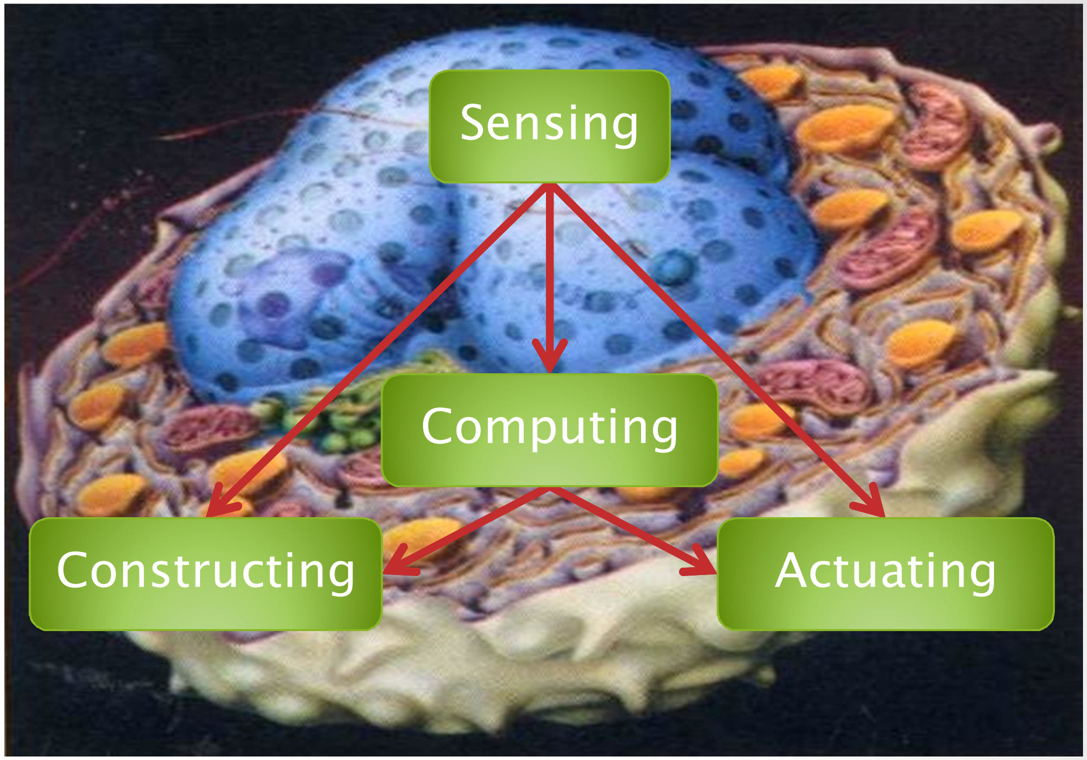
Two-Domain DNA Strand Displacement

Luca Cardelli
Microsoft Research

Tokyo, 2010-06-19
http://lucacardelli.name

Nanoscale Engineering

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

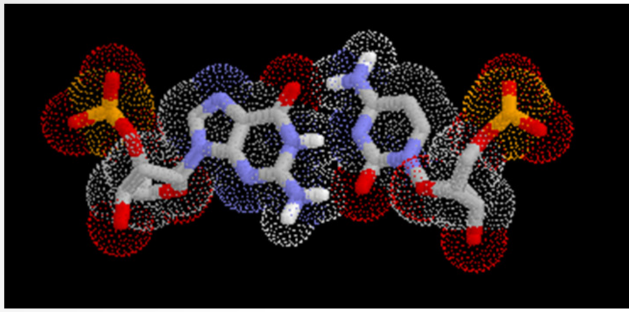


Nucleic Acids can do all this.

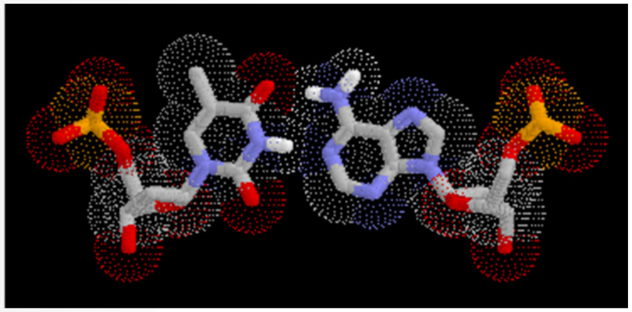
And interface to biology.
And are programmable.

Strand Displacement Basics

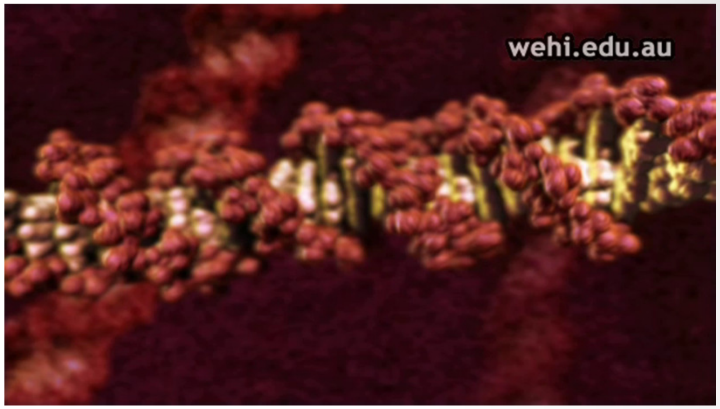
DNA



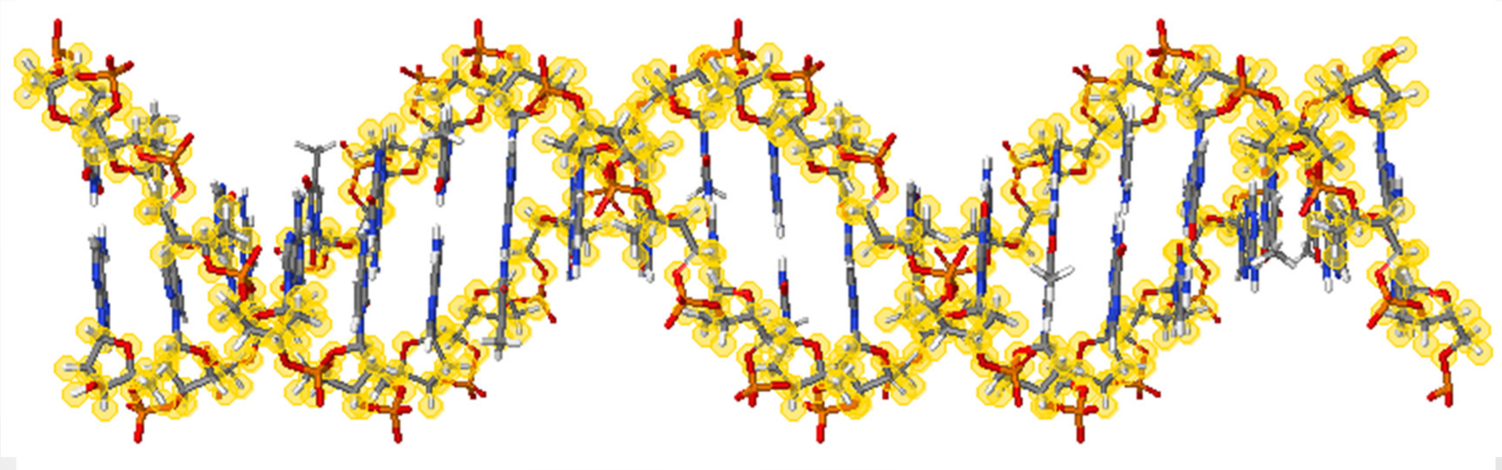
GC Base Pair
Guanine-Cytosine



TA Base Pair
Thymine-Adenine

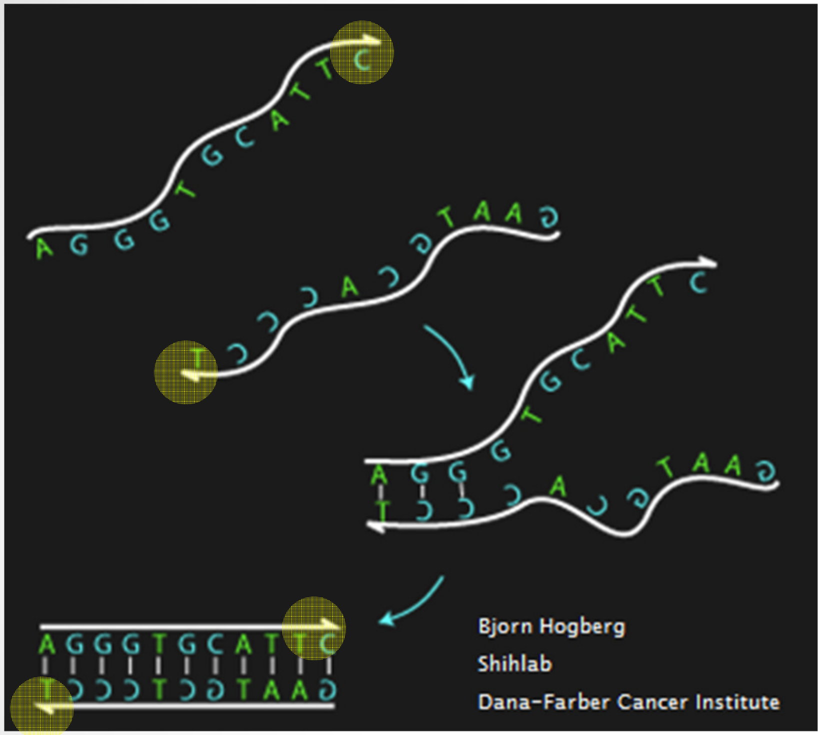


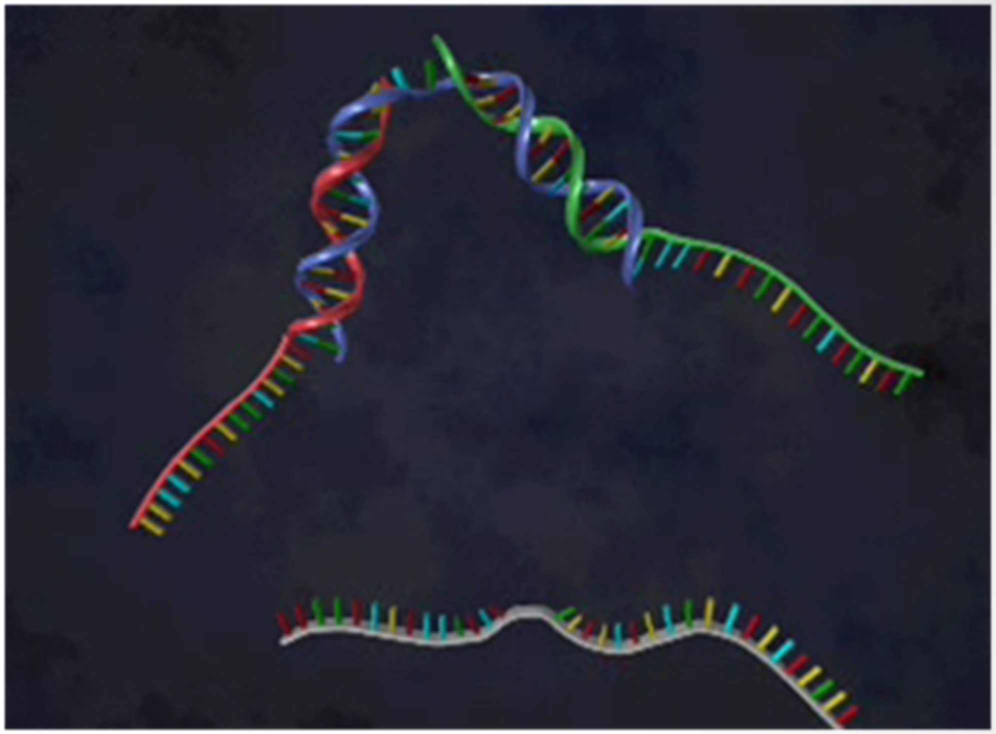
[Interactive DNA Tutorial](http://www.biosciences.bham.ac.uk/labs/minchin/tutorials/dna.html)
(http://www.biosciences.bham.ac.uk/labs/minchin/tutorials/dna.html)



Sequence of Base Pairs (GACT alphabet)

Hybridization





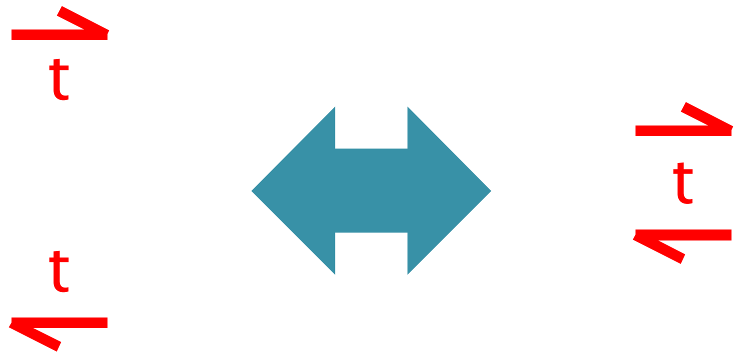
- Strands with **opposite orientation** and **complementary base pairs** stick to each other (Watson-Crick duality).
- This is all we are going to use
  - We are not going to exploit DNA replication, transcription, translation, **restriction and ligation enzymes**, etc., which enable other classes of tricks.

Domains

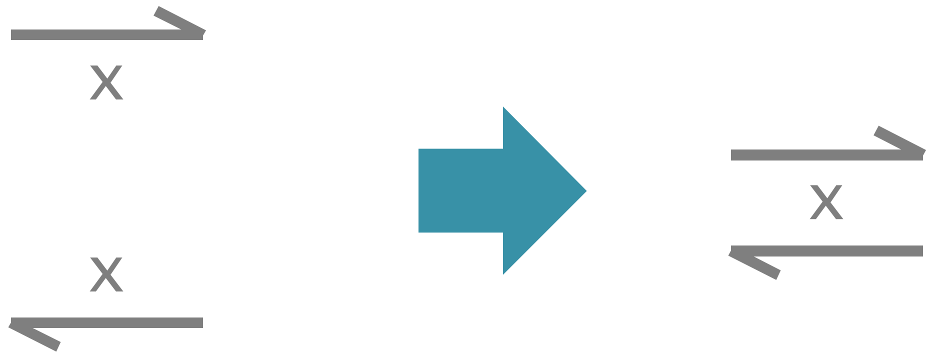
- Subsequences on a DNA strand are called domains.
- PROVIDED they are “independent” of each other.

- 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.
- How to choose 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.

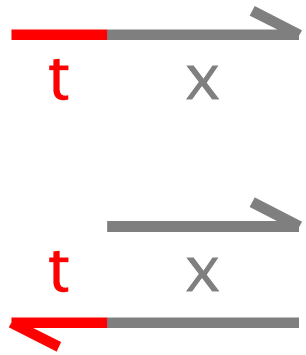
Short Domains



Long Domains

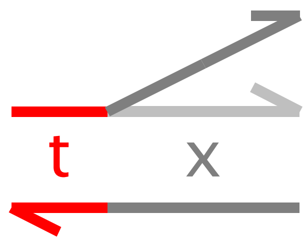


Strand Displacement



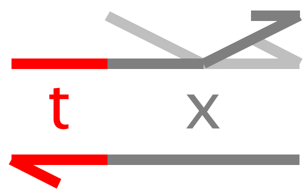
“Toehold Mediated”

