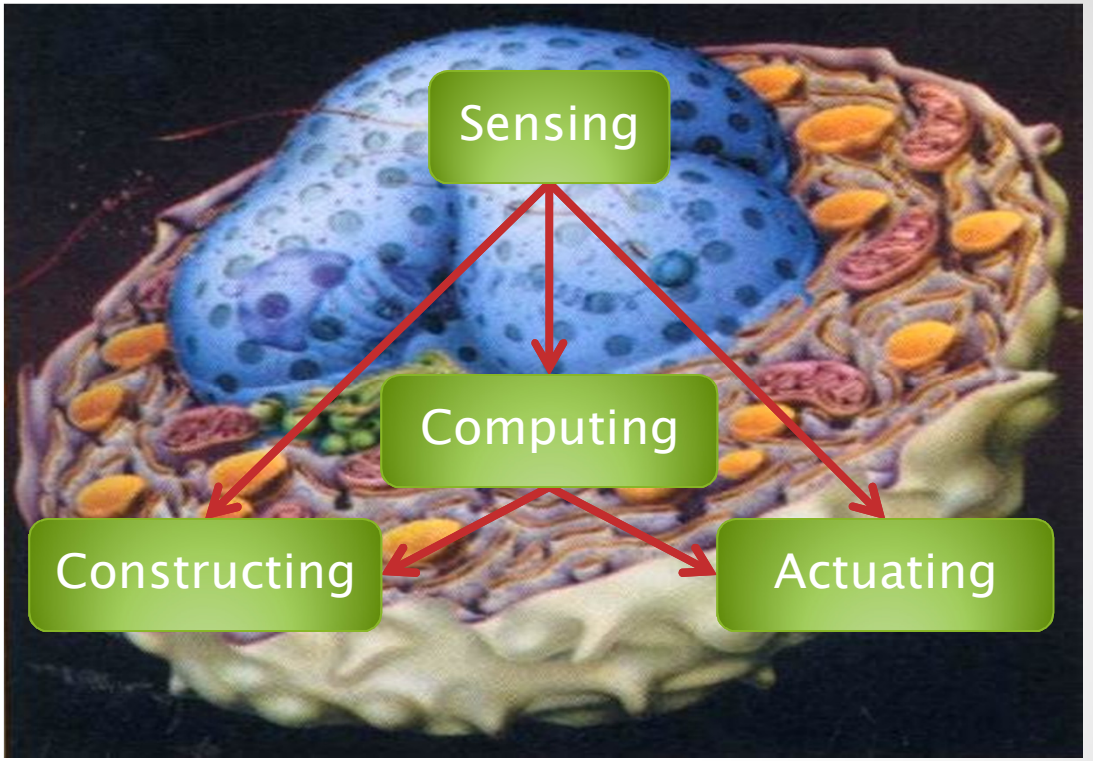
Two-Domain DNA Strand Displacement

Luca Cardelli
Microsoft Research

DCM Edinburgh, 2010-07-09
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

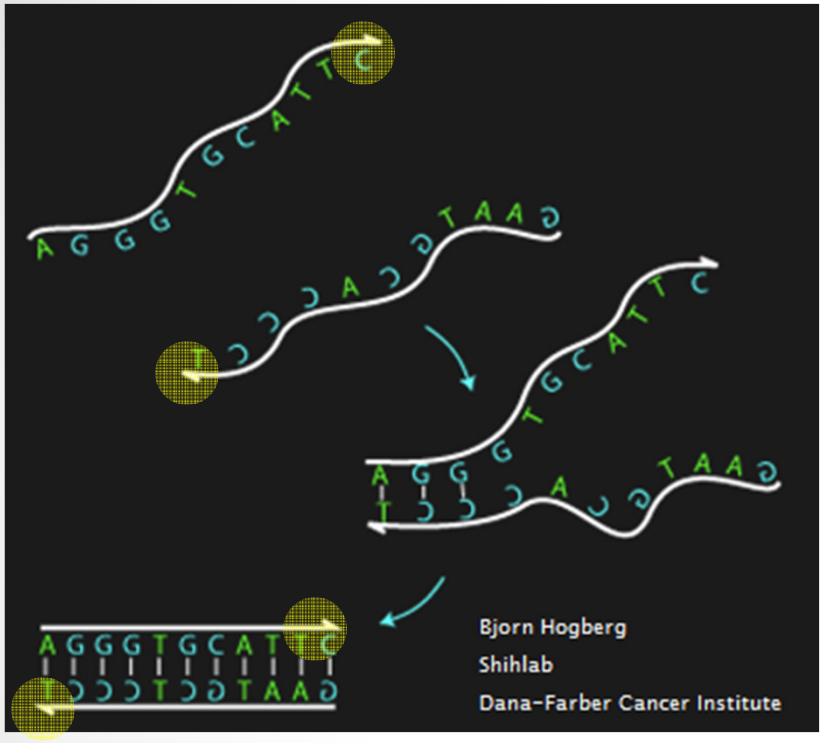


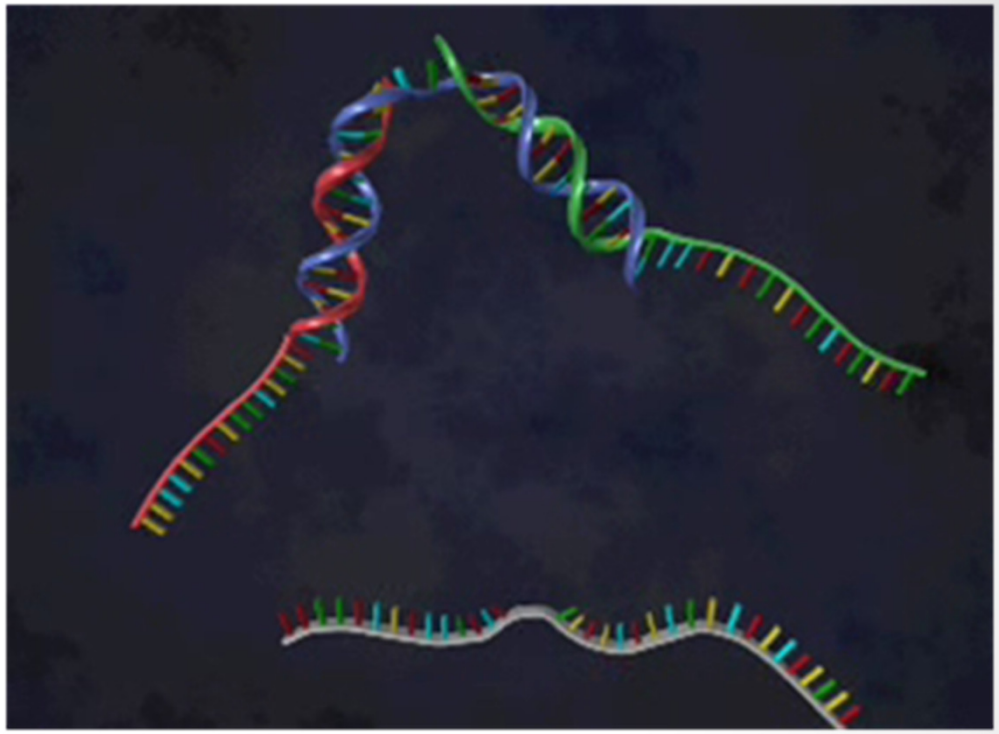
Nucleic Acids can do all this.

And interface to biology.
And are programmable.

Strand Displacement Basics

DNA 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.

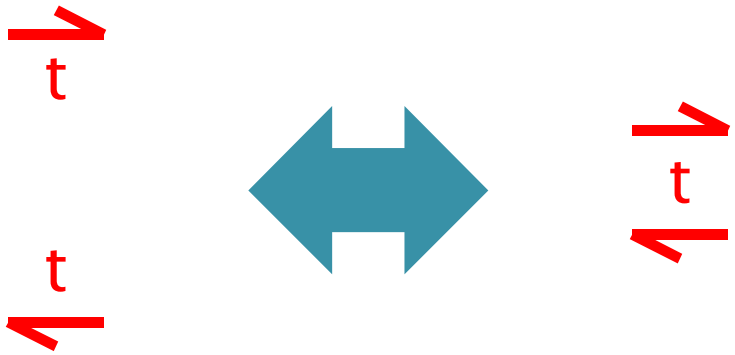
Domains

- 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.

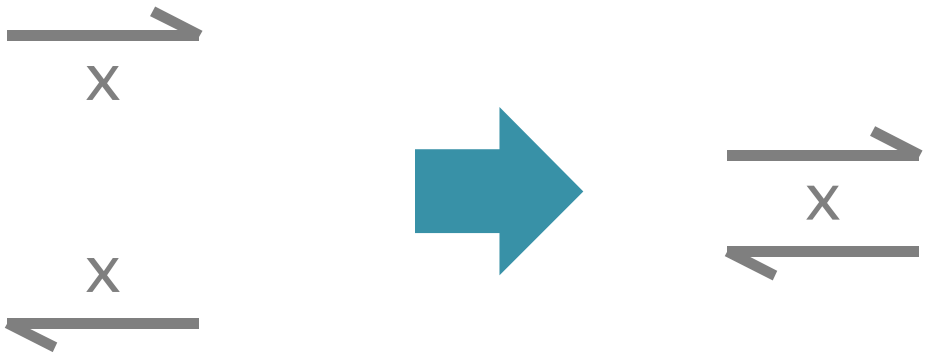
- Choosing domains (subsequences) that are suitably independent is a tricky issue that is still somewhat of an open problem (with a vast literature). But it can work in practice.

Short Domains



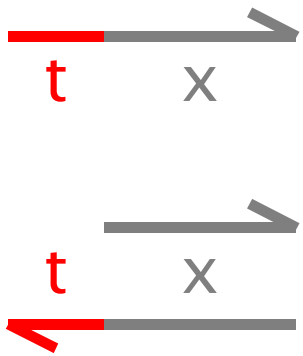
Reversible Hybridization

Long Domains



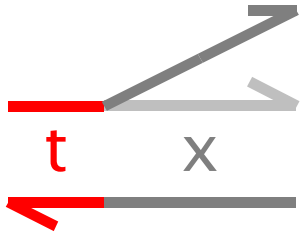
Irreversible Hybridization

Strand Displacement



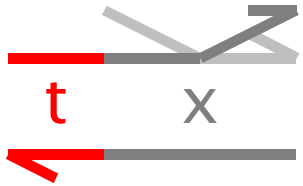
"Toehold Mediated"

