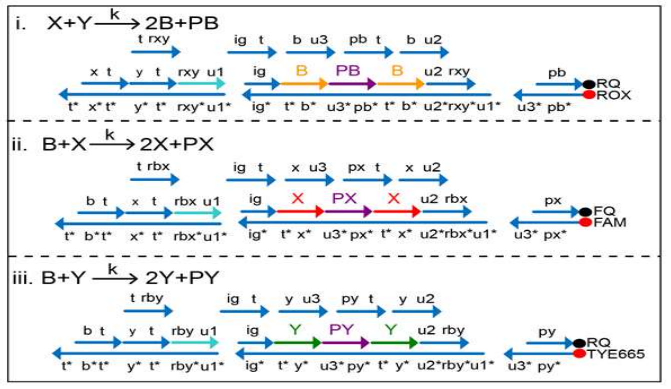
DNA Implementation of the Approximate Majority algorithm

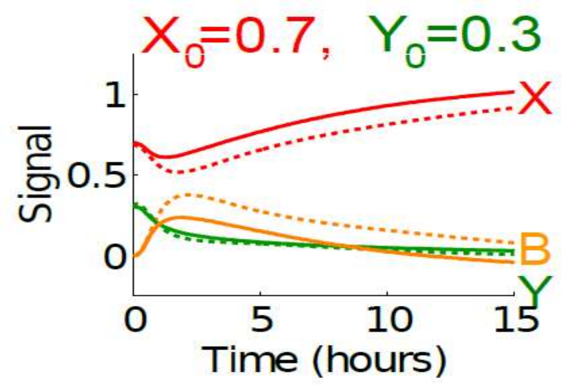


nature nanotechnology

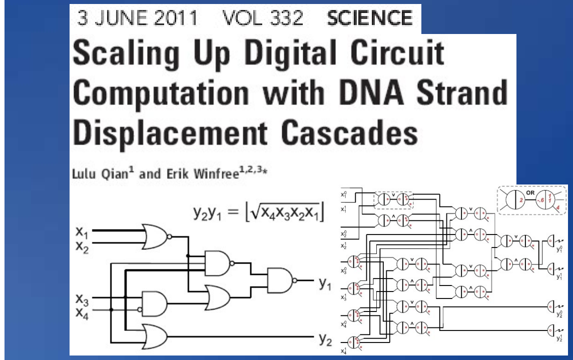
Programmable chemical controllers made from DNA

Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik & Georg Seelig


i. X+Y $\xrightarrow{k}$ 2B+PB
t rxy
ig t b u3 pb t b u2
x t y t rxy u1 ig B PB B u2 rxy pb
RQ
ROX
t* x* t* y* t* rxy* u1* ig* t* b* u3* pb* t* b* u2* rxy* u1* u3* pb*
ii. B+X $\xrightarrow{k}$ 2X+PX
t rbx
ig t x u3 px t x u2
b t x t rbx u1 ig X PX X u2 rbx px
FQ
FAM
t* b* t* x* t* rbx* u1* ig* t* x* u3* px* t* x* u2* rbx* u1* u3* px*
iii. B+Y $\xrightarrow{k}$ 2Y+PY
t rby
ig t y u3 py t y u2
b t y t rby u1 ig Y PY Y u2 rby py
RQ
TYE665
t* b* t* y* t* rby* u1* ig* t* y* u3* py* t* y* u2* rby* u1* u3* py*



Some Large-scale Circuits (so far…)

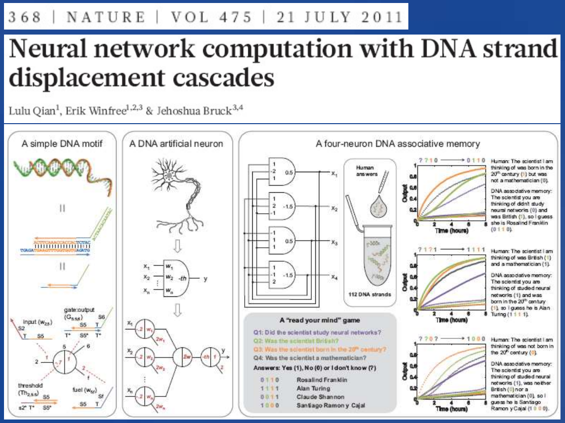


3 JUNE 2011 VOL 332 SCIENCE

Scaling Up Digital Circuit Computation with DNA Strand Displacement Cascades

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

Computing the square root of a 4-bit number



368 | NATURE | VOL 475 | 21 JULY 2011

Neural network computation with DNA strand displacement cascades

Lulu Qian[1], Erik Winfree[1,2,3] & Jehoshua Bruck[3,4]

A "read your mind" game

Q1: Did the scientist study neural networks?
Q2: Was the scientist British?
Q3: Was the scientist born in the 20th century?
Q4: Was the scientist a mathematician?
Answers: Yes (1), No (0) or I don't know (?)

0 1 1 0	Rosalind Franklin
1 1 1 1	Alan Turing
0 0 1 1	Claude Shannon
1 0 0 0	Santiago Ramon y Cajal

Human: The scientist I am thinking of was born in the 20th century (1) but was not a mathematician (0).

DNA associative memory: The scientist you are thinking of didn't study neural networks (0) and was British (1), so I guess she is Rosalind Franklin (0 1 1 0).

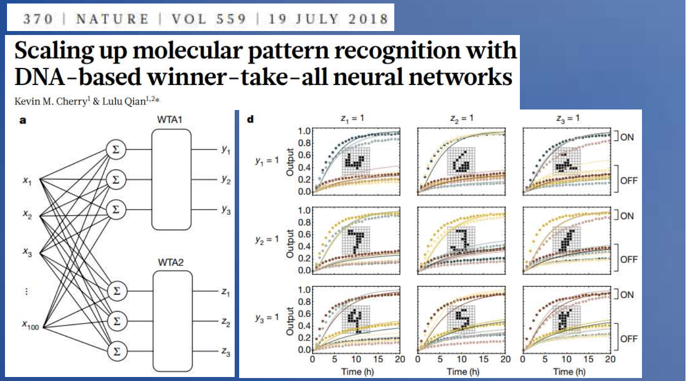
Human: The scientist I am thinking of was British (1) and a mathematician (1).

DNA associative memory: The scientist you are thinking of studied neural networks (1) and was born in the 20th century (1), so I guess he is Alan Turing (1 1 1 1).

Human: The scientist I am thinking of was not born in the 20th century (0).

DNA associative memory: The scientist you are thinking of studied neural networks (1), was neither British (0) nor a mathematician (0), so I guess he is Santiago Ramon y Cajal (1 0 0 0).

