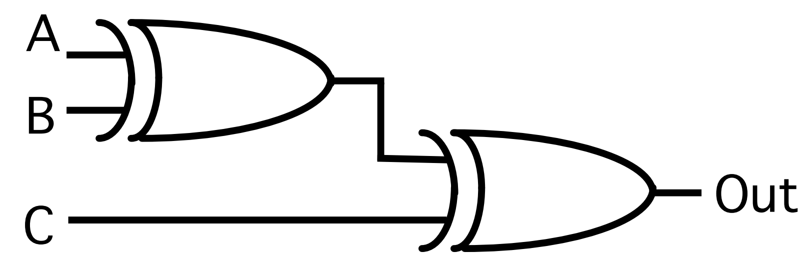
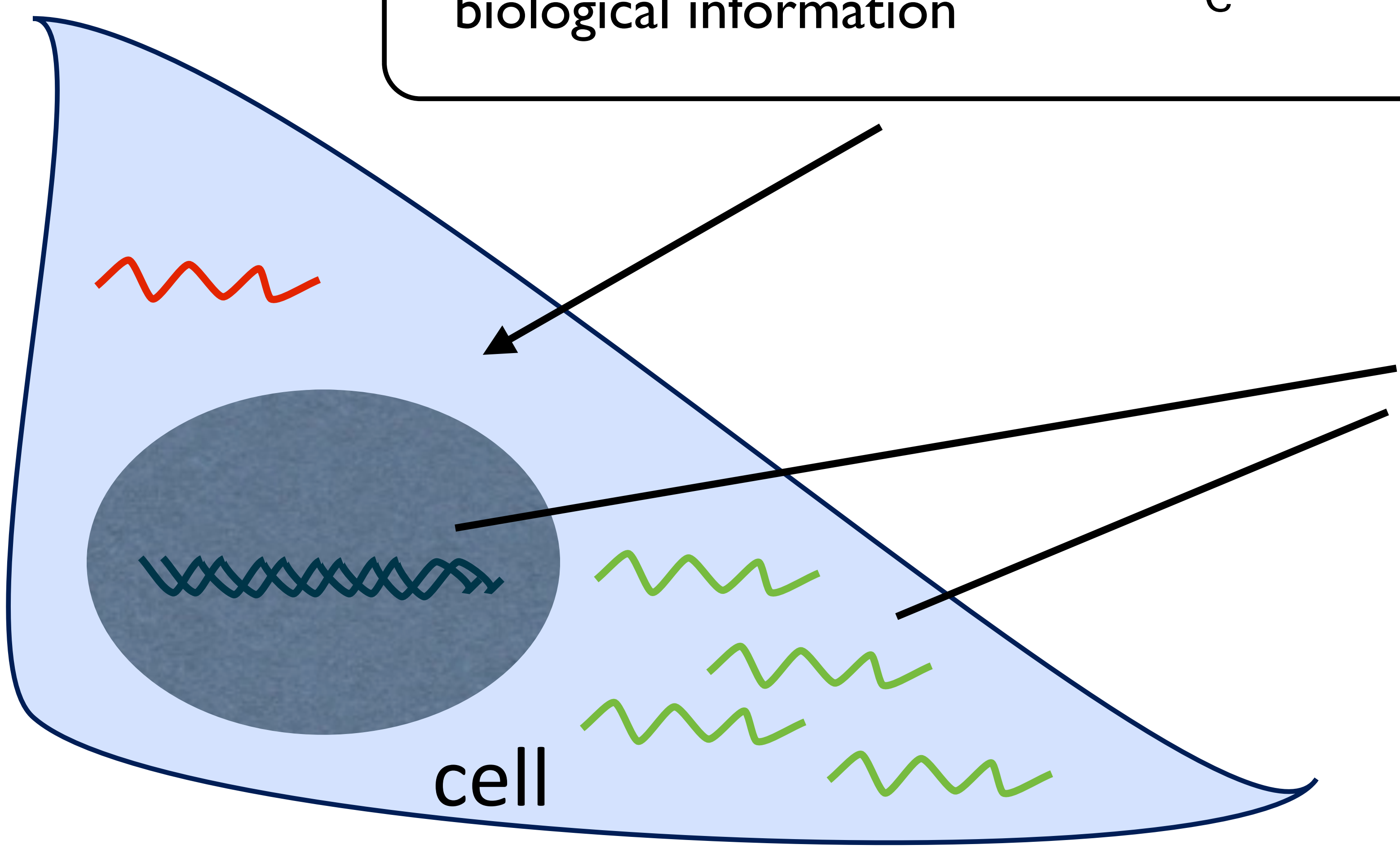
Module 5: Logic circuits with DNA strand displacement (part 1)

CSE590: Molecular programming and neural computation.

Goal: Engineering embedded controllers for biochemical systems

A cell-based "computer" needs to be biocompatible, and sense, analyze and act on biological information





Biological Information is encoded in the sequences and amounts of biomolecules (DNA, RNA, proteins, etc.)

ugagguaguagguuguauaguu
ugagguaguagguugugugguu
ugagguaguagguuguaugguu
agagguaguagguugcauaguu
ugagguaggagguuguauaguu
ugagguaguagauuguauaguu
ugagguaguaguuuguacaguu
ugagguaguaguuugugcuguu

Logic circuits using DNA strand displacement

Q: Why digital logic? Biology is not digital.

A: Because adherence to digital logic design has enabled incredibly complex, manmade information technology. We don't need to do exactly what biology does.

Q: Why DNA strand displacement?

A: Because it's a surprisingly powerful building block.

Basic rules

Short domains bind **reversibly**

5′-TTAAGT-3′

3′-AATTCA-5′

⇌

5′-TTAAGT-3′
3′-AATTCA-5′

Long domains bind **irreversibly**

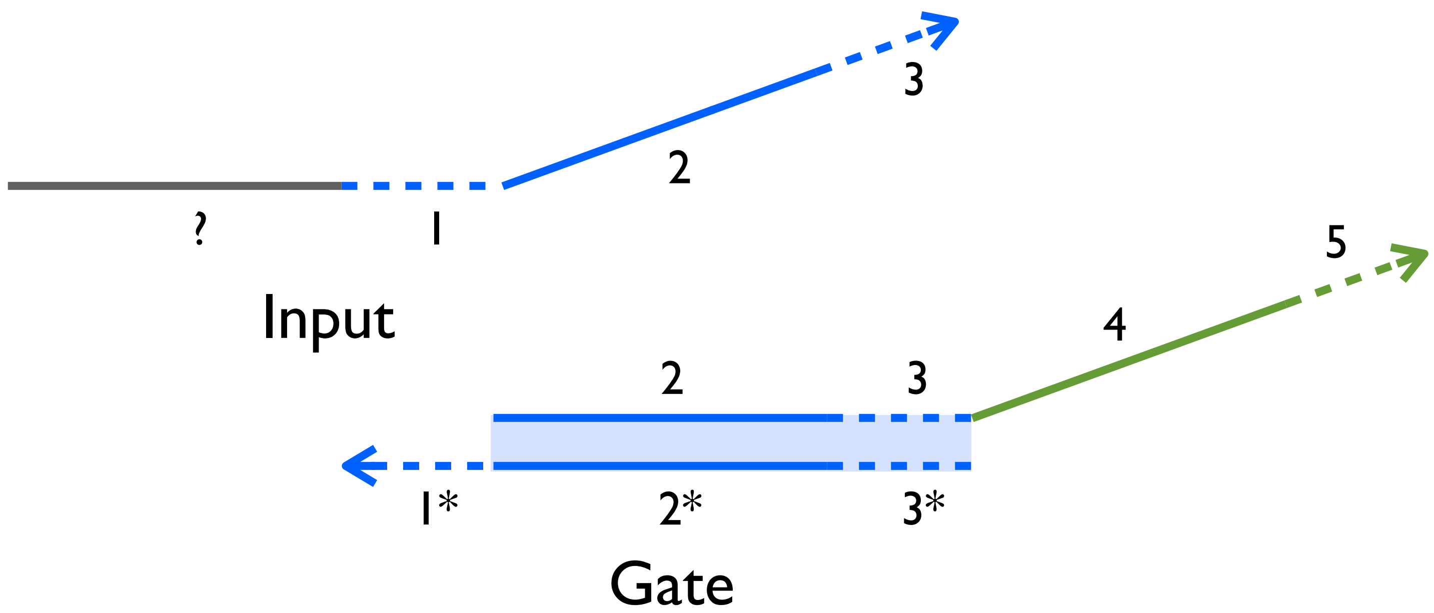
5′-TTAAGTCTAGGTGGGTTTCT-3′

3′-AATTCAGATCCACCCAAAGA-5′

→

5′-TTAAGTCTAGGTGGGTTTCT-3′
3′-AATTCAGATCCACCCAAAGA-5′

DNA strand displacement mechanism

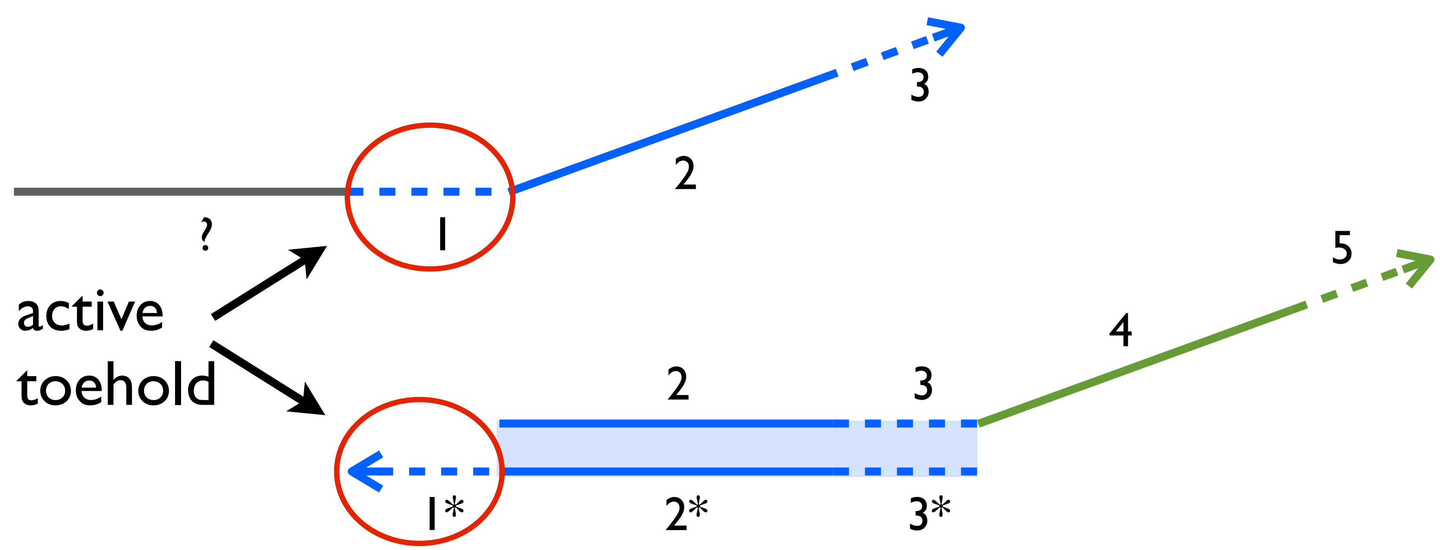


RNA sequence:

5'-AAUUCAGAUCCACCCAAAGAG-3'

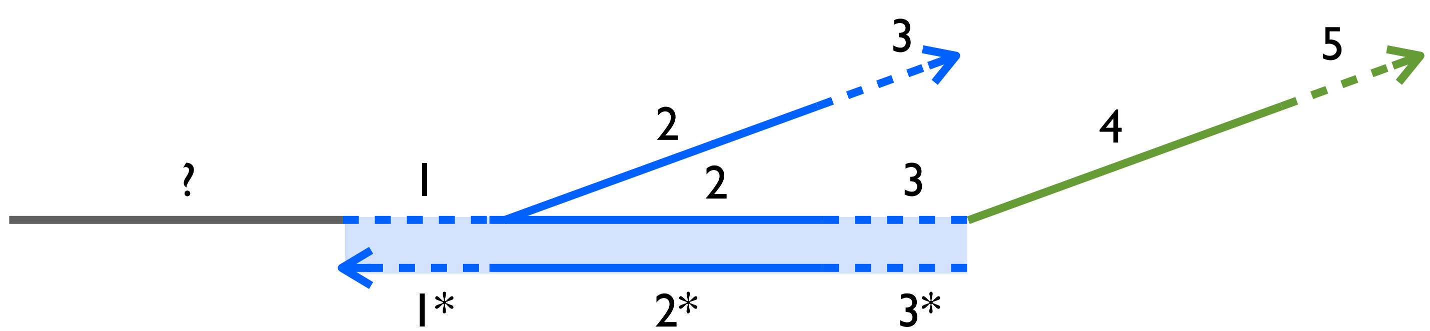
For a review see D.Y. Zhang and G. Seelig, Nature Chemistry (2011)

DNA strand displacement mechanism



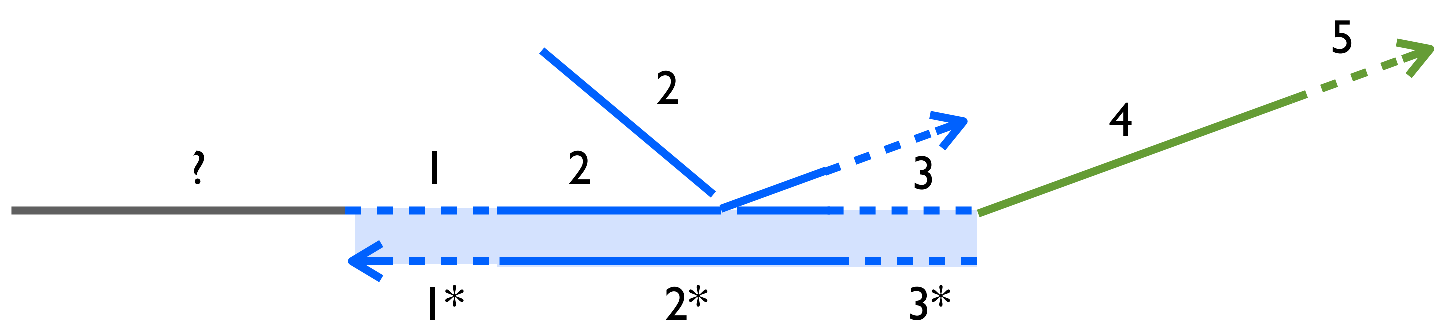
For a review see D.Y. Zhang and G. Seelig, Nature Chemistry (2011)

DNA strand displacement mechanism



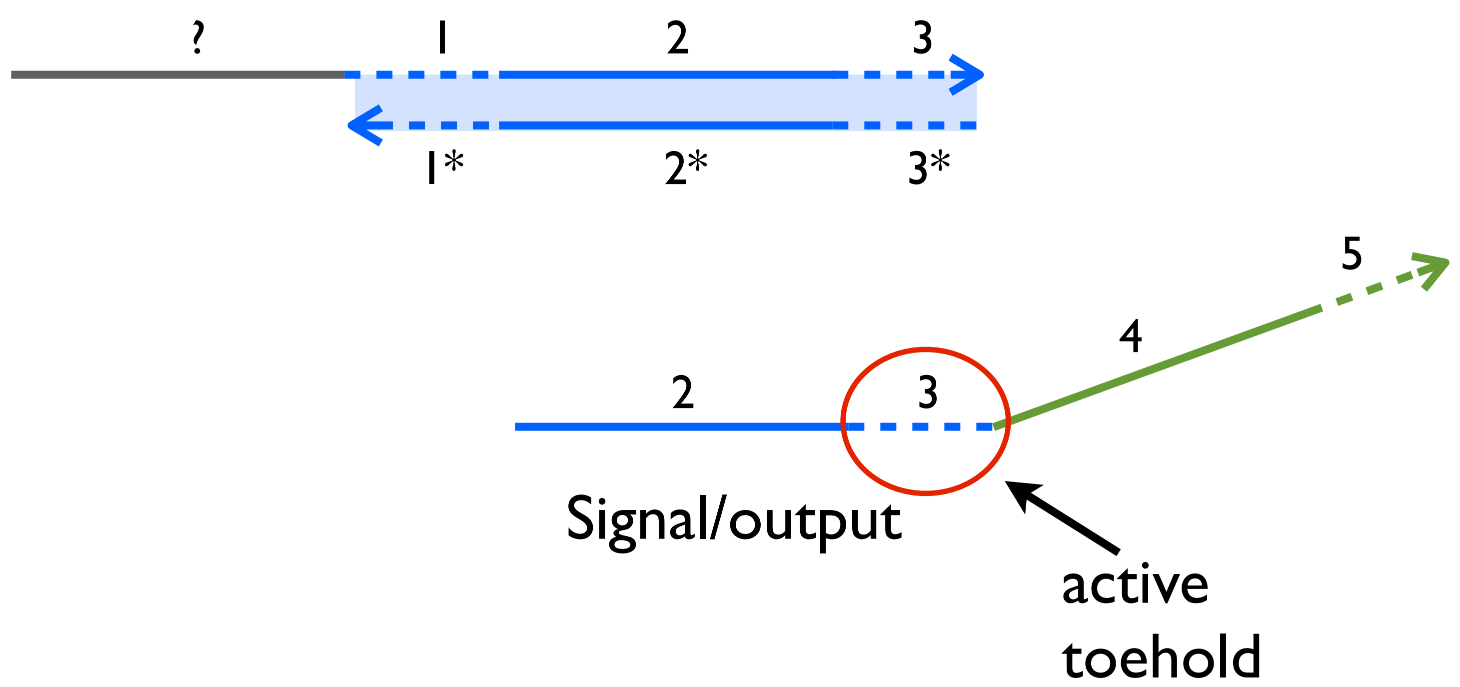
Strand displacement is initiated at the single-stranded toeholds. Toehold binding is a reversible process.