Strand Displacement



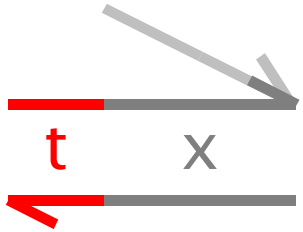
Toehold Binding

Strand Displacement



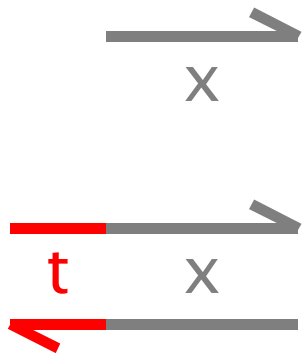
Branch Migration

Strand Displacement



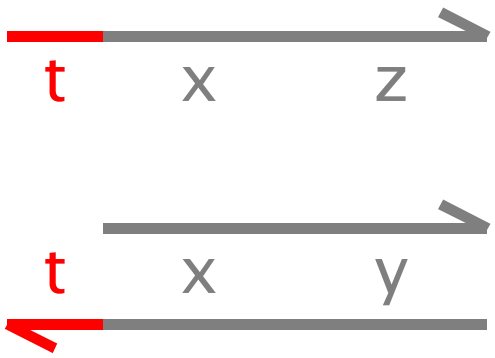
Displacement

Strand Displacement

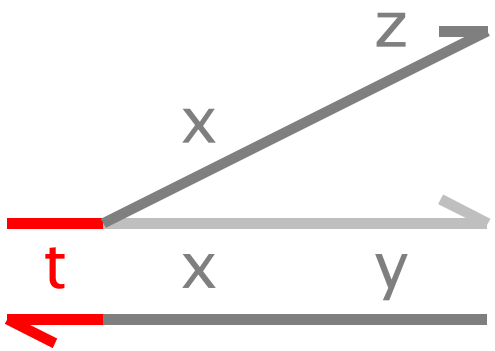


Irreversible release

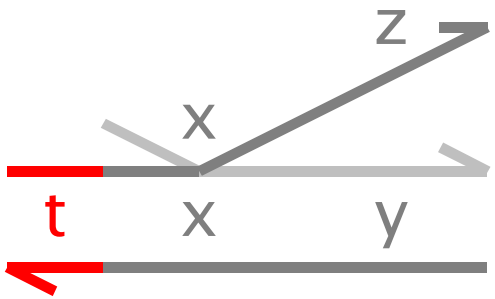
Bad Match



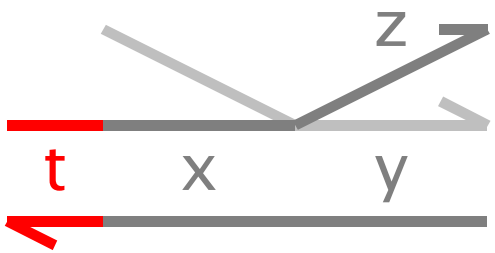
Bad Match



Bad Match



Bad Match

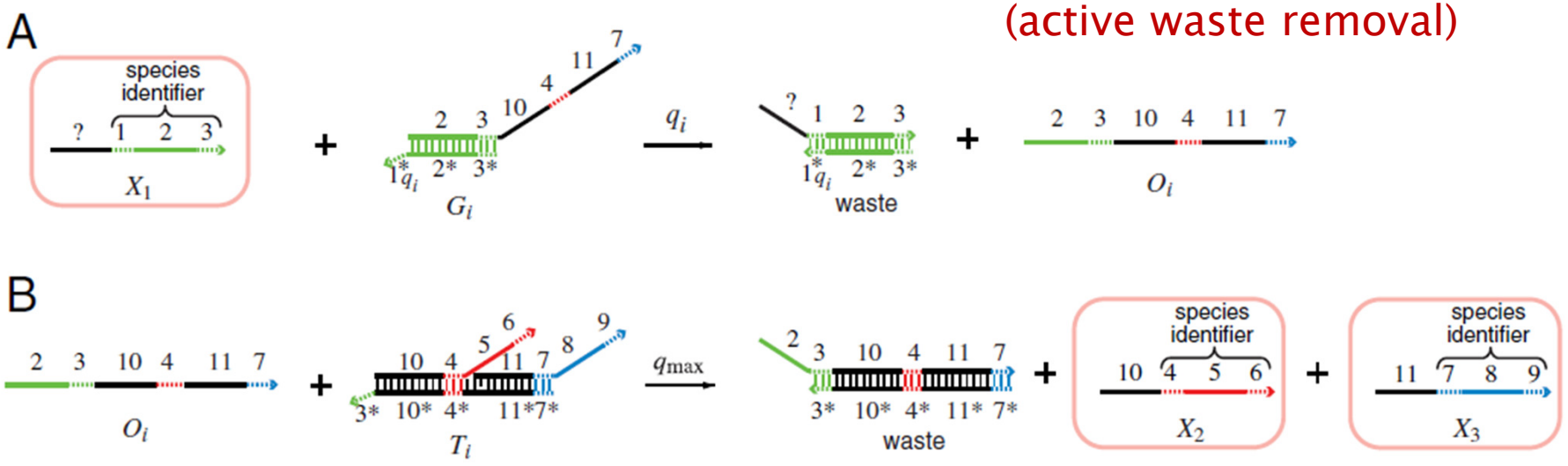


Cannot proceed
Hence will undo

Signals & Gates

Four-Domain Architecture

No “garbage collection”
(active waste removal)



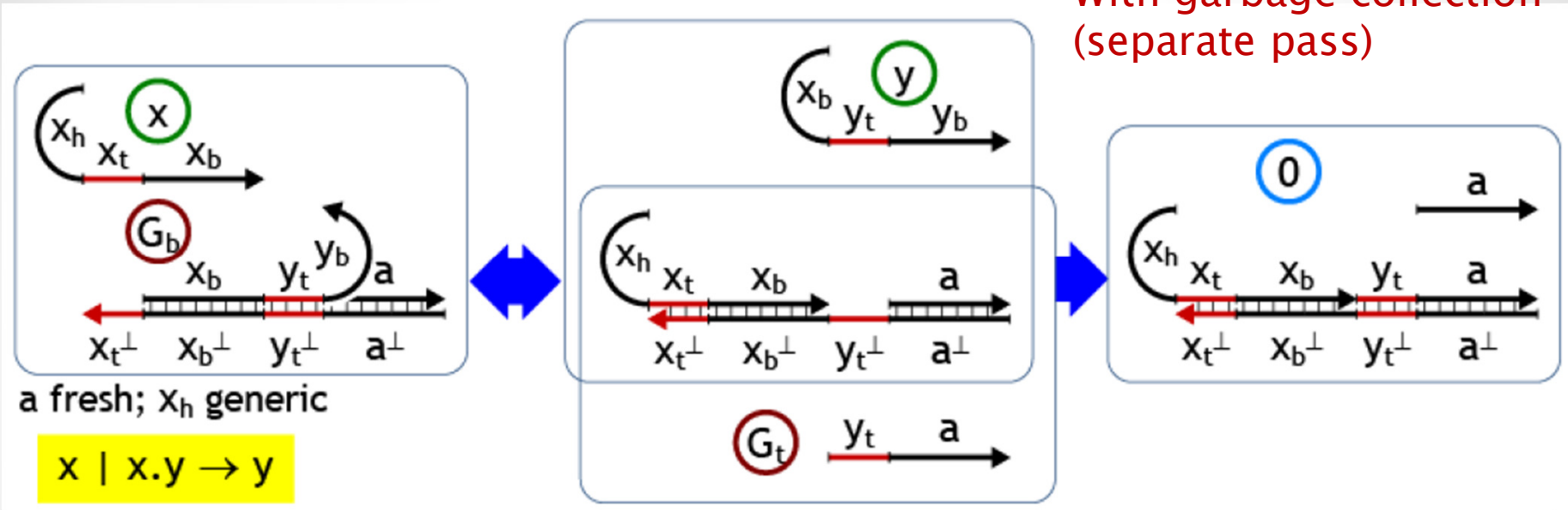
DNA as a universal substrate for chemical kinetics

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

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

Three-Domain Architecture

With garbage collection (separate pass)

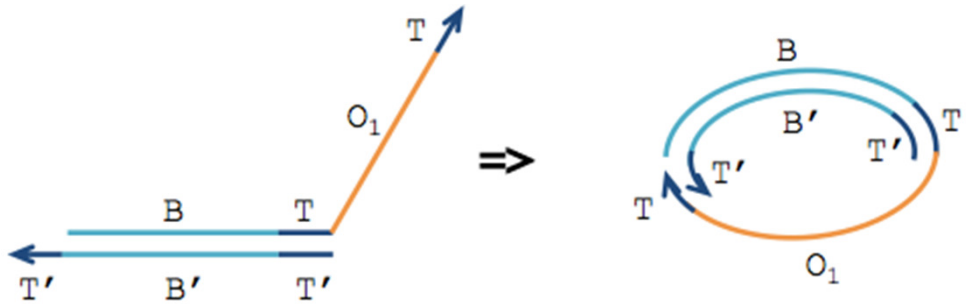


Strand Algebras for DNA Computing

Luca Cardelli

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

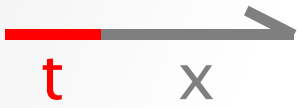
“Lulu’s Trouble”



(from D.Soloveichik)

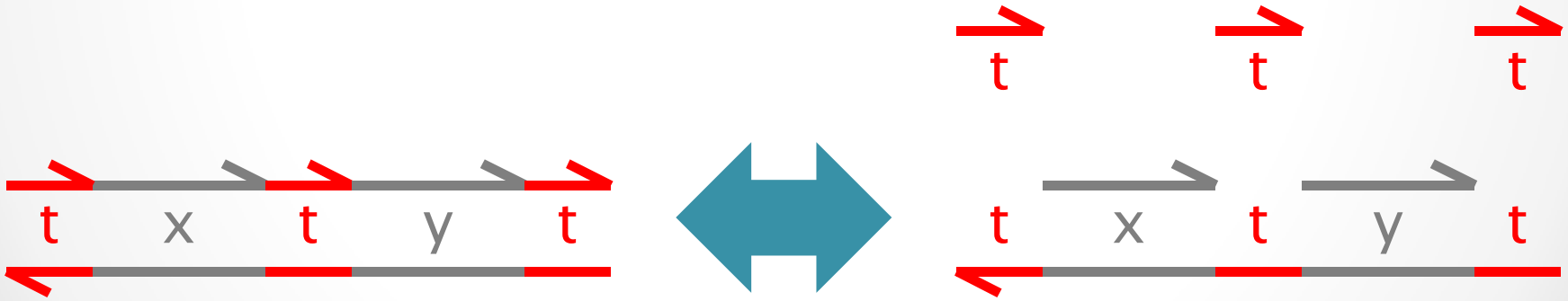
Two-Domain Architecture

- Signals: 1 toehold + 1 recognition region

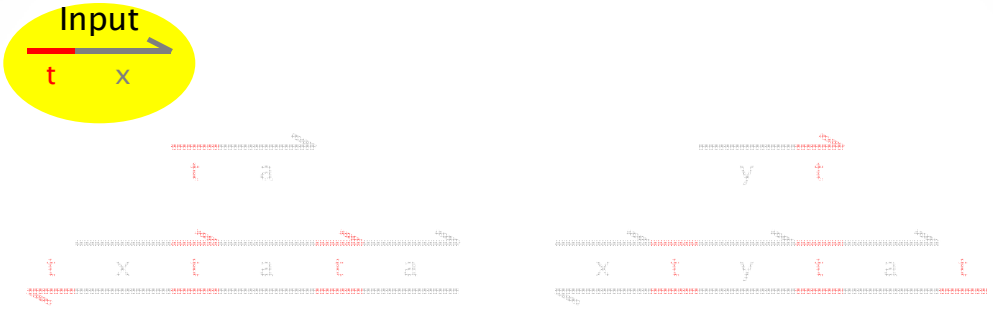


Garbage collection
"built into" the gates

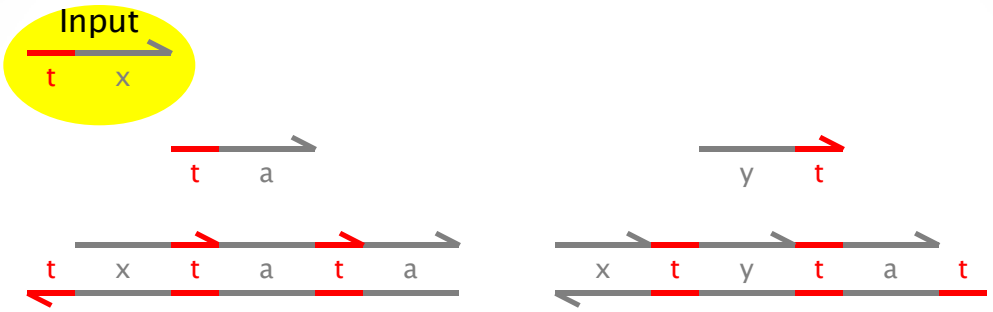
- Gates: "top-nicked double strands"
(or equivalently double strands with open toeholds)



Transducer x→y

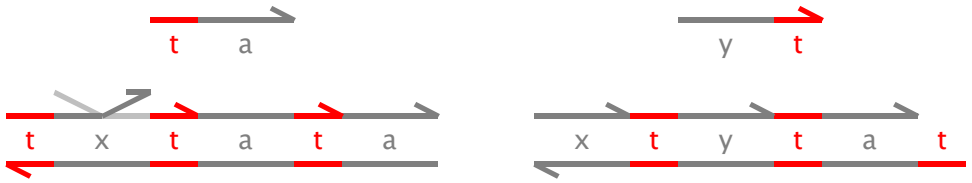


Transducer x→y



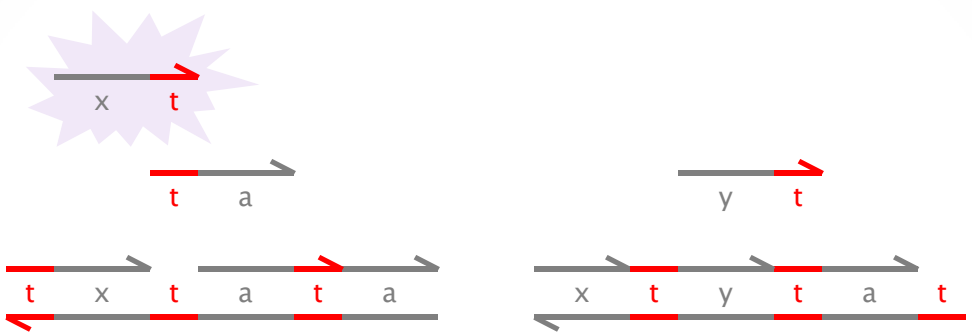
ta is a *private* signal (a different ‘a’ for each xy pair)