Strand Displacement



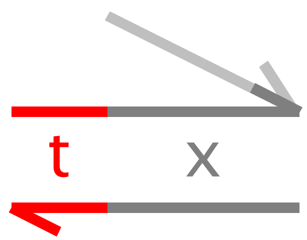
Toehold Binding

Strand Displacement



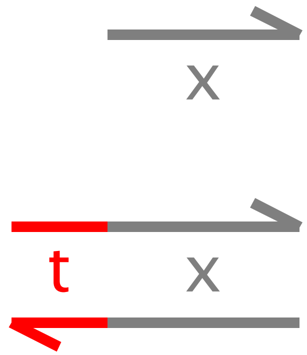
Branch Migration

Strand Displacement



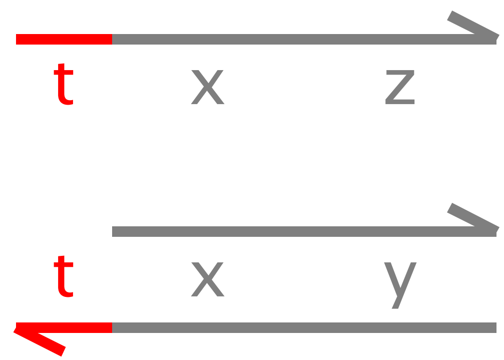
Displacement

Strand Displacement

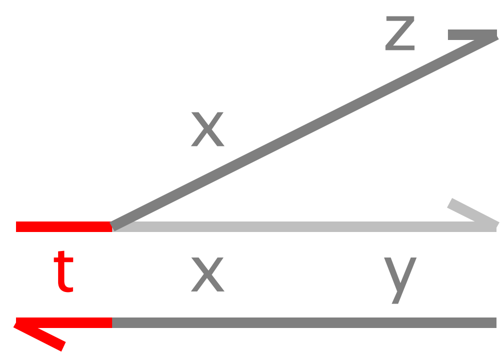


Irreversible

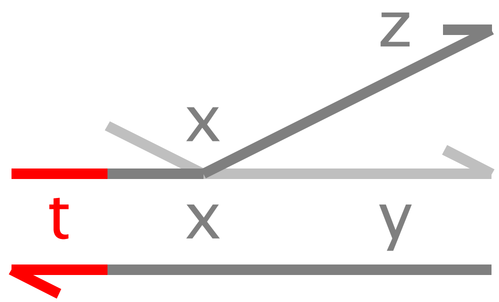
Bad Match



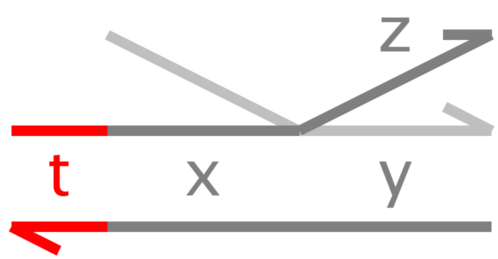
Bad Match



Bad Match

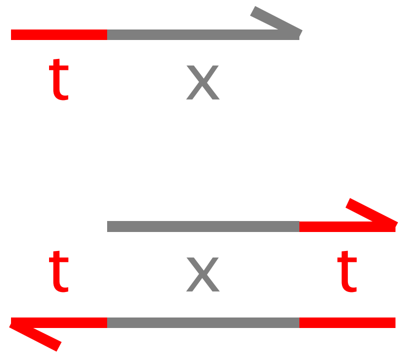


Bad Match

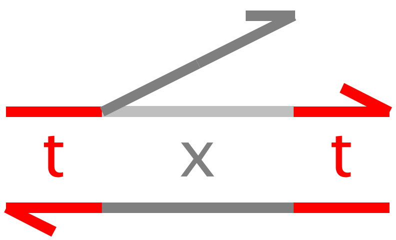


Cannot proceed
Hence will undo

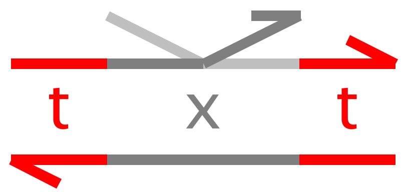
Toehold Exchange



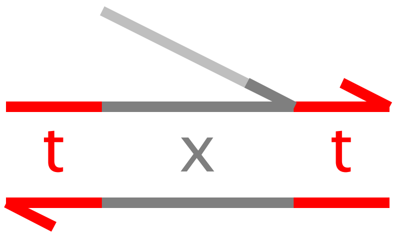
Toehold Exchange



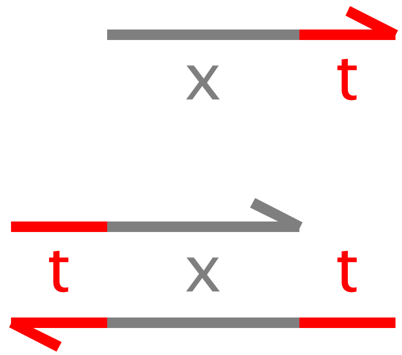
Toehold Exchange



Toehold Exchange

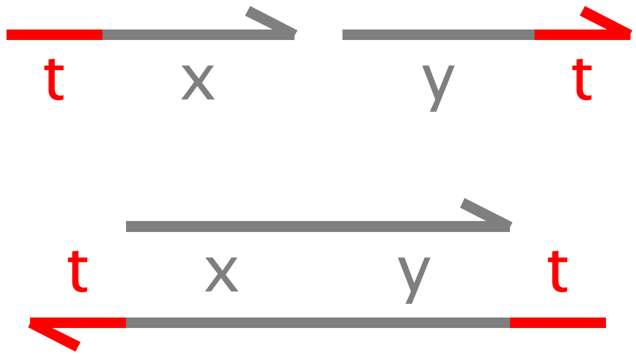


Toehold Exchange

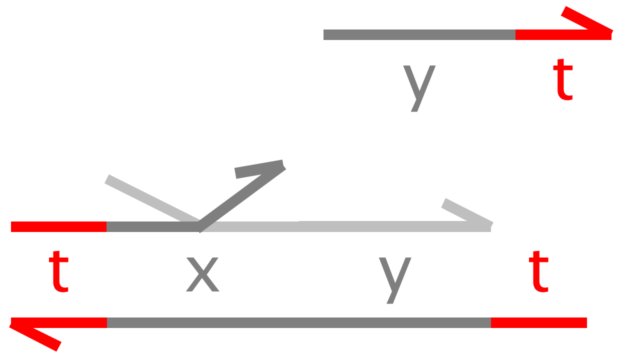


Reversible

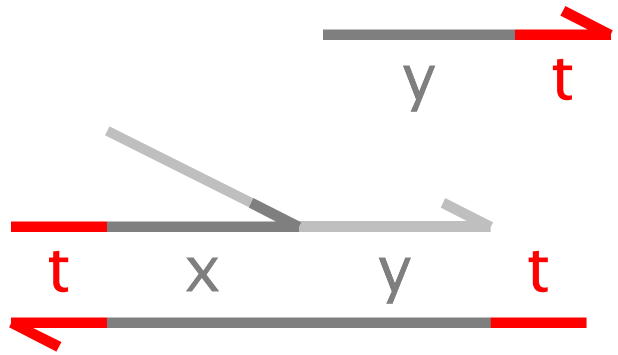
Cooperative Displacement



Cooperative Displacement

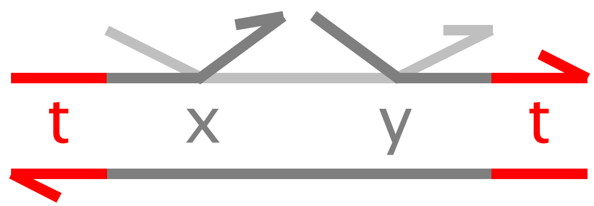


Cooperative Displacement

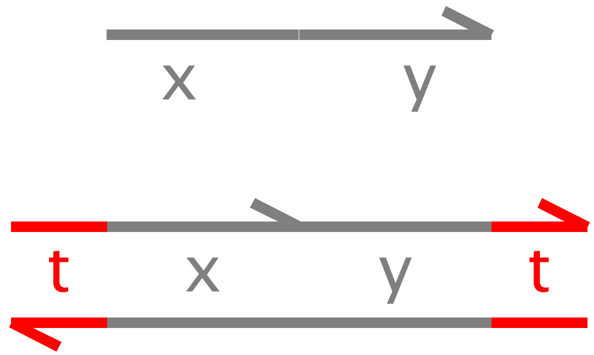


Single input
will reverse

Cooperative Displacement

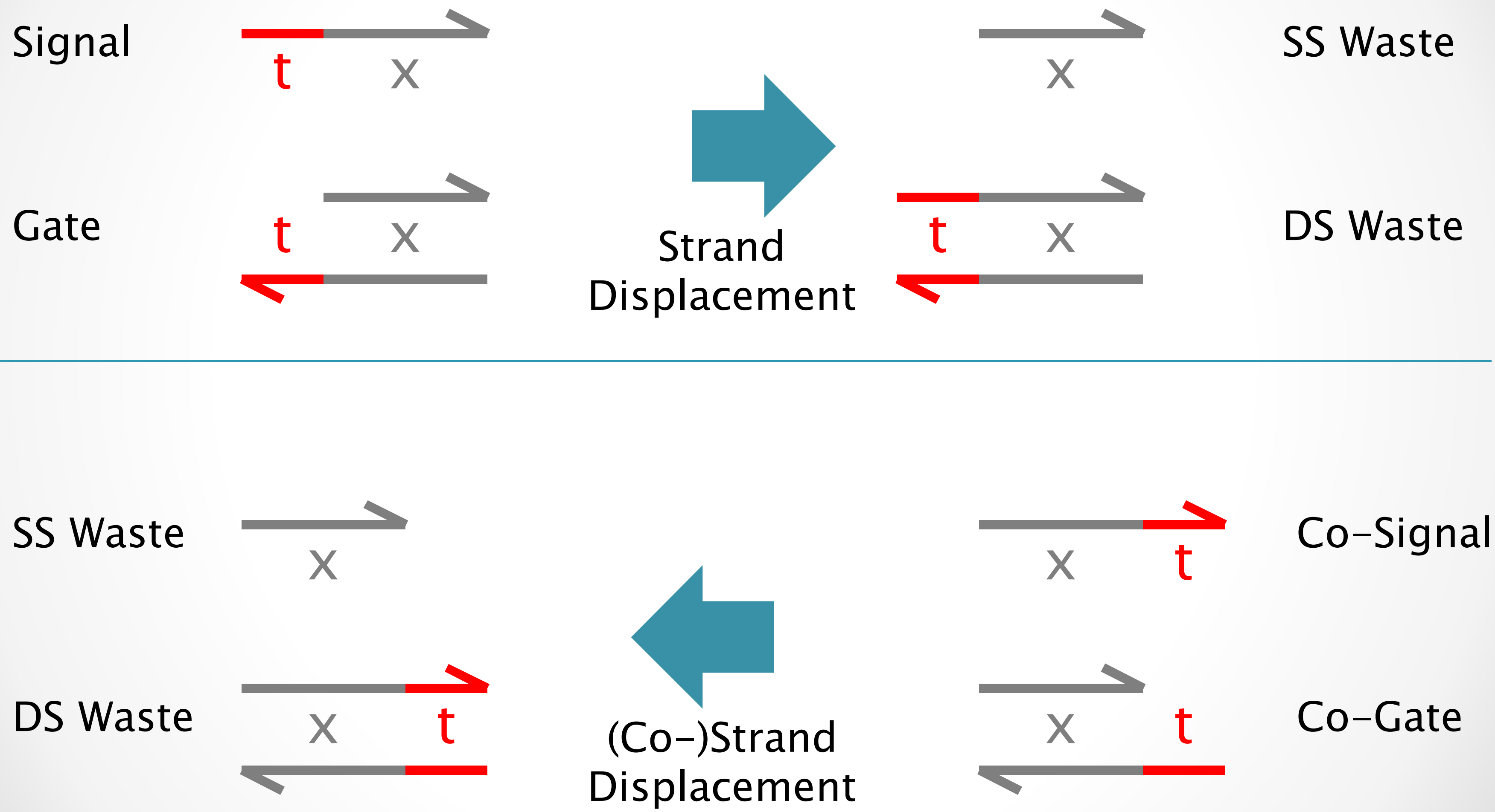


Cooperative Displacement



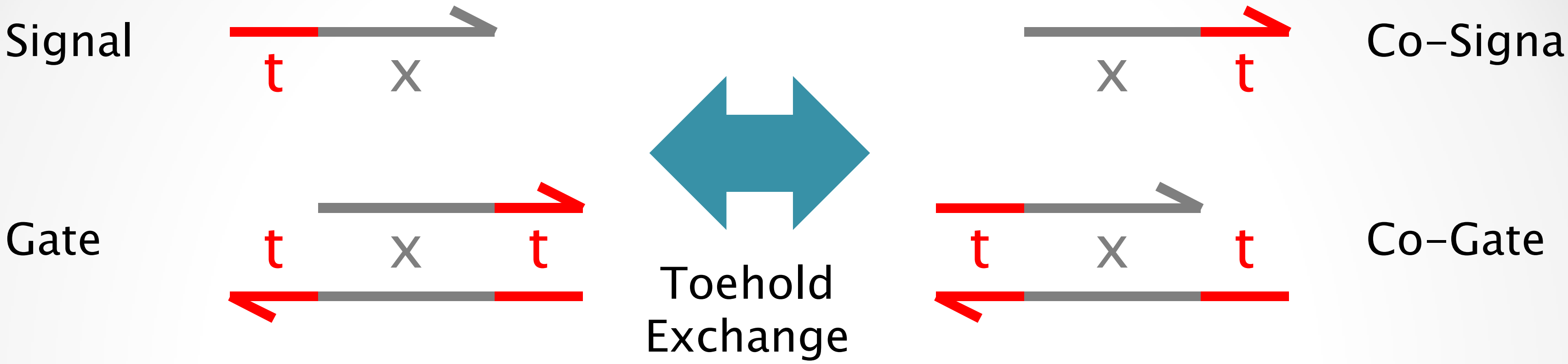
Double input
is irreversible

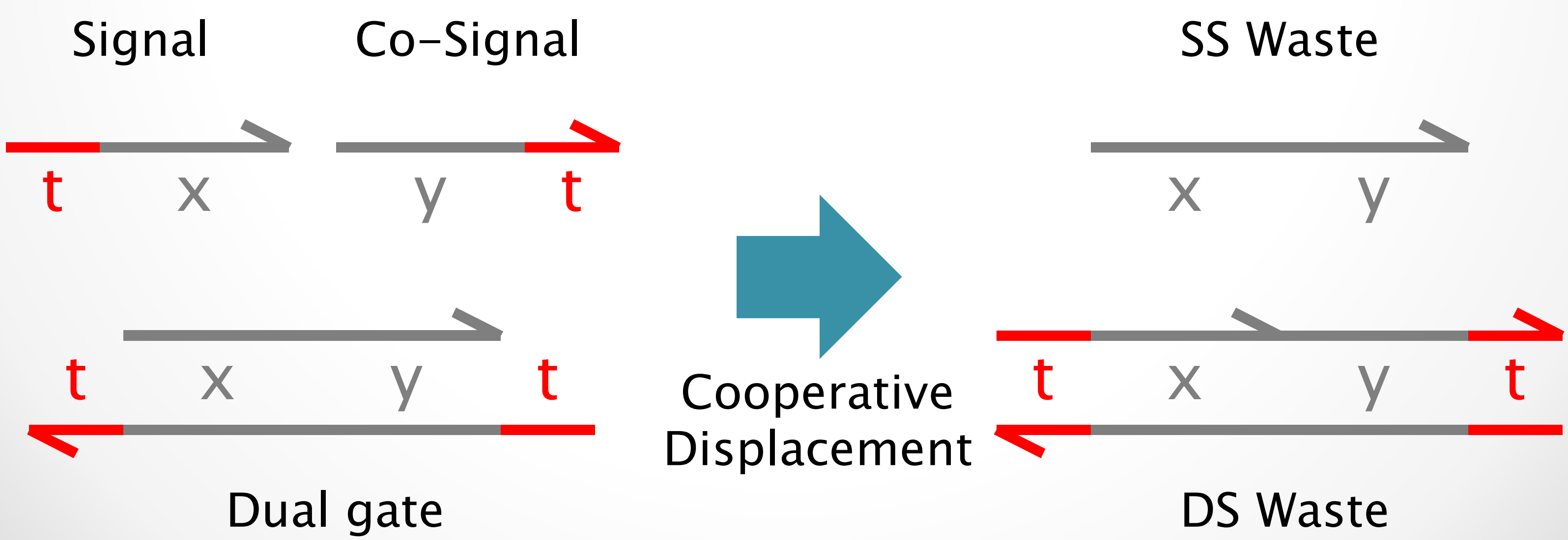
Summary (1)



Signal
t
x
Gate
t
x
Strand Displacement
SS Waste
x
DS Waste
t
x
SS Waste
x
DS Waste
x
t
(Co-)Strand Displacement
Co-Signal
x
t
Co-Gate
x
t

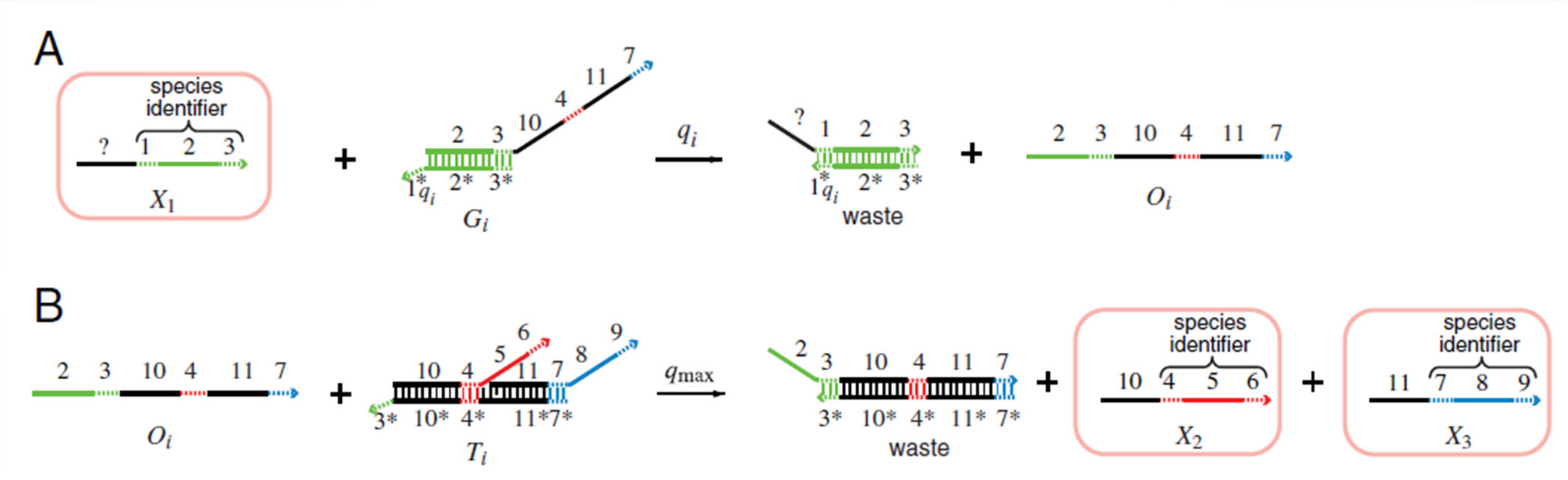
Summary (2)


Signal
t
x
Co-Signal
x
t
Gate
t
x
t
Co-Gate
t
x
t
Toehold
Exchange


Signal
Co-Signal
t
x
y
t
SS Waste
x
y
t
x
y
t
Cooperative
Displacement
t
x
y
t
Dual gate
DS Waste