DNA strand displacement mechanism



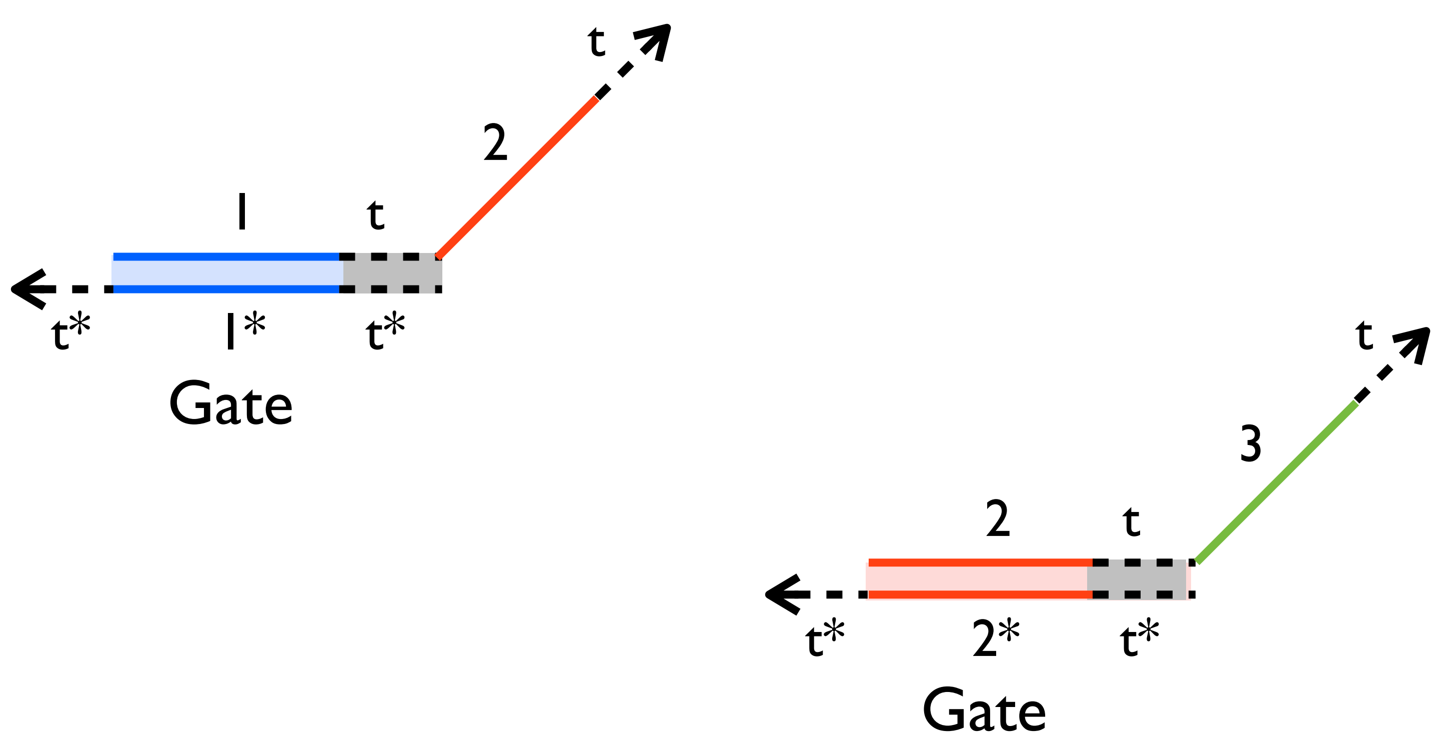
Strand displacement proceeds through a branch migration. Branch migration is a random walk.

DNA strand displacement mechanism



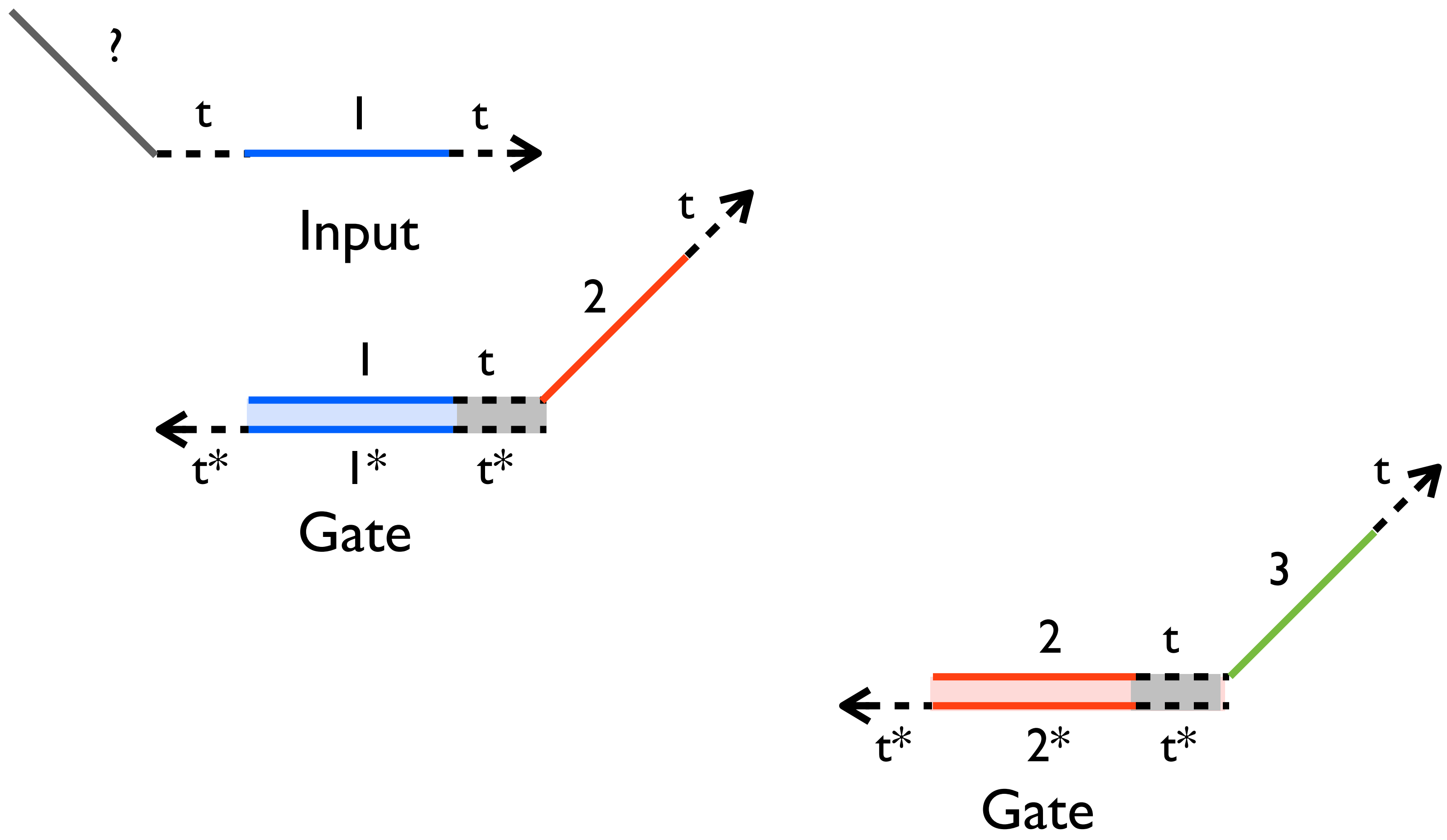
Release of the output strand is (almost) irreversible in the absence of a toehold for the reverse reaction.

Signals can propagate through multiple layers



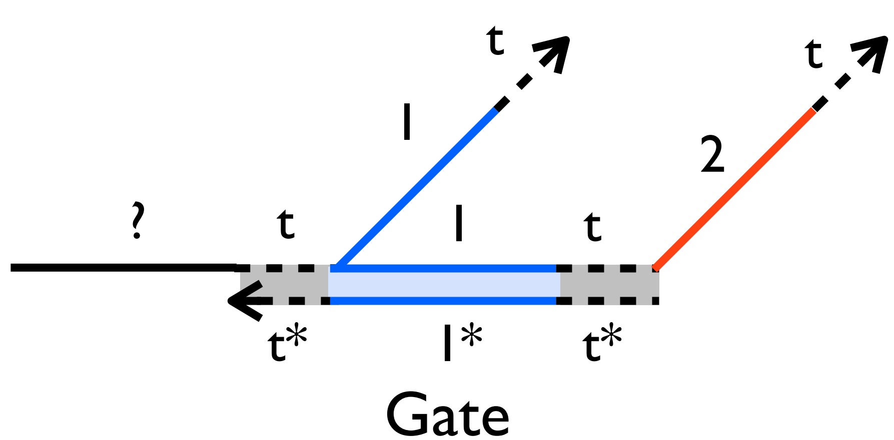
The sequences of inputs and outputs can be completely independent.

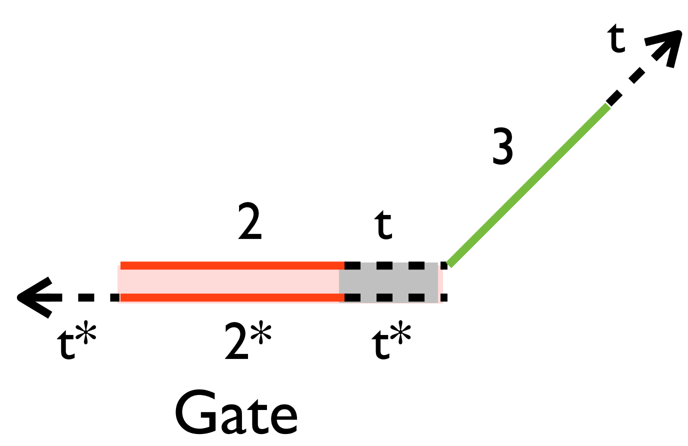
Signals can propagate through multiple layers



The sequences of inputs and outputs can be completely independent.

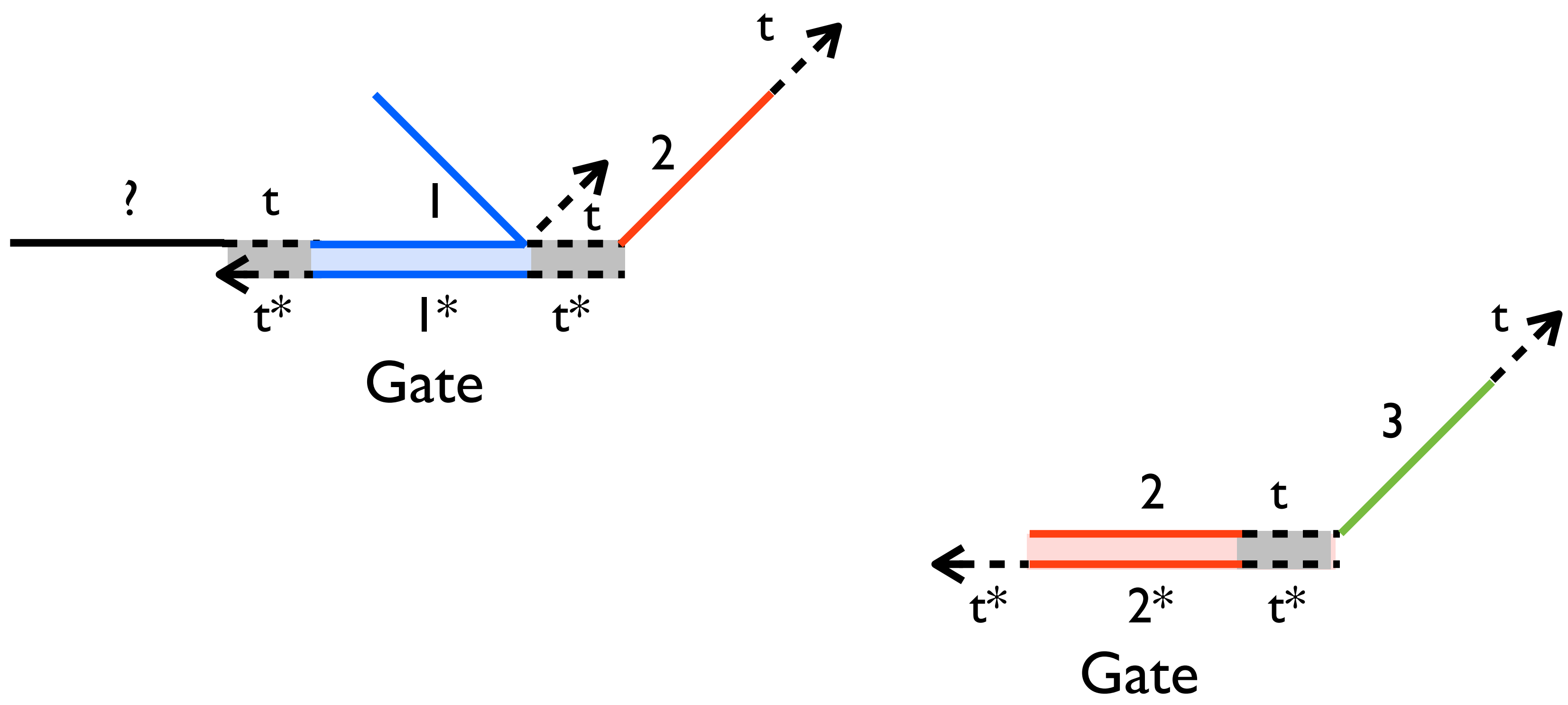
Signals can propagate through multiple layers





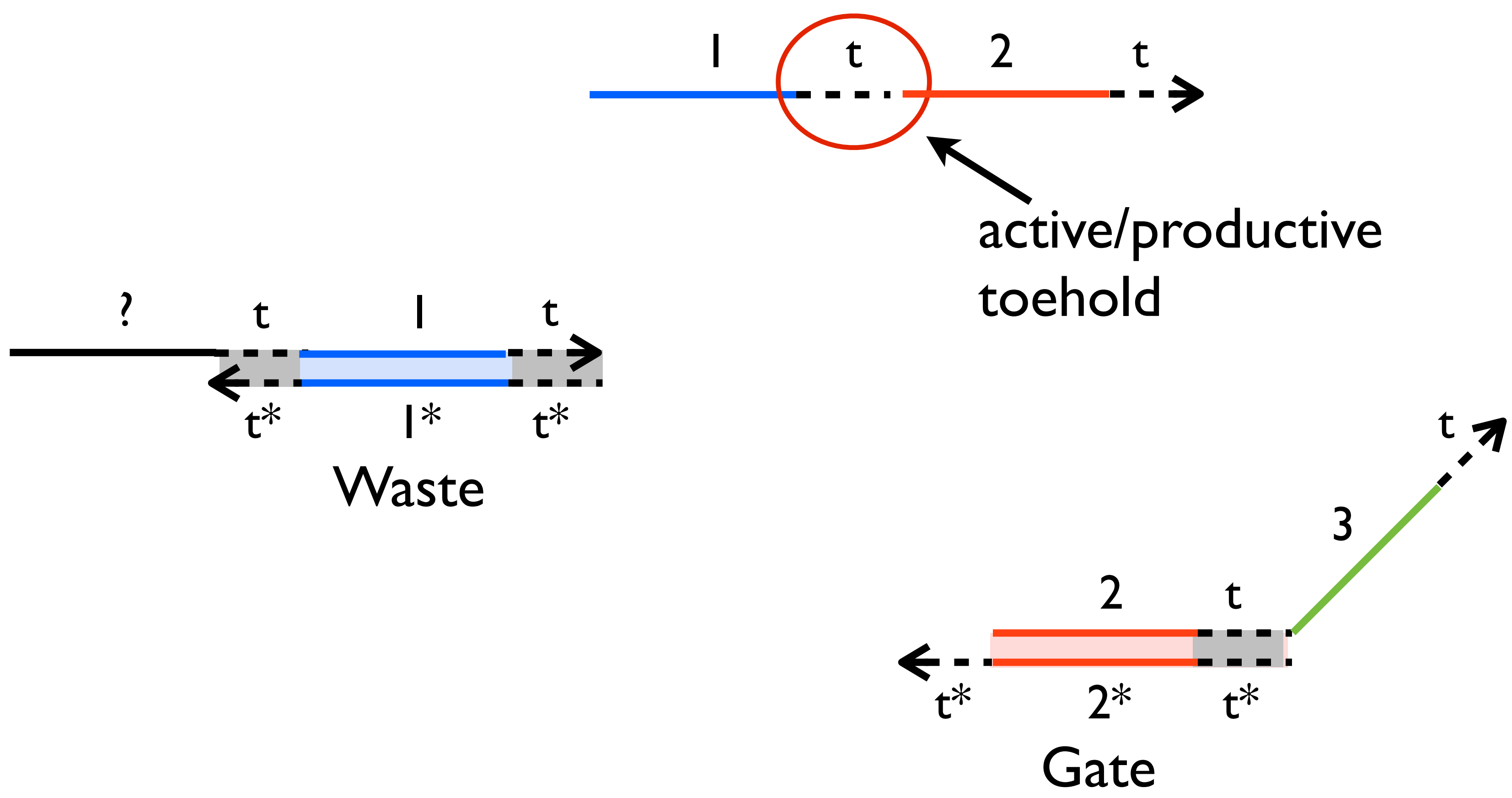
The sequences of inputs and outputs can be completely independent.