Transducer x→y

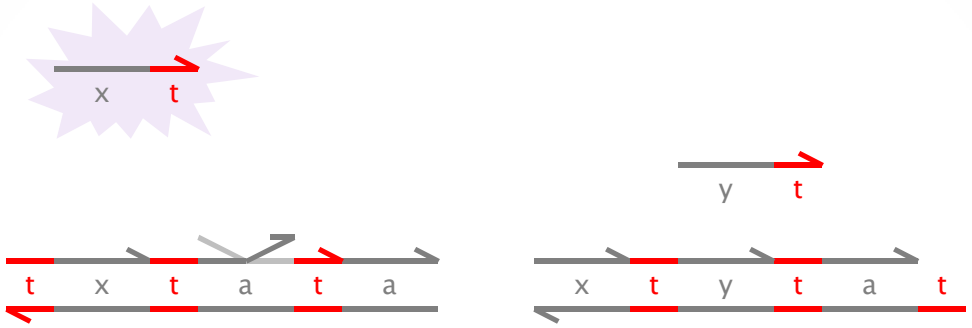


Transducer x→y

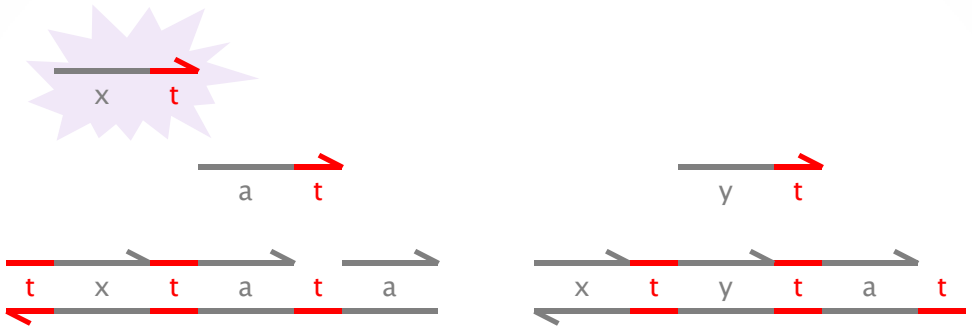
Active waste



Transducer x→y

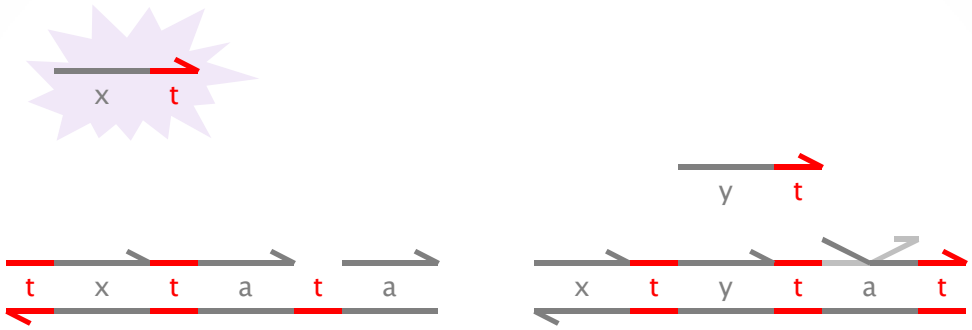


Transducer x→y

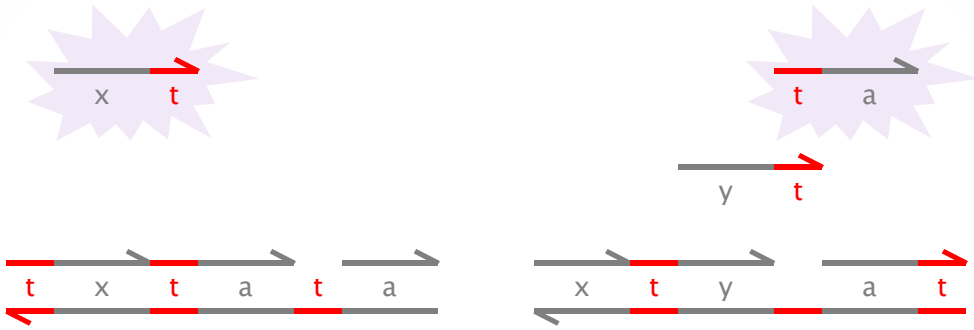


So far, a **tx** *signal* has produced an **at** *cosignal.*
But we want signals as output, not cosignals.

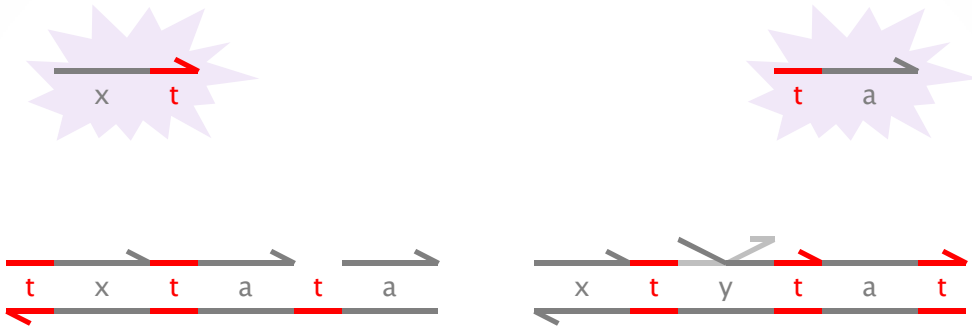
Transducer x→y



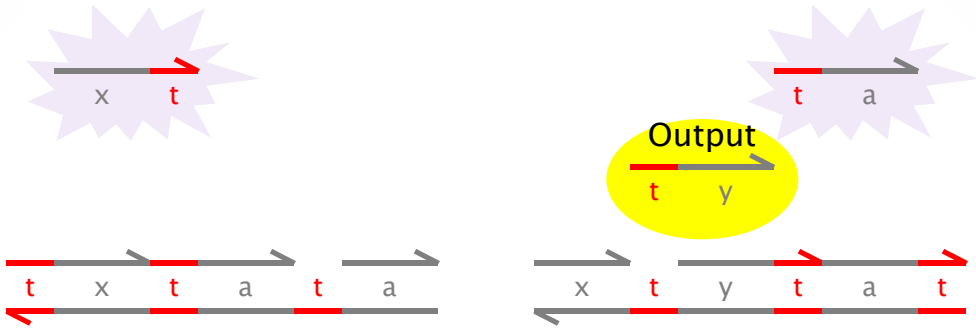
Transducer x→y



Transducer x→y



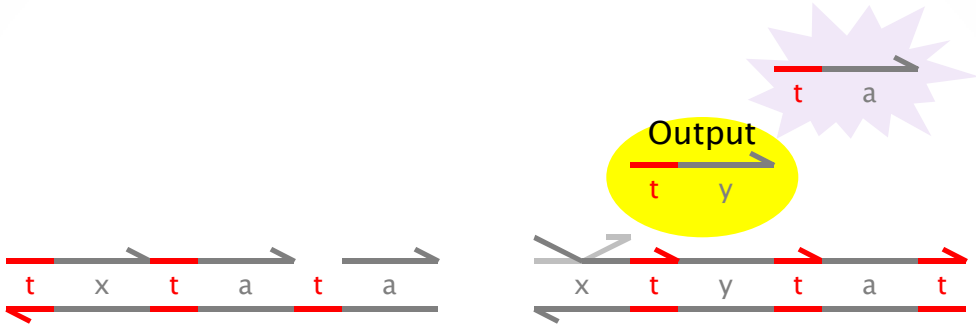
Transducer x→y



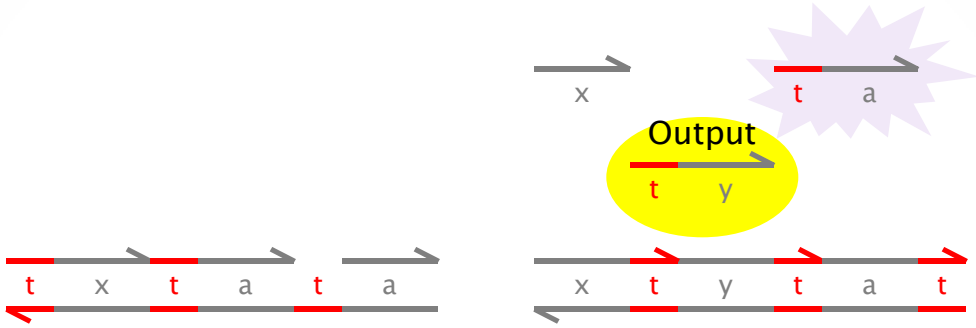
Here is our output **ty** *signal*.

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).