Signals & Gates

Four-Domain Signals

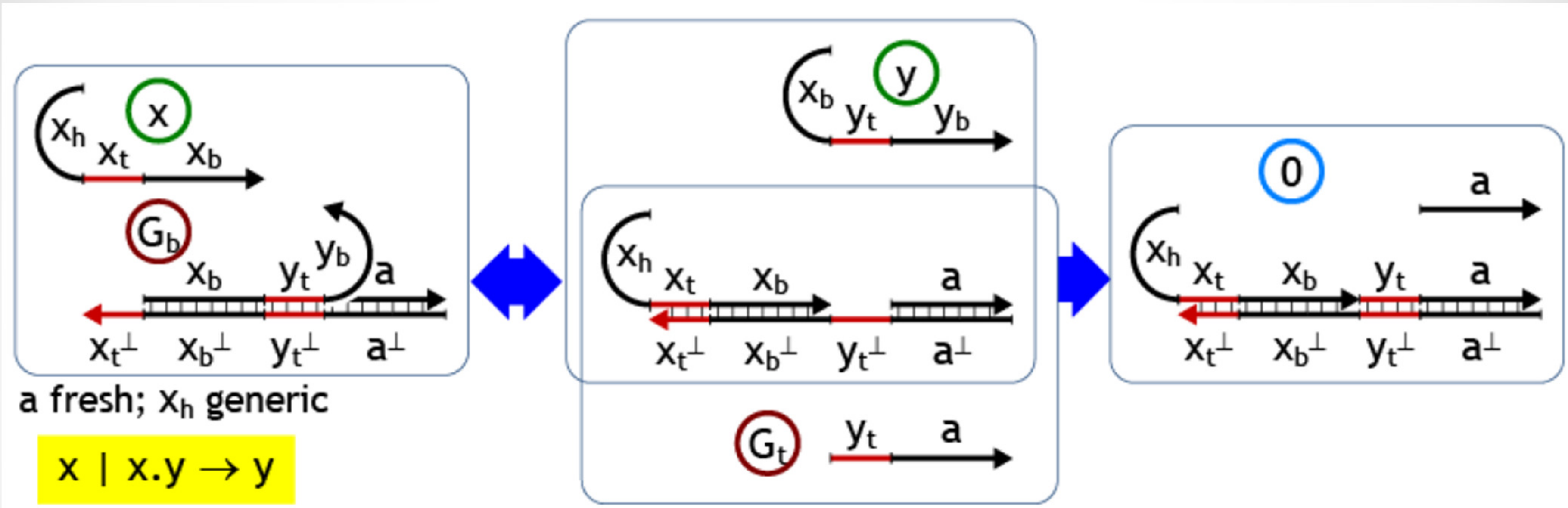


DNA as a universal substrate for chemical kinetics

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 Signals

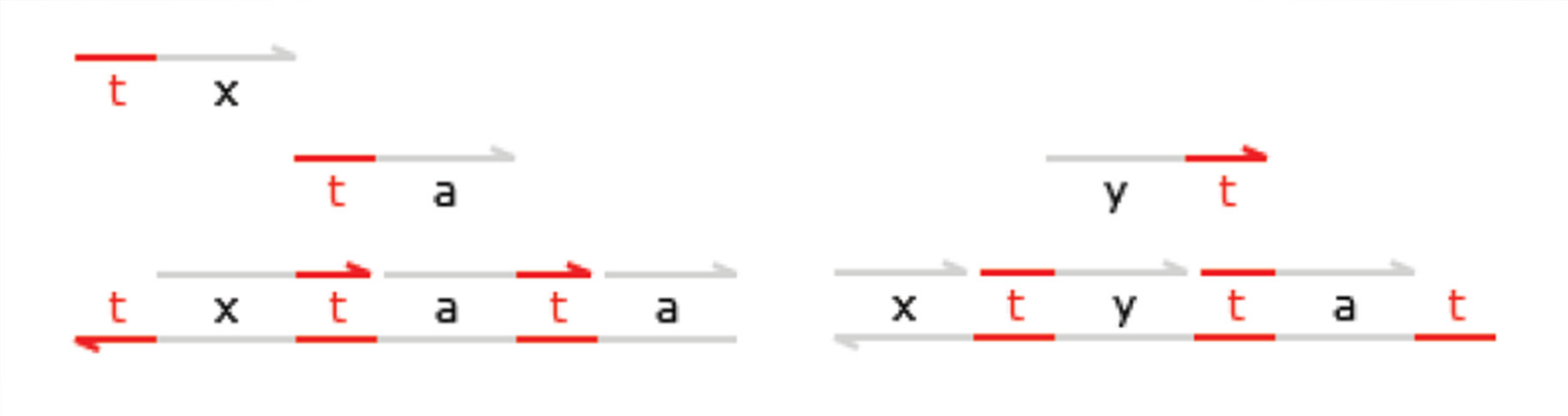


Strand Algebras for DNA Computing

Luca Cardelli

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

Two-Domain Signals



Two-Domain DNA Strand Displacement

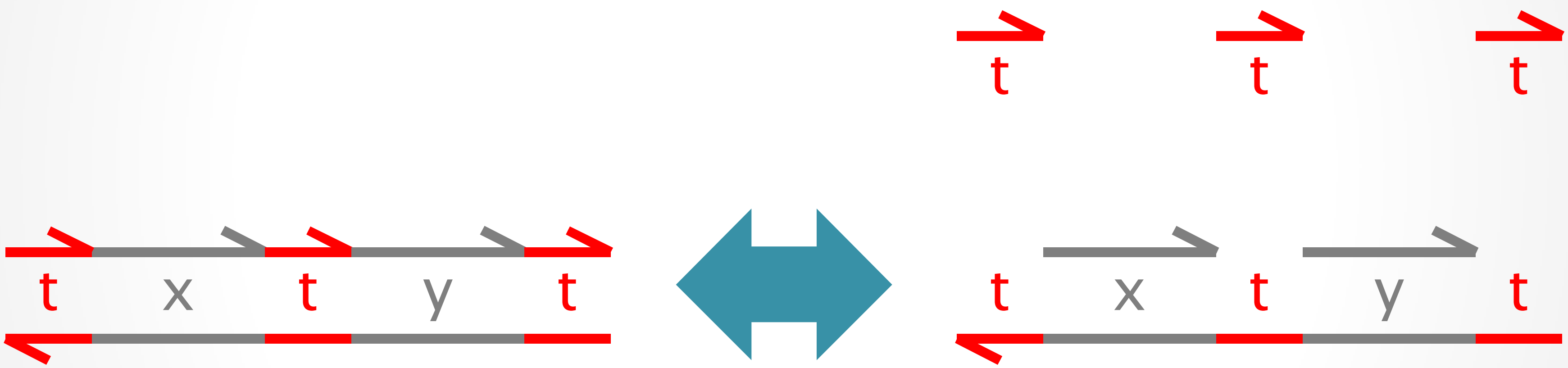
Luca Cardelli

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

Top-Nicked Double Strands

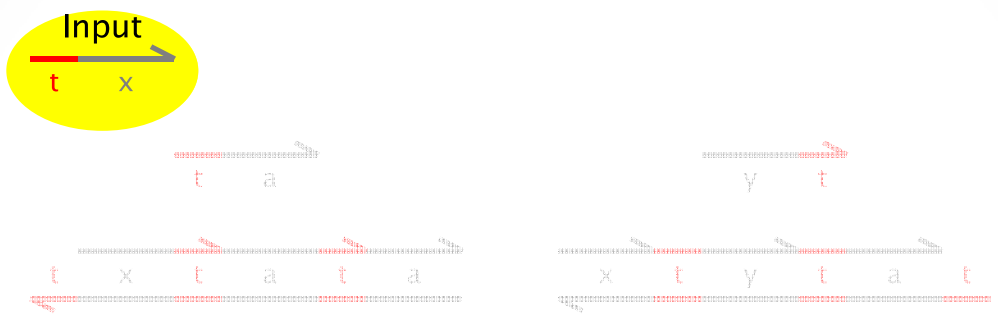
Signals have a simple structure: just two domains.

Gates have a simple structure:
'top-nicked' double-stranded DNA with no 'frills'.

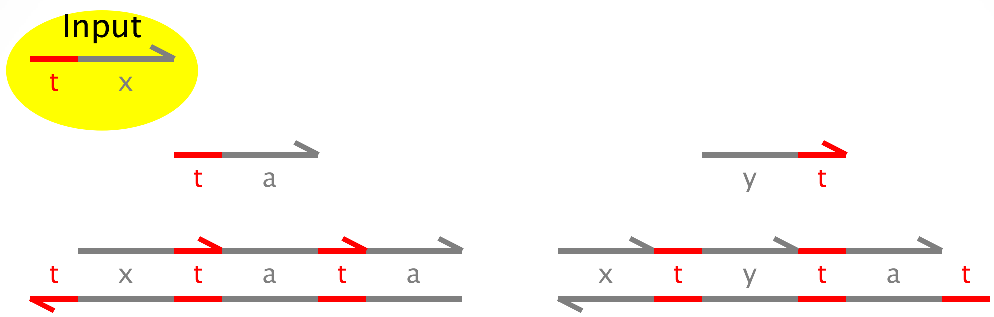


A top-nicked double-strand is 'equivalent'
to a double strand with open toeholds.
These situations shall not be distinguished.

Transducer x→y

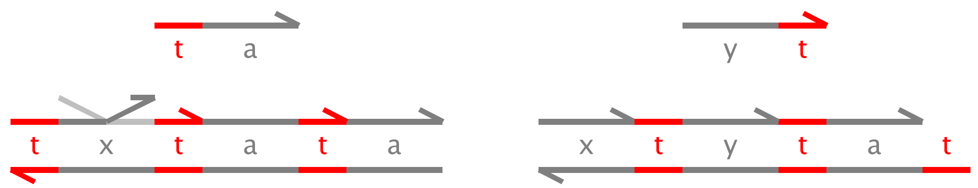


Transducer x→y

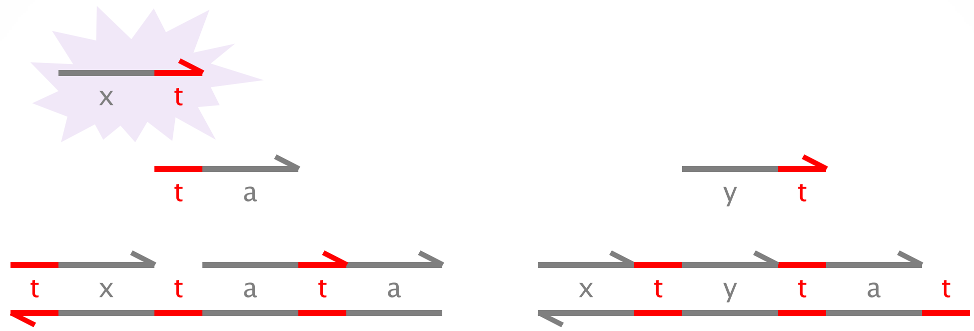


ta is a *private* signal (a different ‘a’ for each xy pair)

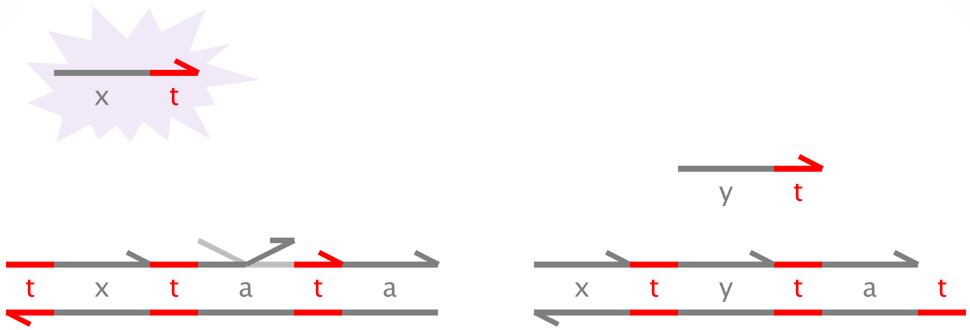
Transducer x→y



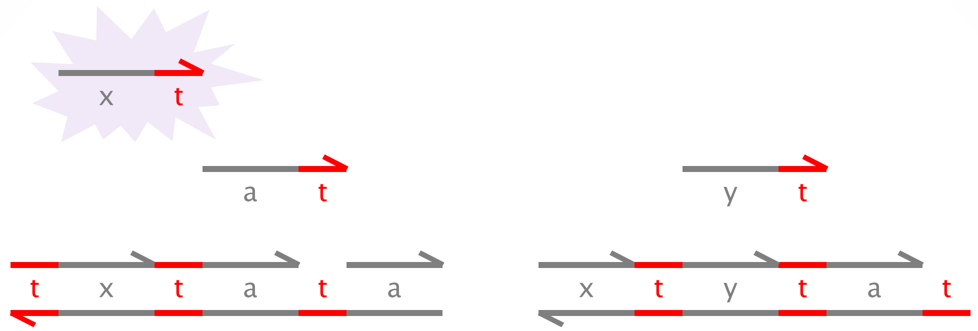
Transducer x→y



Transducer x→y

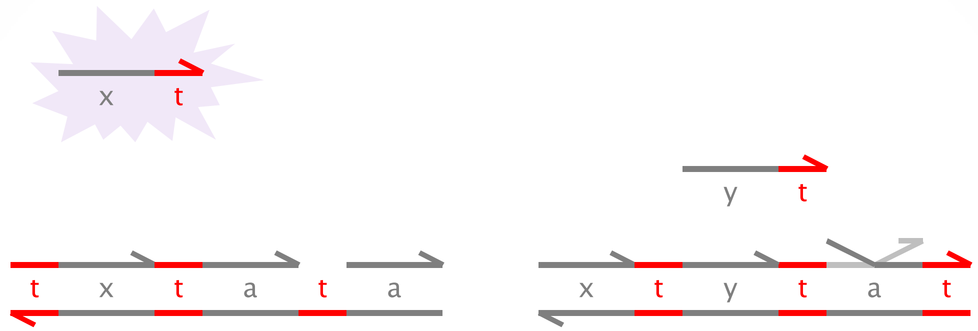


Transducer x→y

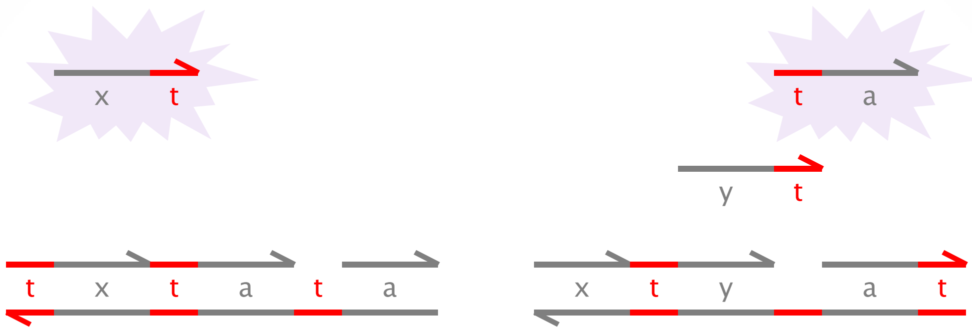


So far, a **tx** *signal* has produced an **at** *cosignal.*
But we want signals as output, not cosignals.

Transducer x→y

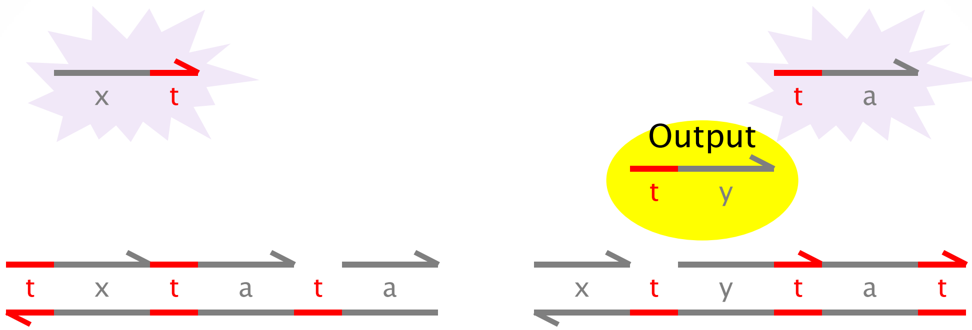


Transducer x→y



Transducer x→y

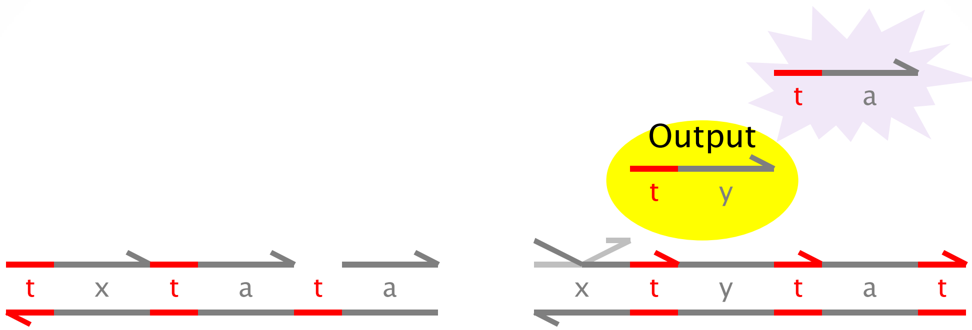
Transducer x→y



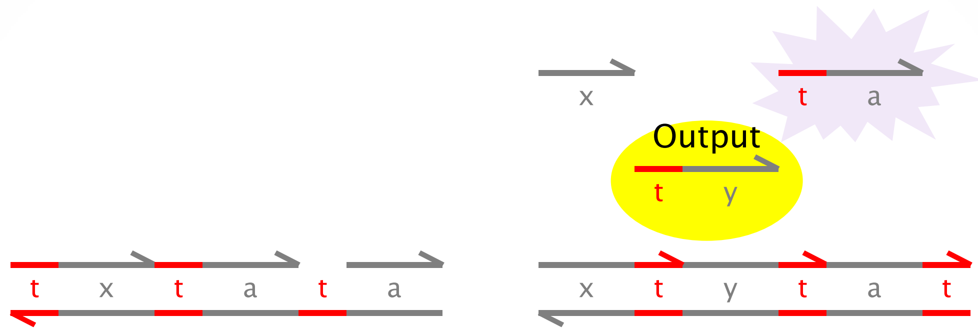
Here is our output **ty** *signal*.

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).