Signals can propagate through multiple layers



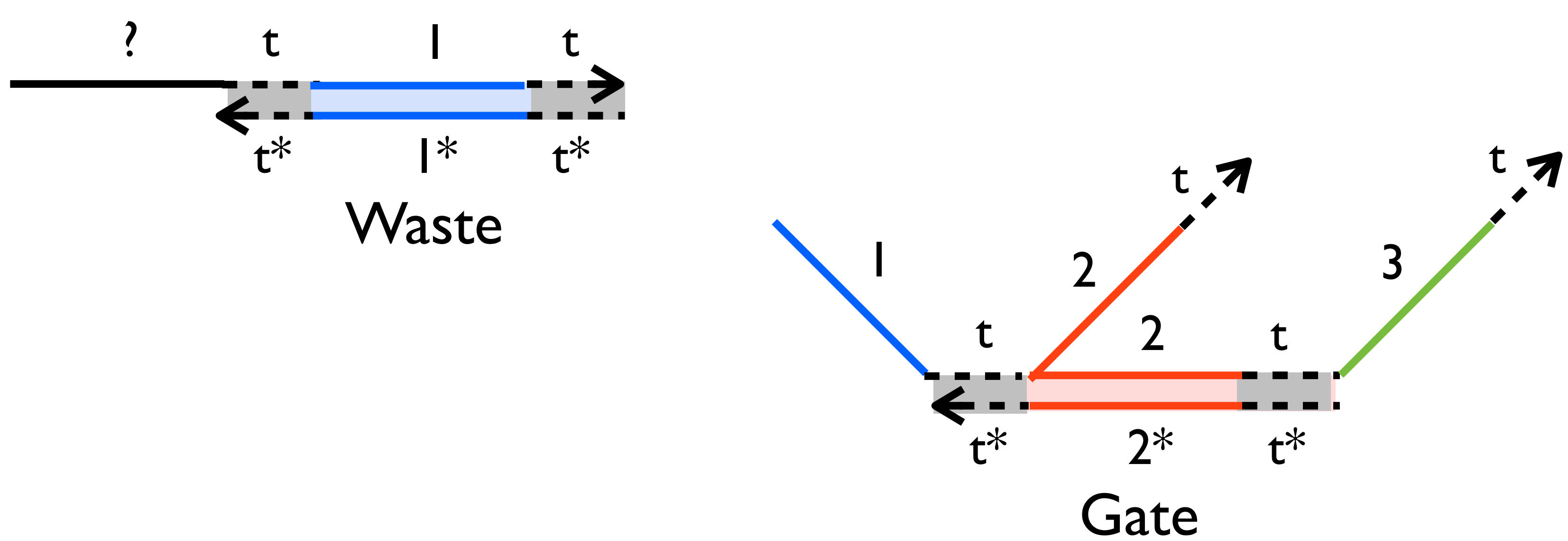
The sequences of inputs and outputs can be completely independent.

Signals can propagate through multiple layers



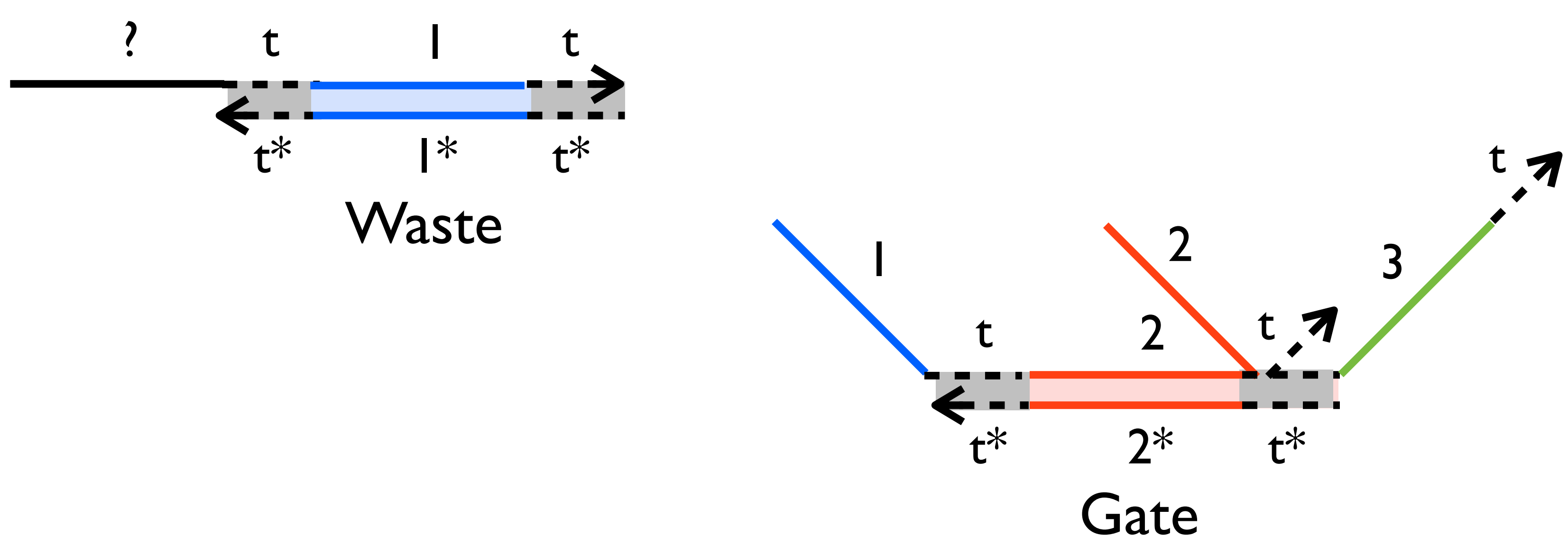
The sequences of inputs and outputs can be completely independent.

Signals can propagate through multiple layers



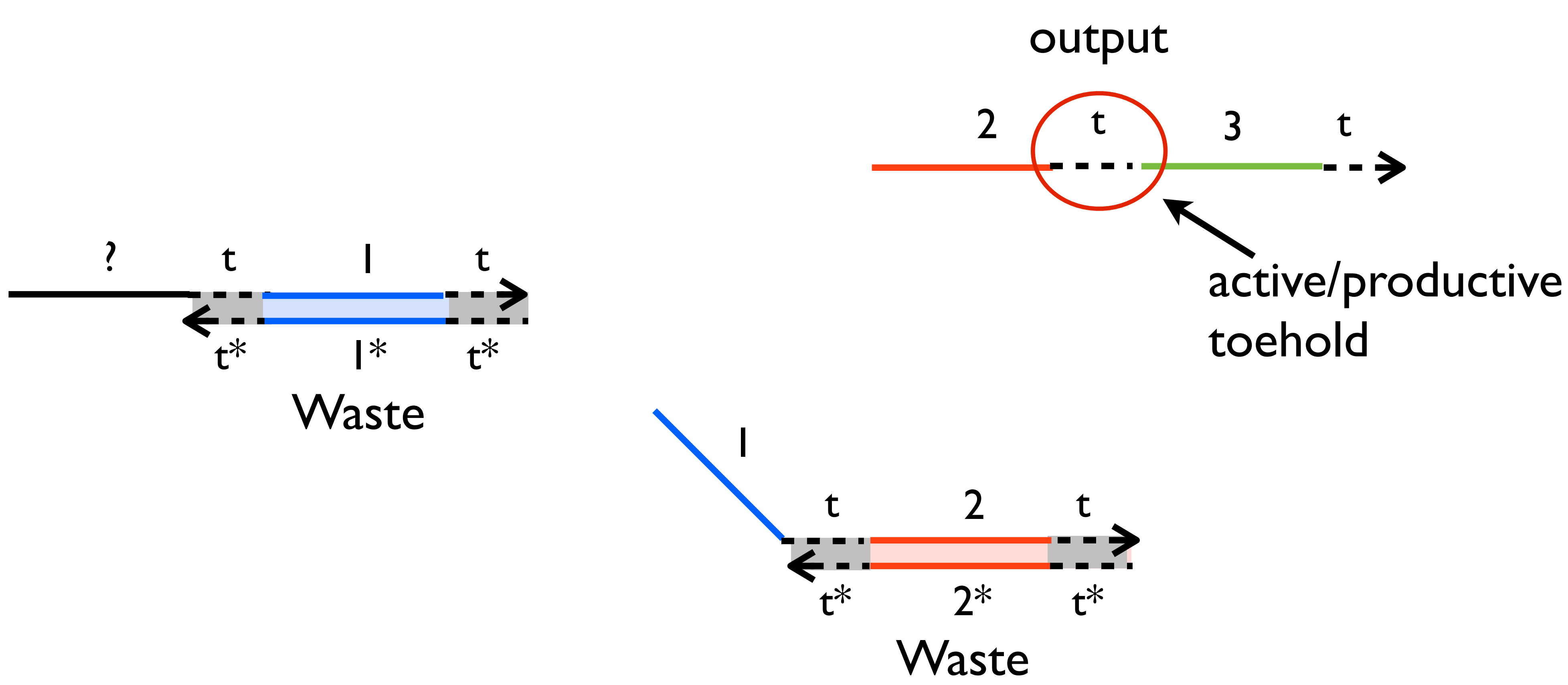
The sequences of inputs and outputs can be completely independent.

Signals can propagate through multiple layers



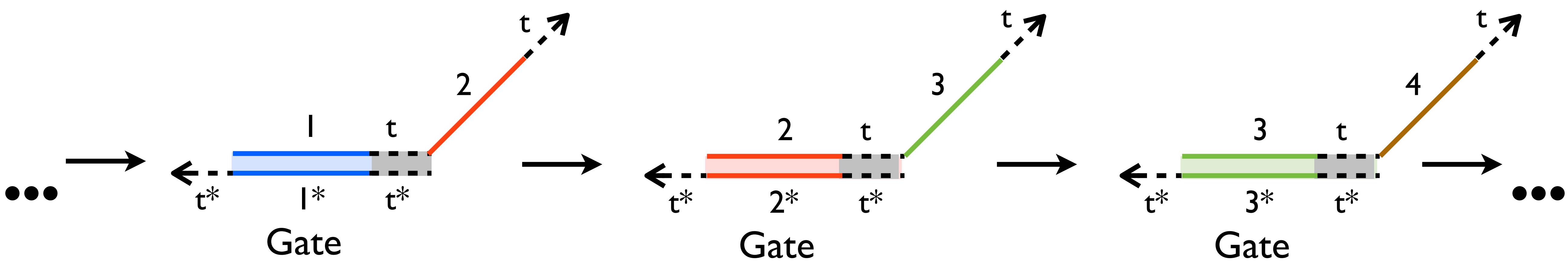
The sequences of inputs and outputs can be completely independent.

Signals can propagate through multiple layers



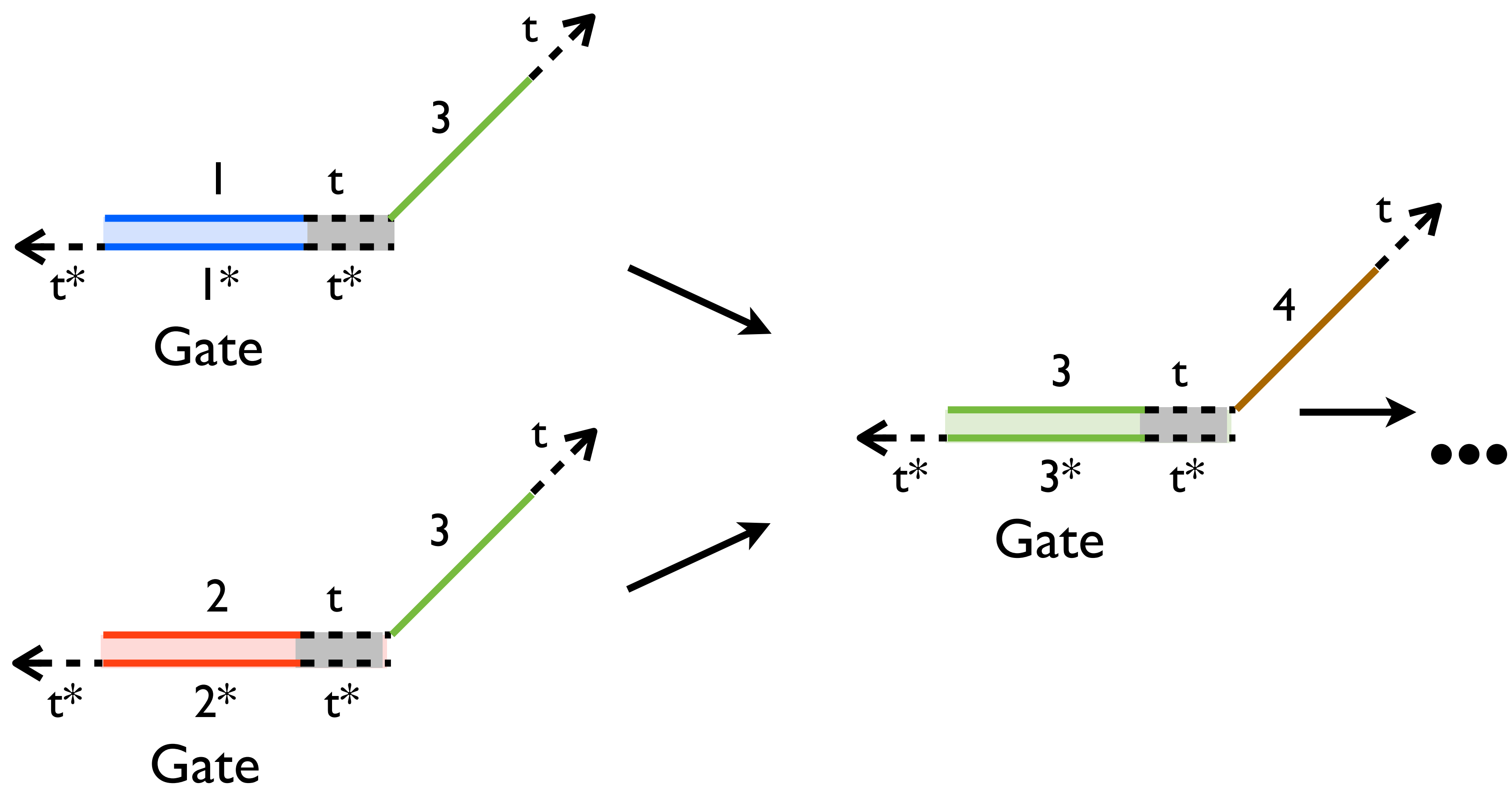
The sequences of inputs and outputs can be completely independent.

Signals can propagate through multiple layers

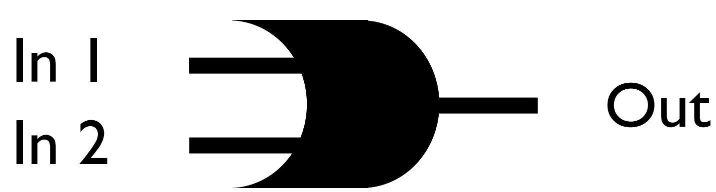


The sequences of inputs and outputs can be completely independent.