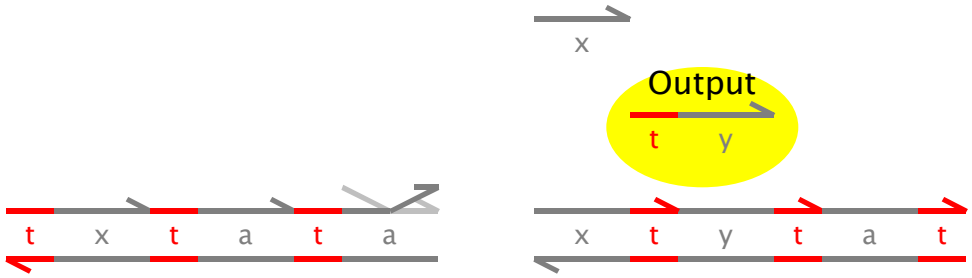
Transducer x→y



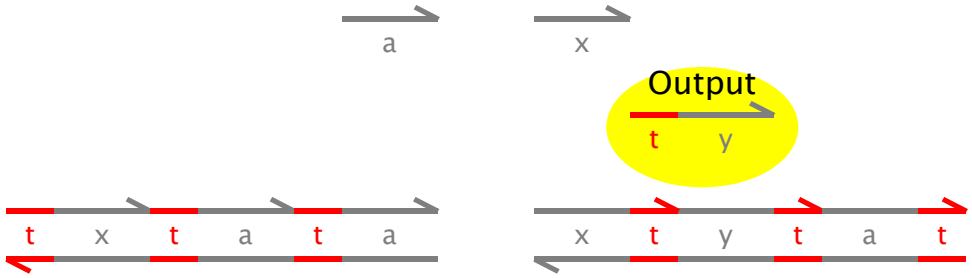
Transducer x→y



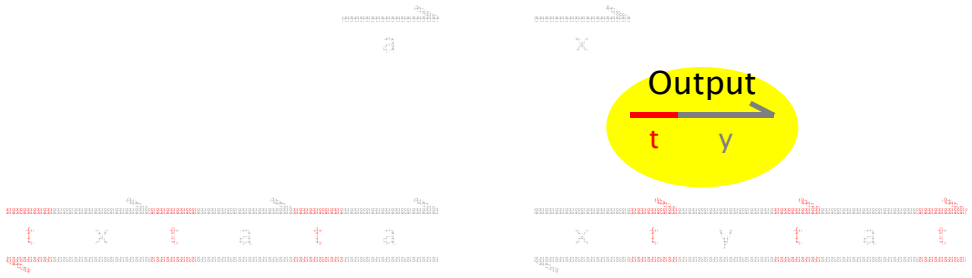
Transducer x→y



Transducer x→y



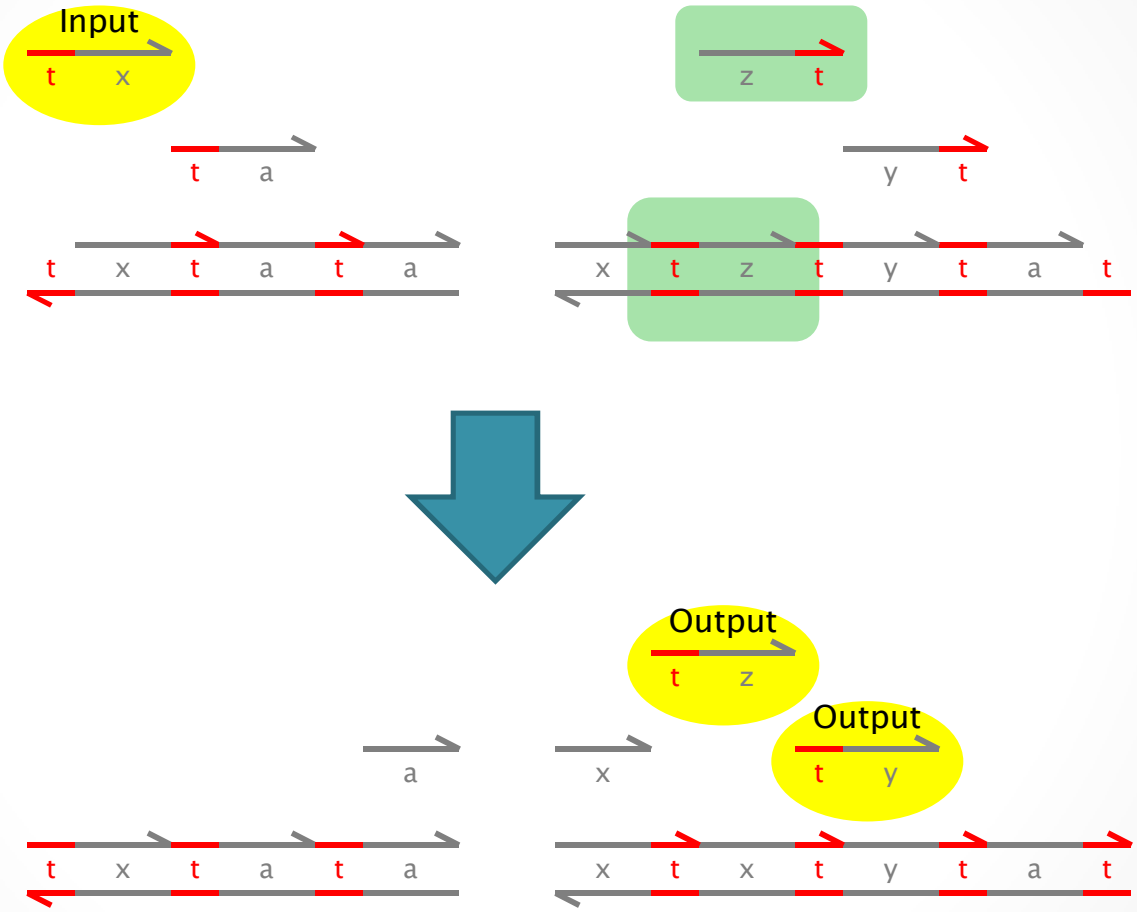
Transducer x→y



Done.

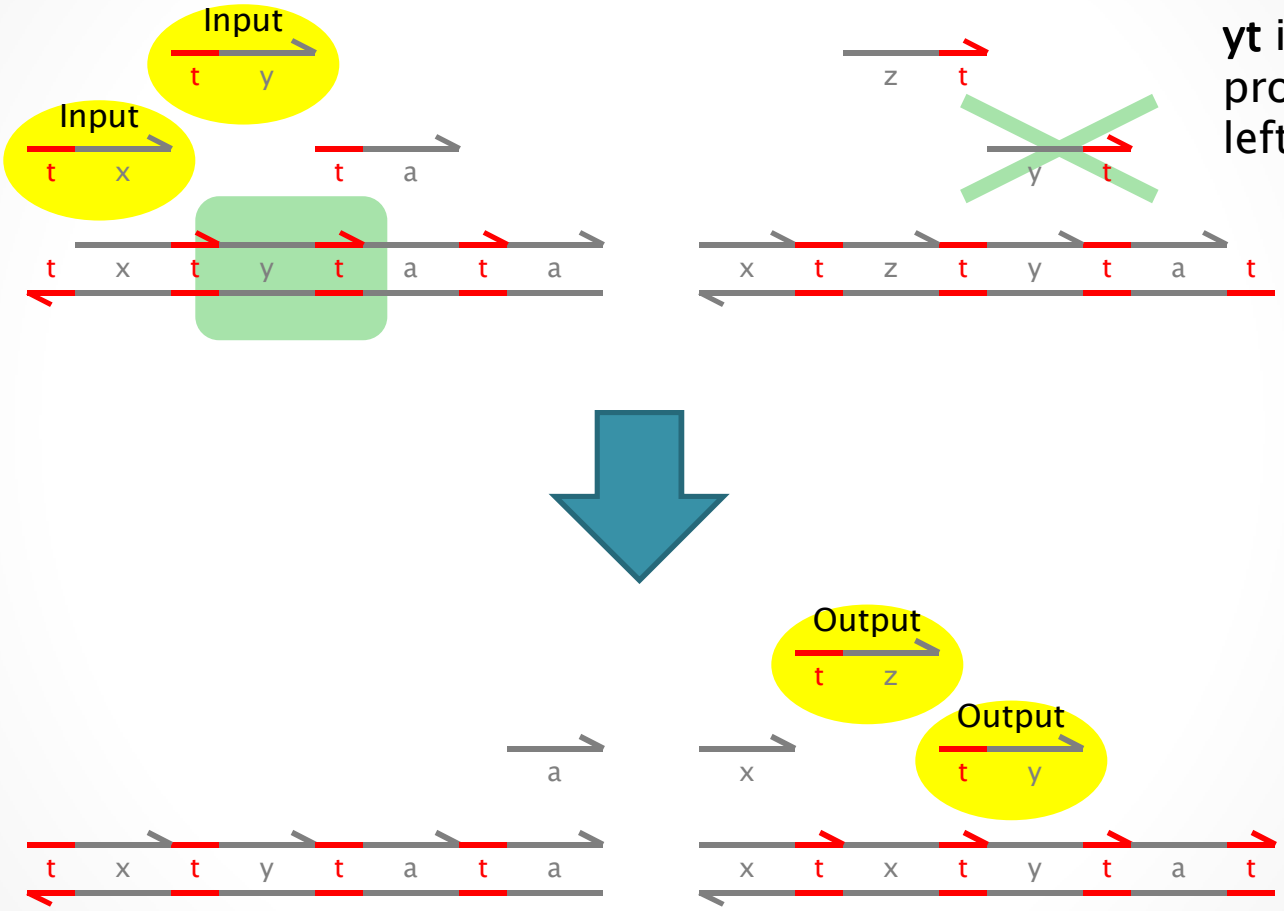
Note the **tata** motif and how it helps in collection.

Fork x→y+z



(Amplifier: x→x+x)

Catalyst x+y→y+z



yt is kindly provided by the left hand side.

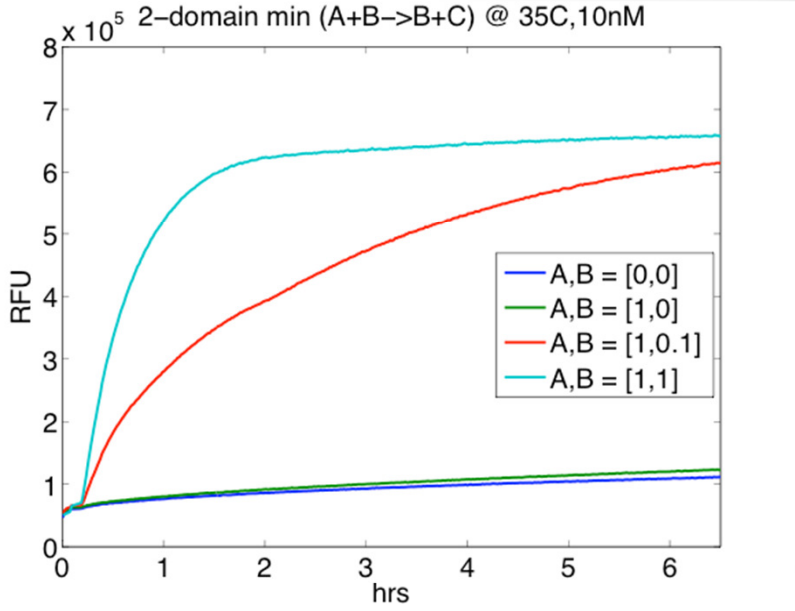
(Autocatalyst: x+y→y+y)

Experiments

Georg Seelig, Matt Olson

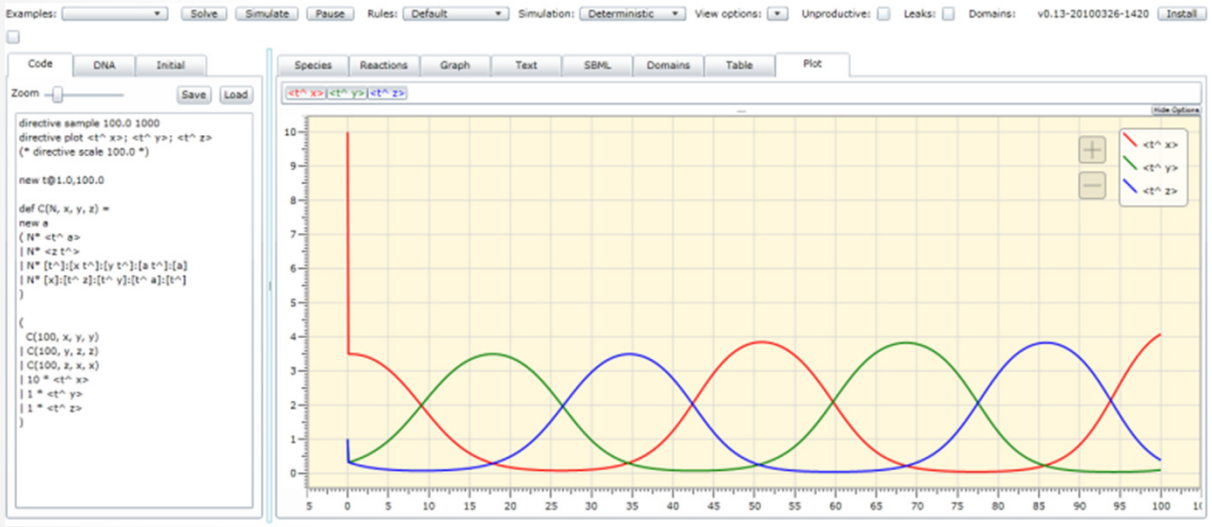
$A + B \rightarrow B + C$

This image cannot currently be displayed.