Classifying 4 distinct 4-bit patterns via 4 neurons



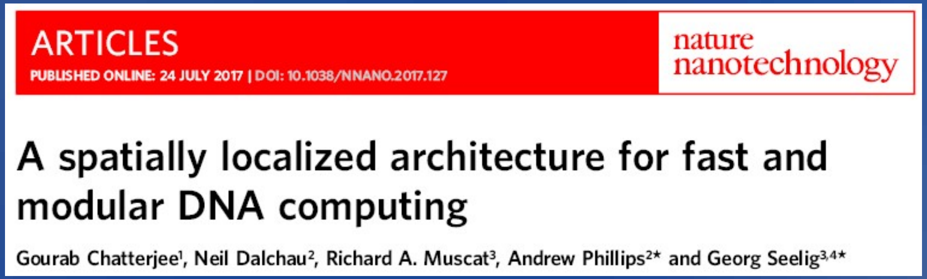
370 | NATURE | VOL 559 | 19 JULY 2018

Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks

Kevin M. Cherry[1] & Lulu Qian[1,2]*

Classifying 9 distinct 100-bit patterns via WTA networks

Scaling up: DNA Circuit Boards



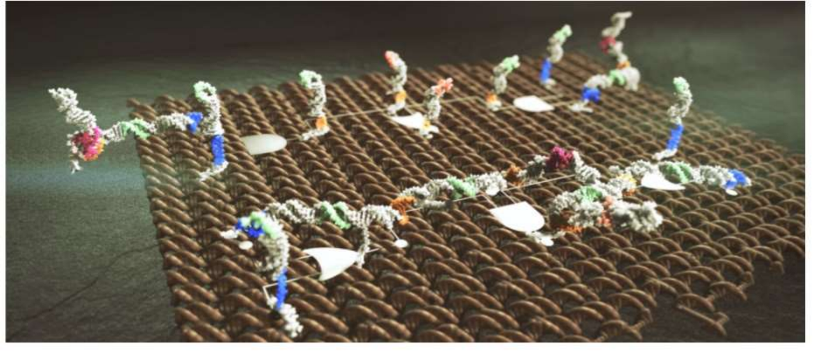
ARTICLES

PUBLISHED ONLINE: 24 JULY 2017 | DOI: 10.1038/NNANO.2017.127

nature nanotechnology

A spatially localized architecture for fast and modular DNA computing

Gourab Chatterjee[1], Neil Dalchau[2], Richard A. Muscat[3], Andrew Phillips[2]* and Georg Seelig[3,4]*

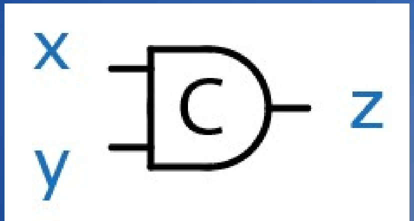


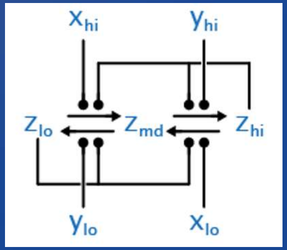
The first computational circuit boards made of DNA

https://www.microsoft.com/en-us/research/blog/researchers-build-nanoscale-computational-circuit-boards-dna

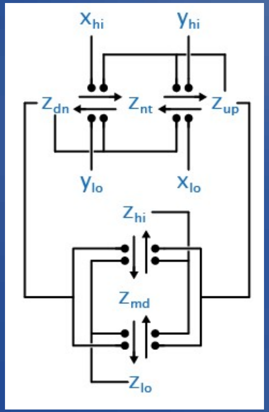
Avoiding Clocks

- Muller C-Element
  - A Boolean gate
  - When x = y then z = x = y, otherwise z remembers its *last* state.





Core C-Element
(AM with external inputs)



Full C-Element with output rectified by another AM

Chemical Reaction Network Designs for Asynchronous Logic Circuits.
Luca Cardelli, Marta Kwiatkowska, Max Whitby.
Natural Computing Journal.

Part 3

Detecting Molecular Events

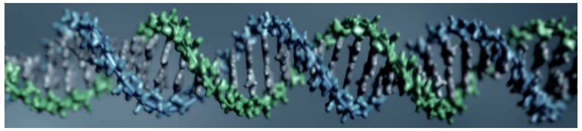
Preorder Recorder

- Detecting molecular events is very difficult and very important
- In science we want to know “what’s going on?”
- In bioengineering we want to know “what when wrong?”
- We often want to know the *order* of events to help determine *causation*

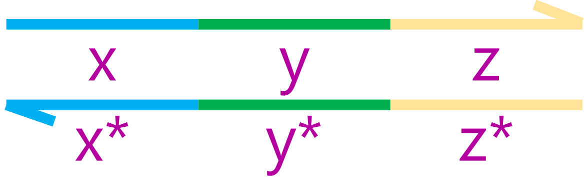
- We discuss a “preorder recorder” algorithm that reads out the *preorder of first-occurrence* of a set of events in a chemical soup, where an event is the appearance of a DNA/RNA strand in the soup
- These events could be DNA circuit signals, or naturally transcribed RNA, or DNA/RNA transduced in response to e.g. presence of certain proteins

DNA Domains

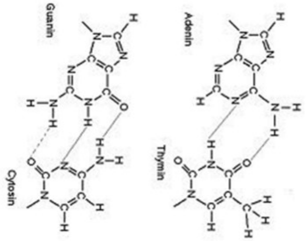
- Subsequences on a DNA strand are called domains
  - *provided* they are “independent” of each other



CTTGAGAATCGGATATTTCGGATCGCGATTAAATCAAATG



A T G C
| | | |
T A C G

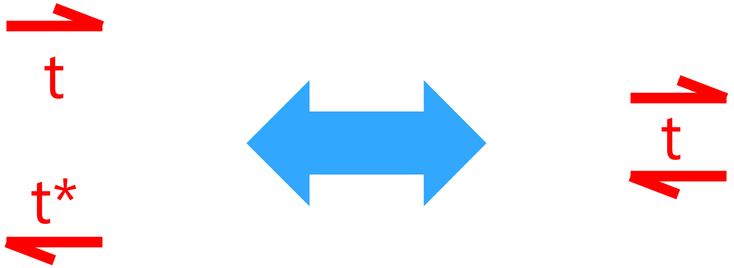


- Differently named domains must not hybridize
  - with each other, with each other’s complement, with subsequences of each other, with concatenations of other domains (or their complements), etc.