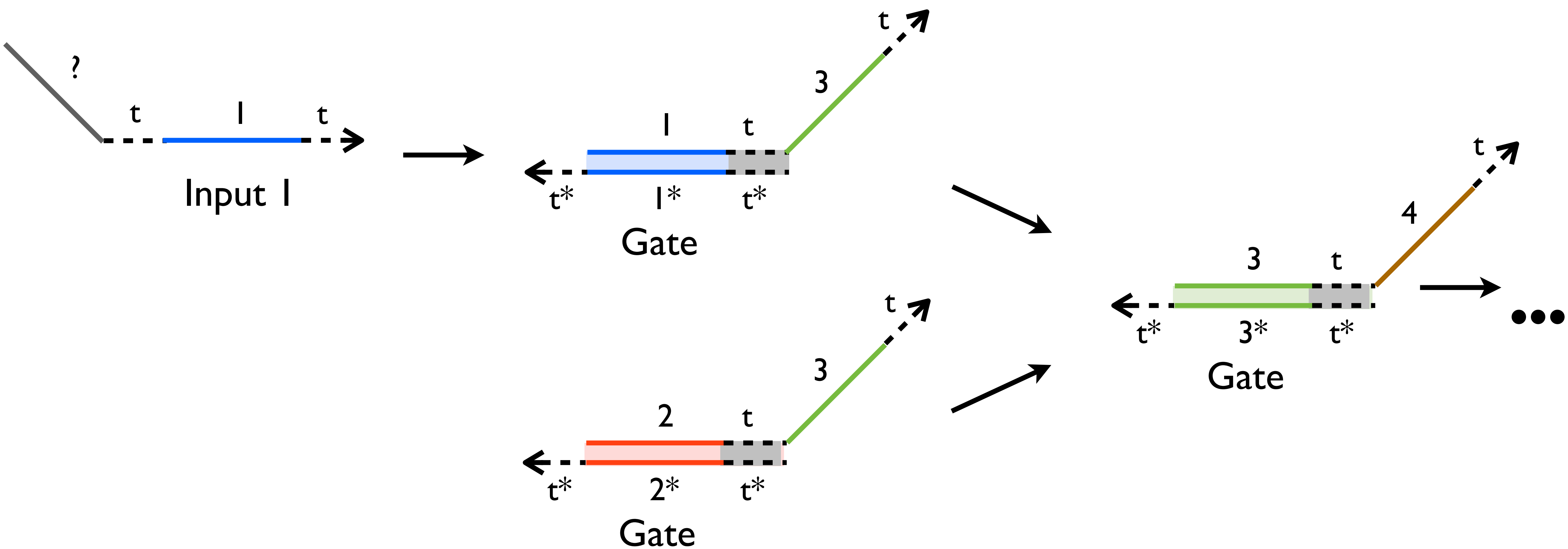
OR logic / fan-in



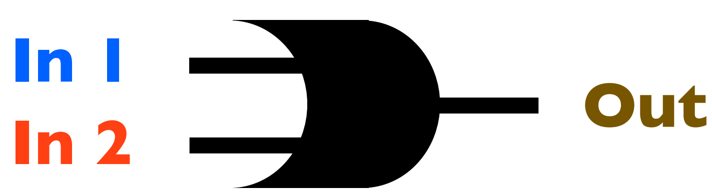
In 1	In 2	Out
0	0	0



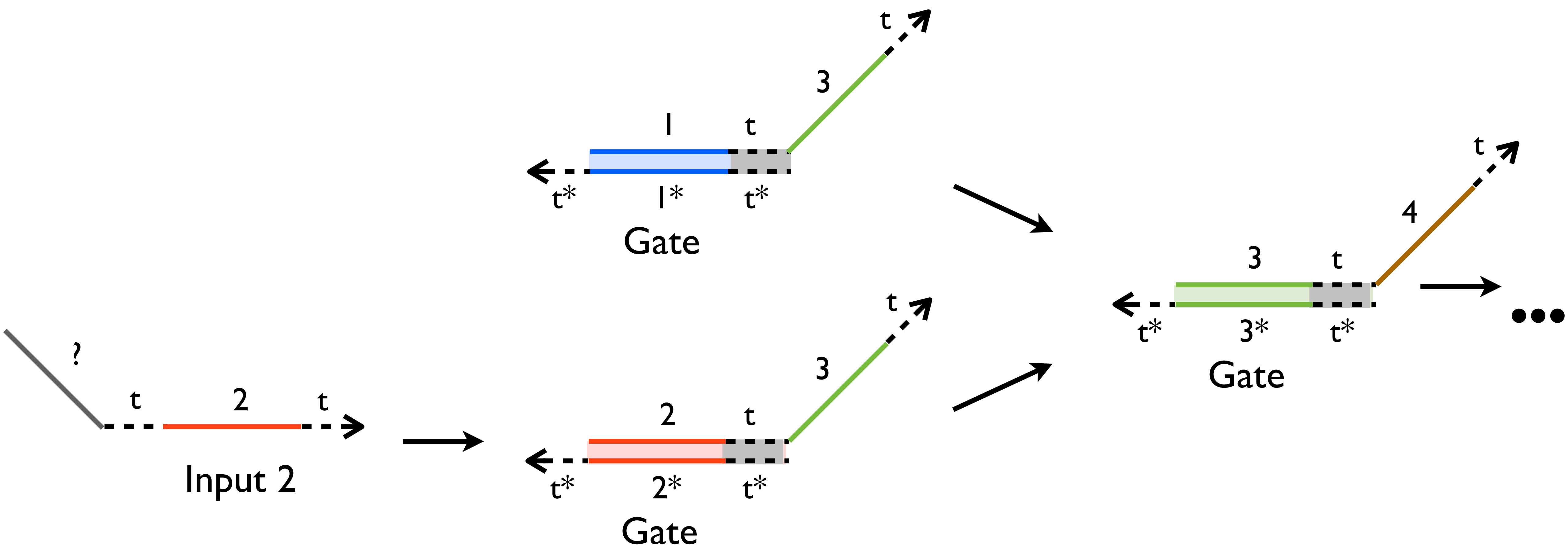
OR logic / fan-in



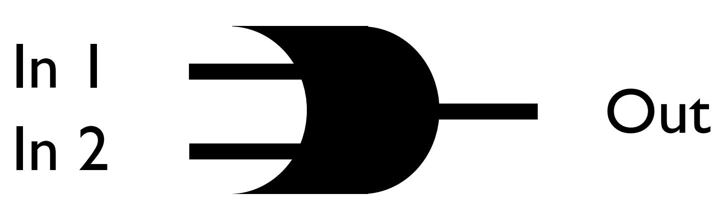
In 2	In 1	Out
0	0	0
0	1	1



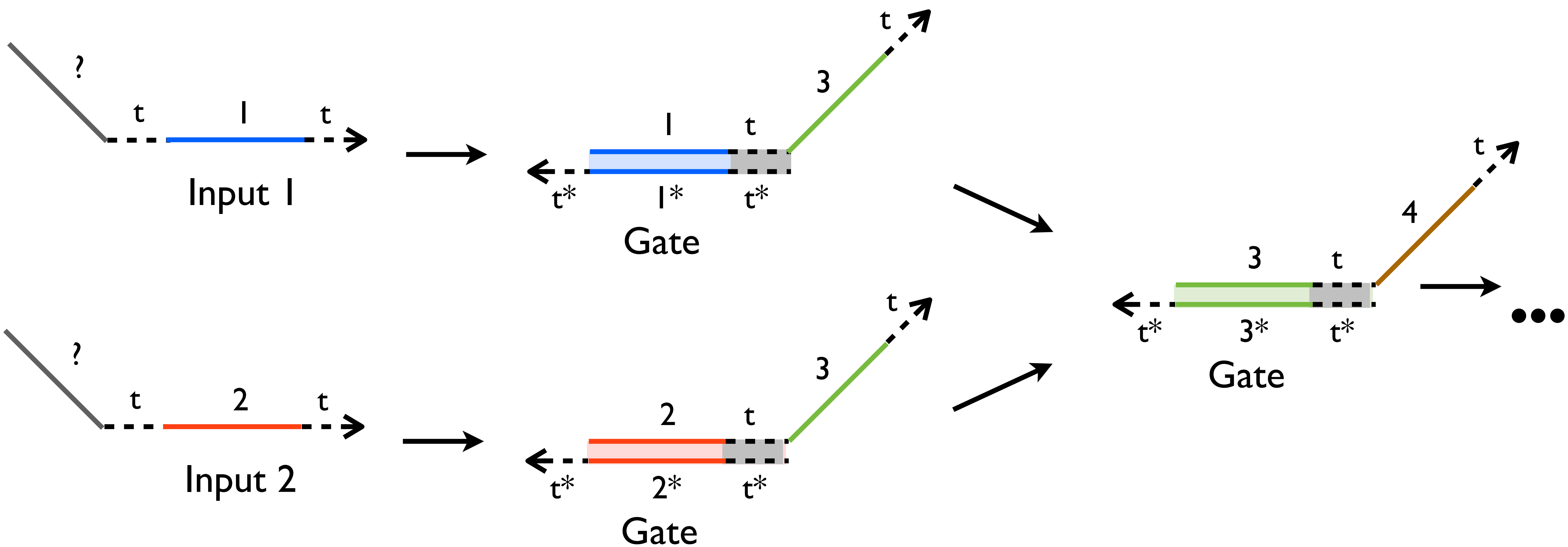
OR logic / fan-in



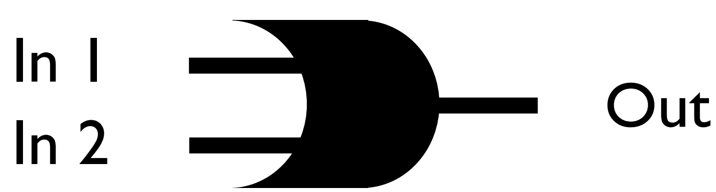
In 2	In 1	Out
0	0	0
0	1	1
1	0	1



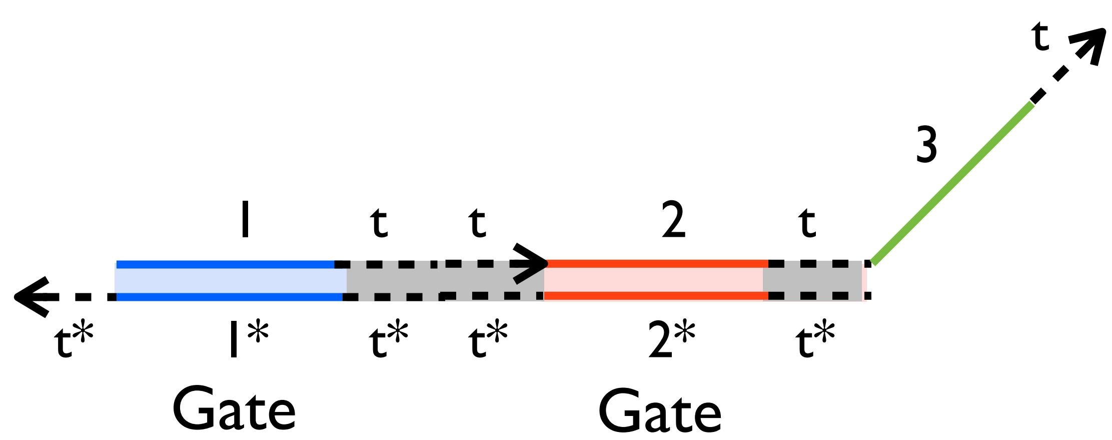
OR logic / fan-in

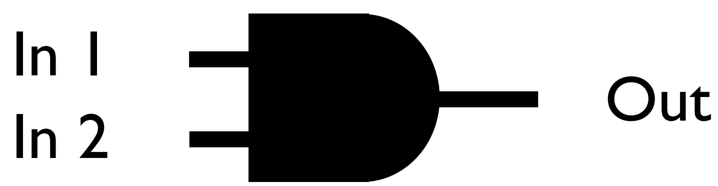


In 2	In 1	Out
0	0	0
0	1	1
1	0	1
1	1	1



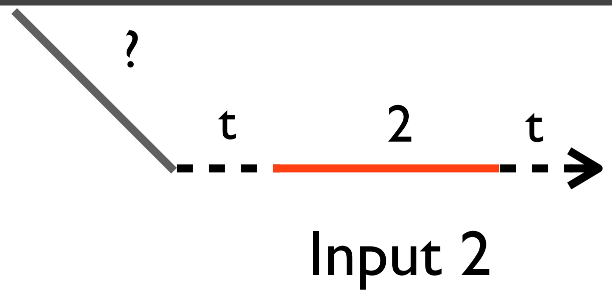
AND logic

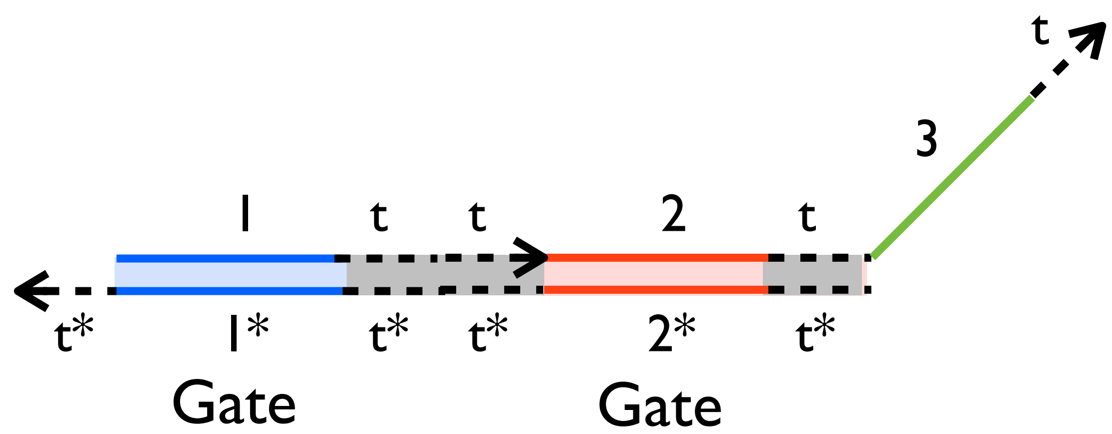


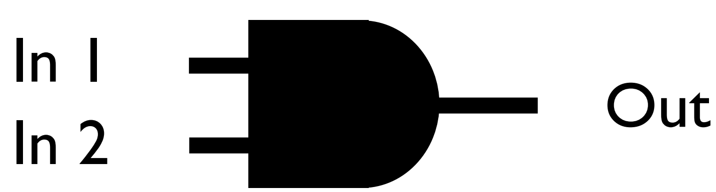


In 2	In 1	Out
0	0	0

AND logic

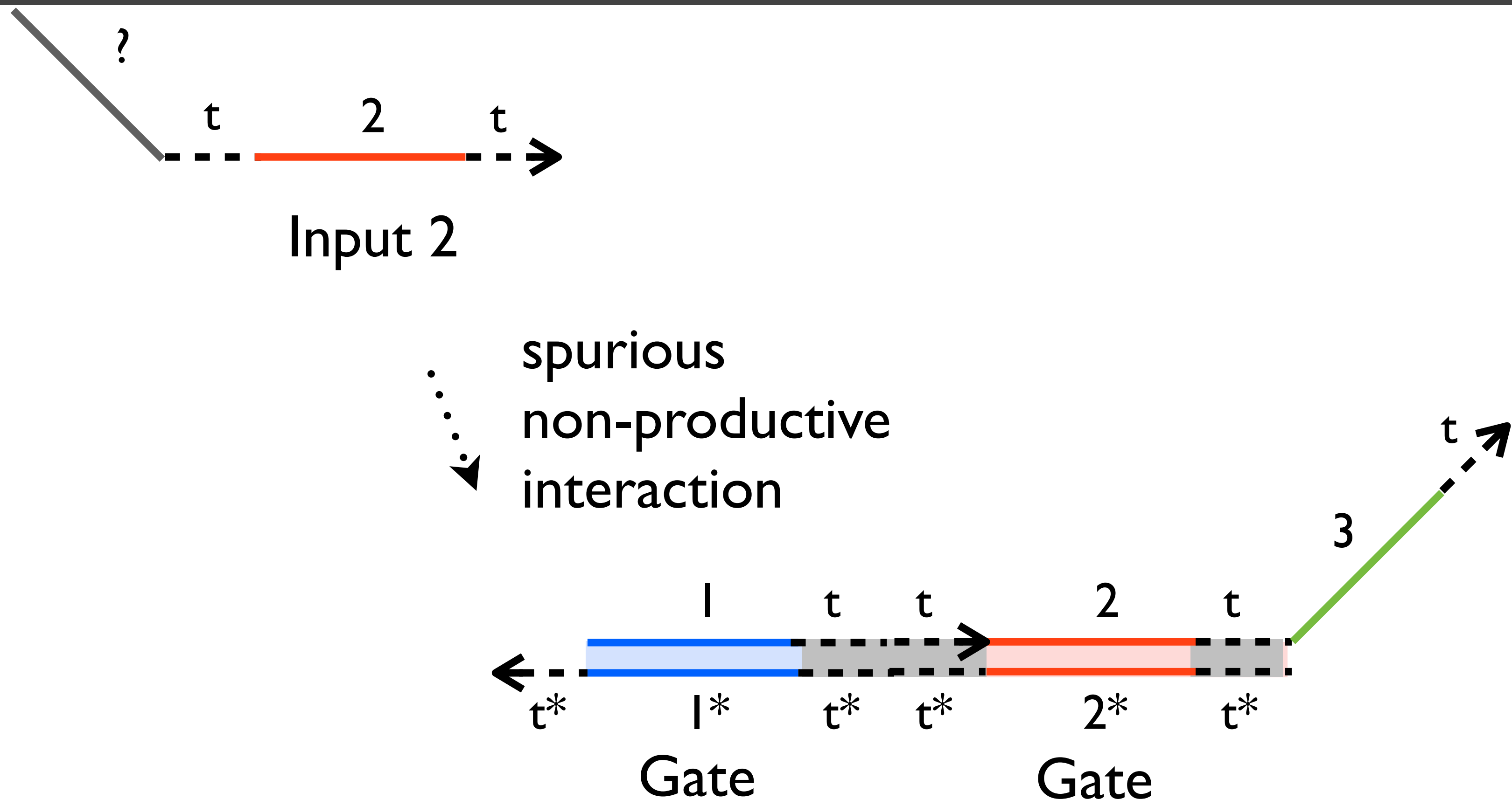




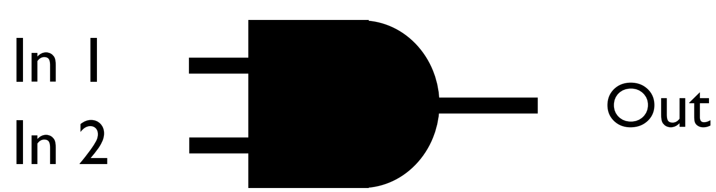