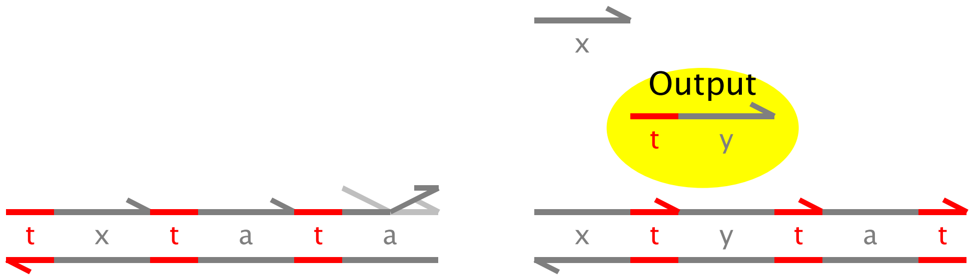
Transducer x→y



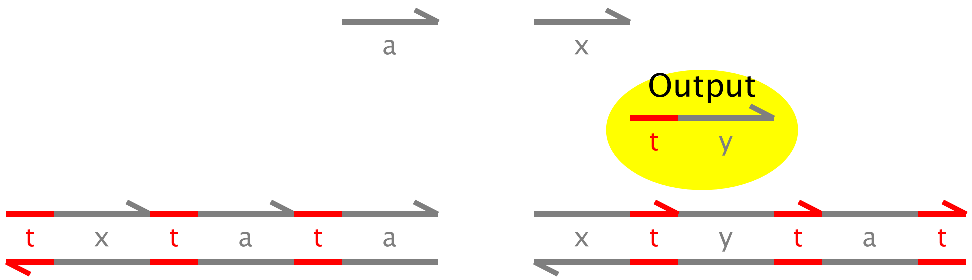
Transducer x→y



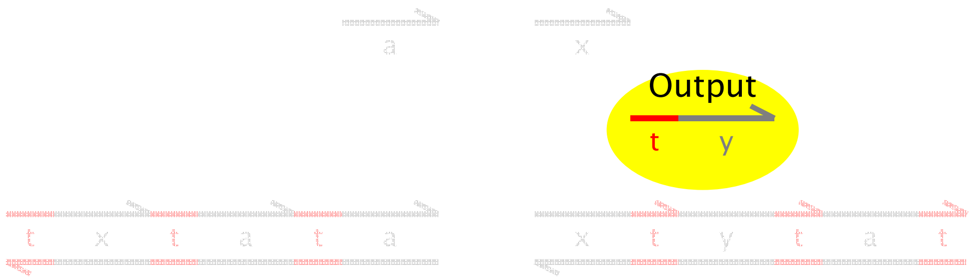
Transducer $x \rightarrow y$



Transducer x→y



Transducer x→y

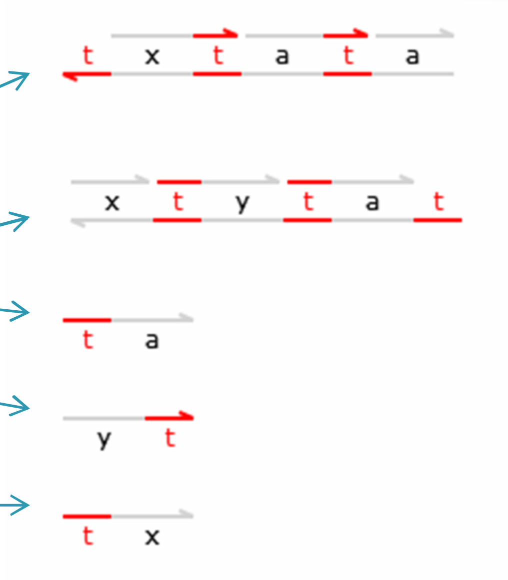


Done.

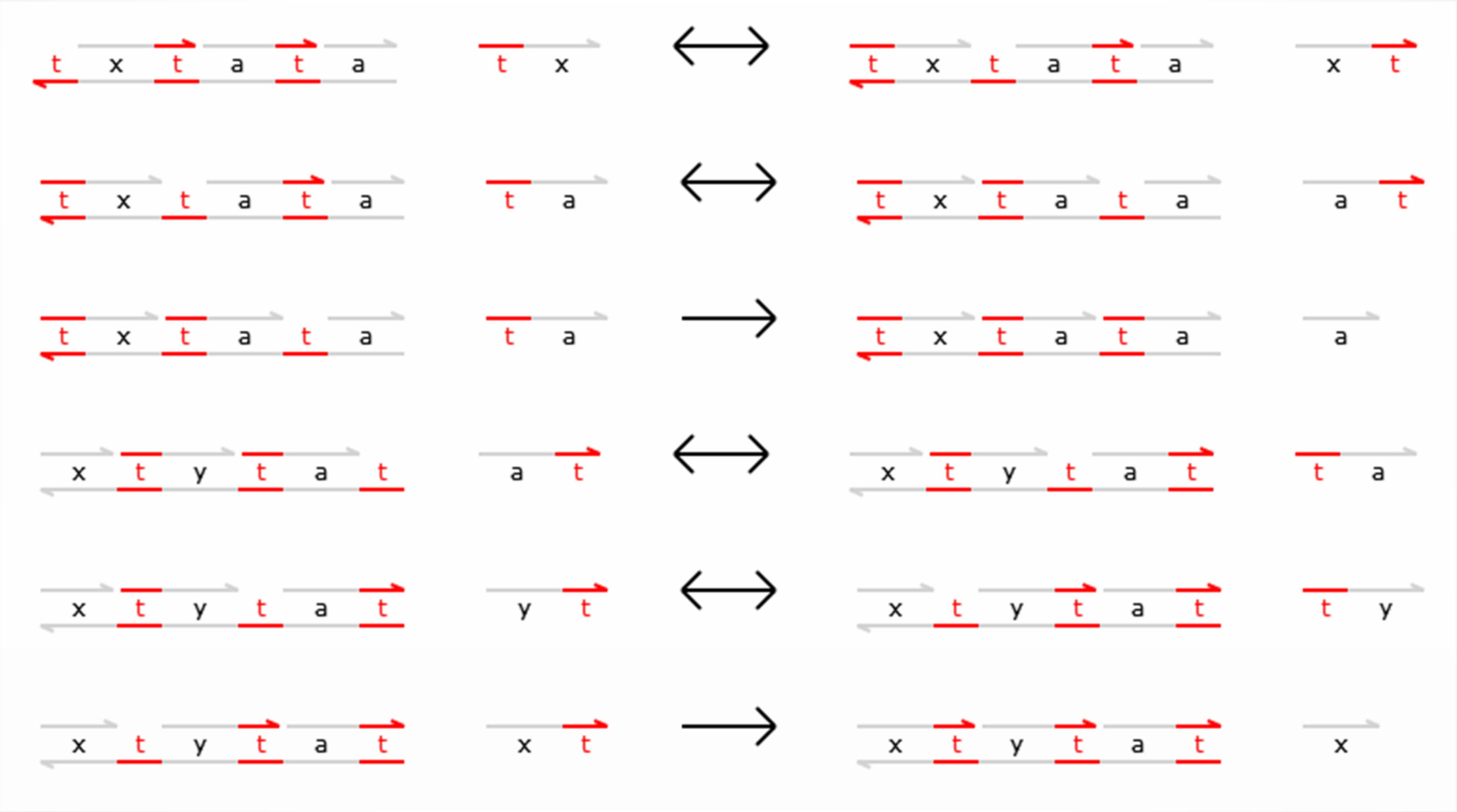
Note the **tata** motif and how it helps in collection.

The Transducer in DSD

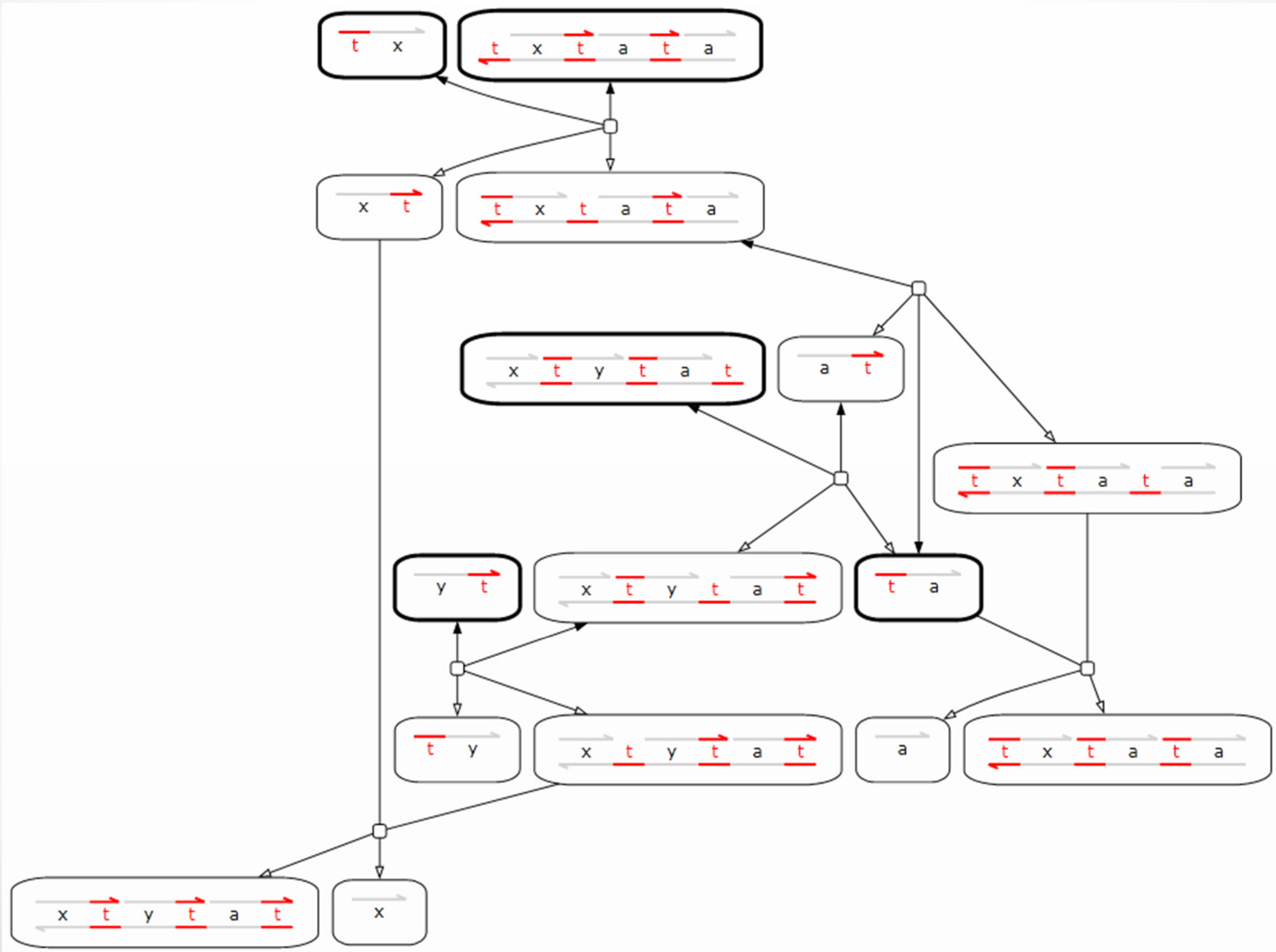
```
directive sample 50.0
directive plot <t^ x>; <t^ y>
directive scale 1.0
new t@1.0,1.0

def Tr(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^
)

( Tr(10, x, y)
| 1* <t^ x>
)
```