Domain Kinetics

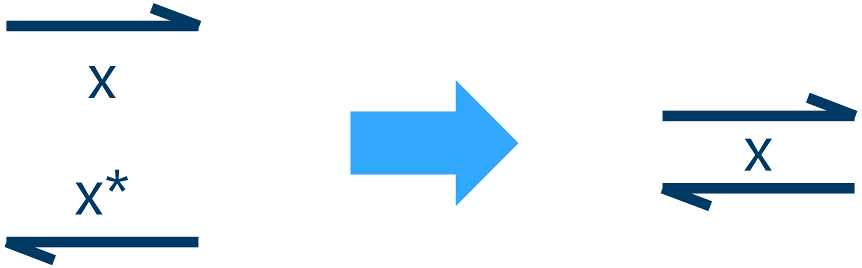
- "Short" Domains

Reversible Hybridization

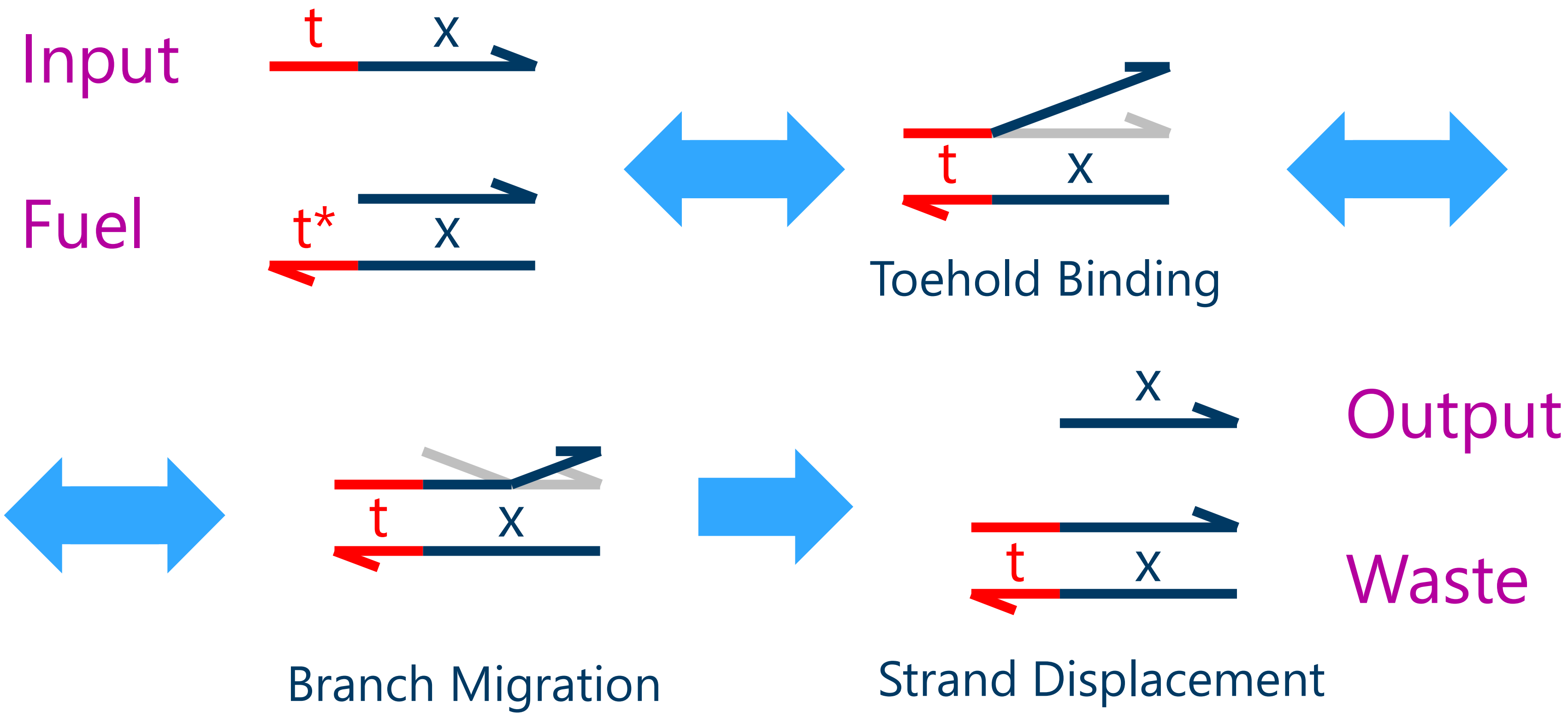


- "Long" Domains

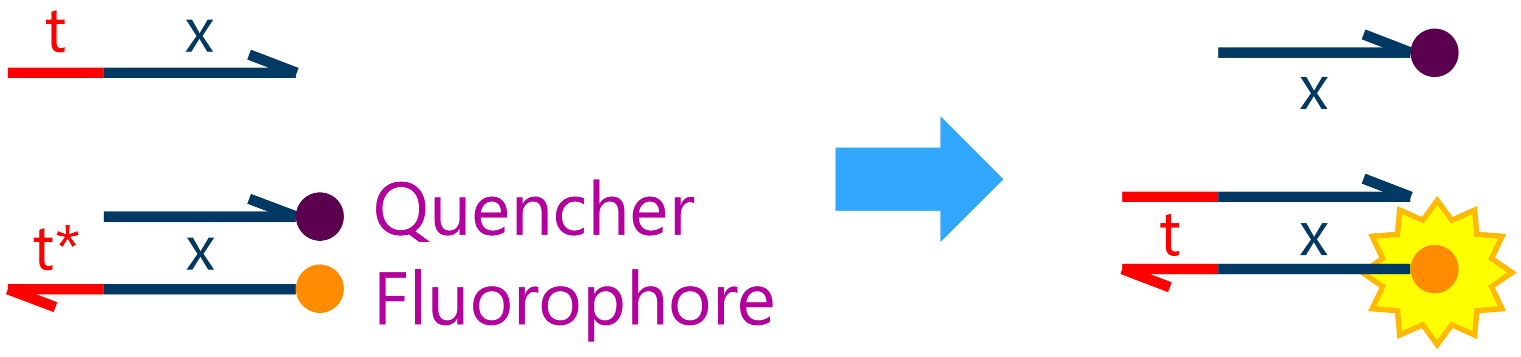
Irreversible Hybridization



DNA Strand Displacement



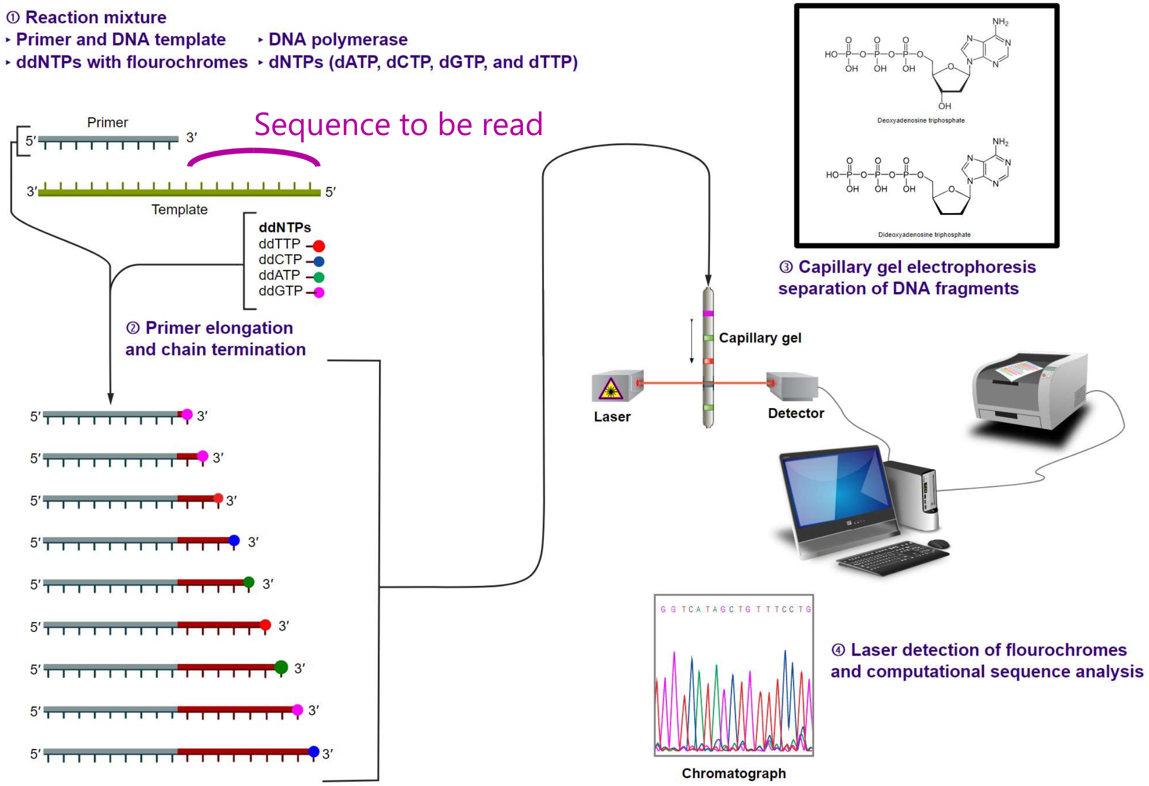
Fluorescence Readout



How to Read DNA (Output)