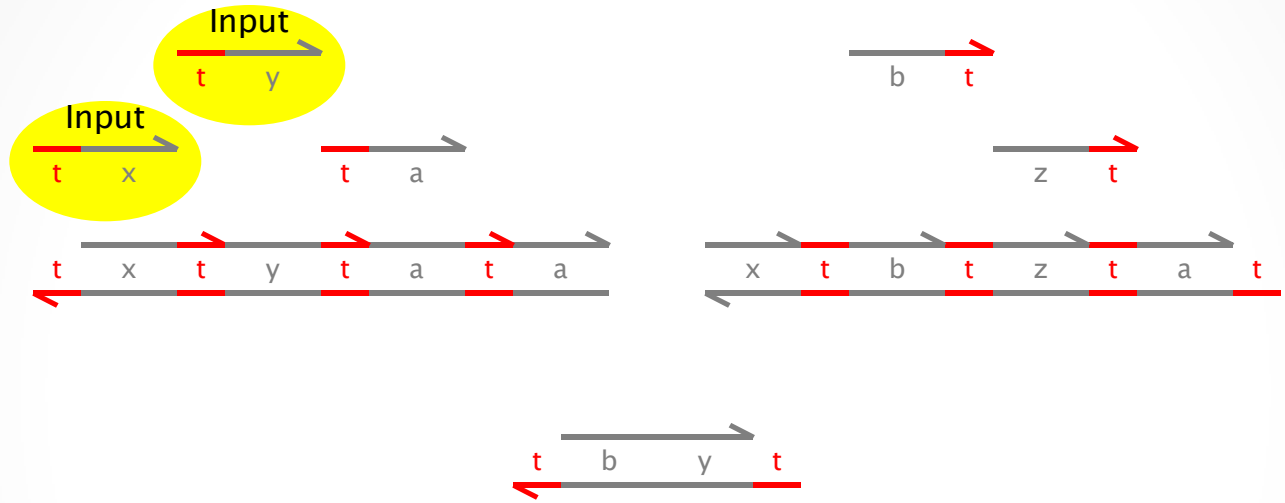
Autocatalytic Oscillator

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

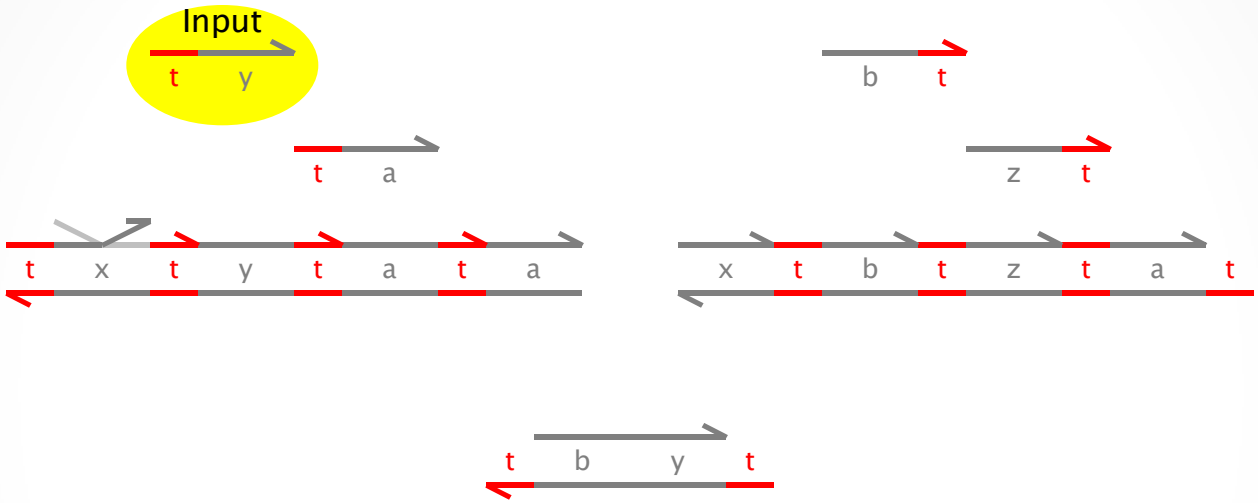



```
directive sample 100.0 1000
directive plot <t^ x>; <t^ y>;
<t^ z>
(* directive scale 100.0 *)

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>
)
```

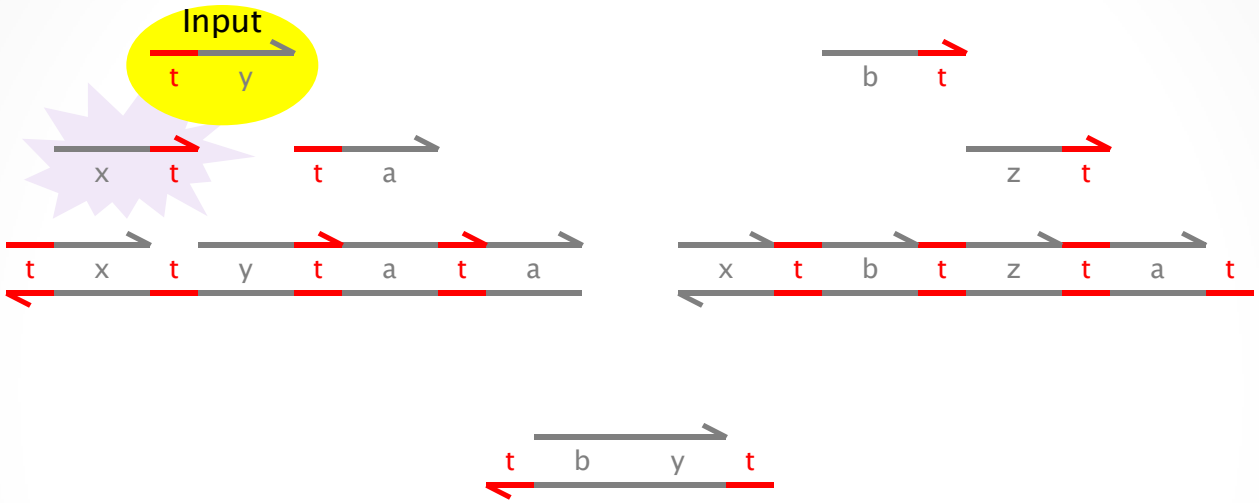
Join x+y→z



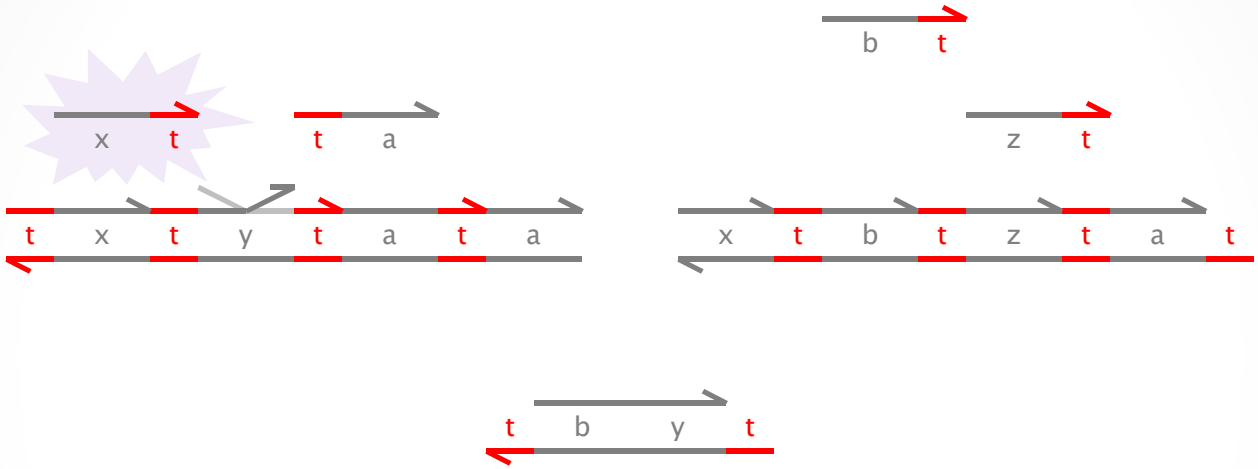
Join x+y→z



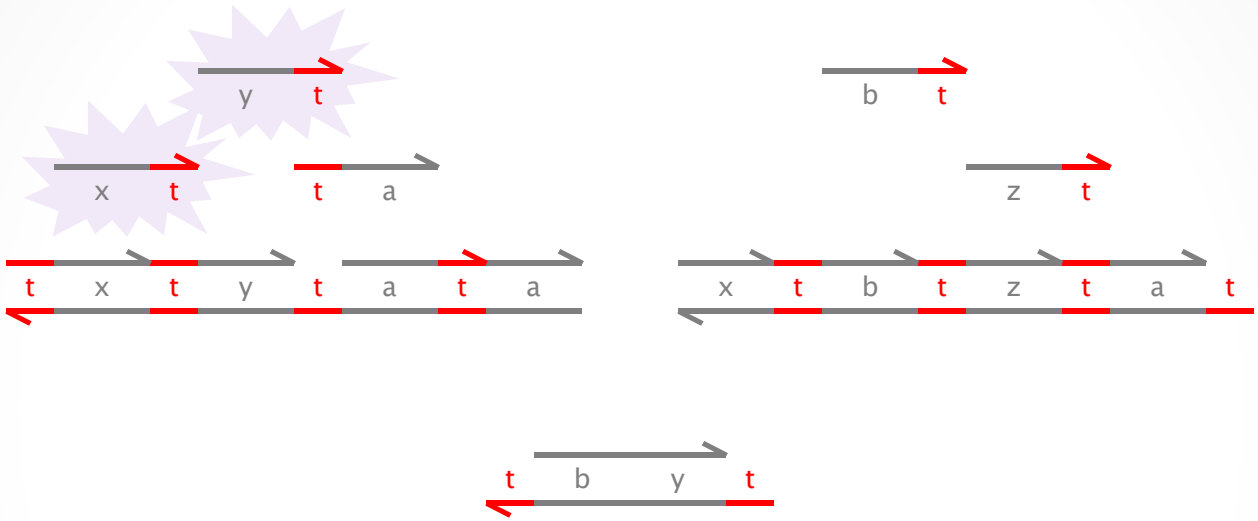
Join x+y→z



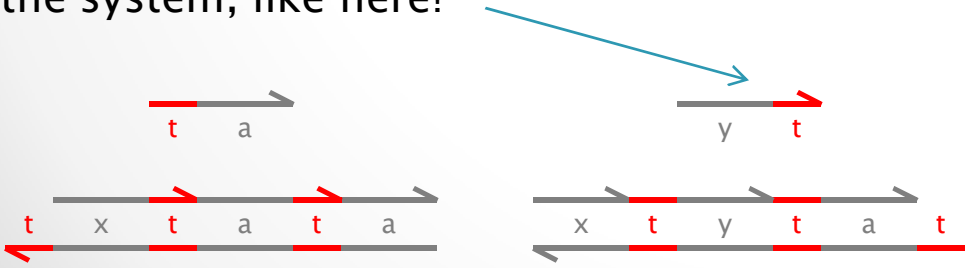
Join x+y→z



Join x+y→z



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

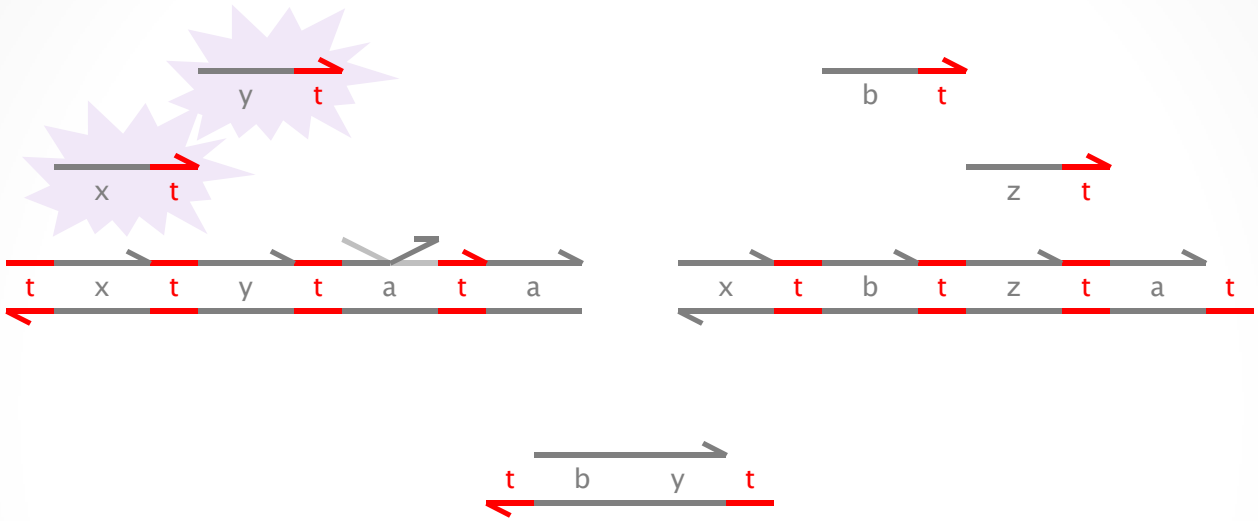


Transducer x→y

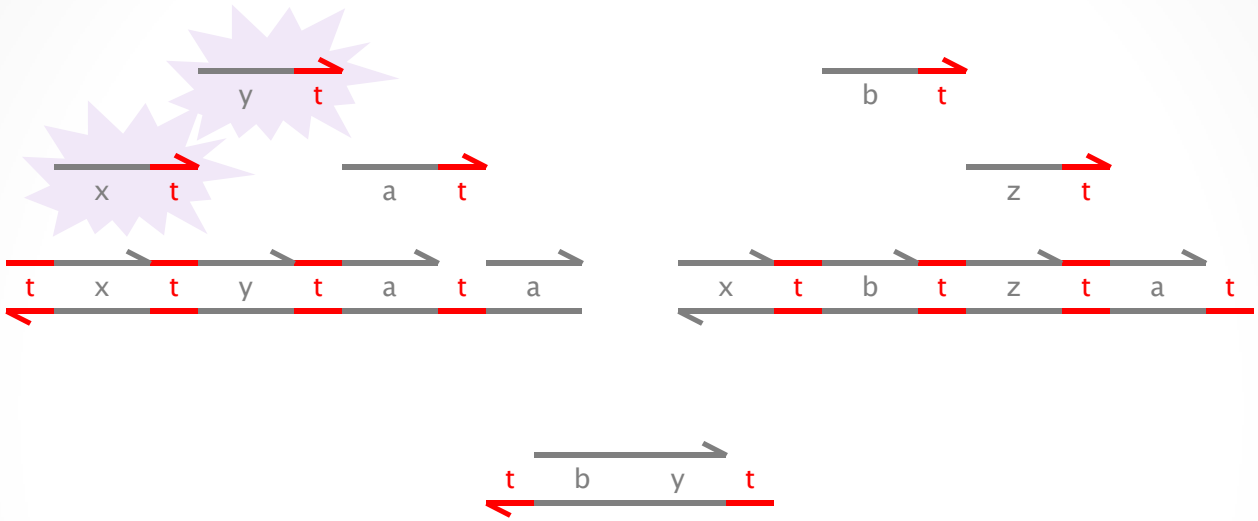
Instead, the collection of **yt** must be triggered only by a signal signifying that an x+y→z gate has fired. That signal is **tb**, which will trigger the collection of **yt** after output **tz** is produced.