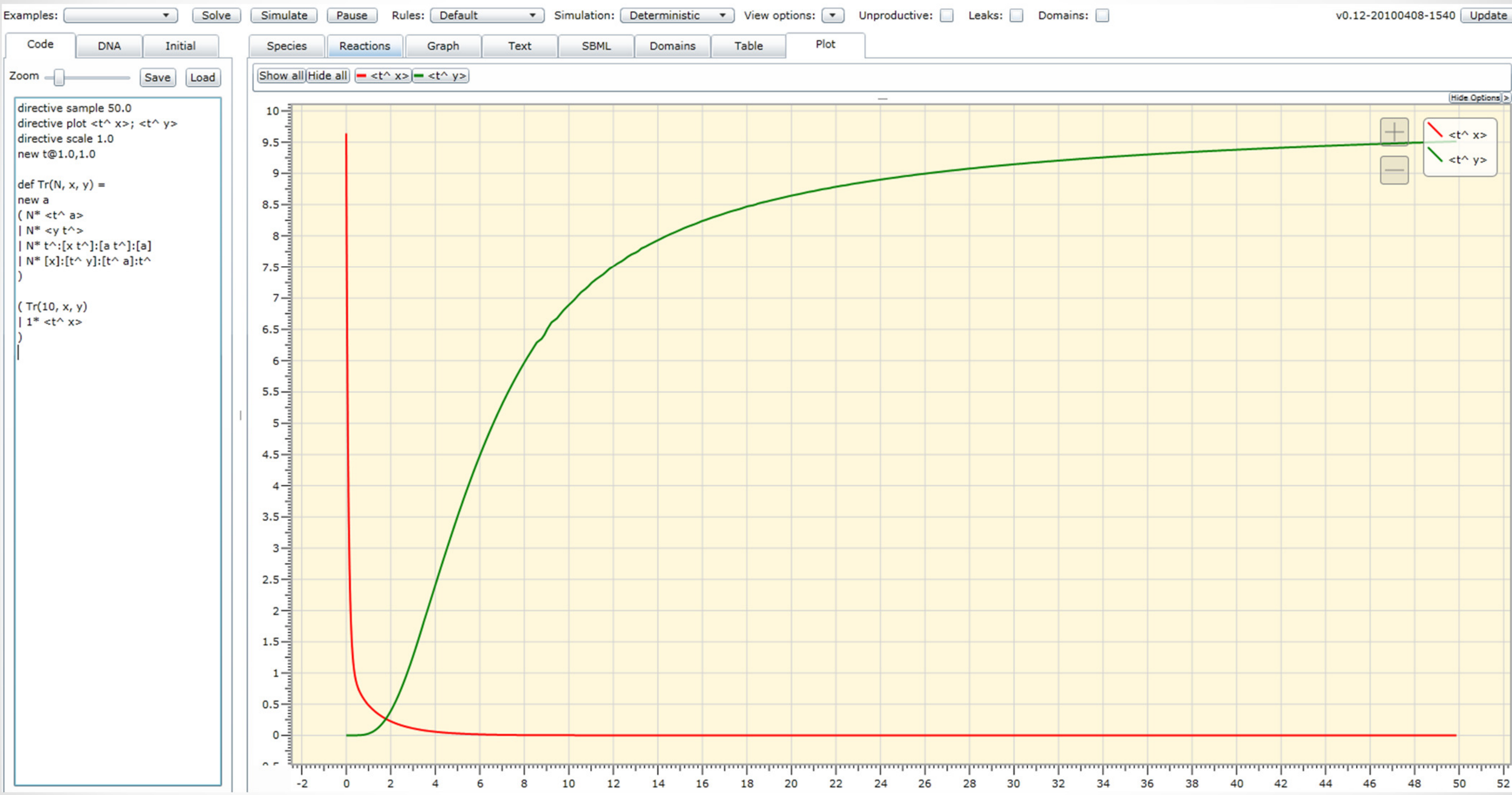
Transducer Reactions



Transducer Reaction Graph

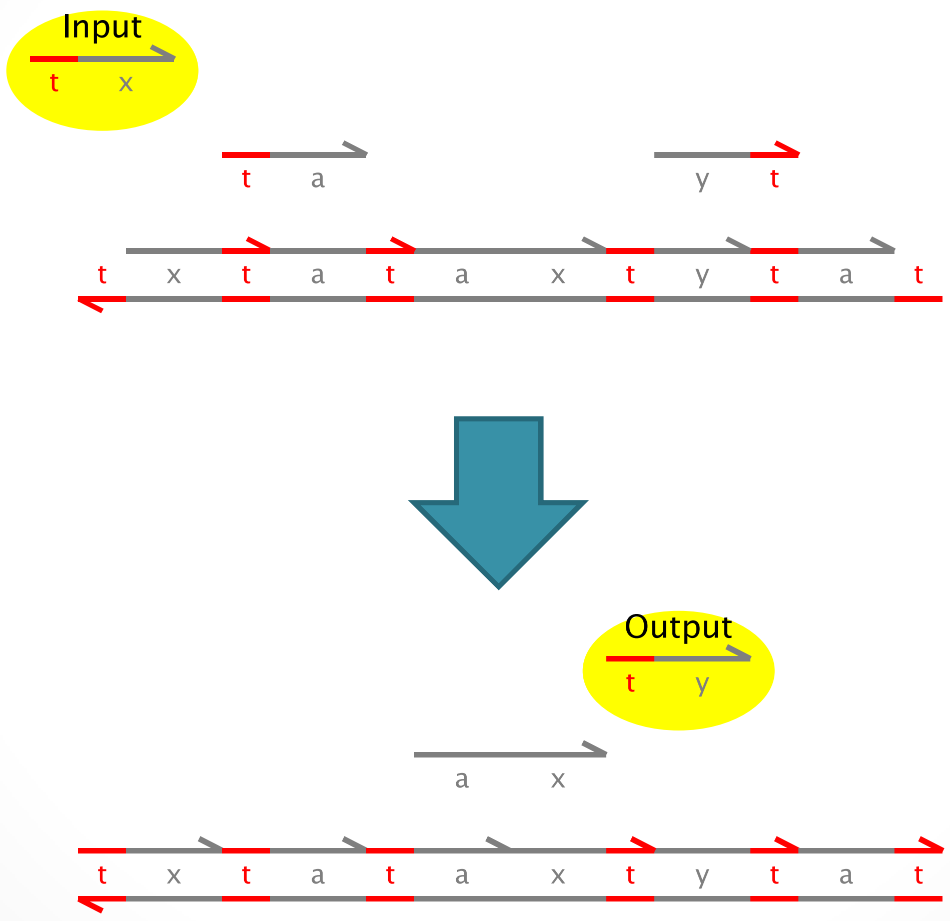


Transducer Simulation



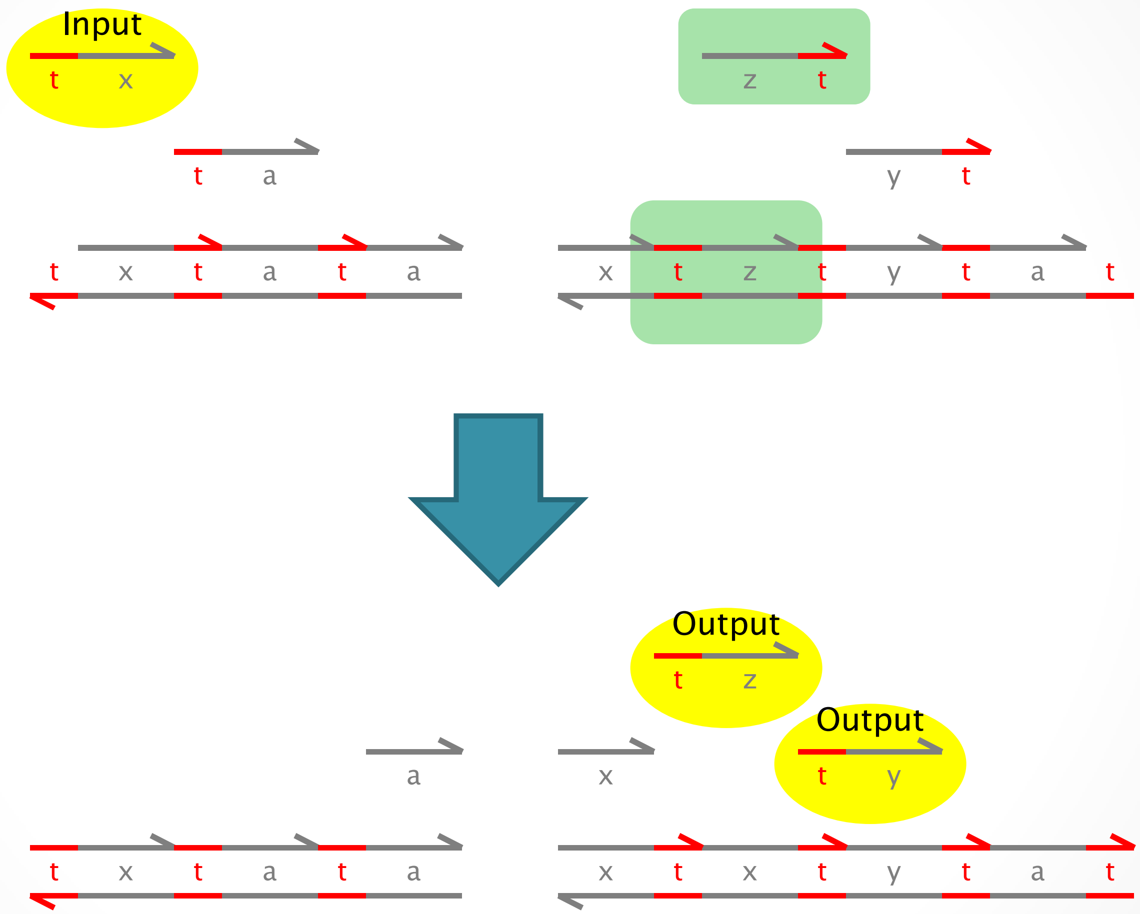
Transducer Variation

Single backbone, using cooperative displacement to remove garbage.



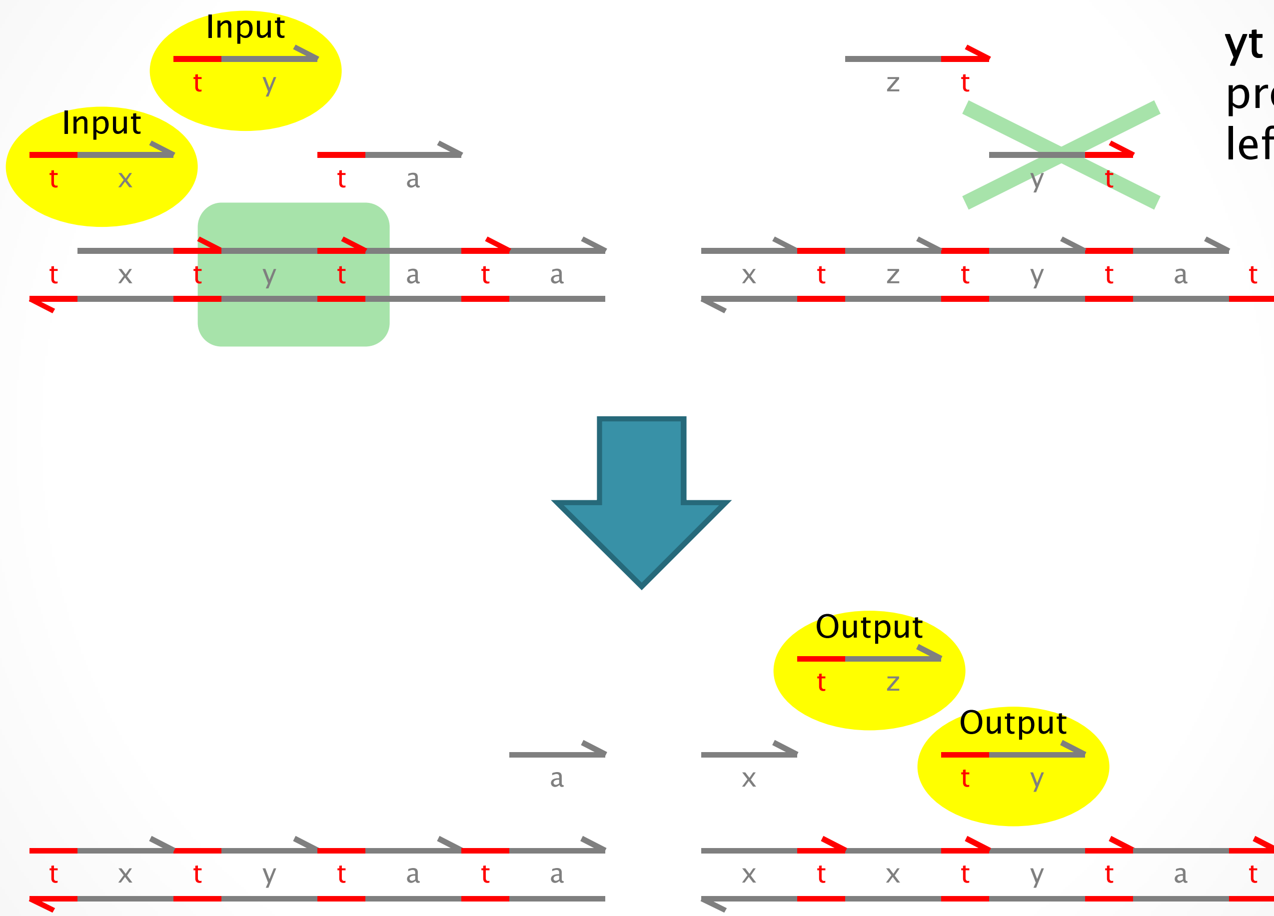
Note: garbage collection by cooperative displacement is optional for the transducer, but becomes essential later.

Fork x→y+z



(Amplifier: x→x+x)

Catalyst x+y→y+z

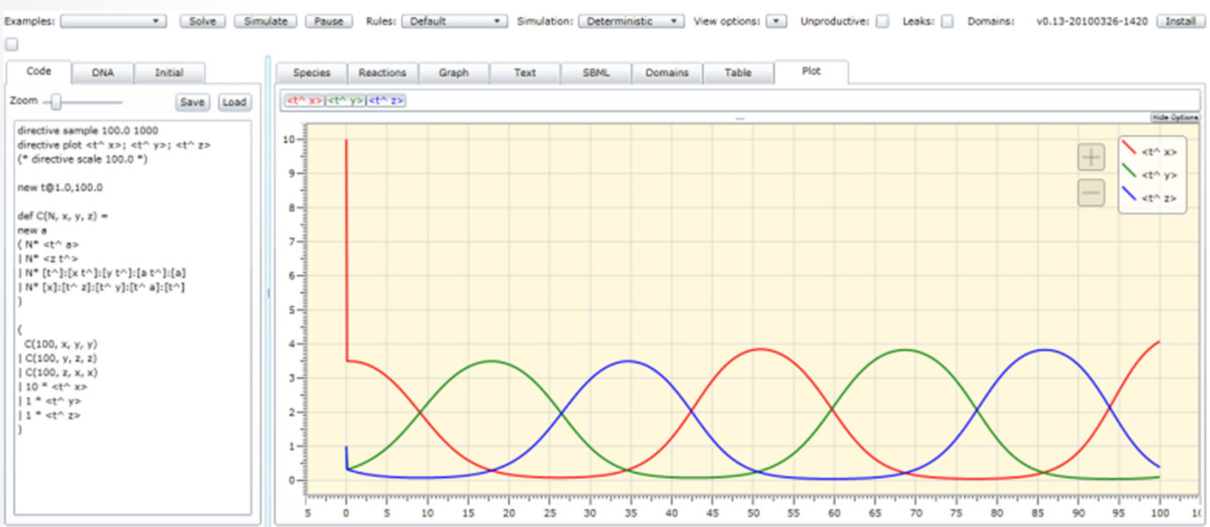


yt is kindly provided by the left hand side.

(Autocatalyst: x+y→y+y)

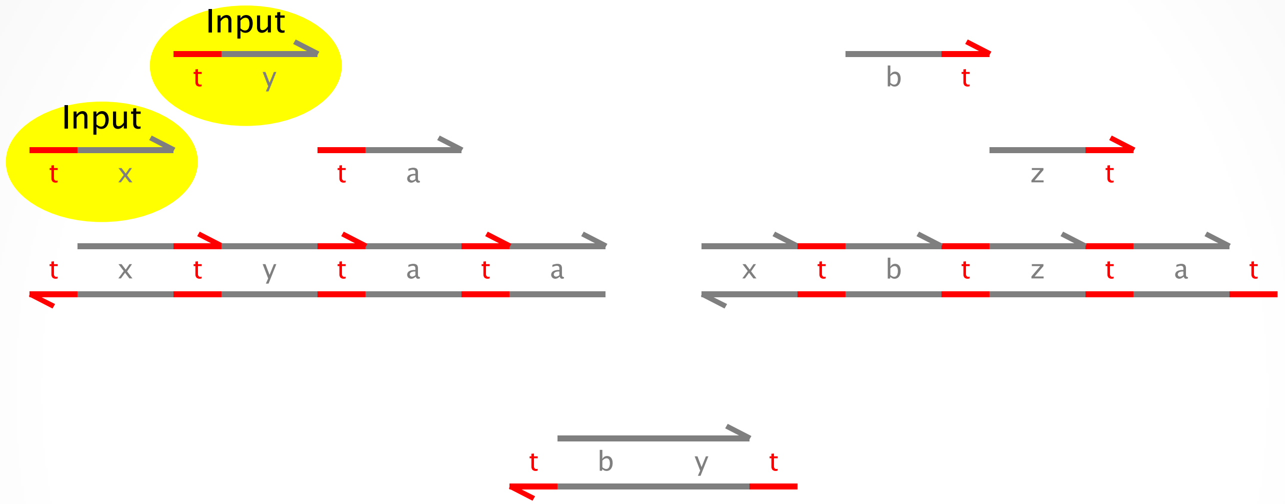
Autocatalytic Oscillator

$x+y \rightarrow y+y$
$y+z \rightarrow z+z$
$z+x \rightarrow x+x$

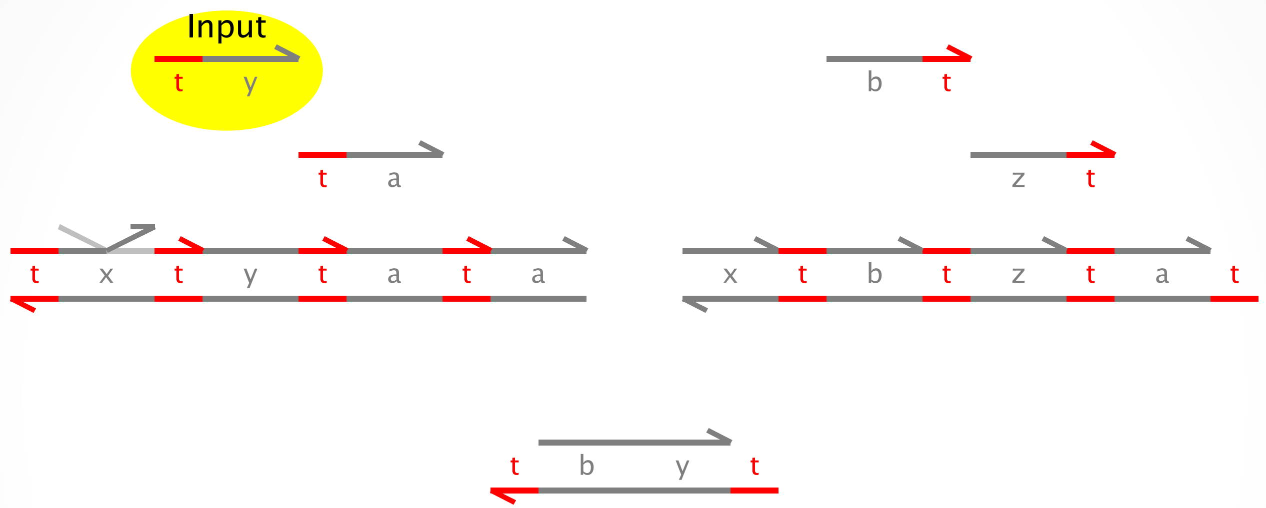



```
directive sample 100.0 1000
directive plot <t^ x>; <t^ y>;
<t^ z>
(* directive scale 100.0 *)

new t@1.0,100.0

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

(
  C(100, x, y, y)
| C(100, y, z, z)
| C(100, z, x, x)
| 10 * <t^ x>
| 1 * <t^ y>
| 1 * <t^ z>
)
```