- Fluorescence Readout
  - Limited redout capability: 3/4 "colors" of output.
  - Output can be read continuously over time
- Atomic Force Microscope Readout
  - Detecting shapes and patterns
  - Comprehensive view of the results
- Sequencing Readout
  - At the end of a computation, sequence the strand types left in the soup
  - Output is a multiset of strand types (each with a real-valued concentration)

Sanger Sequencing



① Reaction mixture
- Primer and DNA template
- ddNTPs with flourochromes
- DNA polymerase
- dNTPs (dATP, dCTP, dGTP, and dTTP)

② Primer elongation and chain termination

③ Capillary gel electrophoresis separation of DNA fragments

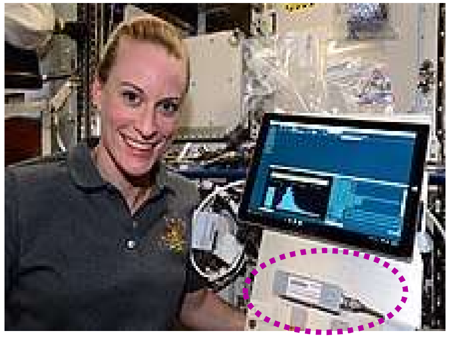
④ Laser detection of flourochromes and computational sequence analysis

The Sanger (chain-termination) method for DNA sequencing. (1) A primer is annealed to a sequence, (2) Reagents are added to the primer and template, including: DNA polymerase, dNTPs, and a small amount of all four dideoxynucleotides (ddNTPs) labeled with fluorophores. During primer elongation, the random insertion of a ddNTP instead of a dNTP terminates synthesis of the chain because DNA polymerase cannot react with the missing hydroxyl. This produces all possible lengths of chains. (3) The products are separated on a single lane capillary gel, where the resulting bands are read by a imaging system. (4) This produces several hundred thousand nucleotides a day, data which require storage and subsequent computational analysis

https://en.wikipedia.org/wiki/Sanger_sequencing

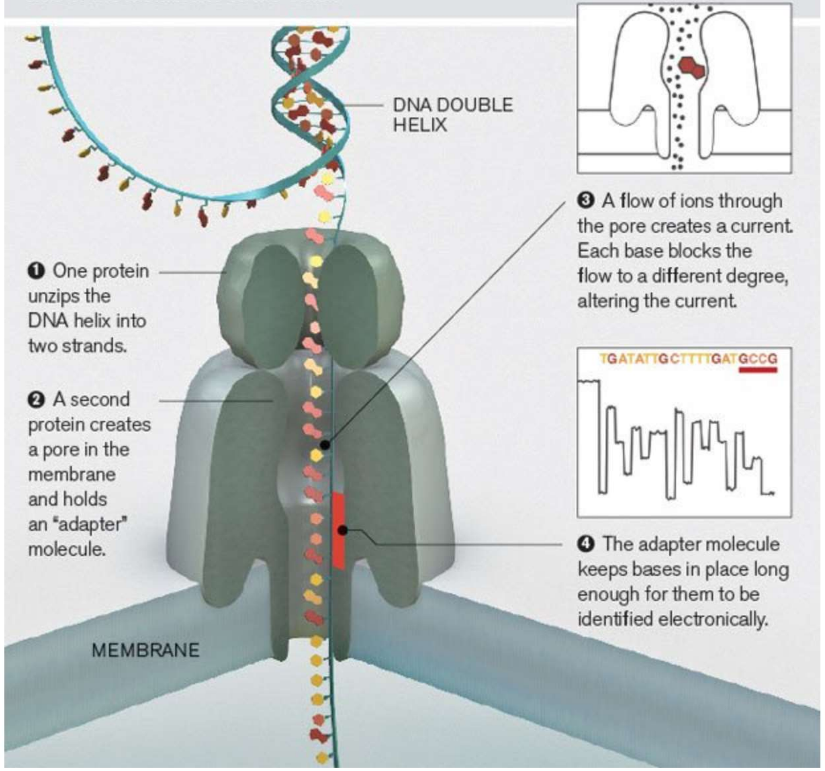
High Throughput Sequencing

- Sequencing by Synthesis
  - Like Sanger sequencing, but done in parallel on a "lawn" of single strands, removing the fluorophores at each step to carry on.
- Nanopore Sequencing
  - ~ 200 *single* different DNA molecules sequenced in parallel



American astronaut Kate Rubins with a MinION sequencer on the ISS in August 2016.
https://en.wikipedia.org/wiki/Oxford_Nanopore_Technologies

DNA can be sequenced by threading it through a microscopic pore in a membrane. Bases are identified by the way they affect ions flowing through the pore from one side of the membrane to the other.



❶ One protein unzips the DNA helix into two strands.

❷ A second protein creates a pore in the membrane and holds an "adapter" molecule.

❸ A flow of ions through the pore creates a current. Each base blocks the flow to a different degree, altering the current.

❹ The adapter molecule keeps bases in place long enough for them to be identified electronically.

http://www2.technologyreview.com/news/427677/nanopore-sequencing/

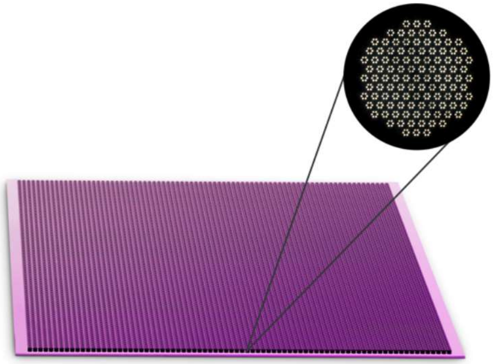
How to Write DNA (Gates + Input)

- Synthesizing DNA using silicon microfabrication technology

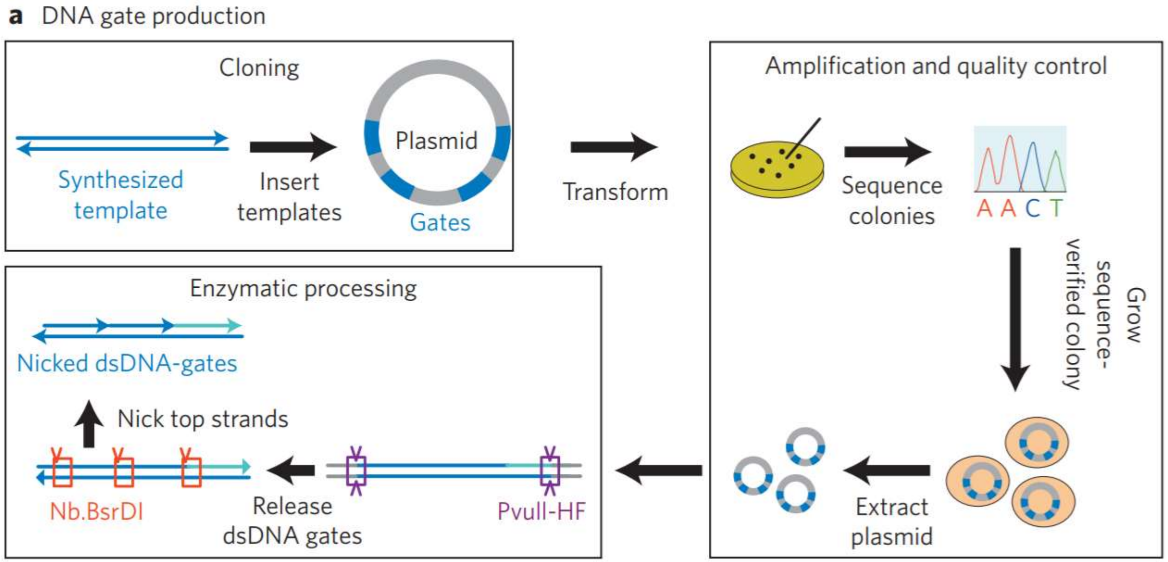
Twist Bioscience developed a proprietary semiconductor-based synthetic DNA manufacturing process featuring a high-throughput silicon platform that allows us to miniaturize the chemistry necessary for DNA synthesis. This miniaturization allows us to reduce the reaction volumes by a factor of 1,000,000 while increasing throughput by a factor of 1,000, enabling the synthesis of 9,600 genes on a single silicon chip at full scale. Traditional synthesis methods produce a single gene in the same physical space using a 96-well plate.

