

Strand Algebras for DNA Computing

Luca Cardelli

Microsoft Research

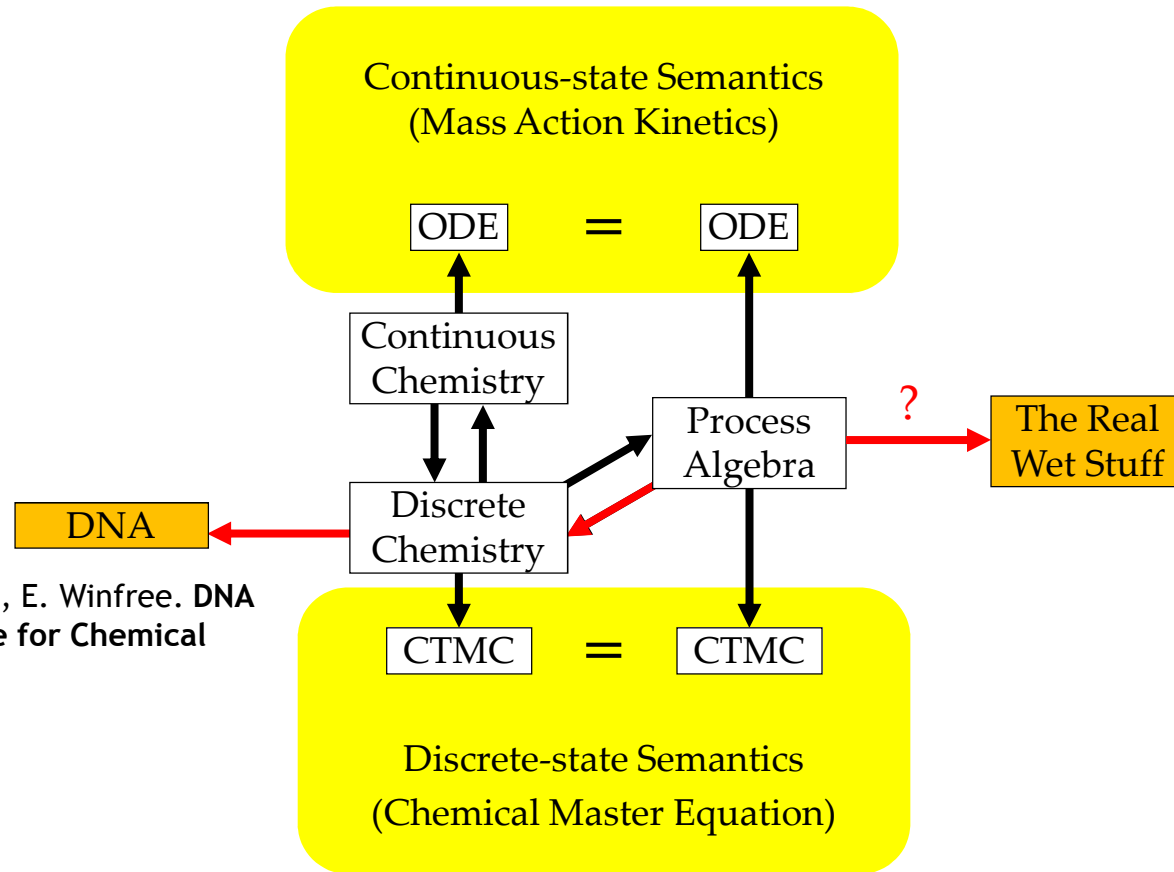
DNA 15 Conference
Fayetteville, 2009-06-10

<http://lucacardelli.name>

How Process Algebra fits in DNA Computing

- Electronics has *electrons*
 - All electrons are the same
 - All you can do is see if you have *few* ('False') or *lots* ('True') of electrons
 - Hence Boolean logic is at the basis of digital circuit design
 - Symbolic and numeric computation has to be encoded above that
 - But mostly we want to compute with symbols and numbers, not with Booleans
- DNA computing has *symbols* (DNA words)
 - DNA words are not all the same
 - Symbolic computation can be done *directly*
 - We can also directly use molecular concurrency
- Process Algebra as the 'Boolean Algebra' of DNA Computing
 - What are the 'gates' of symbolic concurrent computation?
 - That's what Process Algebra is about
 - (Process Algebra comes from the theory of concurrent systems)

Molecules as Automata (DNA14 Invited Talk)