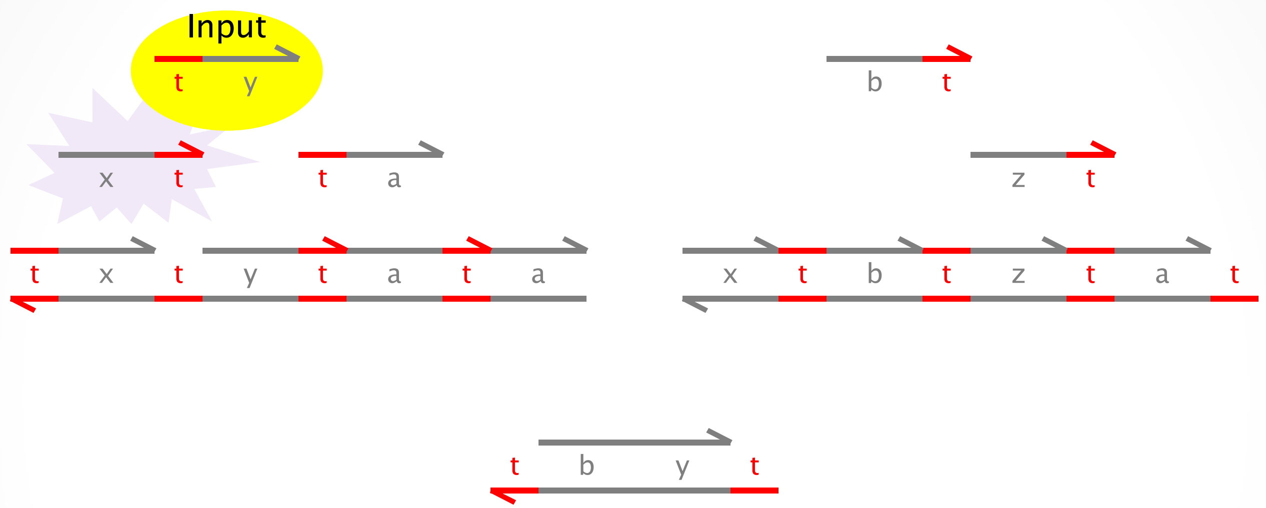
Join x+y→z



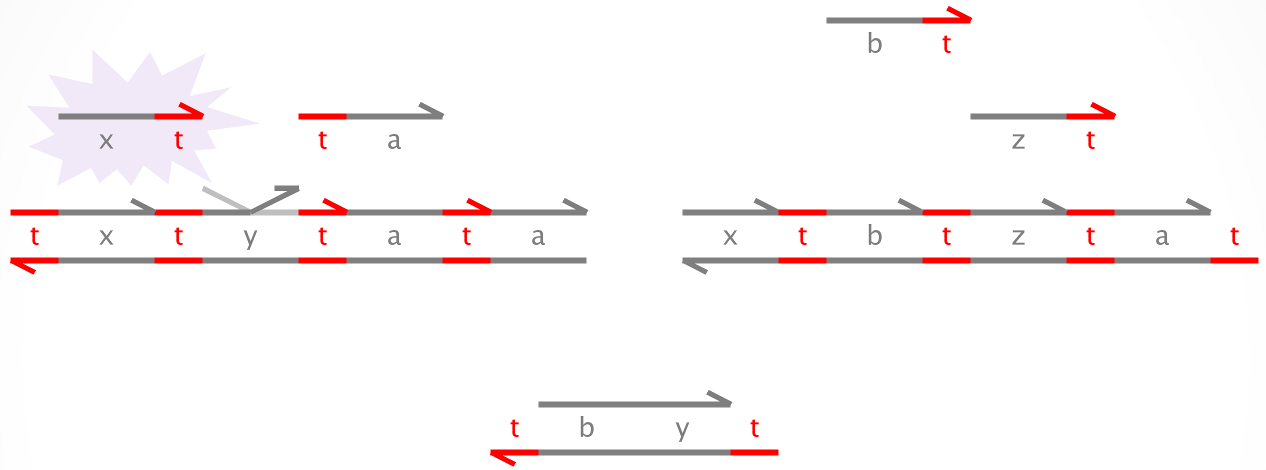
Join x+y→z



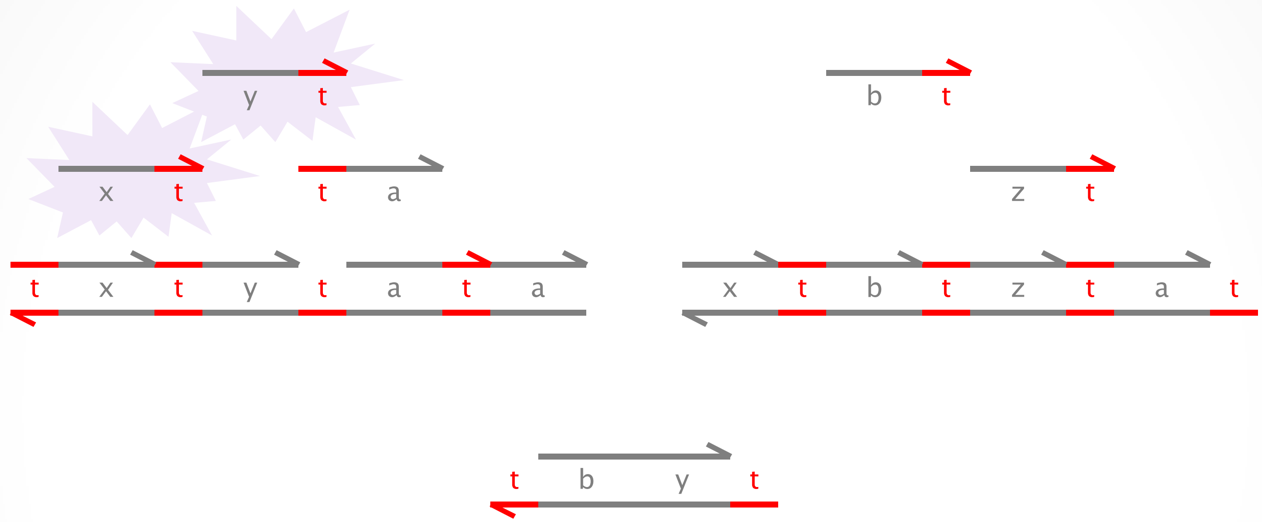
Join x+y→z



Join x+y→z



Join $x+y\rightarrow z$



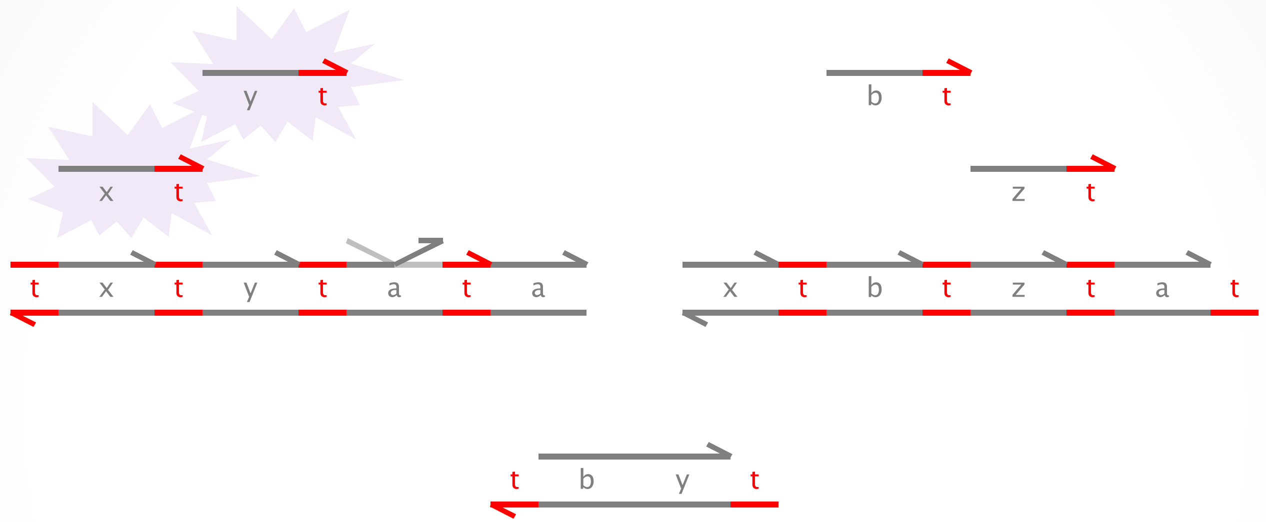
We cannot have a collector just waiting for **yt**, because there may be innocent **yt** elsewhere in the system, like here!

Transducer $x\rightarrow y$

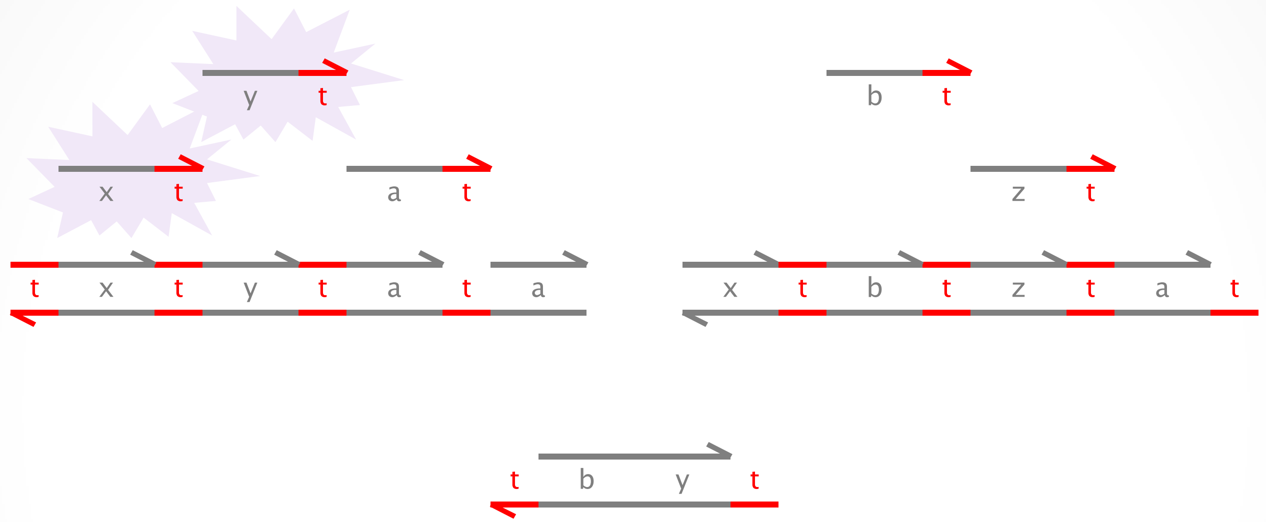
Instead, the collection of **yt** must be triggered only by a signal signifying that an $x+y\rightarrow z$ gate has fired. That signal is **tb**, which will trigger the collection of **yt** after output **tz** is produced.

bt is a *private* signal
(a different 'b' for each xyz triple)