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

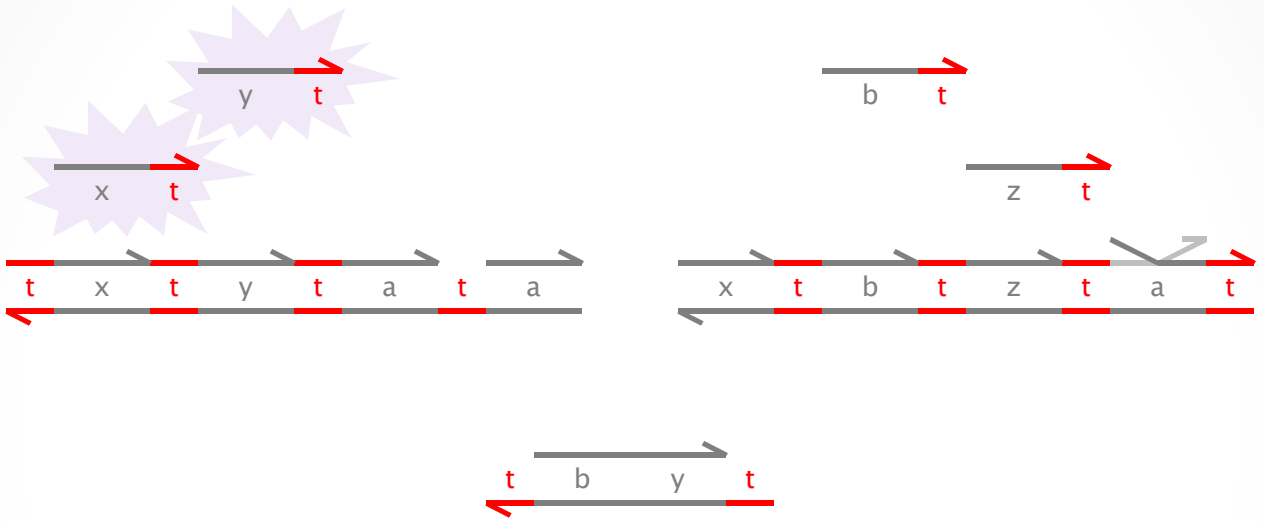
Join x+y→z



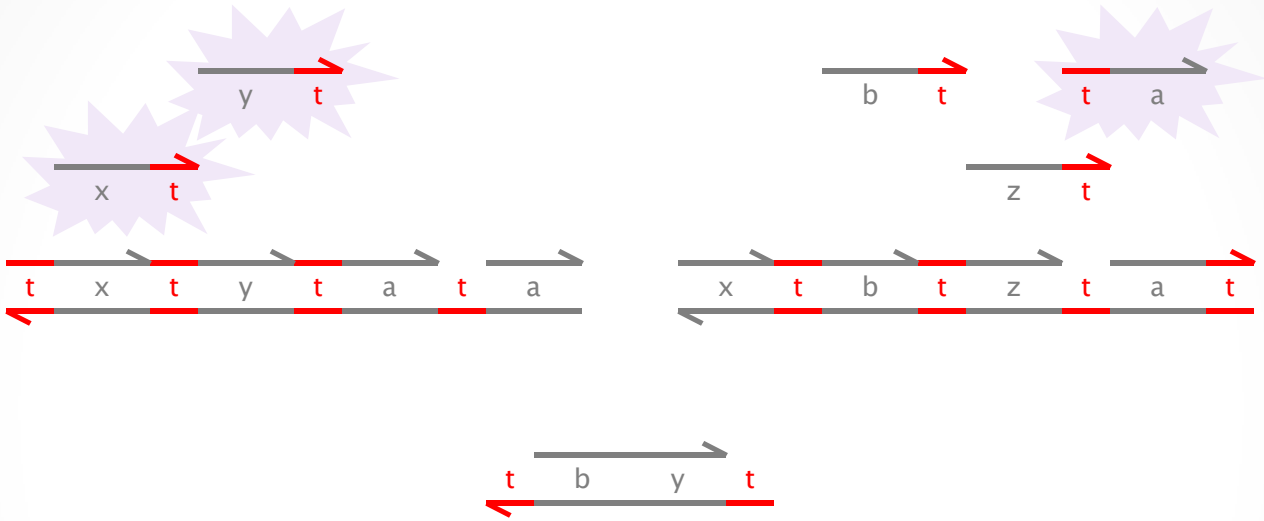
Join x+y→z



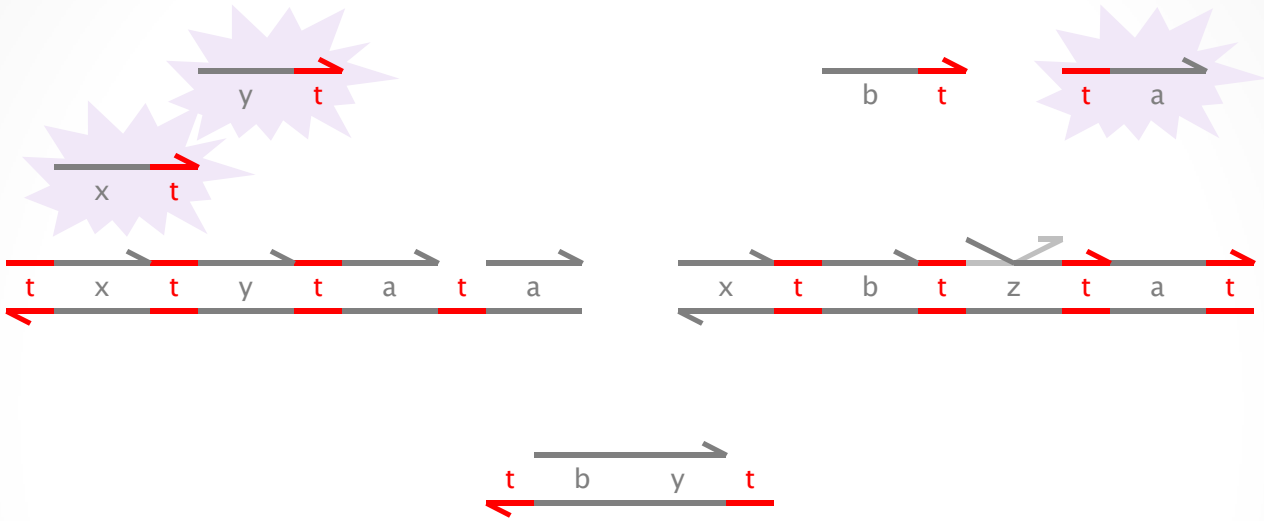
Join x+y→z



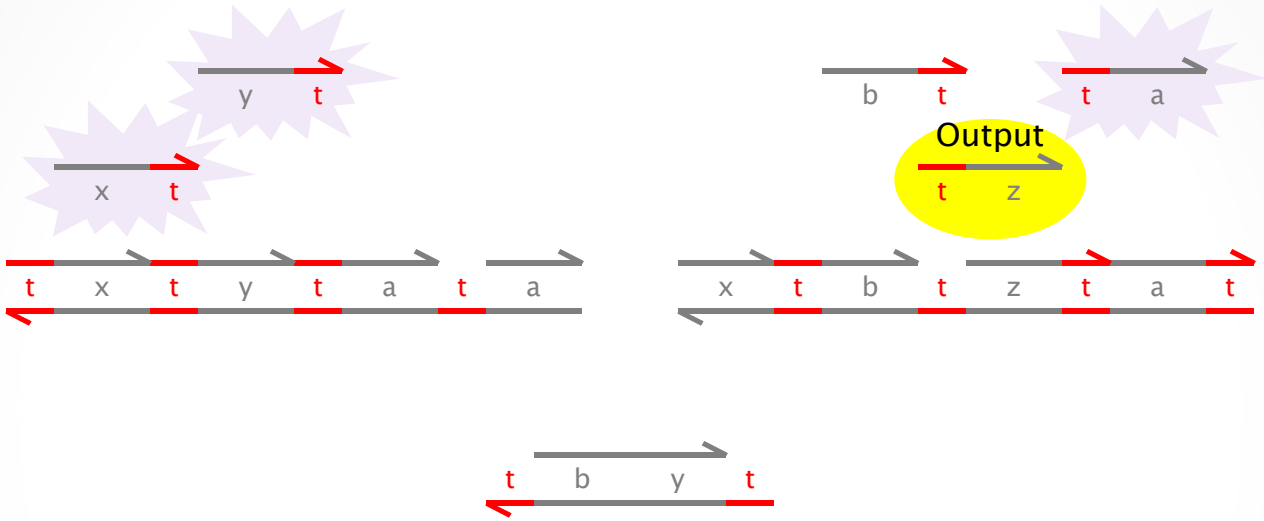
Join x+y→z



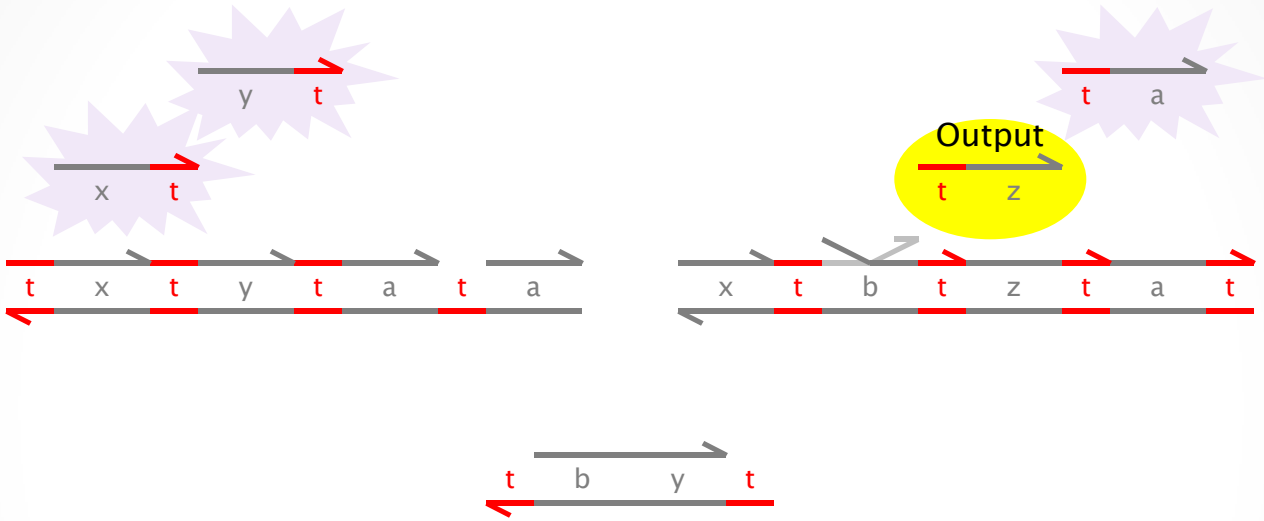
Join x+y→z



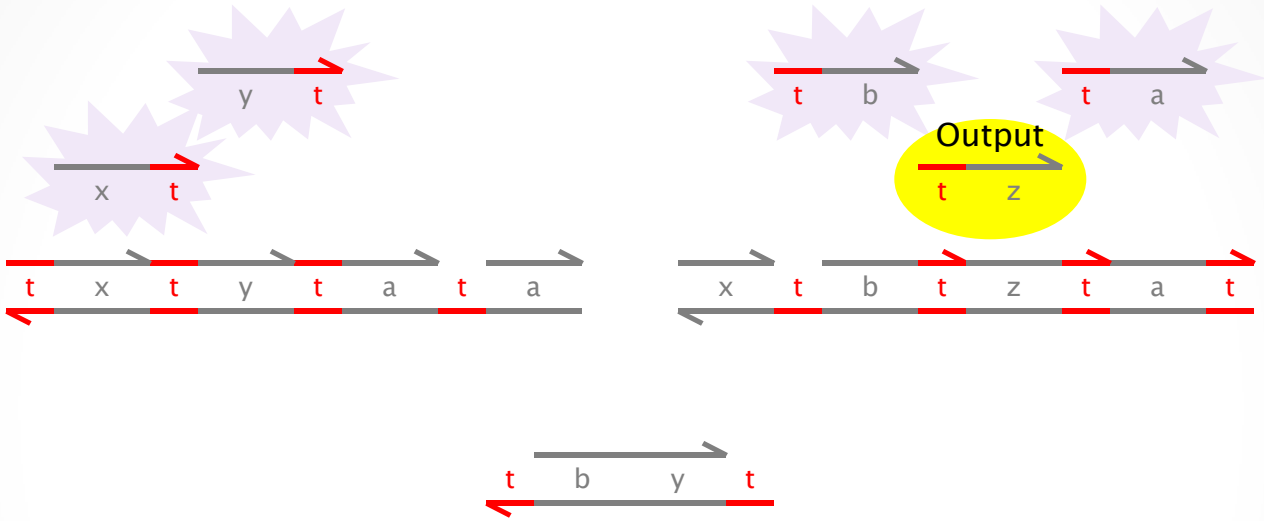
Join x+y→z



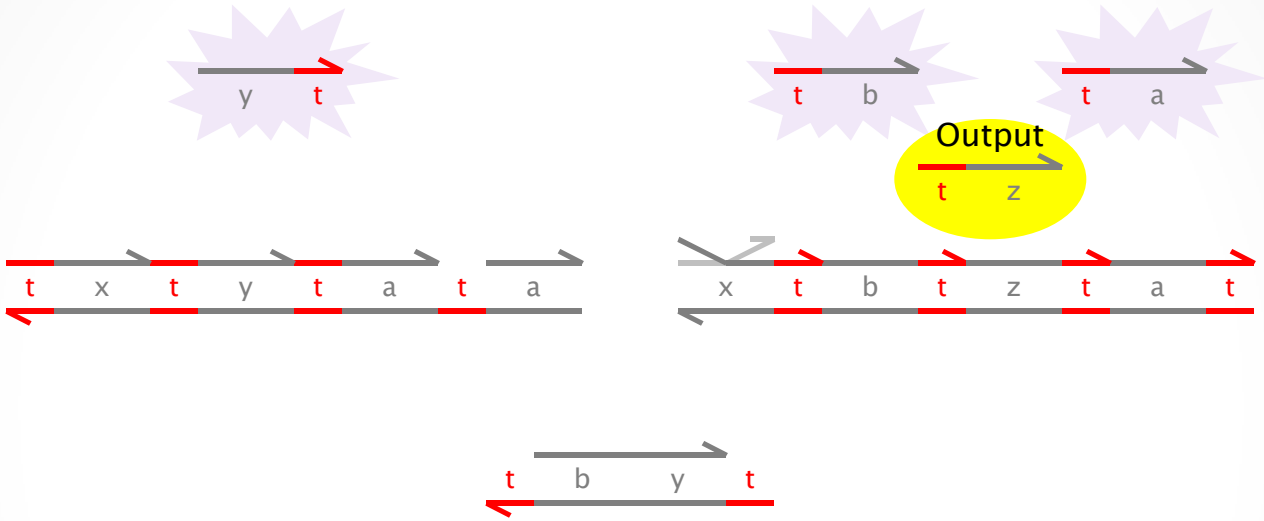
Join x+y→z



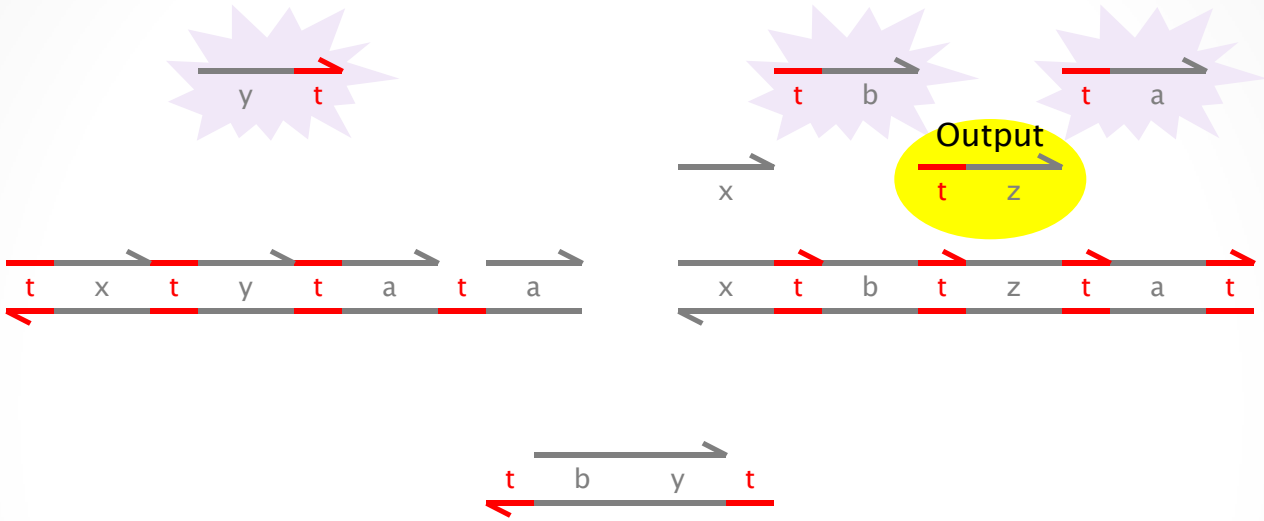
Join x+y→z



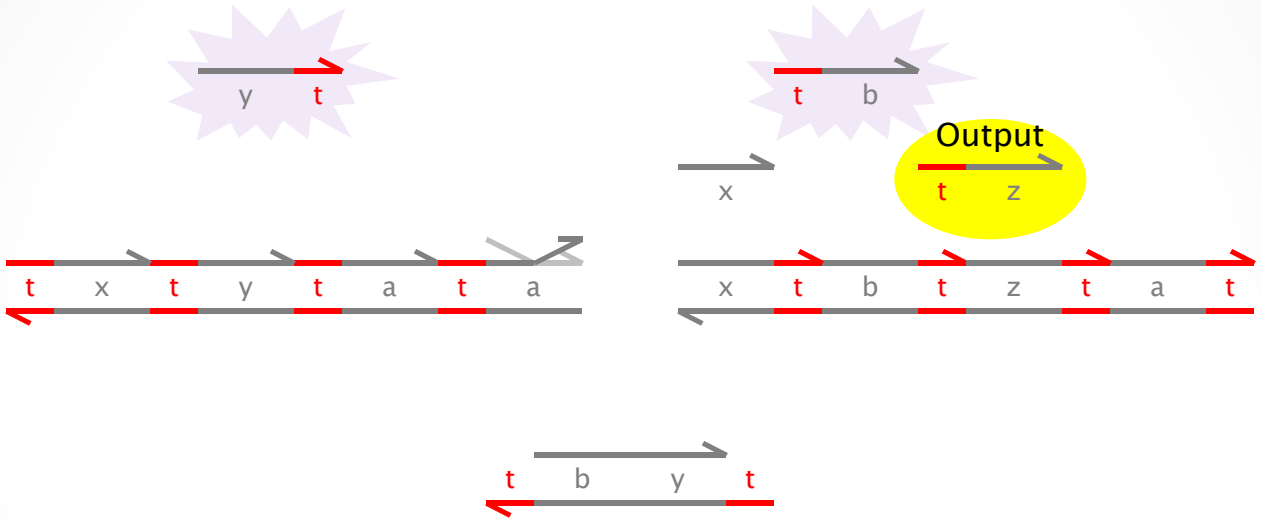
Join x+y→z



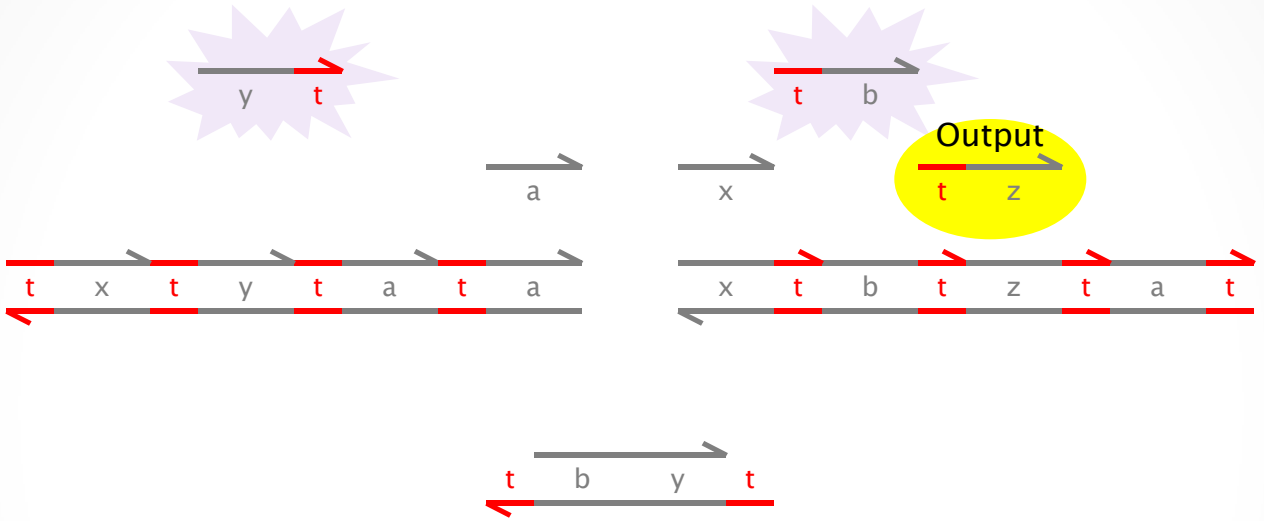
Join x+y→z



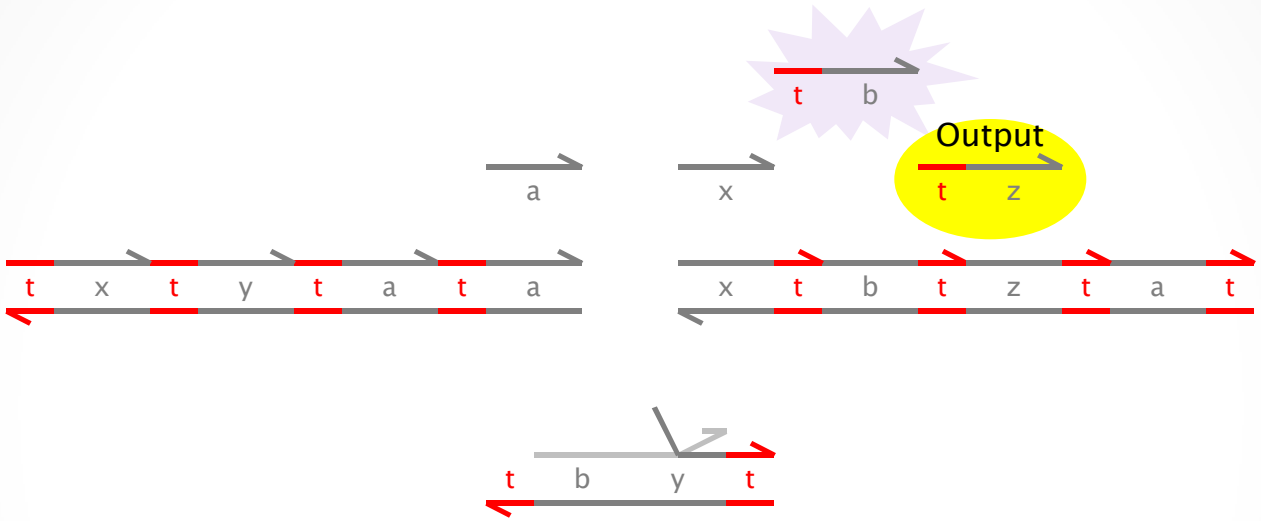
Join x+y→z



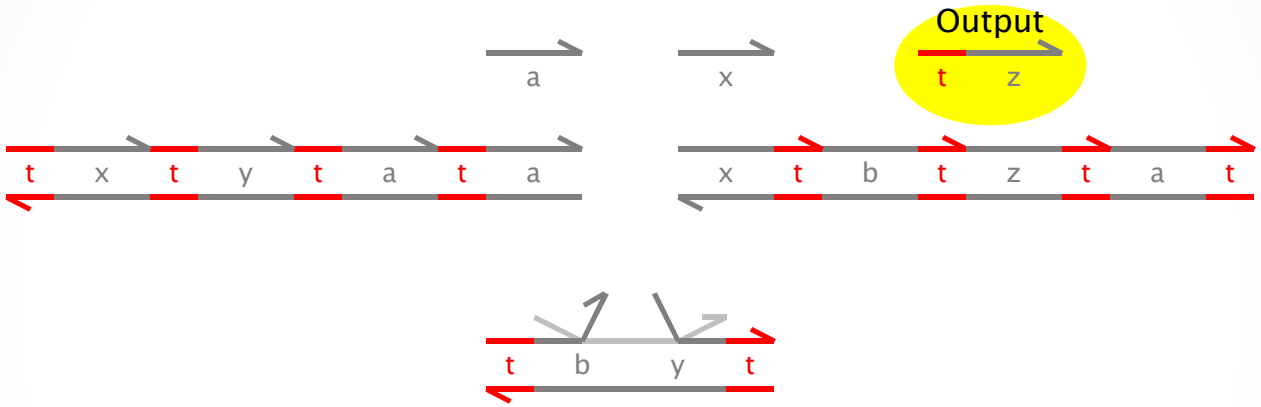
Join x+y→z



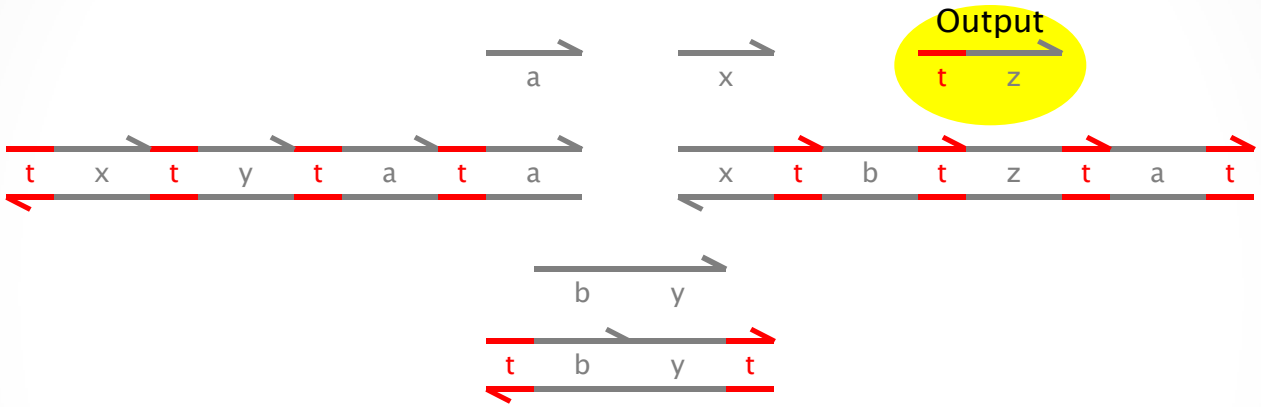
Join x+y→z



Join x+y→z



Join x+y→z



General n×m Join-Fork

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

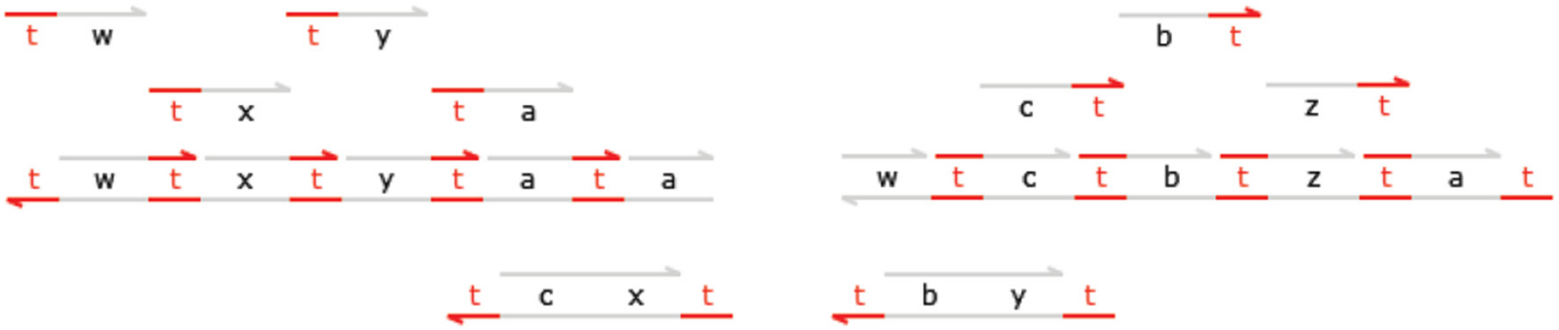
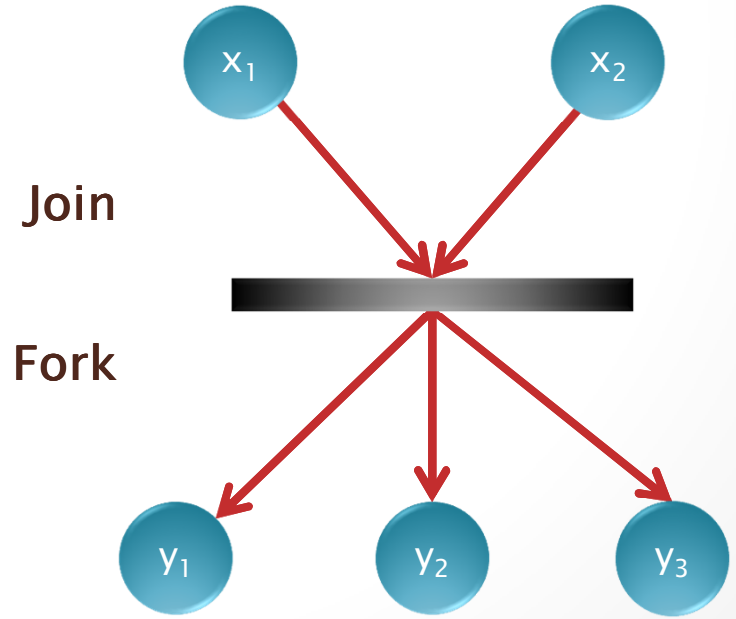


Figure 9: 3-Join $J_{wxyz} \mid tw \mid tx \mid ty \rightarrow tz$: initial state plus inputs tw, tx, ty.

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.



Verification

• • •

Verification Issues

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

Correctness

- The spec of a transducer: $T_{xy} + tx \rightarrow ty$
  - Is it true at all?
  - Is it true *possibly,* or *necessarily,* or *probabilistically (measure 1)?*
  - Is it true in the context of a *population of identical transducers?*
  - Is it true *in all possible contexts?*