In 2	In 1	Out
0	0	0

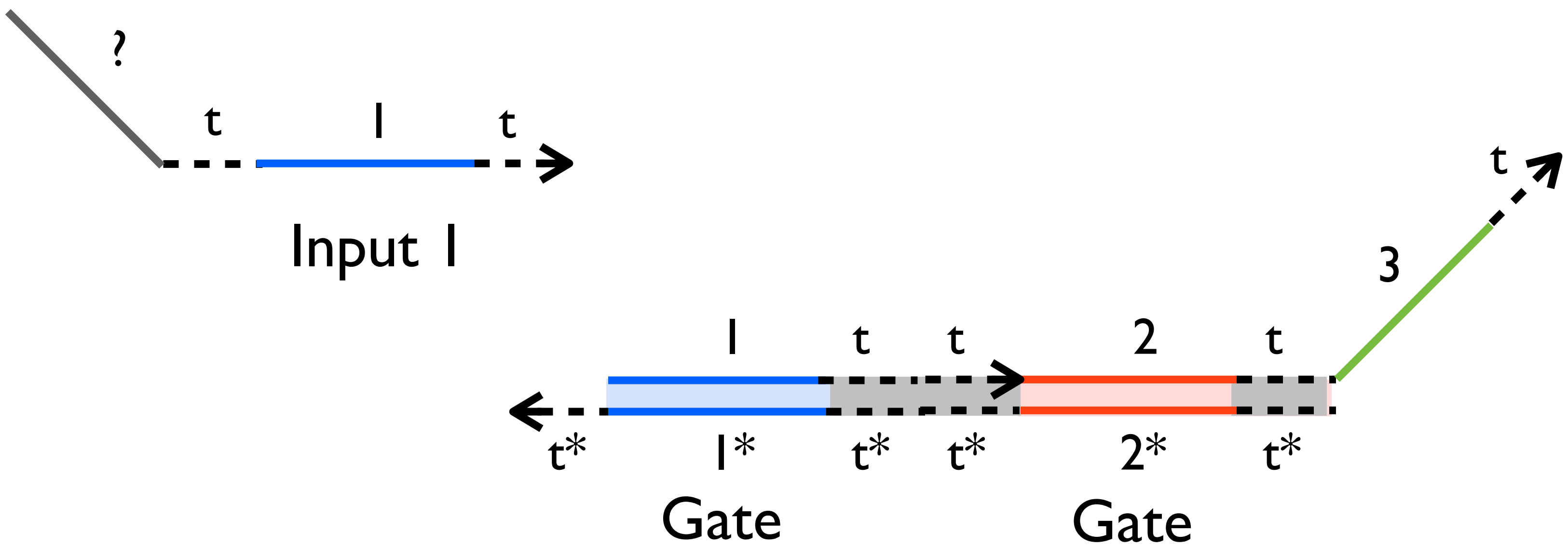
AND logic

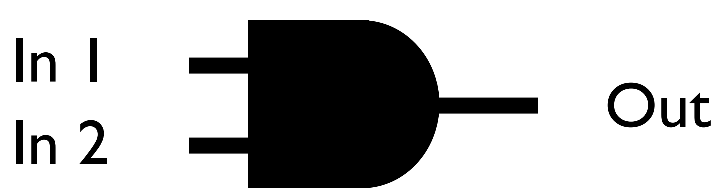


In 2	In 1	Out
0	0	0
1	0	0



AND logic

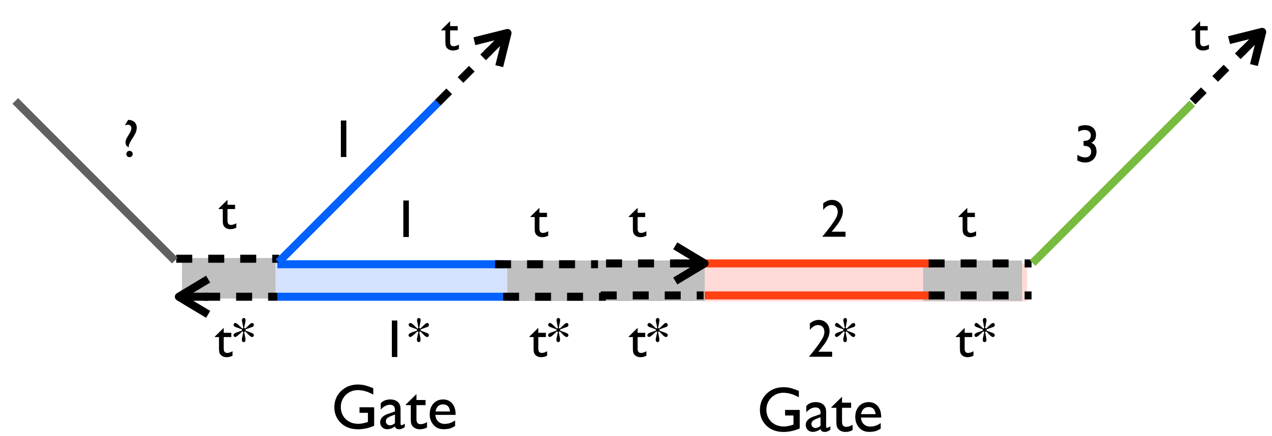




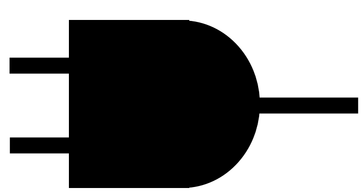
In 2	In 1	Out
0	0	0
1	0	0

AND logic

Input 1



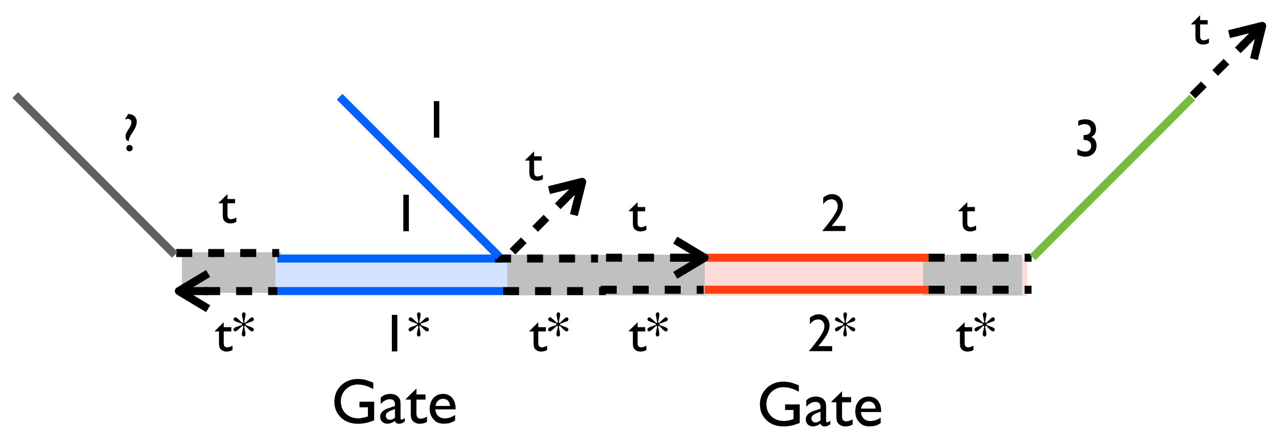
In 1
In 2

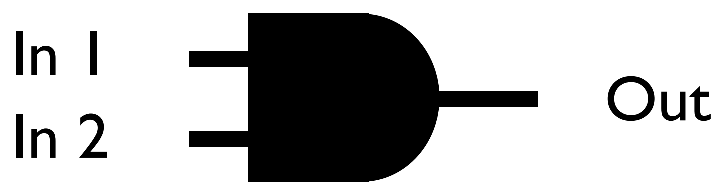


Out

In 2	In 1	Out
0	0	0
1	0	0

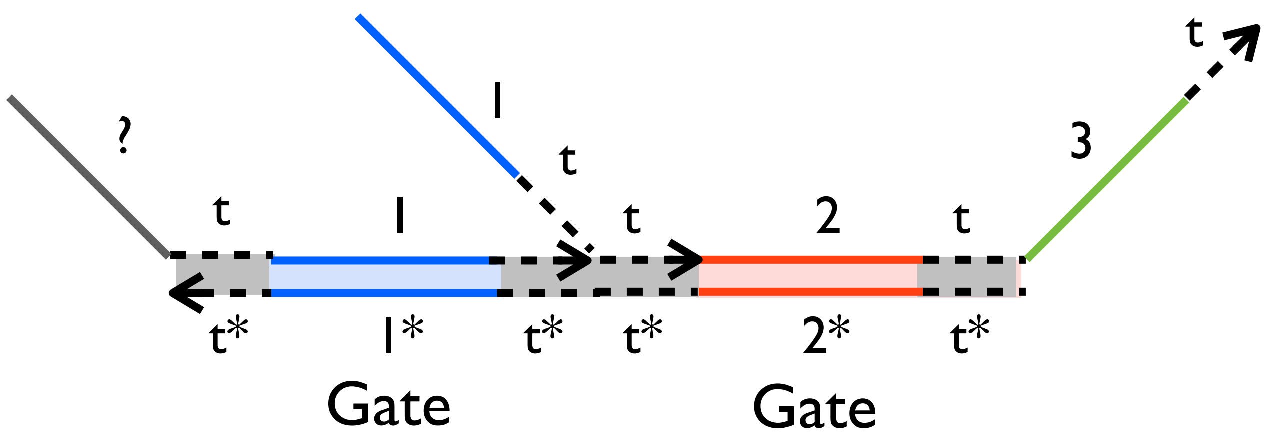
AND logic

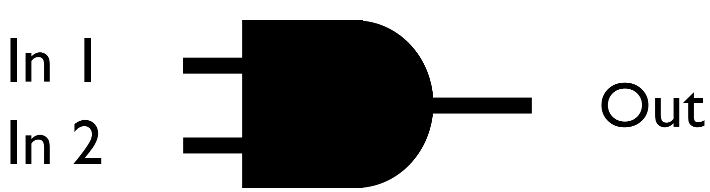




In 2	In 1	Out
0	0	0
1	0	0

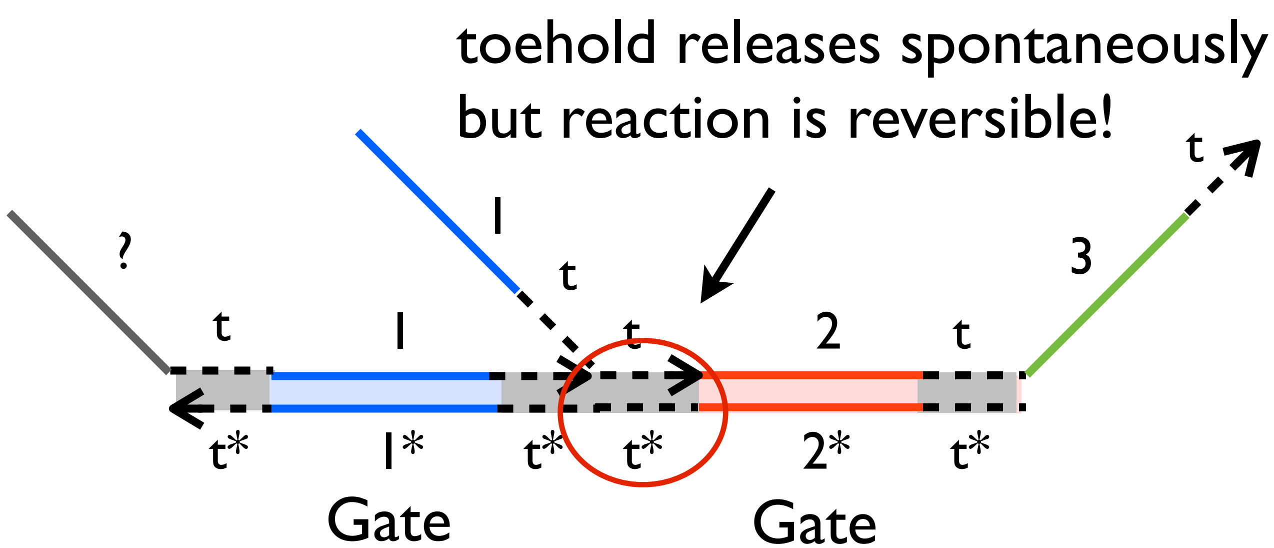
AND logic

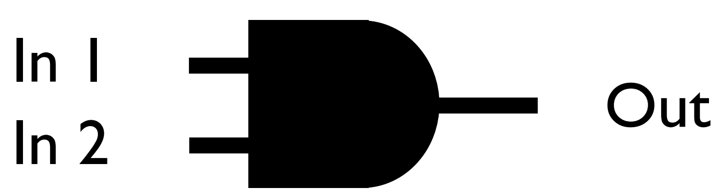




In 2	In 1	Out
0	0	0
1	0	0

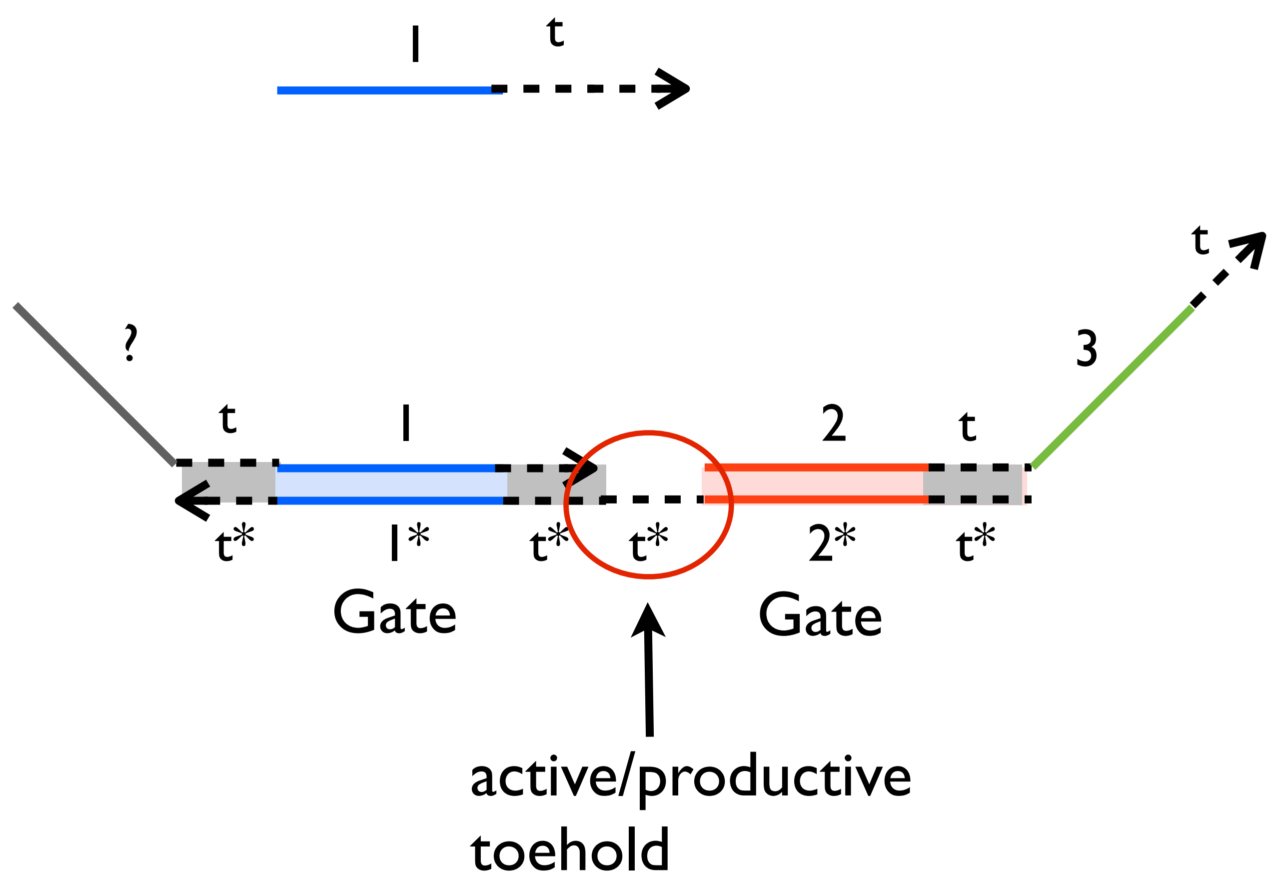
AND logic

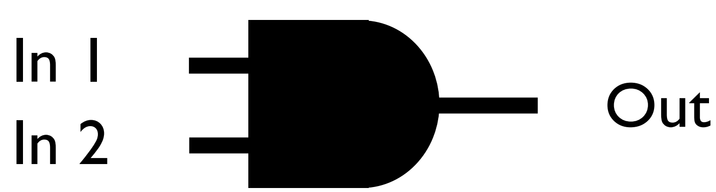




In 2	In 1	Out