=> DNA Storage

TWIST BIOSCIENCE

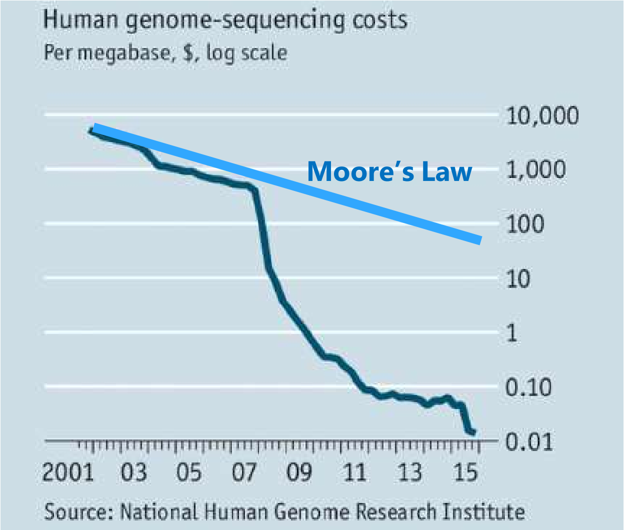


Cloning



- Standard technique, but not normally used to produce "computational" DNA.

The Pace of Biotechnology



Human genome-sequencing costs

Per megabase, $, log scale

Source: National Human Genome Research Institute

The Pace and Proliferation of Biological Technologies

March 4, 2004 by Rob Carlson

- How can we take full advantage of this, for DNA-based algorithms?

Many DNA strand displacement computational schemes are "Universal"

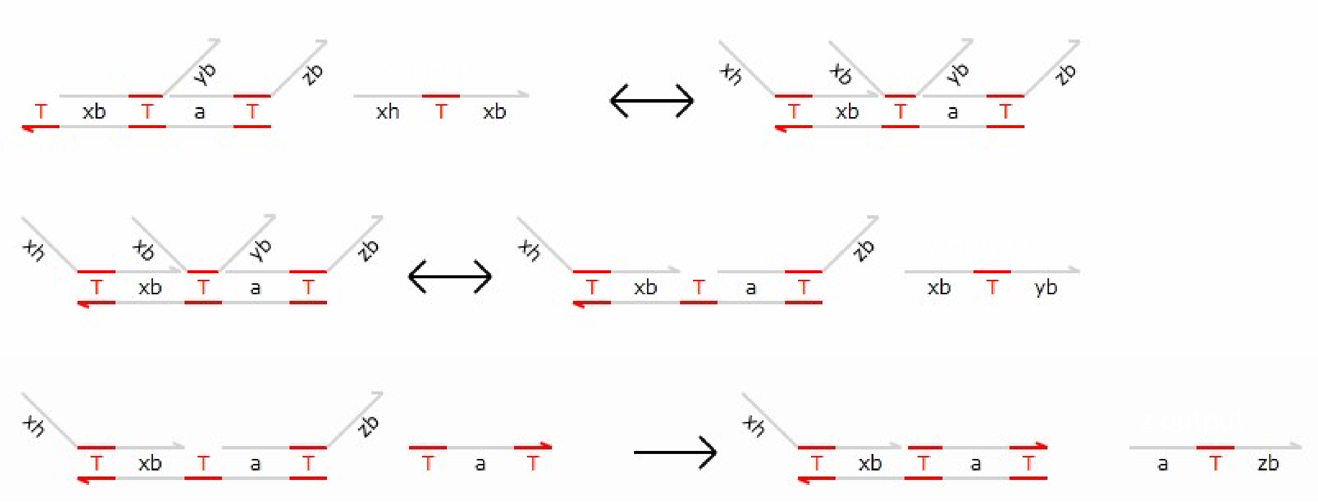
- 4-domain, 3-domain, 2-domain, split-domain …

- Can be used to systematically compile arbitrary finite chemical reaction networks to DNA molecules that exhibit (approximately) the same kinetics.

- But not all can be written by cloning and read by sequencing.

A Typical 3-domain Scheme

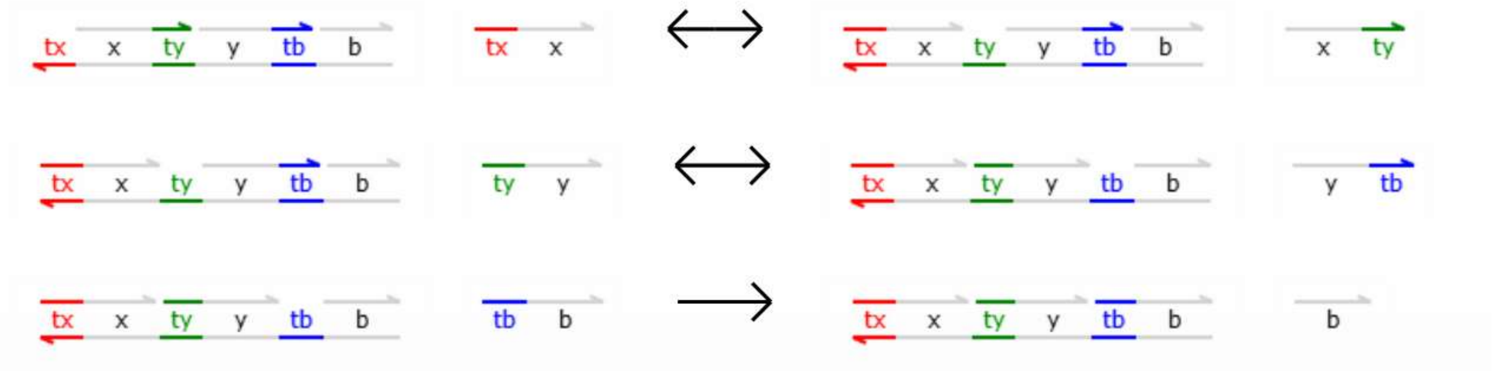
2-input "join"



"Non-clonable, non-sequenceable"