  - Is it *(more)* true for large populations?
  - Is it true for infinite populations (continuous limit)?

Nick Algebra

Nick Algebra

S ::= t.x ⋮ x.t — single strand

D ::= ø ⋮ t ⋮ x ⋮ t.x ⋮ x.t ⋮ x.x ⋮ D†D — double strand

U ::= S ⋮ D ⋮ U|U ⋮ (νx)U — soup

S

D

nick operator

Algebraic Equality

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

$\underline{D_1}\dagger(\underline{D_2}\dagger\underline{D_3}) = (\underline{D_1}\dagger\underline{D_2})\dagger\underline{D_3}$
$\text{ø}\dagger\underline{D} = \underline{D}\dagger\text{ø} = \underline{D}$

$U_1|(U_2|U_3) = (U_1|U_2)|U_3$
$U_1|U_2 = U_2|U_1$
$\text{ø}|U = U|\text{ø} = U$

$(\nu x)U = (\nu y)(U\{y/x\})$ if $y \notin pd(U)$
$(\nu x)\text{ø} = \text{ø}$
$(\nu x)(U_1|U_2) = U_1|(\nu x)U_2$ if $x \notin pd(U_1)$
$(\nu x)(\nu y)U = (\nu y)(\nu x)U$

Reduction

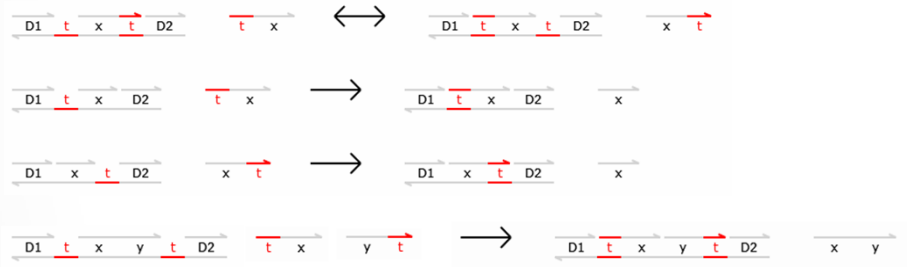
$\underline{D_1\,\dagger t\,\dagger x t\,\dagger D_2} \mid tx \quad \leftrightarrow \quad \underline{D_1\,\dagger t x\,\dagger t\,\dagger D_2} \mid xt$ exchange

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

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

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

i.e.:



$D \;\rightarrow\; \emptyset$ if 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).
  - $U \rightarrow^{\forall} U$ means that U is *reversible*.
  - If U_2 is the only terminal state then $U_1 \rightarrow^{\forall} U_2$ means that U_1 *must* reduce to U_2.

Gate Definitions