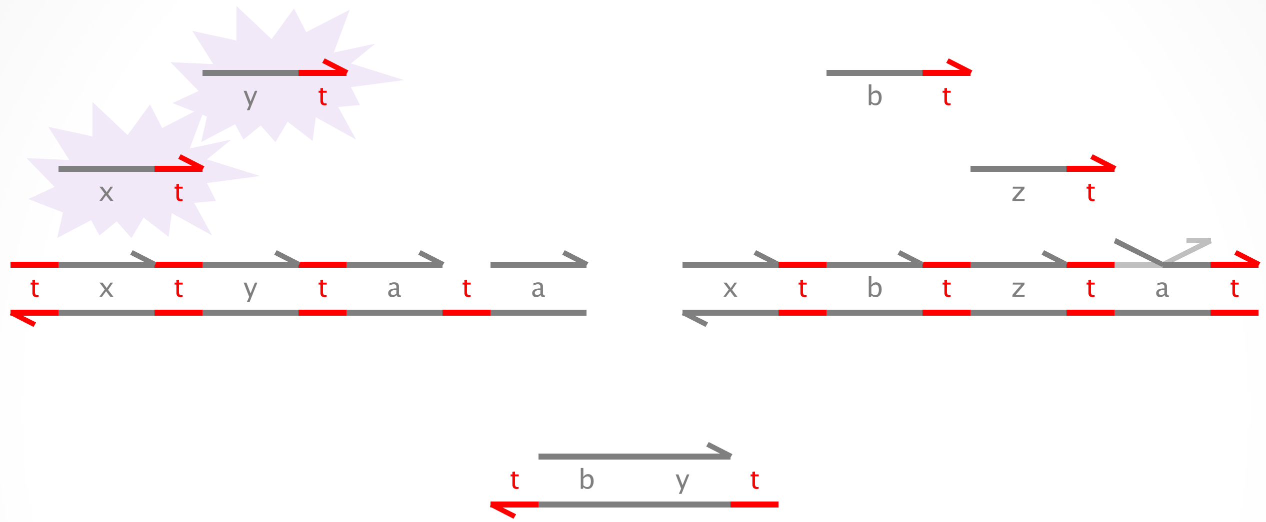
Join $x+y \rightarrow z$



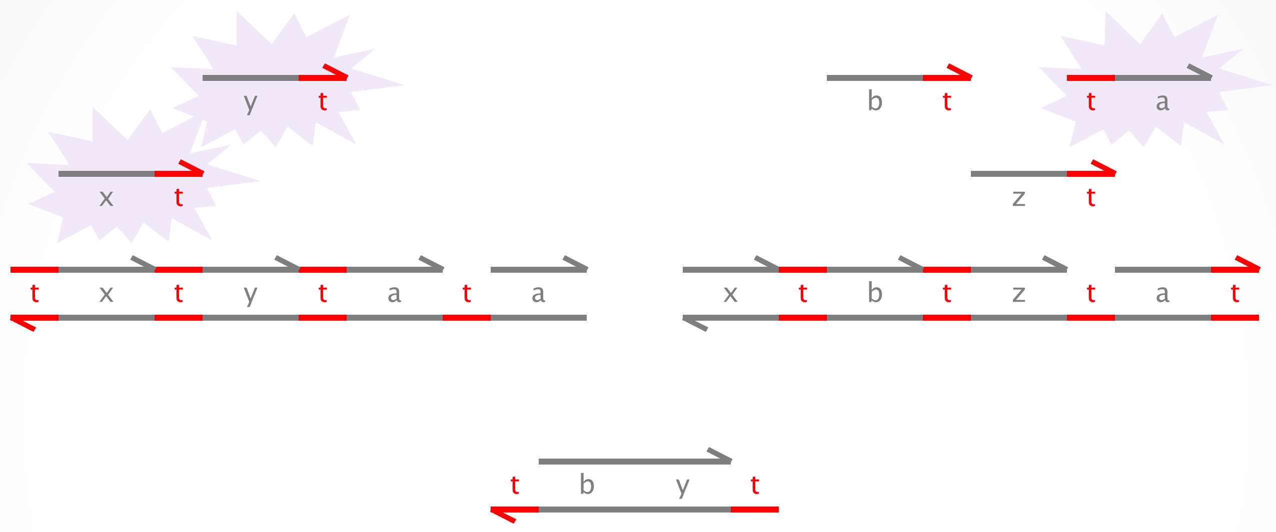
Join x+y→z



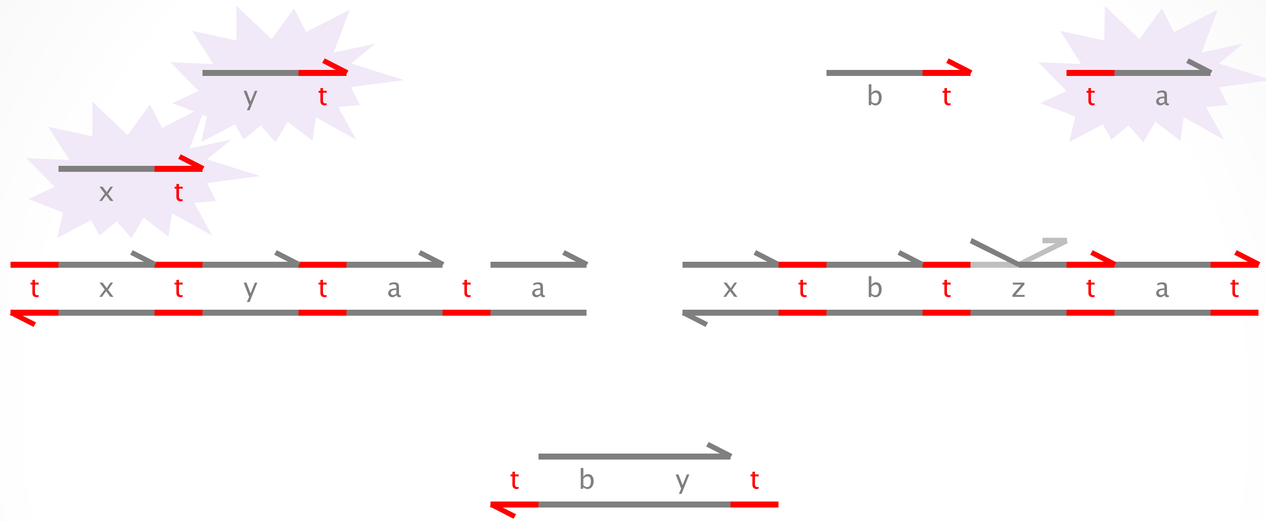
Join x+y→z



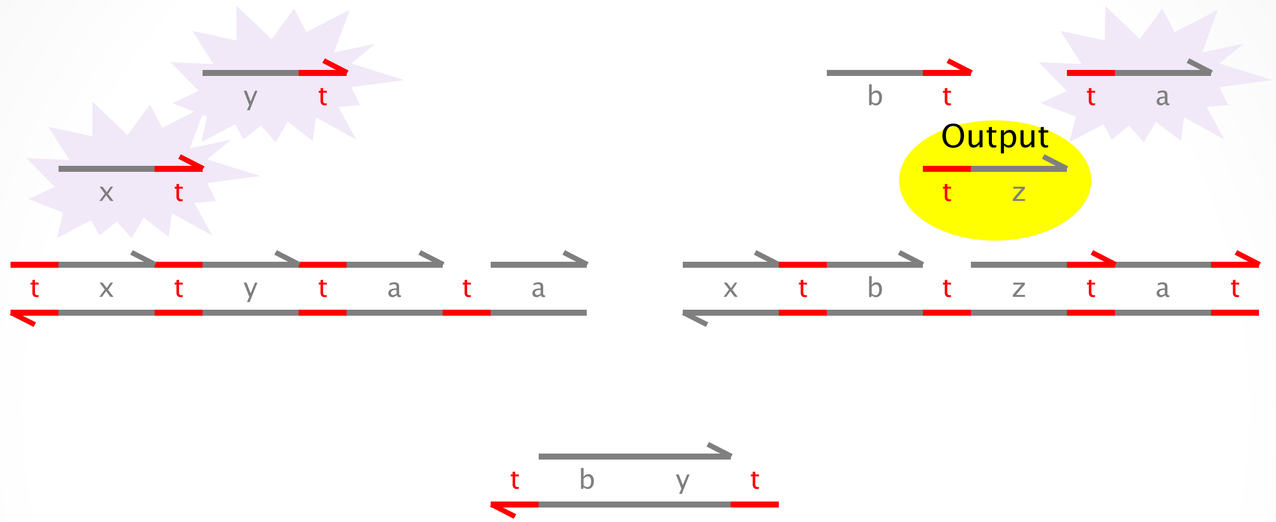
Join x+y→z



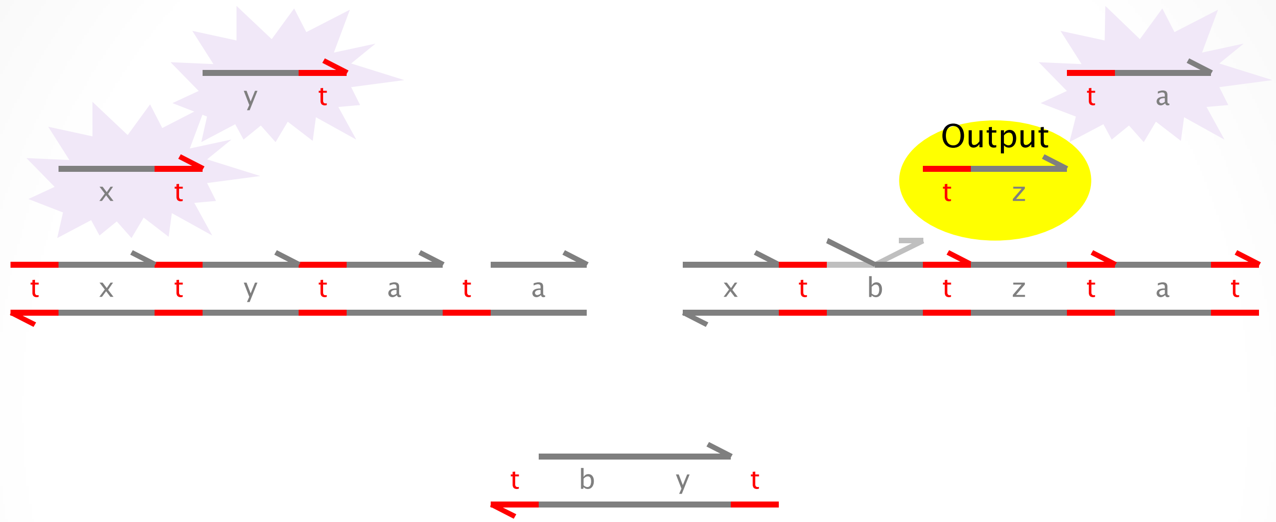
Join x+y→z



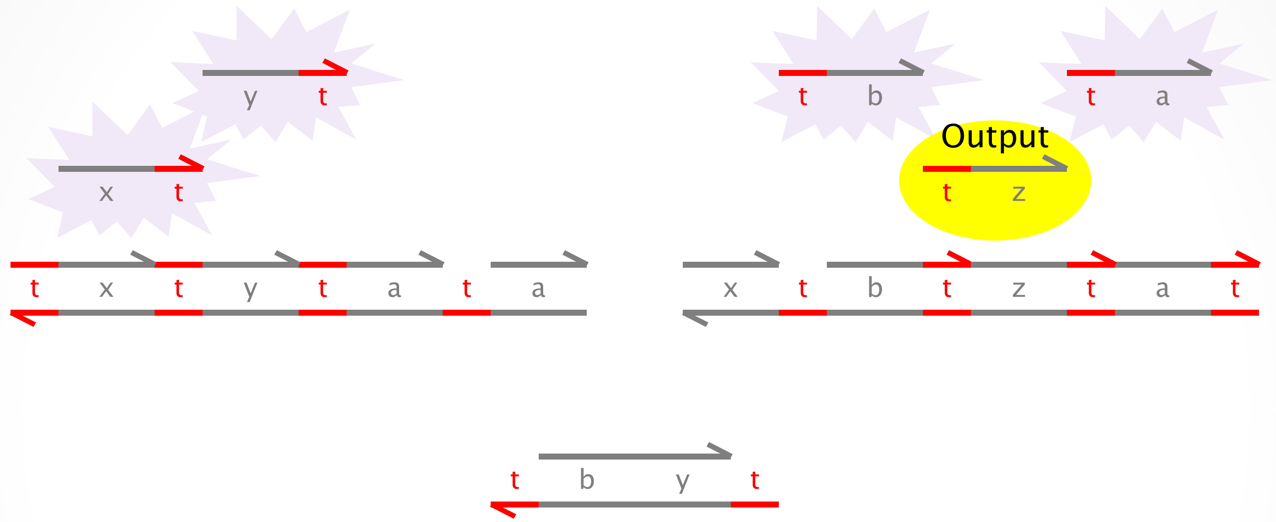
Join x+y→z



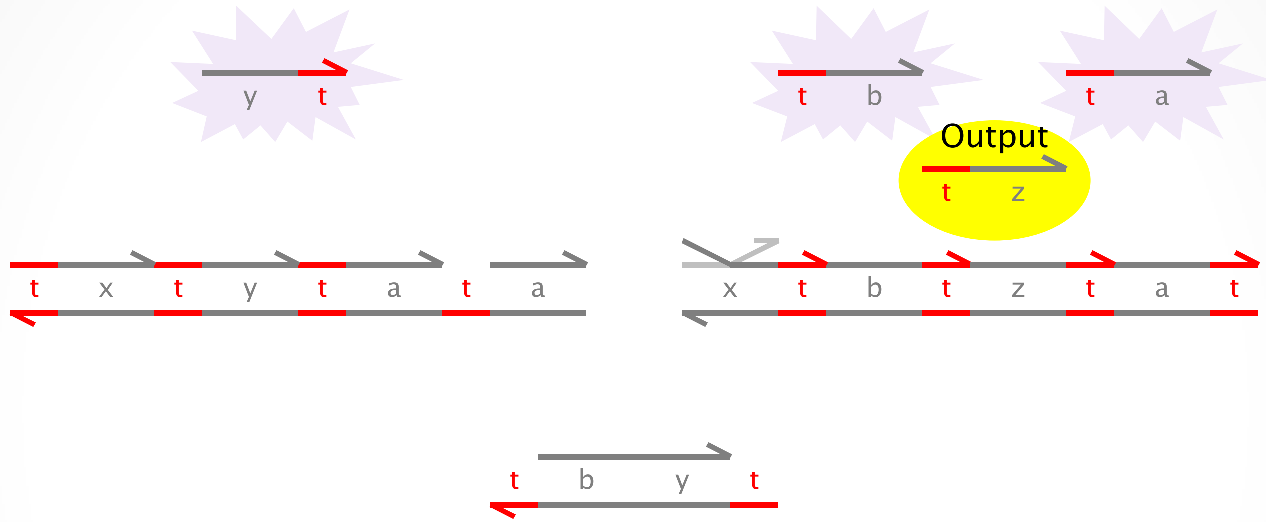
Join $x+y \rightarrow z$



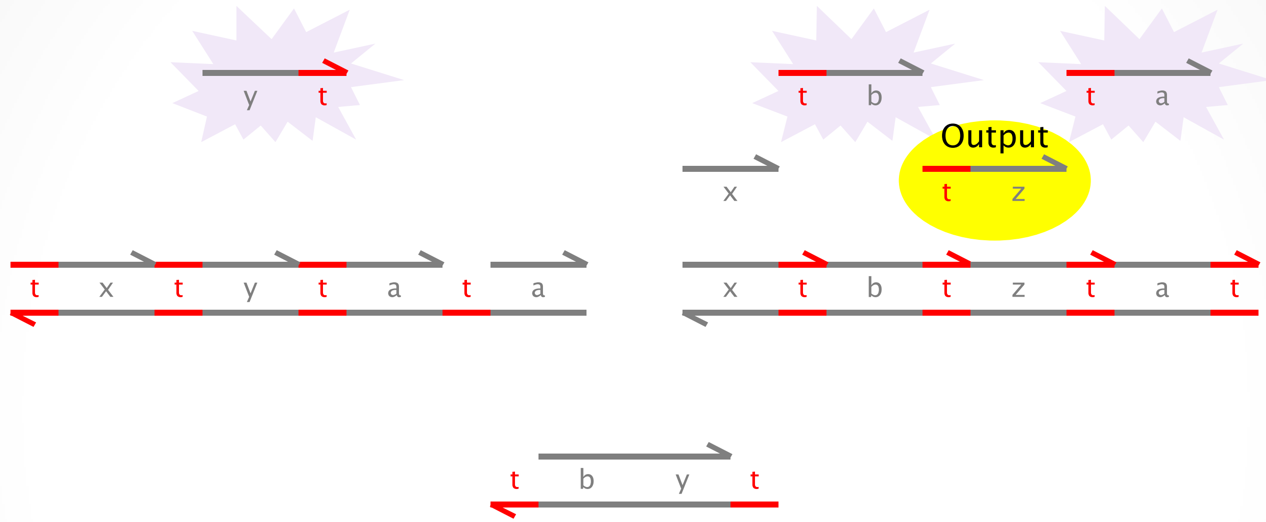
Join x+y→z



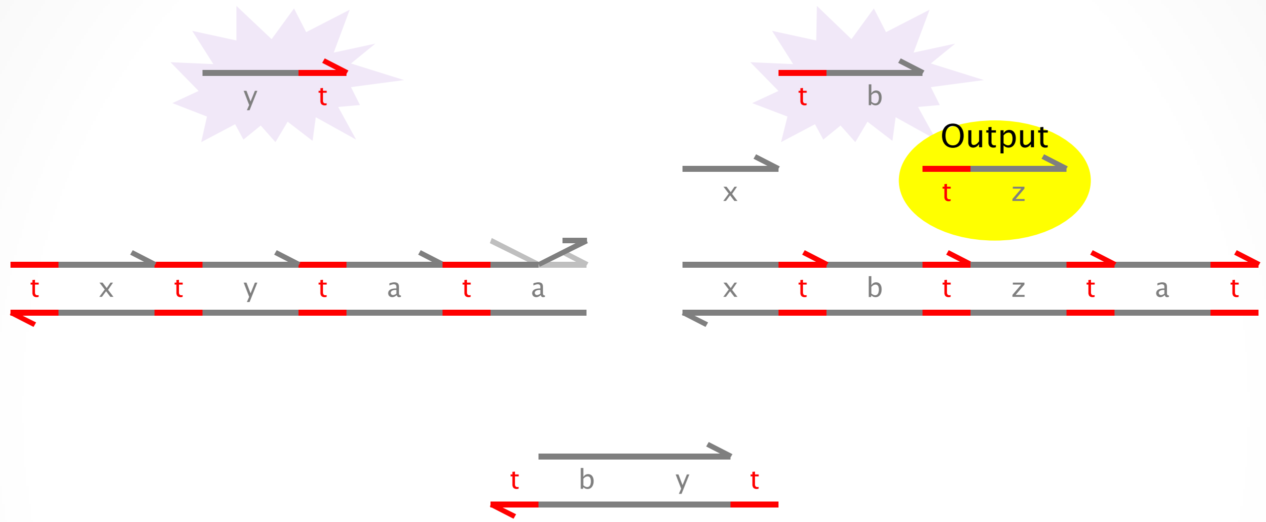
Join x+y→z



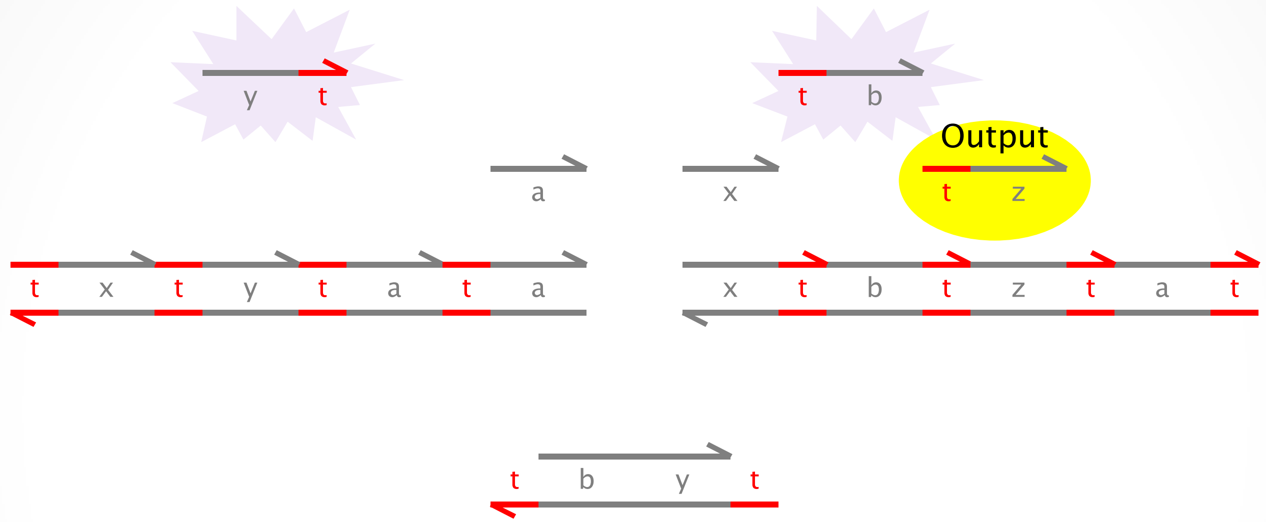
Join $x+y\to z$



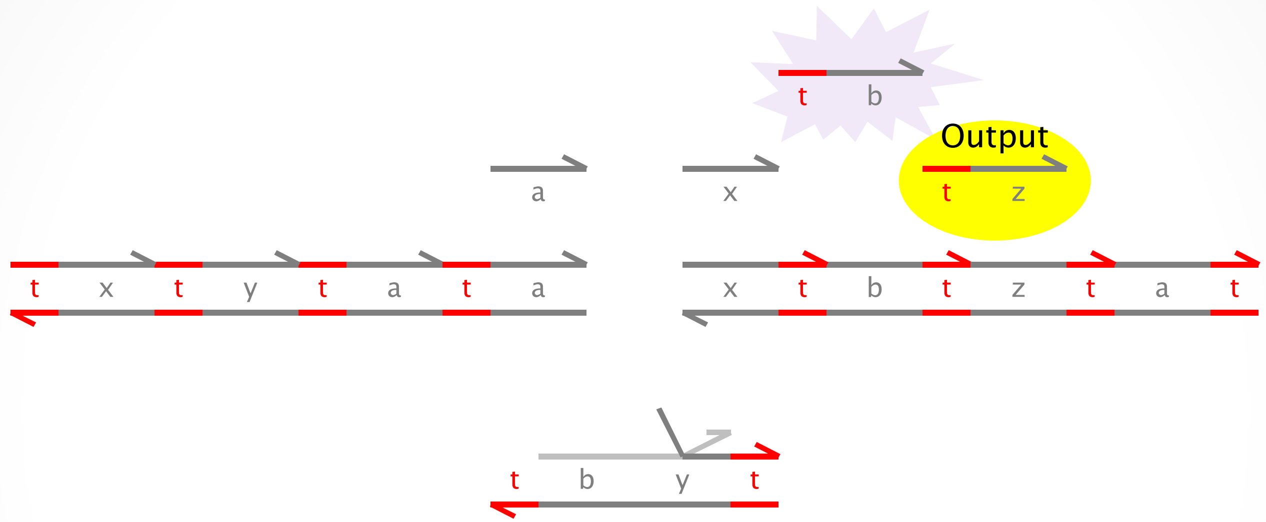
Join x+y→z



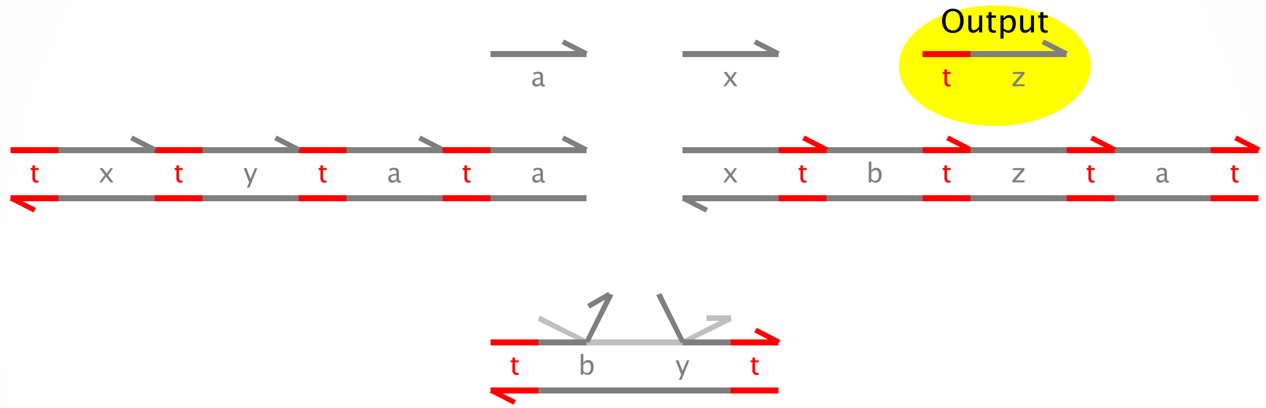
Join x+y→z



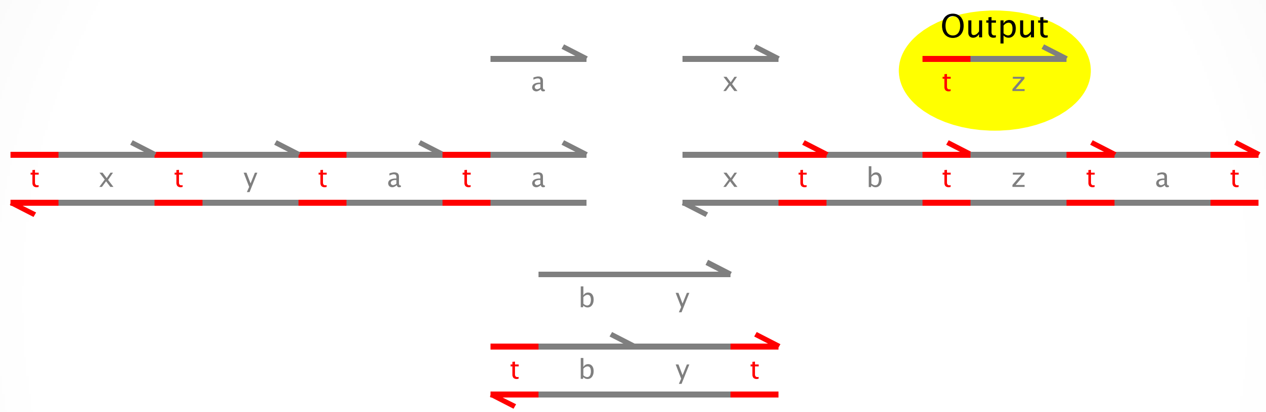
Join x+y→z



Join $x+y \rightarrow z$



Join $x+y \rightarrow z$



General n×m Join–Fork

- Easily generalized to 3+ inputs (with 2+ collectors) etc.
- Easily generalized to 2+ outputs (like Fork) etc.

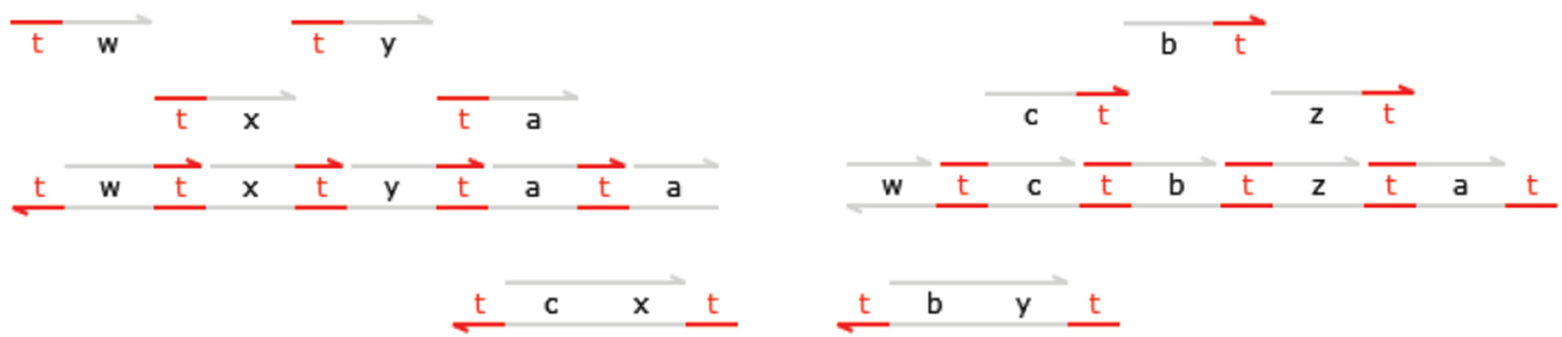
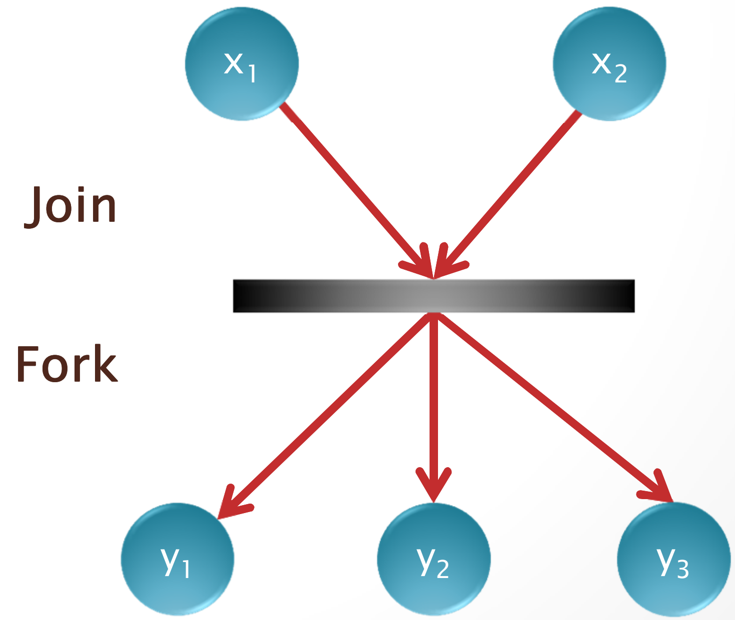


Figure 9: 3-Join $J_{wxyz} \mid tw \mid tx \mid ty \rightarrow tz$: initial state plus inputs tw, tx, ty.