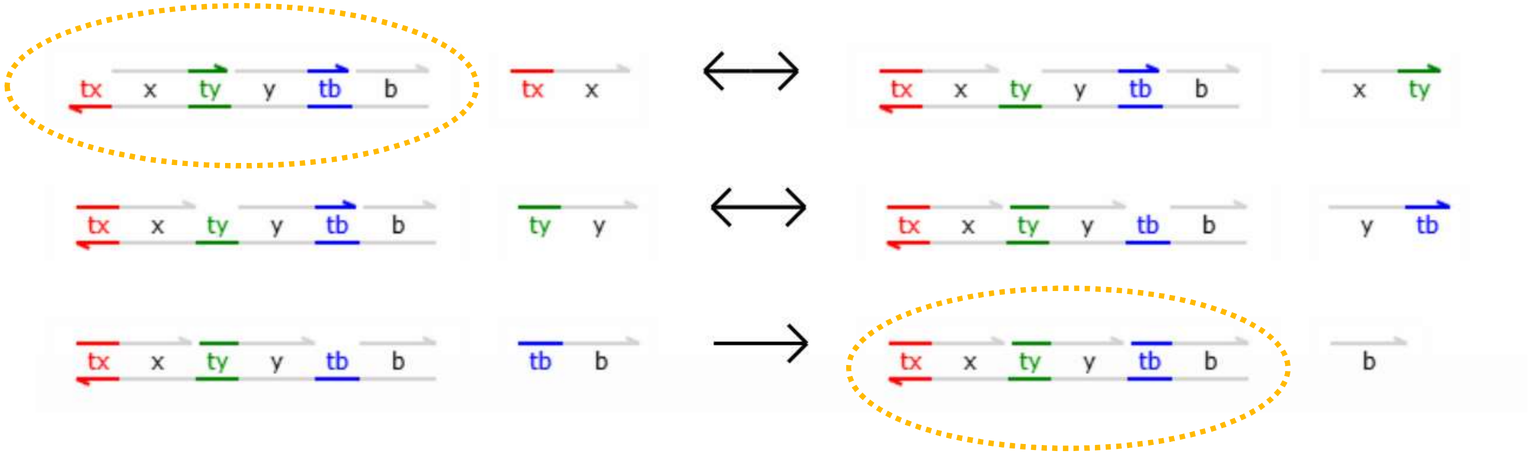
A 2-domain Scheme

2-input "join"



Clonable but not Sequenceable

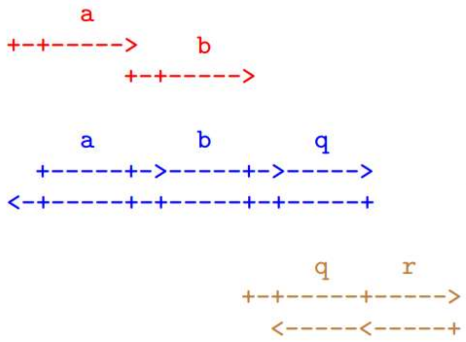
Sequencing (of double strands) must be preceded by *polymerase extension* (to remove single-stranded gaps) and *ligation* (to remove nicks)



Sequenceable Join gate

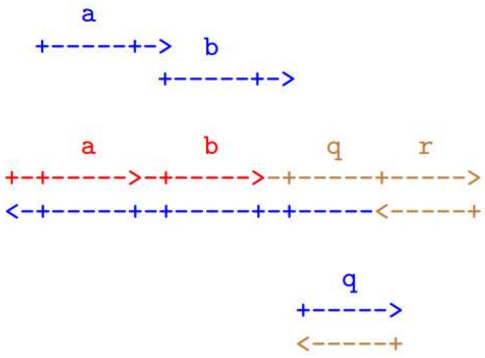
[Georg Seelig & Yuan-Jyue Chen]

A 2-input join gate, join(a,b):



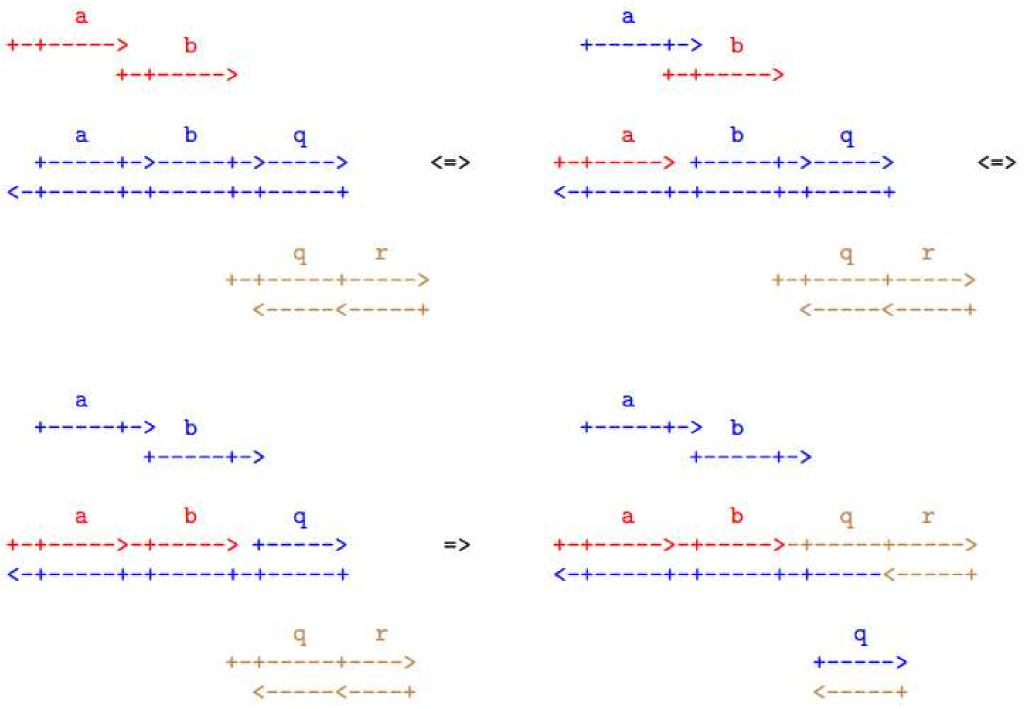
Two-domain gate architecture [L.Cardelli 2013]
based on double stranded DNA (no secondary structure)
hence gates can be sequenced by standard means

if a, b are present *together*, then after full activation:



an "abqr+q" read (after ligation) reveals there was activation of join(a,b), hence both a and b occurred. Otherwise, we would read "abq+qr".

Join Gate activation steps



```
     a                                           a
+-+------>    b                               +------+->   b
         +-+------>                                  +-+------>

     a        b        q                         a        b        q
  +------+->------+->------>      <=>       +-+------> +------+->------>      <=>
<-+------+-+------+-+------+                <-+------+-+------+-+------+

                       q     r                                     q      r
                 +-+-------+----->                            +-+-------+----->
                    <------<-----+                               <------<-----+


     a                                           a
  +------+->  b                                 +------+->  b
           +------+->                                  +------+->

     a        b        q                         a        b        q      r
+-+------>-+------> +------>       =>       +-+------>-+------>-+------+----->
<-+------+-+------+-+------+                <-+------+-+------+-+------<-----+

                       q     r                                      q
                 +-+-------+---->                                +------>
                    <------<----+                                <------+
```

Sequence the soup: an "abqr" read indicates that both "a" and "b" were present.