0	0	0
1	0	0

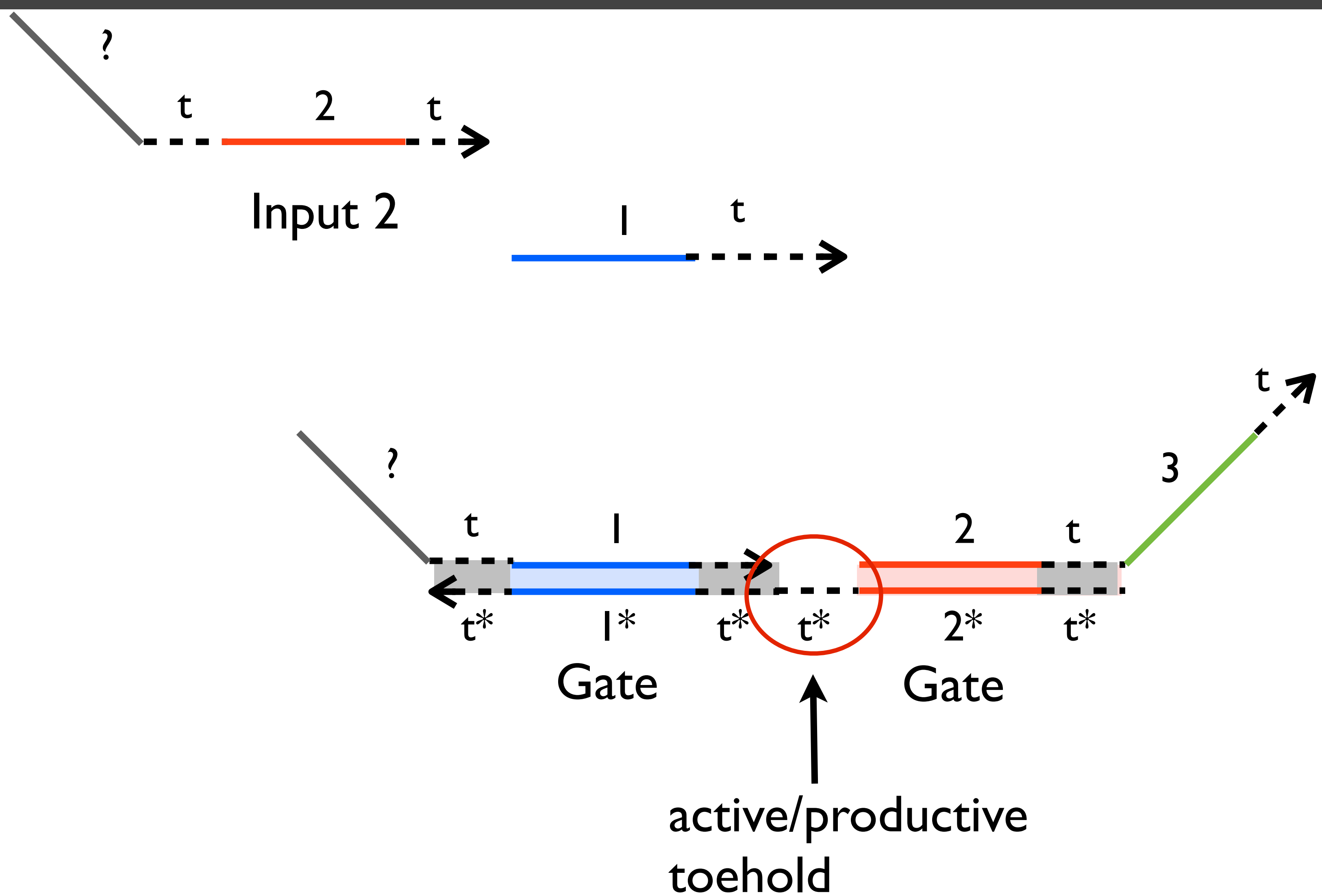
AND logic



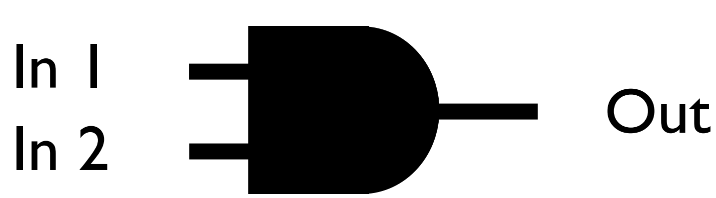


In 2	In 1	Out
0	0	0
0	1	0
1	0	0

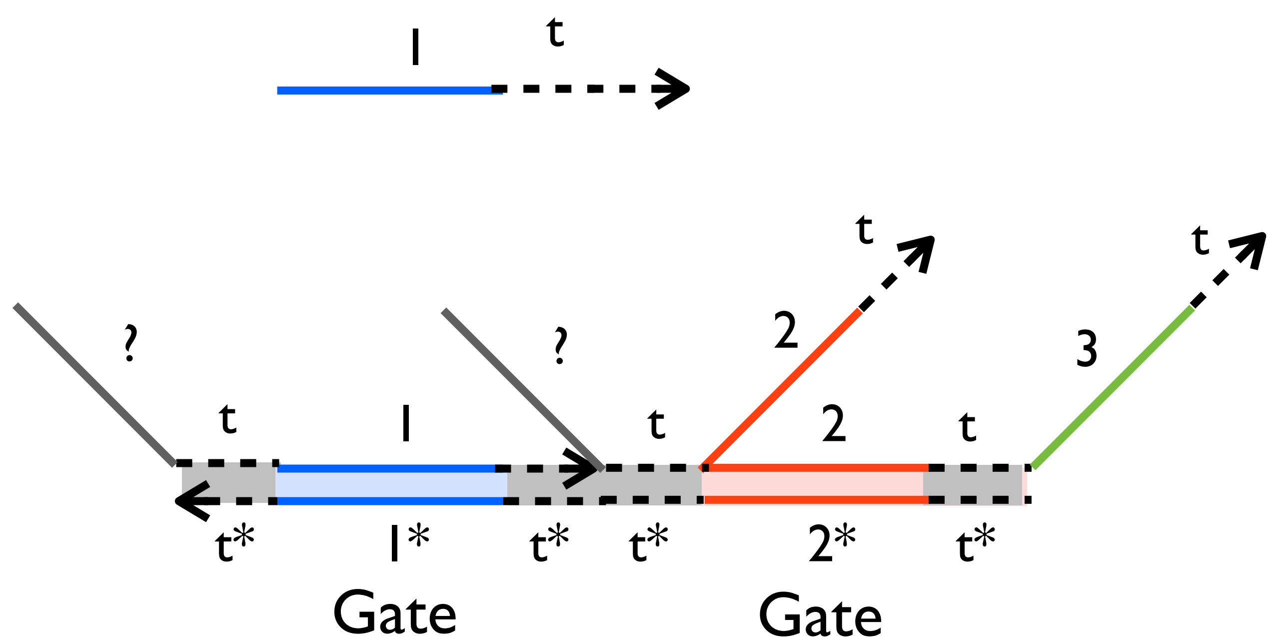
AND logic

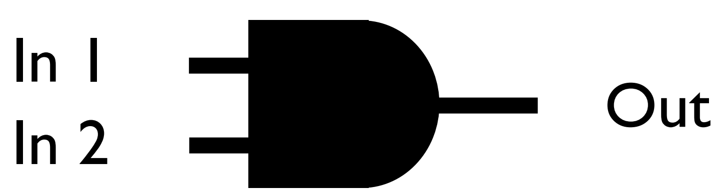


In 2	In 1	Out
0	0	0
0	1	0
1	0	0



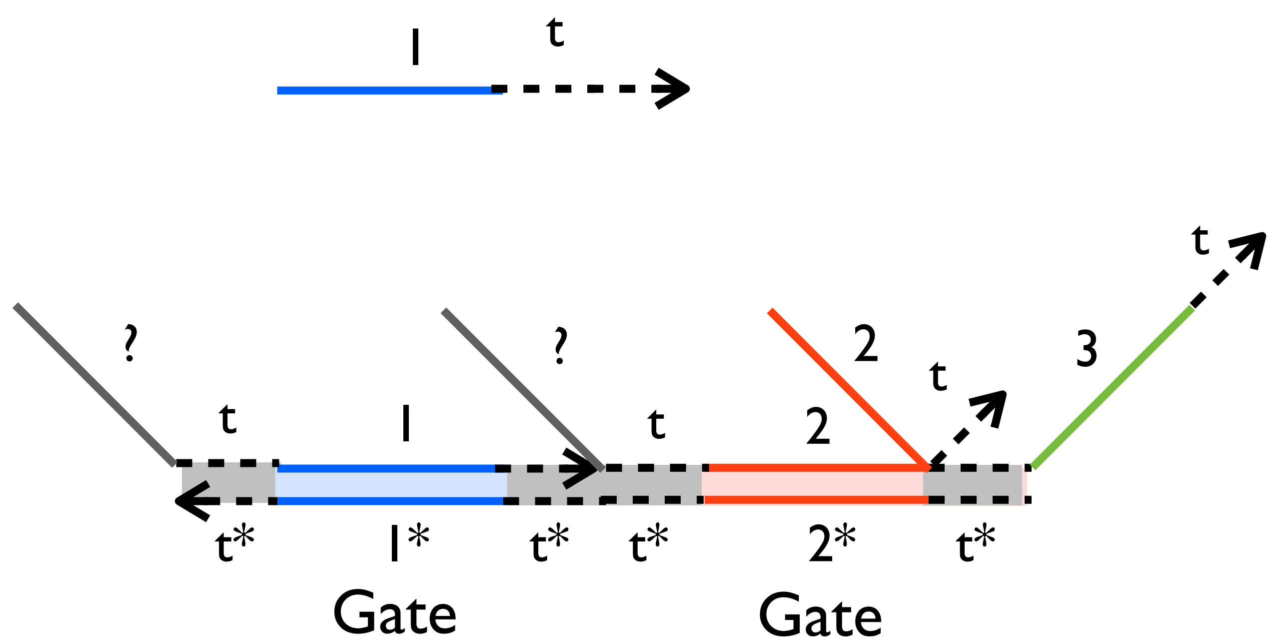
AND logic

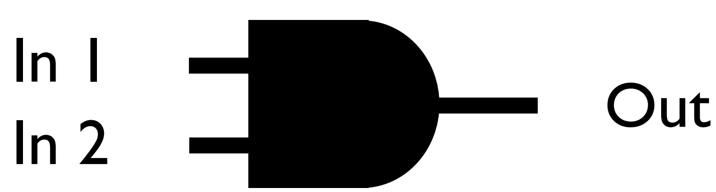




In 2	In 1	Out
0	0	0
0	1	0
1	0	0

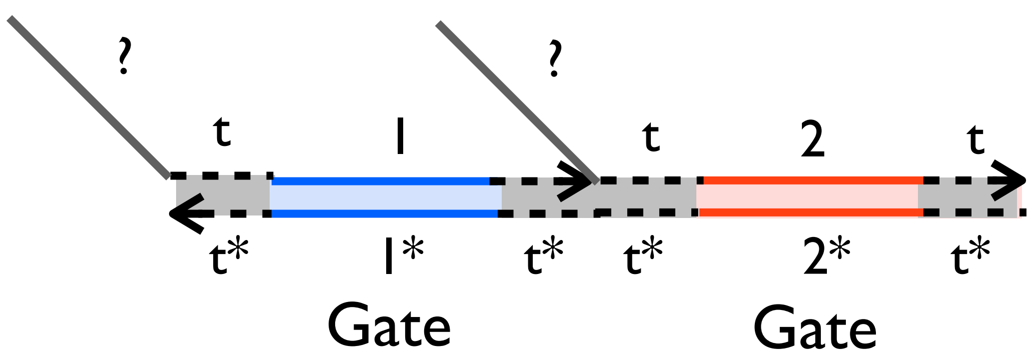
AND logic

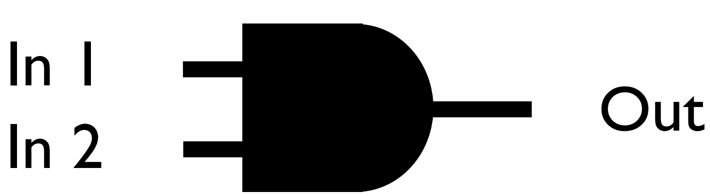




In 2	In 1	Out
0	0	0
0	1	0
1	0	0

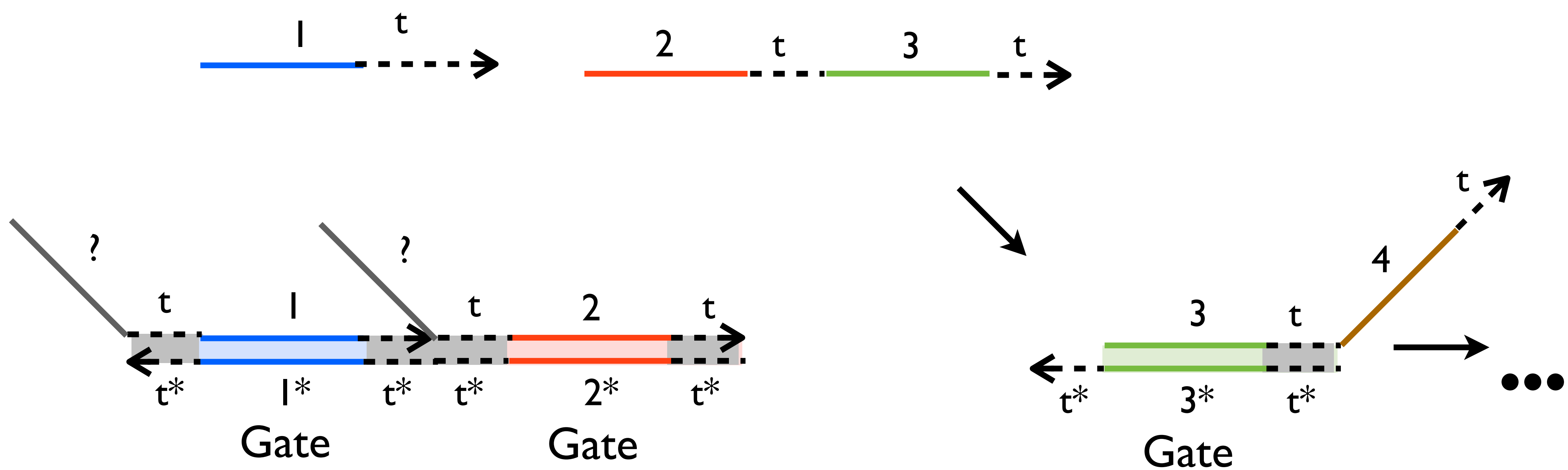
AND logic

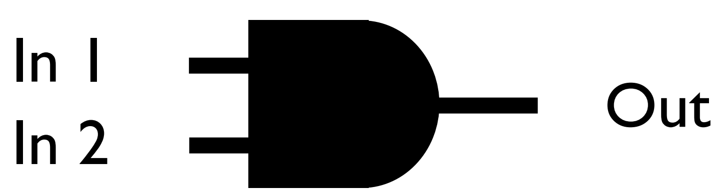




In 2	In 1	Out
0	0	0
0	1	0
1	0	0
1	1	1

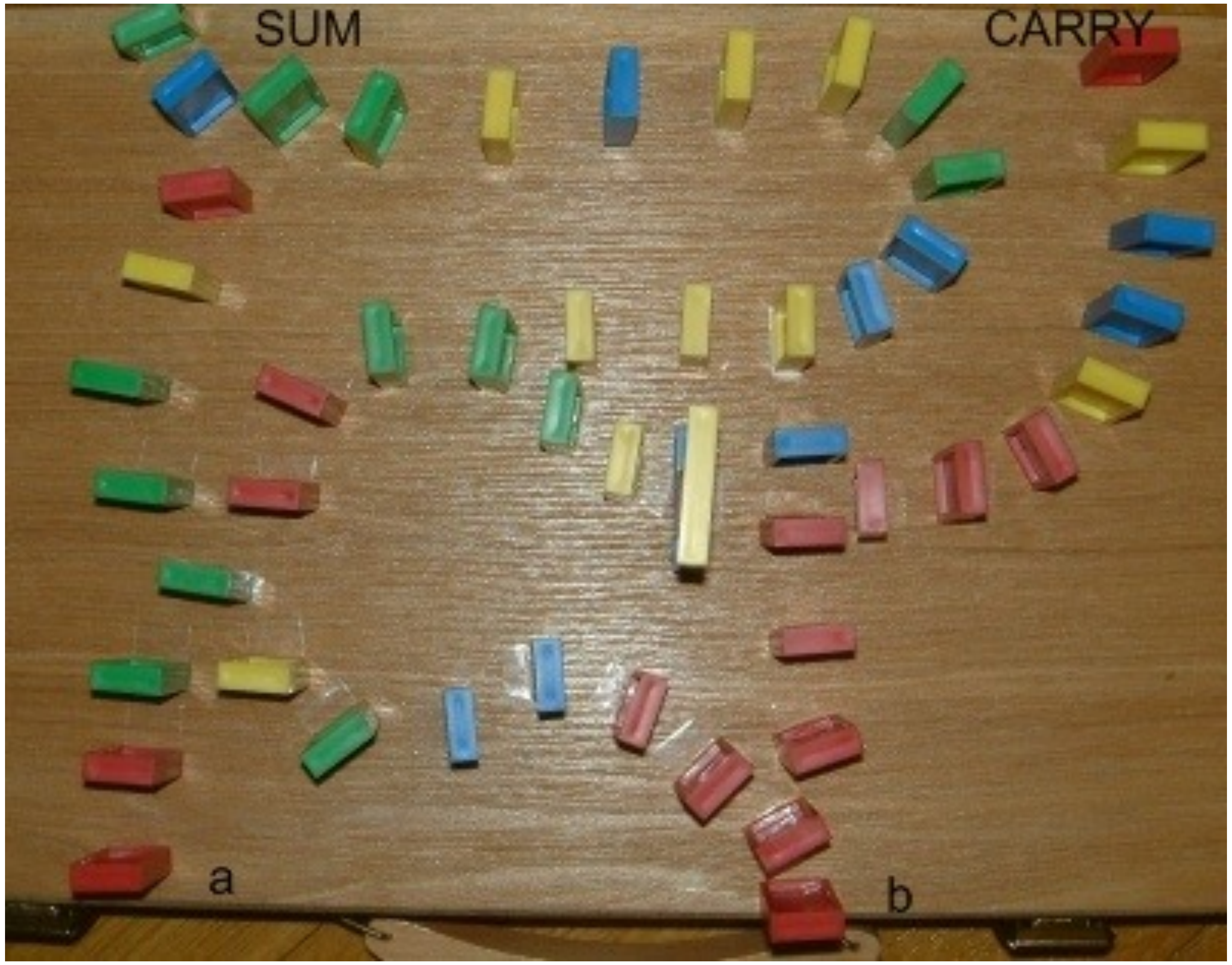
AND logic





In 2	In 1	Out
0	0	0
0	1	0
1	0	0
1	1	1

Why is NOT difficult?