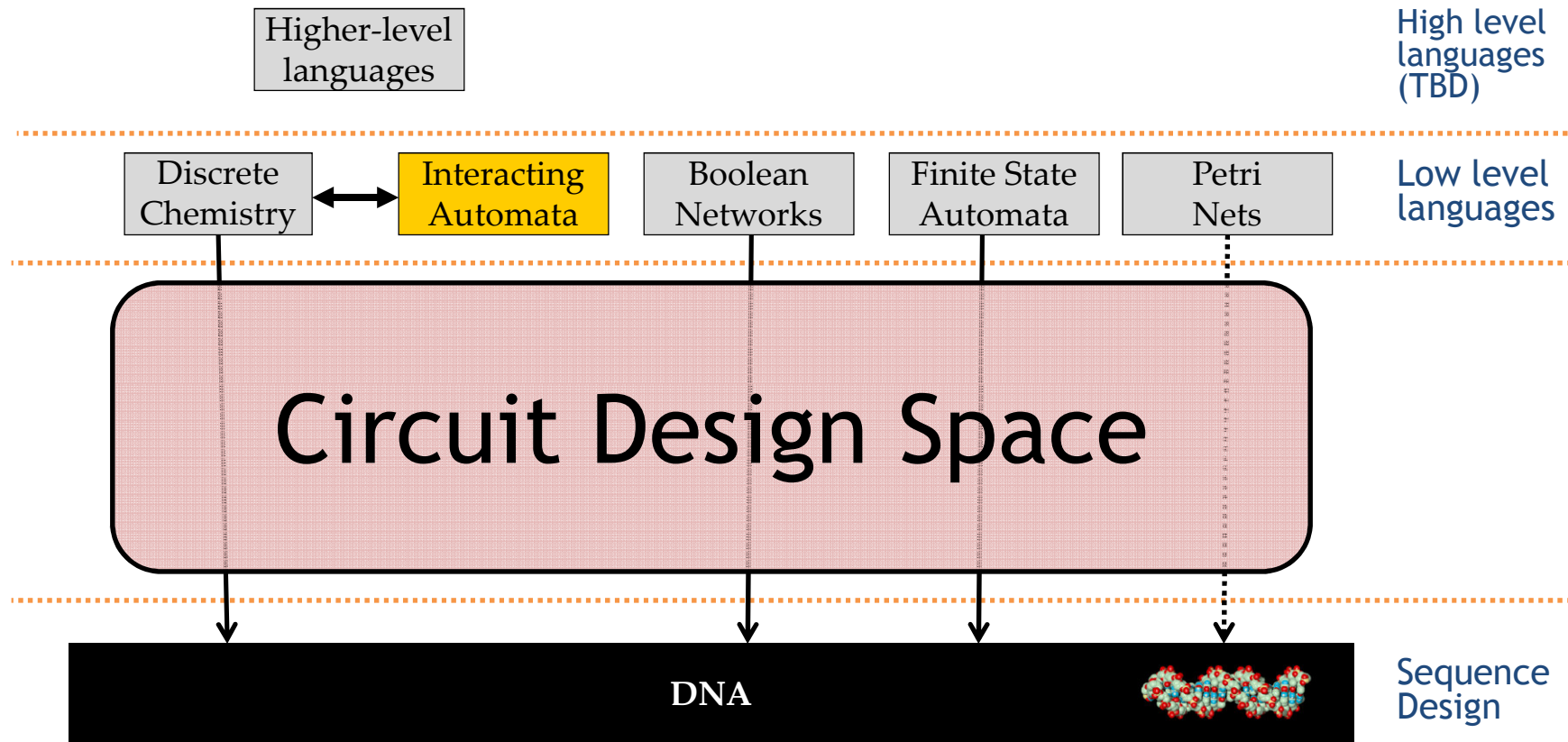
D. Soloveichik, G. Seelig, E. Winfree. **DNA as a Universal Substrate for Chemical Kinetics**. Proc. DNA14.