What we can use

- Technologies to write (synthesize) whole sets of DNA strands in parallel
- Technologies to read (sequence) whole sets of DNA strands in parallel
- An architecture to do computation on DNA strands and produce sequenceable results
- Hence … highly concurrent algorithms!

Coincidence Recorder

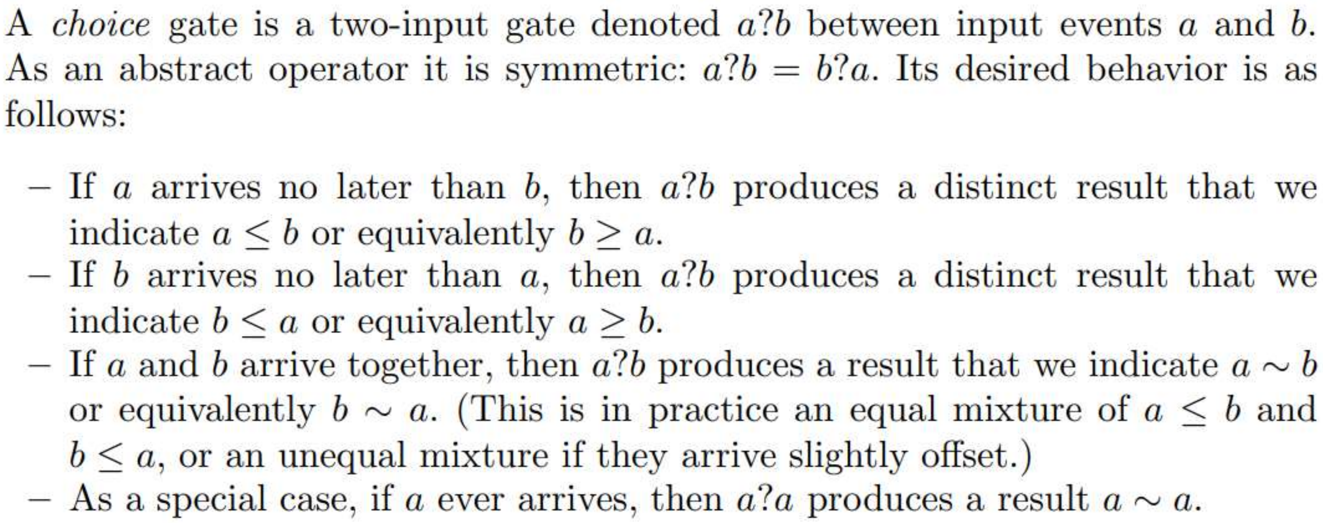
Goal: determine which pairs of a set of events were present *together* in the pot.

Algorithm:
At the beginning, add *all* the pairs join(x,y) for x,y in Events.
At the end, sequence the whole pot.
End.

N.B. join(x,x) tells us if x was ever present.

N^2 algorithm: great, we make "good use" of high-throughput synthesis and sequencing!
It uses no power when events are not present (it does not record *timing*, only *coincidence*).

Choice gate Specification



A *choice* gate is a two-input gate denoted $a?b$ between input events a and b. As an abstract operator it is symmetric: $a?b = b?a$. Its desired behavior is as follows:

- If a arrives no later than b, then $a?b$ produces a distinct result that we indicate $a \leq b$ or equivalently $b \geq a$.
- If b arrives no later than a, then $a?b$ produces a distinct result that we indicate $b \leq a$ or equivalently $a \geq b$.
- If a and b arrive together, then $a?b$ produces a result that we indicate $a \sim b$ or equivalently $b \sim a$. (This is in practice an equal mixture of $a \leq b$ and $b \leq a$, or an unequal mixture if they arrive slightly offset.)
- As a special case, if a ever arrives, then $a?a$ produces a result $a \sim a$.

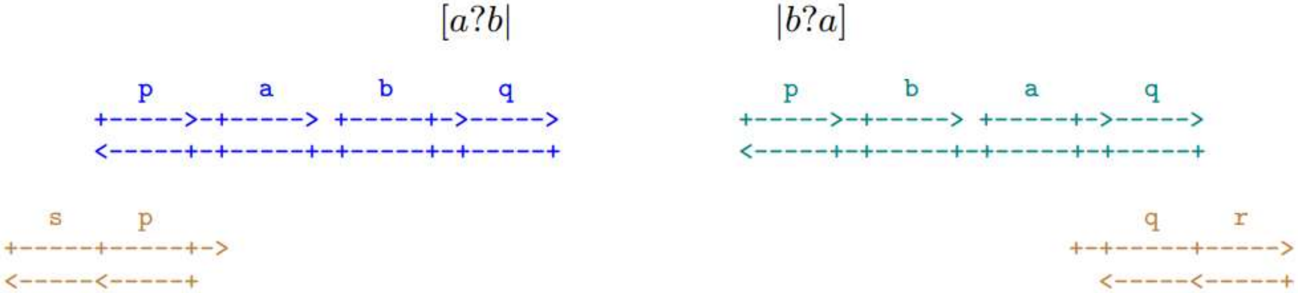
That is, we want to implement the CRN:

"a?b" + a -> a + "a≤b"
"a?b" + b -> b + "b≤a"

But by the general scheme in Part 2
this would not be sequenceable
(and would require too many distinct domains)

Sequenceable Choice gate

$$a?b = [a?b| + |b?a] = [b?a| + |a?b] = b?a.$$



$[a?b|$

```
   p       a       b       q
+----->-+-----> +-----+->----->
<-----+-+-----+-+-----+-+-----+
```

$|b?a]$

```
   p       b       a       q
+----->-+-----> +-----+->----->
<-----+-+-----+-+-----+-+-----+
```

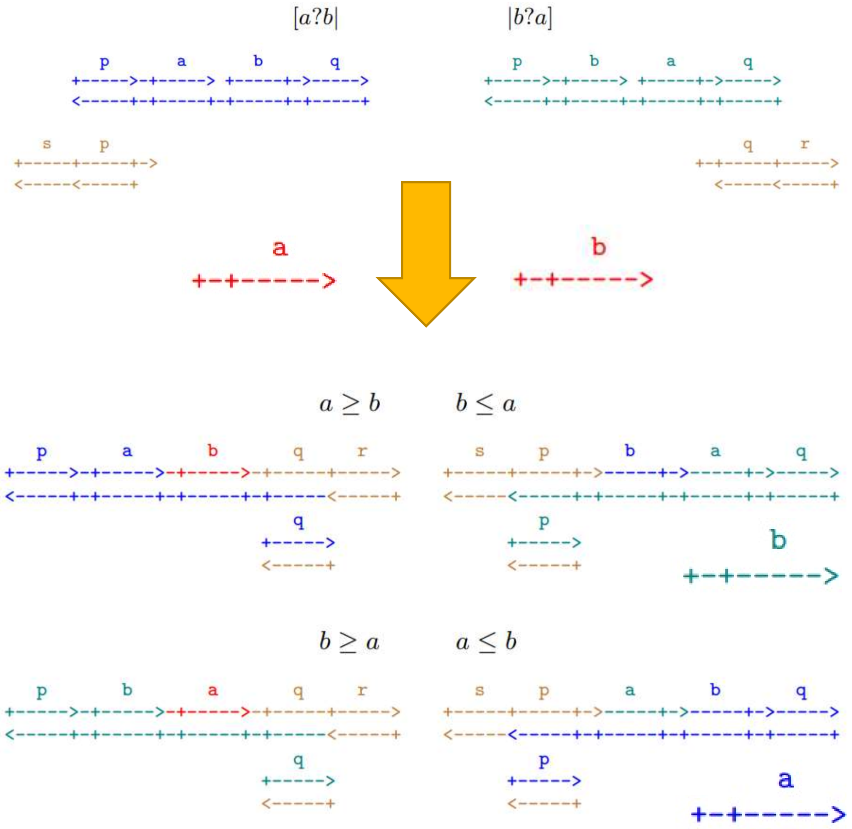
```
  s     p
+-----+-----+->
<-----<-----+
```

```
       q     r
+-+-----+----->
  <-----<-----+
```