Absence of a signal could be "NOT" or could simply mean that computation hasn't occured yet.

Dual-rail logic: AND and OR are sufficient for feed-forward digital circuits

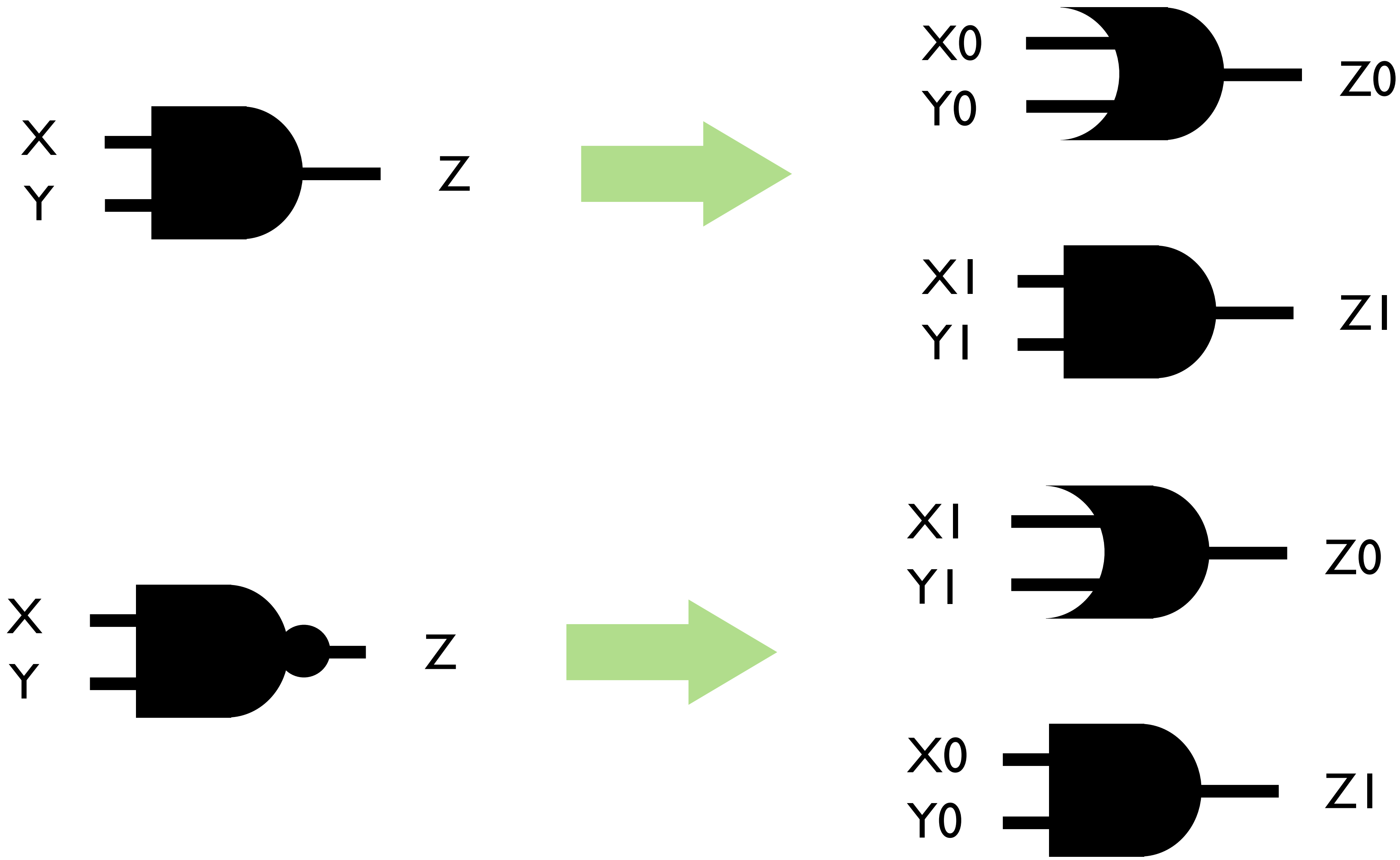
Replace X by the pair (X0,X1):

X0 on: logical "0"

X1 on: logical "1"

X0, X1 off:
not yet computed

X0, X1 on:
error



Single wire circuit using NOT, AND, OR, NAND, ... can be replaced by a dual rail representation using AND and OR only. This implementation requires maximally 2x as many gates.

Differences and similarities between electronic and molecular circuits

1. **Lack of spatial isolation:** All gates and signals diffuse in solution and interact stochastically.

2. **Computation energy and non-reusable gates:** Both inputs and gates are consumed as the circuit is evaluated by cascade reactions, so they cannot be reused.

3. **Data encoding:** Information is encoded in the sequences and concentration of biomolecules.

4. **Lack of clear hardware software separation:** Gates and circuits come pre-programmed for the specific computation they are meant to carry out.

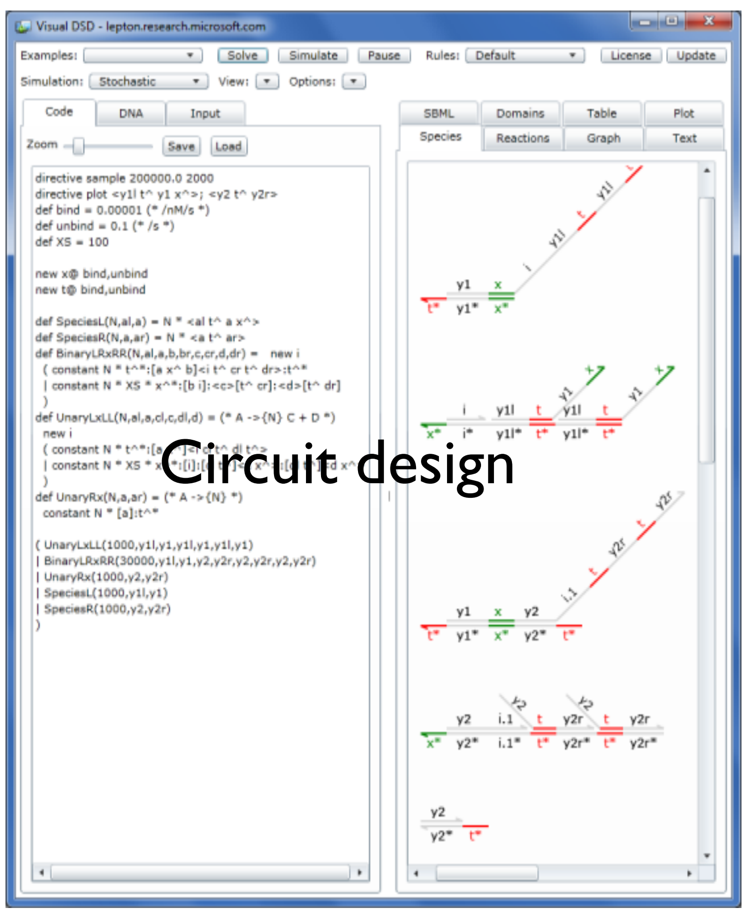
5. **Speed of computation:** A circuits evaluation under typical reaction conditions takes minutes to hours.

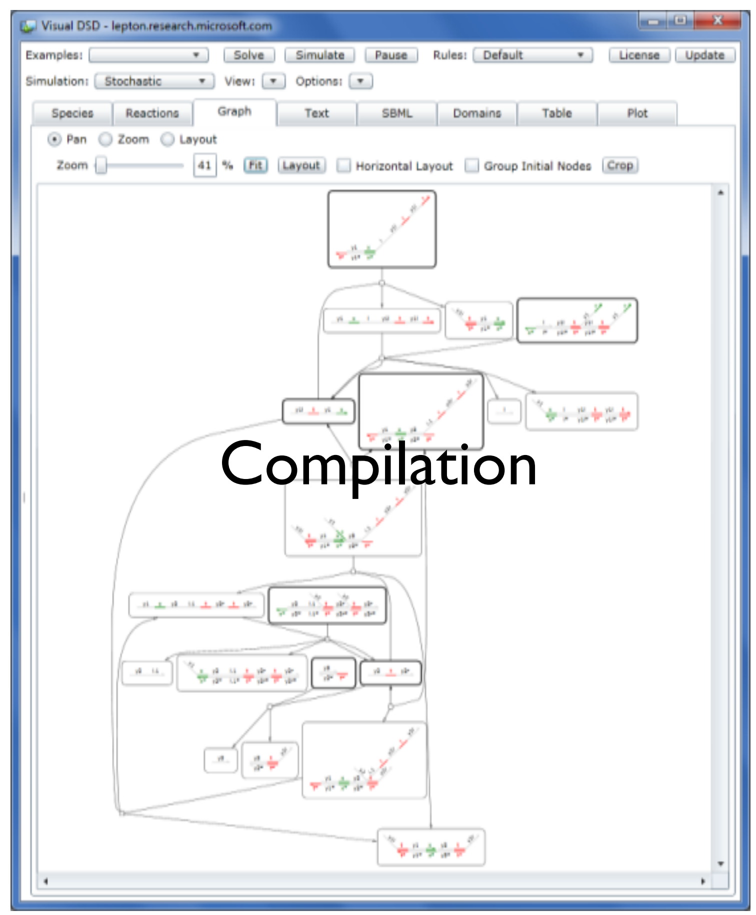
6. **Need for dual-rail logic:** NOT is difficult to implement

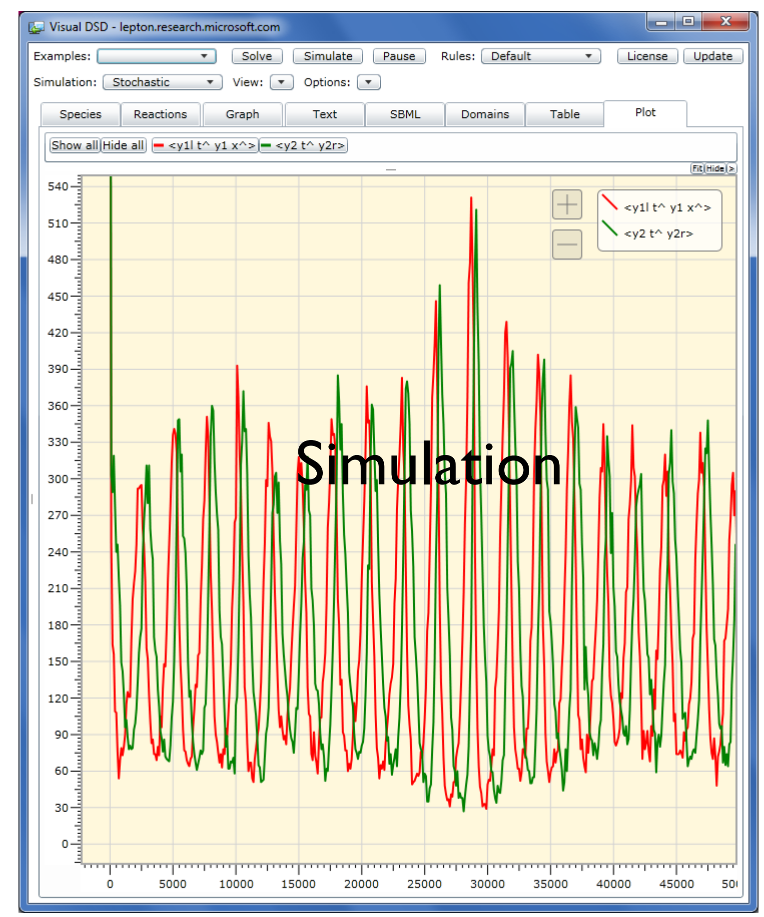
visual DSD: A tool for simulating DNA strand displacement systems

http://research.microsoft.com/en-us/projects/dna/

Use links to "web simulator" and "tutorial" for hw.



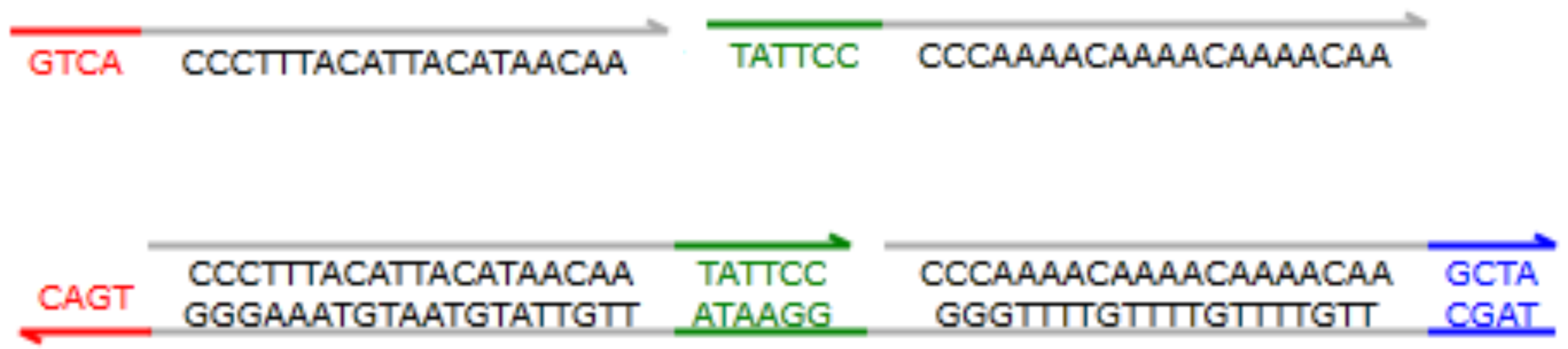




Phillips, Cardelli. Royal Society Interface, 2009

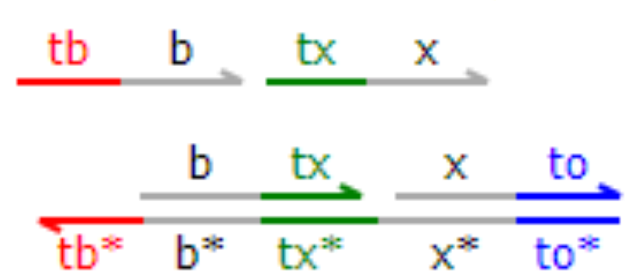
Lakin, Youssef, Polo, Emmott, Phillips. Bioinformatics, 2011

visual DSD: A tool for simulating DNA strand displacement systems



tb --> (5') TACCAA (3')
tx --> (5') TATTCC (3')
to --> (5') GTCA (3')
b --> (5') CCCTTTTCTAAACTAAACAA (3')
x --> (5') CCCAAAACAAAACAAAACAA (3')