L. Cardelli: “On Process Rate Semantics” (TCS)

L. Cardelli: “A Process Algebra Master Equation” (QEST’07)

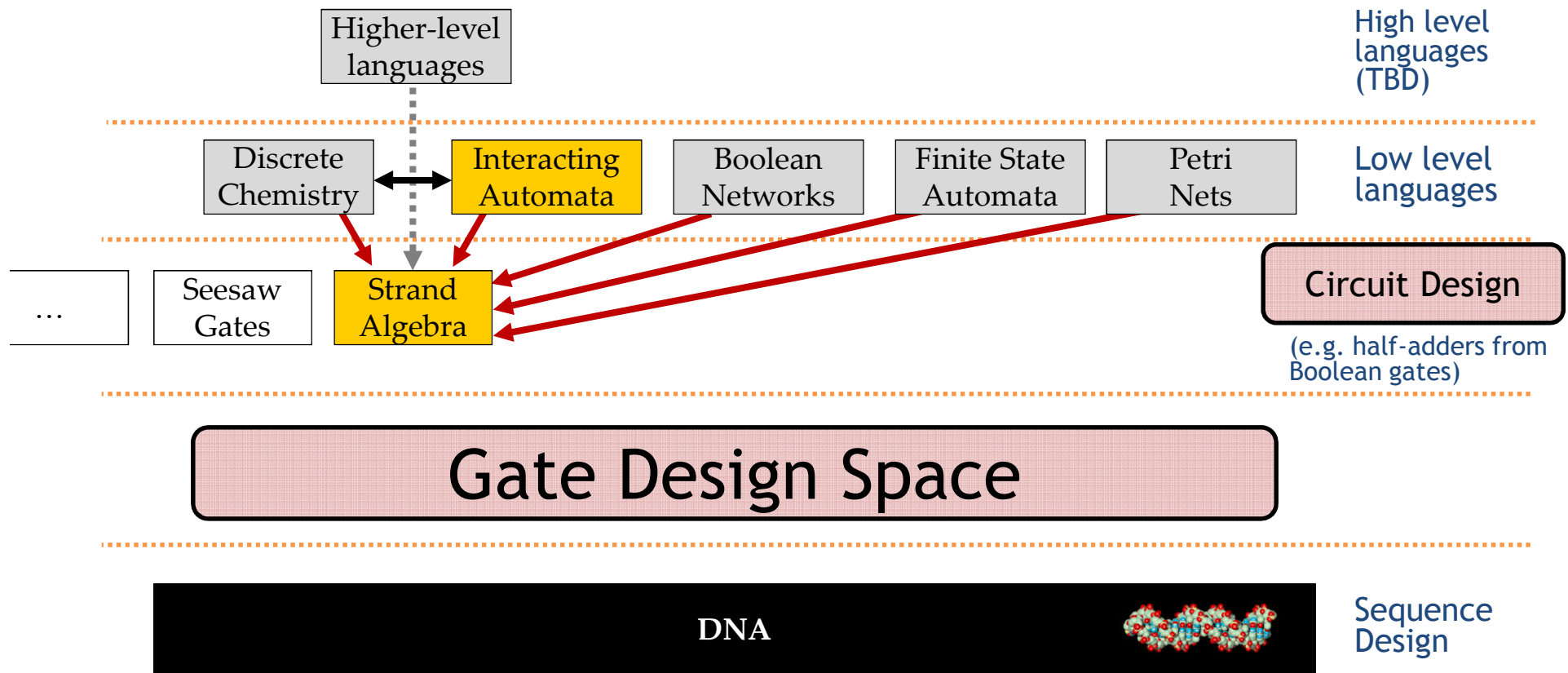
DNA Compilation

Separating Circuit Design from Gate Design