- $T_{xay} = \underline{t^{\dagger}xt^{\dagger}at^{\dagger}a} \mid ta \mid \underline{x^{\dagger}ty^{\dagger}ta^{\dagger}t} \mid yt$
- $T^{n}_{xy} = (\nu a)((T_{xay})^{n})$

- $F_{xayz} = \ldots$
- $F^{n}_{xyz} = (\nu a)((F_{xayz})^{n})$

- $J_{xyaz} = \ldots$
- $J^{n}_{xyz} = (\nu a)((J_{xyaz})^{n})$

Correctness

- Proposition: 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 case analysis and induction on n.

- 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 (more states).
  - Will-correctness for combinations of gates is harder (does not compose and requires analysis of joint state space).
  - We are using modelchecking to verify some of these properties. [Andrew Phillips & David Parker in PRISM]

Interfering Transducers

- Although $T_{xay} \mid T_{yax} \mid tx \not\rightarrow^{\forall} tx$
- We have $T_{xay} \mid T_{yax} \mid tx \mid ty \rightarrow^{\forall} tx \mid ty$
- That means that a large population of such gates in practice does not deadlock easily: each pair of deadlocked gates can be unblocked by another pair correctly producing a ty as an intermediate product.
- Wisdom of the masses: individuals can be wrong, but the population is right. It is very unlikely that a significant fraction of gates ends up being deadlocked.

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.
- Verification issues
  - Verification techniques for gate populations.
  - Are the fork/join gates in Nick Algebra a correct implementation of (Strand Algebra and) Petri nets?