Petri Net Transitions

- Computing power equivalent to Petri Nets (not Turing complete).
- Not completely trivial: gates are consumed by activation, hence a persistent Petri net transition requires a stable population of gates.



Strand Algebra

- An abstract description of signal-gate interactions:

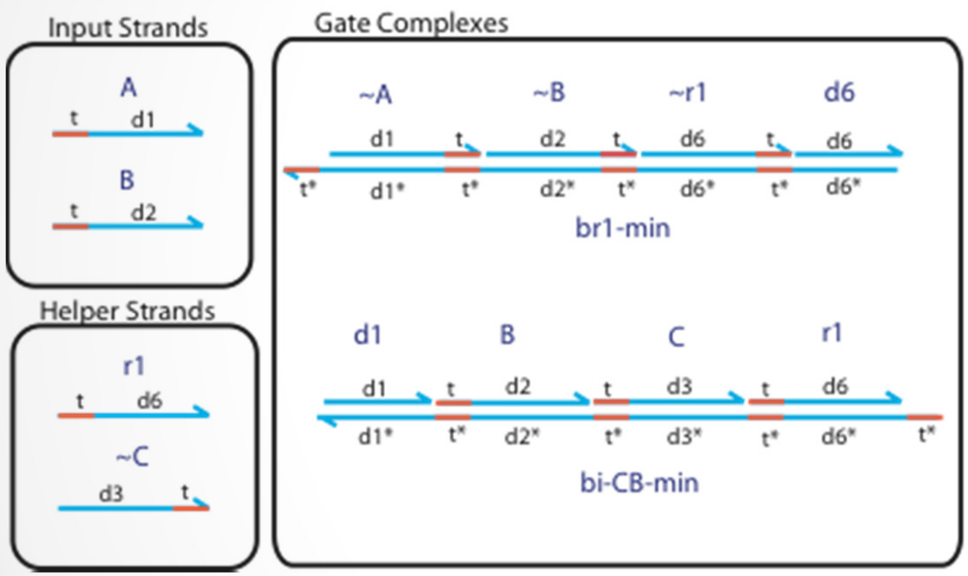
$$x_1 \mid .. \mid x_n \mid [x_1,..,x_n].[y_1,..,y_m] \rightarrow y_1 \mid .. \mid y_m$$

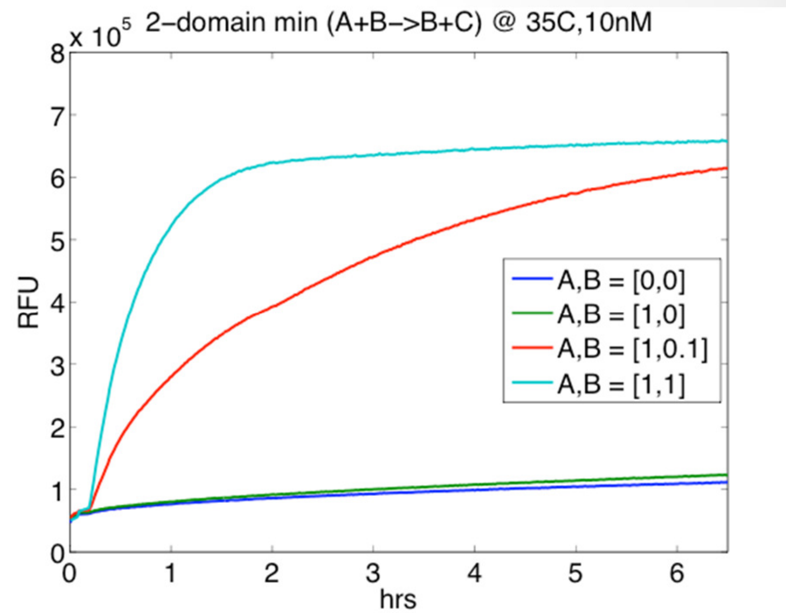
- Strand Algebra is an 'intermediate language'
  - Four-three-two domain gates implement Strand Algebra.
  - Strand Algebra implements Boolean circuits, Petri Nets, FSA, Linear I/O Systems, Interacting Automata, etc.

- Two-domain gates implement Strand Algebra
  - N.B. this is a *conjecture*.


Experiments
Postdoc
Faculty

A+B->B+C





Georg Seelig, Matt Olson

Enzymatic gate production

Make dsDNA from plasmids or using PCR, then digest with
Restriction enzymes and nicking enzymes

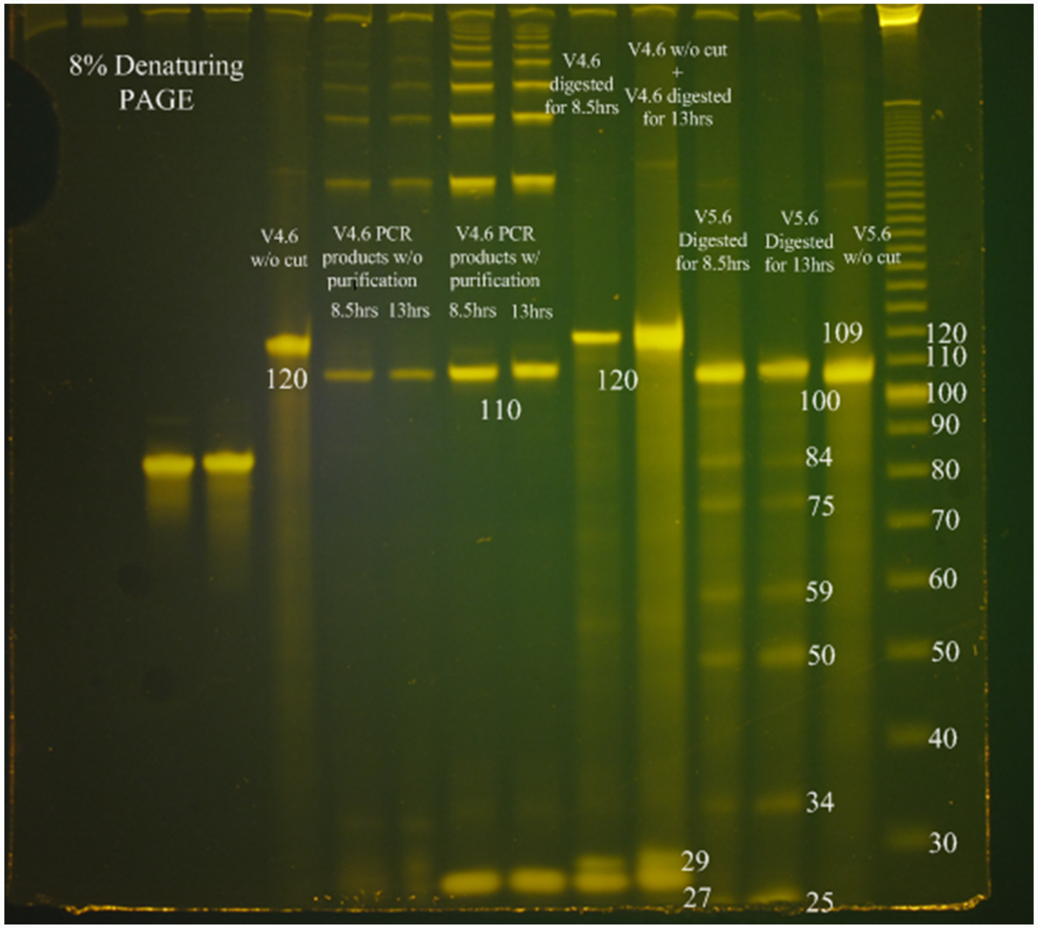
```
5'-NNCACGTCNNNNNNNNNGAGTCATTGCNNNNNNNNNNNNGAGTCATTGCNNNNNNNNNNNNGAGTCATTGCNNNNN
3'-NNGTGCAGNNNNNNNNNCTCAGTAACGNNNNNNNNNNNNCTGTGTAACGNNNNNNNNNNNNCTCAGTAACGNNNNN
```

Digest with nicking enzyme Nt.BstNB:

```
5'-NNCACGTCNNNNNNNNNGAGT CATTGCNNNNNNNNNNNNGAGT CATTGCNNNNNNNNNNNNGAGT CATTGCNNNNN
3'-NNGTGCAGNNNNNNNNNCTCA-GTAACGNNNNNNNNNNNNCTGT-GTAACGNNNNNNNNNNNNCTCA-GTAACGNNNNN
```

Georg Seelig, Yuan-Jyue Chen

Nicking Enzyme Digest



Georg Seelig, Yuan-Jyue Chen

Structural Invariants

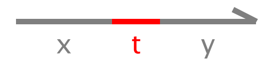
Double Strand Invariant

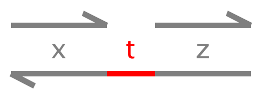
- Using top-nicked double strands *only*
  - The absence of any branching is inherently more trouble-free than branching structures that can tangle and interact in unexpected ways through their protruding single-stranded parts.
  - All double-stranded structures are quiescent (except for receptive toeholds on the bottom strand), eliminating the possibility that the gate themselves may polymerize, or may self-interact.
  - Gates can be produced by any available means of generating double-stranded DNA (e.g. biologically). Top-nicks can be added by restriction enzymes.
  - These structures have a simple syntactical representation and simple reduction rules, which simplify formal verification.
- A structural invariant
  - No double-stranded structure other than top-nicked double strands should exist *through computation*. (Except fleetingly during branch migration.)
  - This imposes restrictions on the allowable single strands.

Single Strand Restrictions

- Sequences of long domains 

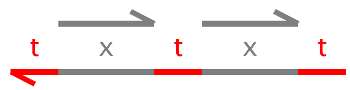
    - Are inert (because all double strands are fully complemented) and can be ignored/forbidden.

- (Sub-)sequences of the form: 

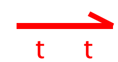
    - Can react with 

      violating the invariant. Forbidden.

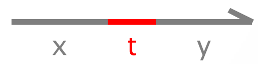
- (Sub-)sequences of the form: 

    - Can react with 

      violating the invariant. Forbidden.

- (Sub-)sequences of the form: 

    - Bind irreversible, hence 

      is as bad as 

      . Forbidden.

- (Sub-)sequences of the form: 

    - Can react with 

      violating the invariant. Forbidden.

Single Strand Invariant

- Hence we are left with: 

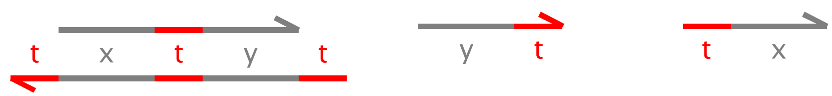
  - The two-domain signals!

- That is, the top-nicked DNA restriction forces the two-domain signal structure.

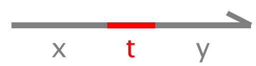
- Now, another structural invariant:
  - No single-stranded structures other than **xt, tx** should exist *through computation*. (Except for sequences of long strands, and single short strands.)
  - This imposes *new* restrictions on the allowable double strands.

Double Strand Restrictions

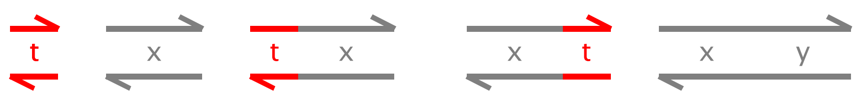
- Nicks must break the top strand into segments of two domains or less.
  - Otherwise,



releases the forbidden



- Hence, we are left with:
  - Double strands that are the bottom-strand concatenations of the double-stranded elements made of at most two domains:



Nick Algebra

• • •

Correctness

- Correctness issues
  - Some domains are supposed to be ‘private’ to some gates
  - Active residuals must be converted to proper waste
  - Interferences between copies of the same gate are possible
  - Interferences between copies of different gates are possible

- How to check correctness?
  - Other than by simulation?

- The spec of a transducer: $T_{xy}+tx \rightarrow ty$
  - Is that true at all?
  - Is that true *possibly* or *necessarily*?
  - Is that true *in all possible contexts*?
  - How do we check these properties?

Nick Algebra

S	$::= \; t.x \;\vdots\; x.t$	single strand
$\underline{D}$	$::= \; ø \;\vdots\; \underline{t} \;\vdots\; \underline{x} \;\vdots\; \underline{t.x} \;\vdots\; \underline{x.t} \;\vdots\; \underline{x.x} \;\vdots\; \underline{D^{\dagger}D}$	double strand
U	$::= \; S \;\vdots\; \underline{D} \;\vdots\; U\vert U \;\vdots\; (\nu x)U$	soup

Algebraic Equality

= is an equivalence relation,
and a congruence over the term syntax

$\underline{D_1}^{\ddagger}(\underline{D_2}^{\dagger}\underline{D_3}) \;=\; (\underline{D_1}^{\dagger}\underline{D_2})^{\dagger}\underline{D_3}$

$ø^{\dagger}\underline{D} \;=\; \underline{D}^{\dagger}ø \;=\; \underline{D}$

$U_1|(U_2|U_3) \;=\; (U_1|U_2)|U_3$

$U_1|U_2 \;=\; U_2|U_1$

$ø|U \;=\; U|ø \;=\; U$

$(\nu x)U \;=\; (\nu y)(U\{y/x\})$ if $y \notin \mathit{pd}(U)$

$(\nu x)ø \;=\; ø$

$(\nu x)(U_1|U_2) \;=\; U_1|(\nu x)U_2$ if $x \notin \mathit{pd}(U_1)$

$(\nu x)(\nu y)U \;=\; (\nu y)(\nu x)U$

Reduction

$\underline{D_1}\,\underline{{}^\dagger t^\dagger x t^\dagger D_2} \mid tx \quad \leftrightarrow \quad \underline{D_1}\,\underline{{}^\dagger t x^\dagger t^\dagger D_2} \mid xt$ exchange

$\underline{D_1}\,\underline{{}^\dagger t^\dagger x^\dagger D_2} \mid tx \;\rightarrow\; \underline{D_1}\,\underline{{}^\dagger t x^\dagger D_2}$ left coverage

$\underline{D_1}\,\underline{{}^\dagger x^\dagger t^\dagger D_2} \mid xt \;\rightarrow\; \underline{D_1}\,\underline{{}^\dagger x t^\dagger D_2}$ right coverage

$\underline{D_1}\,\underline{{}^\dagger t^\dagger x y^\dagger t^\dagger D_2} \mid tx \mid yt \;\rightarrow\; \underline{D_1}\,\underline{{}^\dagger t x^\dagger y t^\dagger D_2}$ cooperation

$\underline{D} \;\rightarrow\; ø$ if $\underline{D}$ not reactive waste

$U_1 \;\rightarrow\; U_2 \quad\Rightarrow\quad U_1 \mid U \;\rightarrow\; U_2 \mid U$ dilution

$U_1 \;\rightarrow\; U_2 \quad\Rightarrow\quad (\nu x)U_1 \;\rightarrow\; (\nu x)U_2$ isolation

$U_1 = U_2,\; U_2 \;\rightarrow\; U_3,\; U_3 = U_4 \quad\Rightarrow\quad U_1 \;\rightarrow\; U_4$ mixing

Reachability

- $U_1 \rightarrow^* U_2$ iff $U_1 \rightarrow \ldots \rightarrow U_2$
  - That is, U_1 *may* reduce to U_2.

- $U_1 \rightarrow^\forall U_2$ iff $\forall U, U_1 \rightarrow^* U \Rightarrow U \rightarrow^* U_2$.
  - That is, U_1 *will* reduce to U_2. (It cannot avoid the possibility of reducing to U_2).

Correctness

- Proposition: Gate may-Correctness

  $T^n_{xy}|tx^n \rightarrow^* ty^n$

  $F^n_{xyz}|tx^n \rightarrow^* ty^n|tz^n$

  $J^n_{xyz}|tx^n|ty^n \rightarrow^* tz^n$

  - Easy induction.

- Proposition: T^1_{xy} Will-Correctness

  $T^1_{xy} \mid tx \rightarrow^{\forall} ty$

  - Exhaustive case analysis enumerating all states of the system.
  - Can be done by hand for T^1_{xy}, and maybe T^2_{xy}, but not really for T^3_{xy} etc.
  - Will-correctness for fork/join is harder.
  - Will-correctness for combinations of gates is harder.
  - We are using modelchecking to verify some of these properties.

Conclusions

- A new architecture for general DNA gates
  - Simple signals, simple gate structures.
  - Self-cleaning: no garbage left by operation (except inert).
  - Enabling new ways of assembling gates.
  - Some experimental evidence that it works.

- A correspondingly simple algebra
  - For verifying gate designs mechanically.
  - For studying expressiveness (does it *really* implement Petri nets?).