(also clonable)

Sequenceable Choice gate outcomes

$a?b$



If b arrives first:

If a arrives first:

Sequencing pattern:

pabqr + spbaq

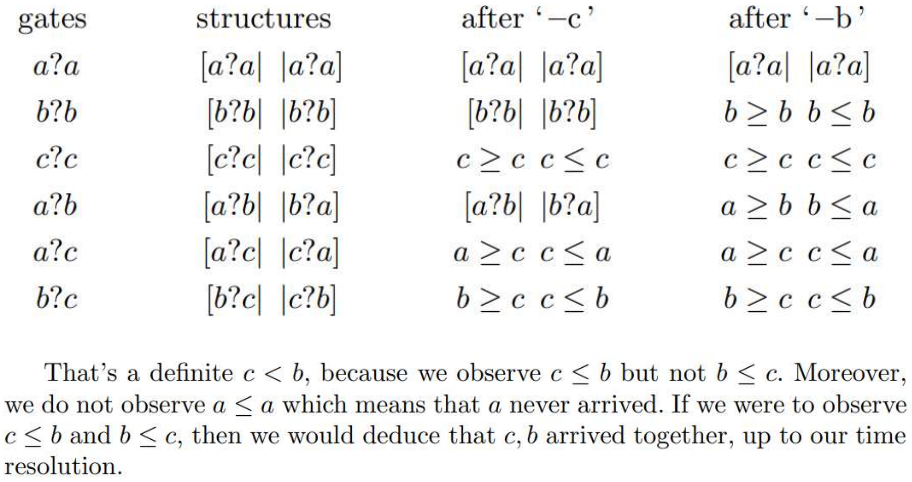
pbaqr + spabq

Preorder Recorder

Goal: Record the preorder of first arrivals of a set of events that occur in a pot.

Algorithm:
At the beginning, add *all* the pairs x?y, for x,y in Events.
At the end, sequence the whole pot and reconstruct the preorder by transitive reduction.
End.

E.g.: Events = {a,b,c}



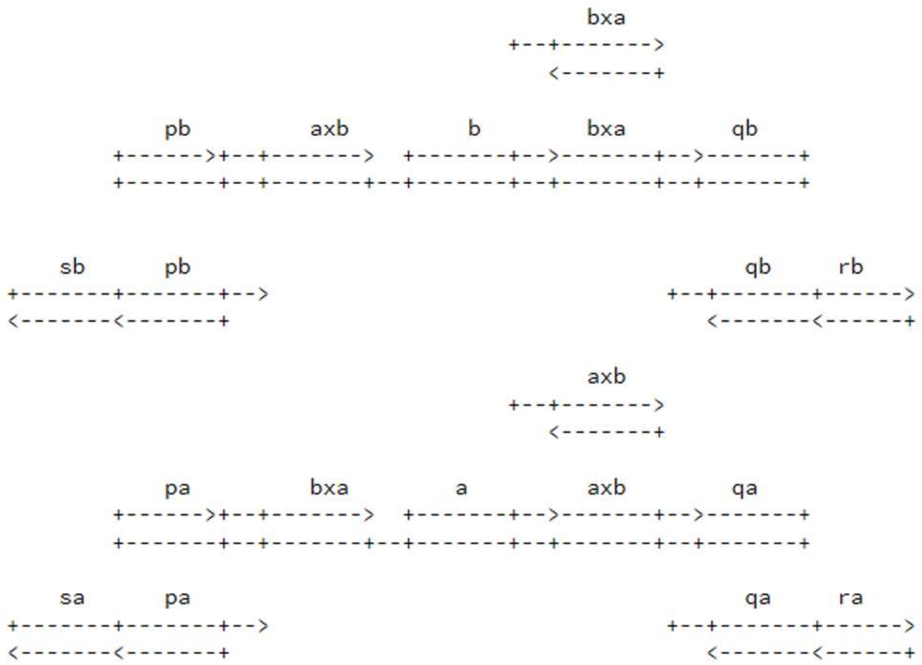
gates	structures	after '−c'	after '−b'
$a?a$	$[a?a\|\ \ \|a?a]$	$[a?a\|\ \ \|a?a]$	$[a?a\|\ \ \|a?a]$
$b?b$	$[b?b\|\ \ \|b?b]$	$[b?b\|\ \ \|b?b]$	$b \geq b\ \ b \leq b$
$c?c$	$[c?c\|\ \ \|c?c]$	$c \geq c\ \ c \leq c$	$c \geq c\ \ c \leq c$
$a?b$	$[a?b\|\ \ \|b?a]$	$[a?b\|\ \ \|b?a]$	$a \geq b\ \ b \leq a$
$a?c$	$[a?c\|\ \ \|c?a]$	$a \geq c\ \ c \leq a$	$a \geq c\ \ c \leq a$
$b?c$	$[b?c\|\ \ \|c?b]$	$b \geq c\ \ c \leq b$	$b \geq c\ \ c \leq b$

That's a definite $c < b$, because we observe $c \leq b$ but not $b \leq c$. Moreover, we do not observe $a \leq a$ which means that a never arrived. If we were to observe $c \leq b$ and $b \leq c$, then we would deduce that c, b arrived together, up to our time resolution.

Correctness

- The choice gate presented here is "faulty by design"
  - There is cross-talk e.g. between a?b and b?c
  - But it turns out this does not hurt the *particular* preorder recorder algorithm
- A proper choice gate can be designed
  - That avoids cross-talk and can be used compositionally in other algorithms
  - But it is more complex and more expensive (O(n^2) distinct domains needed)
- Correctness of the preorder recorder is non-trivial
  - It depends on non-compositional properties of the choice gate
  - It uses n^2 gates, but only O(n) distinct domains. This is important in practice.

Proper Choice Gate



```
                                            bxa
                                      +--+------->
                                         <-------+

            pb         axb         b        bxa        qb
        +------>+--+------->  +-------+-->-------+-->-------+
        +-------+--+-------+--+-------+--+-------+--+-------+

    sb      pb                                          qb     rb
+-------+-------+-->                              +--+-------+------>
<-------<-------+                                    <-------<------+

                                            axb
                                      +--+------->
                                         <-------+

            pa         bxa        a         axb        qa
        +------>+--+------->  +-------+-->-------+-->-------+
        +-------+--+-------+--+-------+--+-------+--+-------+

    sa      pa                                          qa     ra
+-------+-------+-->                              +--+-------+------>
<-------<-------+                                    <-------<------+
```

- axb,bxa are domains uniquely determined by a,b

Conclusions

- Technological advances
  - High-throughput synthesis and sequencing

- Provide new readout opportunities
  - Reading and writing n^2 elements feasibly

- Which can inspire a new class of parallel algorithms
  - Coincidence Recorder, Preorder Recorder, … ???

Sequenceable Event Recorders

Luca Cardelli