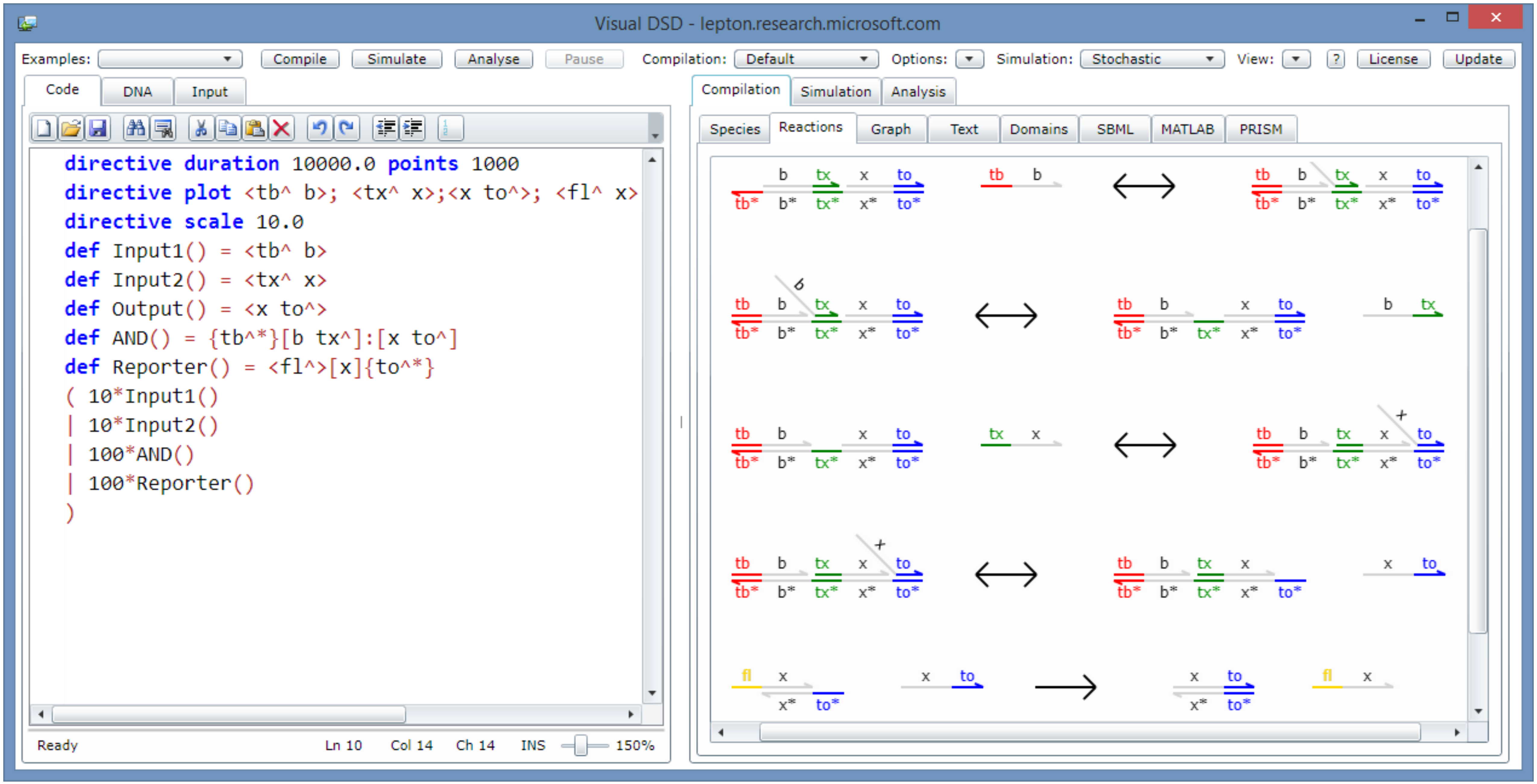
```
<tb^ b> <tx^ x>
```




```
{tb^*}[b tx^]:[x to^]
```

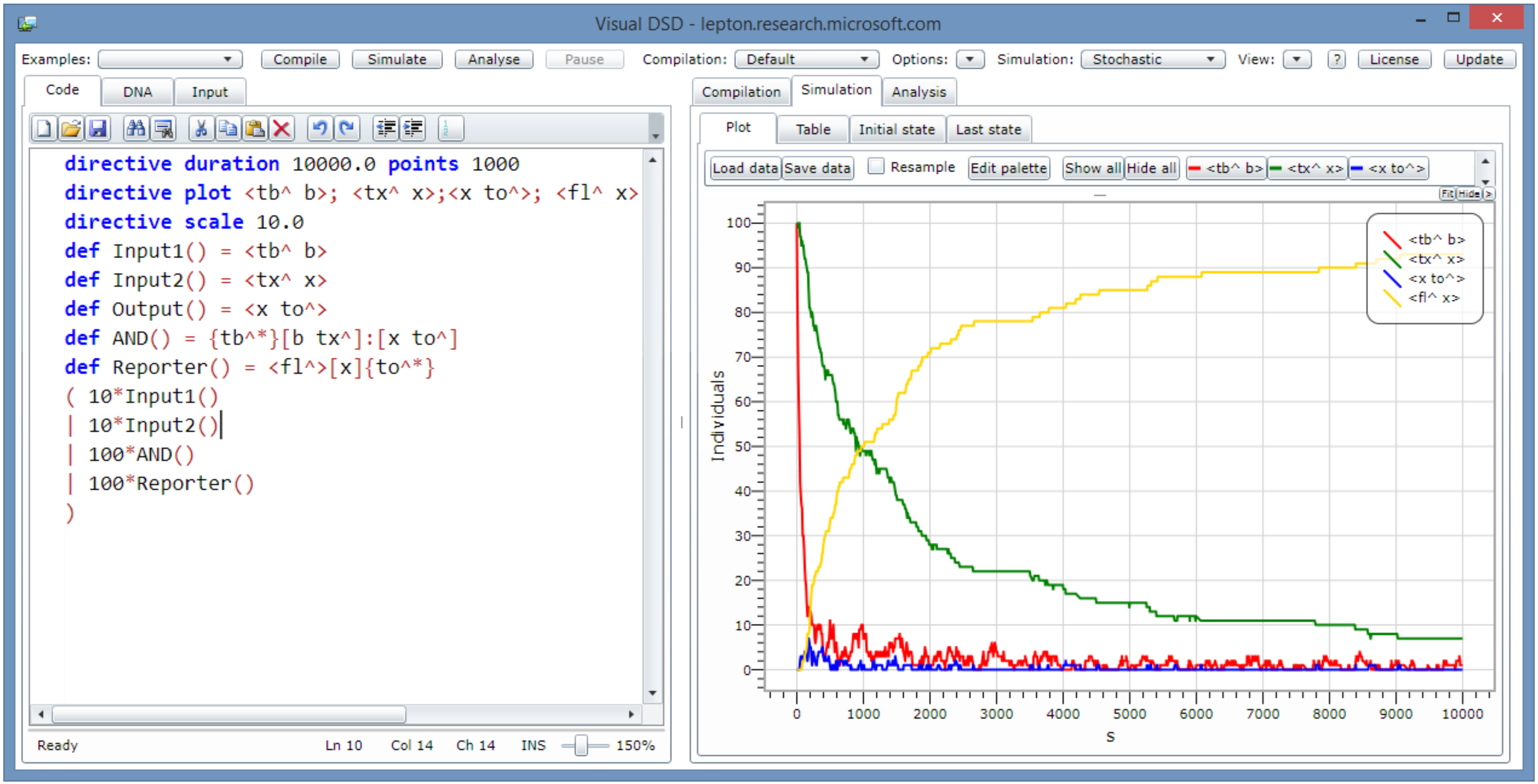

Slide credit: Andrew Phillips (MSR)

visual DSD: A tool for simulating DNA strand displacement systems



Slide credit: Andrew Phillips (MSR)

visual DSD: A tool for simulating DNA strand displacement systems



Slide credit: Andrew Phillips (MSR)

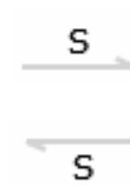
visual DSD: A tool for simulating DNA strand displacement systems

Strand::=

{S} and <S> strands:

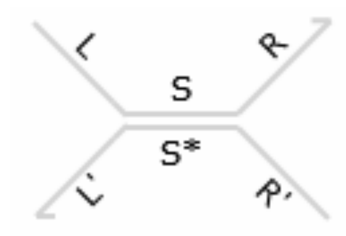
<S> — Upper strand

{S} — Lower strand

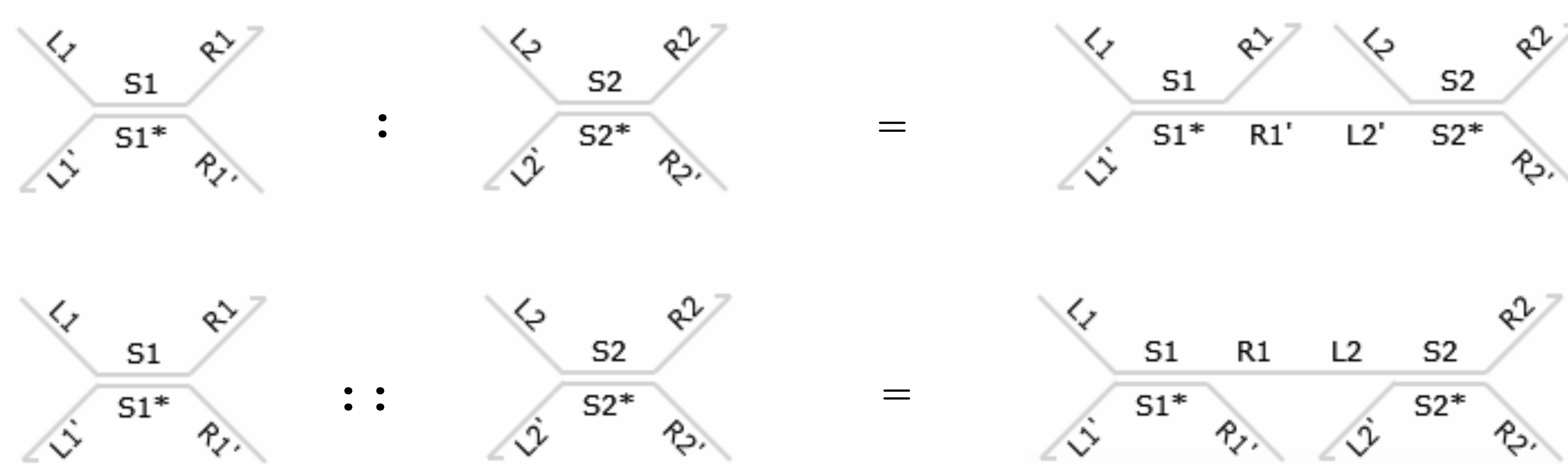


Segment::=

{L′}<L>[S]<R>{R′} — Double stranded complex with overhangs

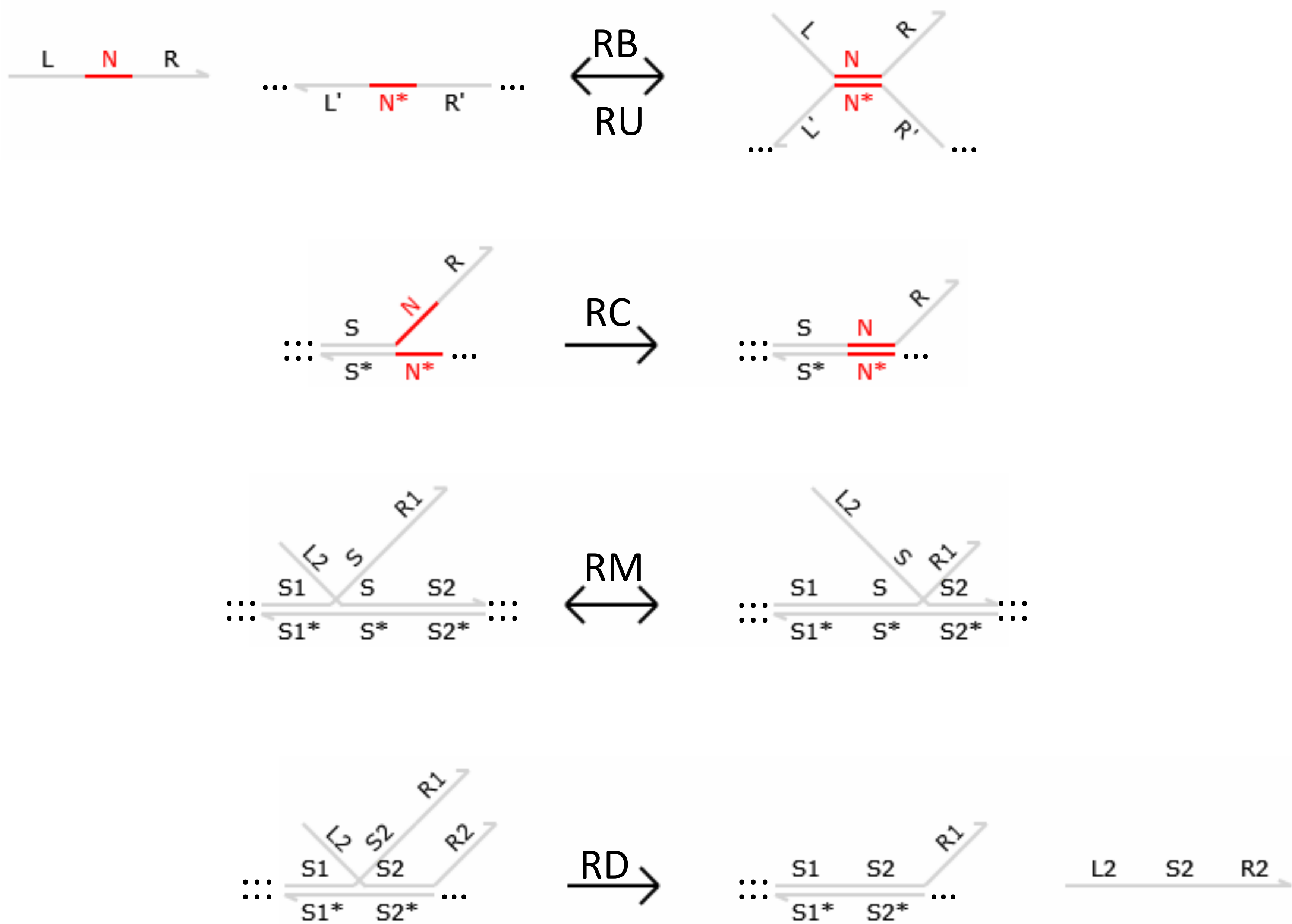


Segment concatenation



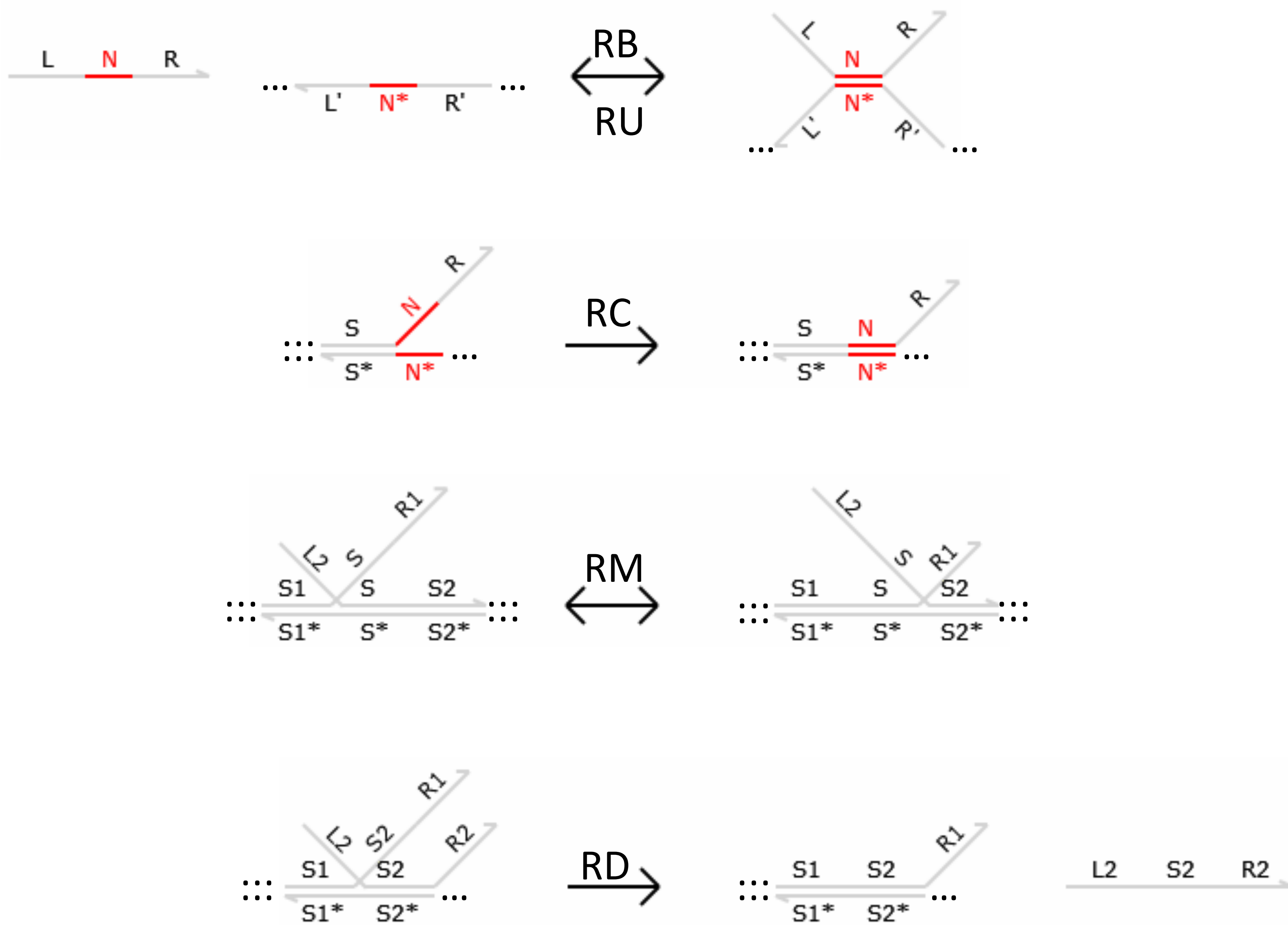
Slide credit: Andrew Phillips (MSR)

visual DSD: Syntax of strands and complexes



Slide credit: Andrew Phillips (MSR)

visual DSD: Reduction rules



Slide credit: Andrew Phillips (MSR)