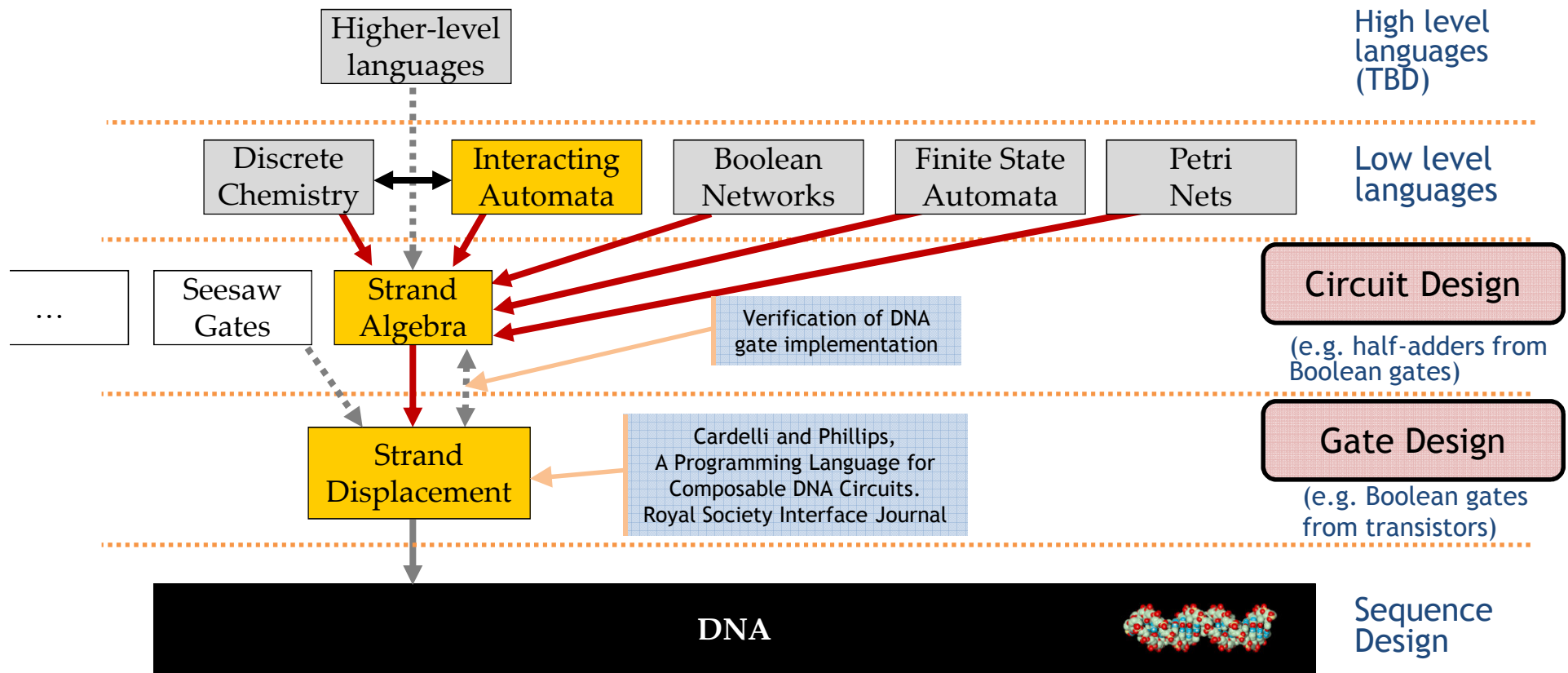
DNA Compilation

Separating Circuit Design from Gate Design



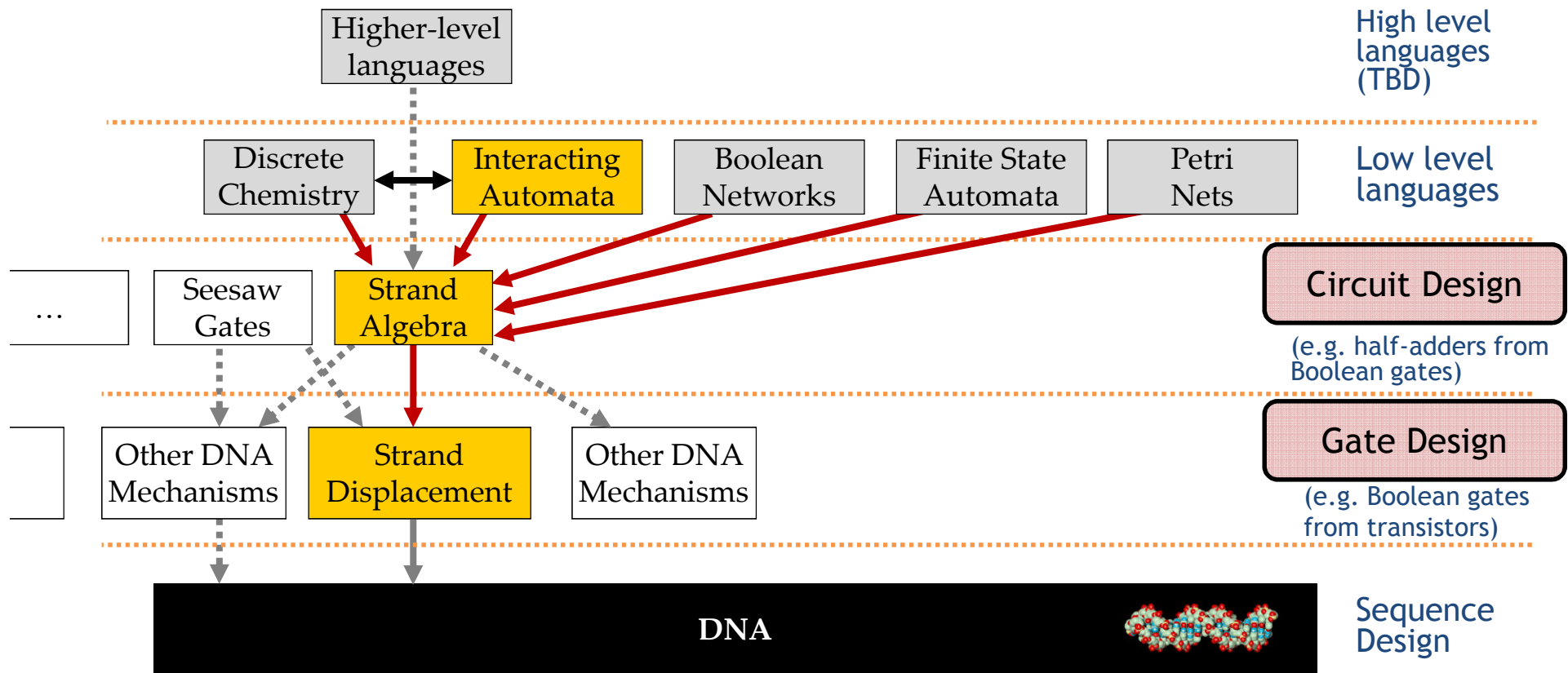
DNA Compilation

Separating Circuit Design from Gate Design



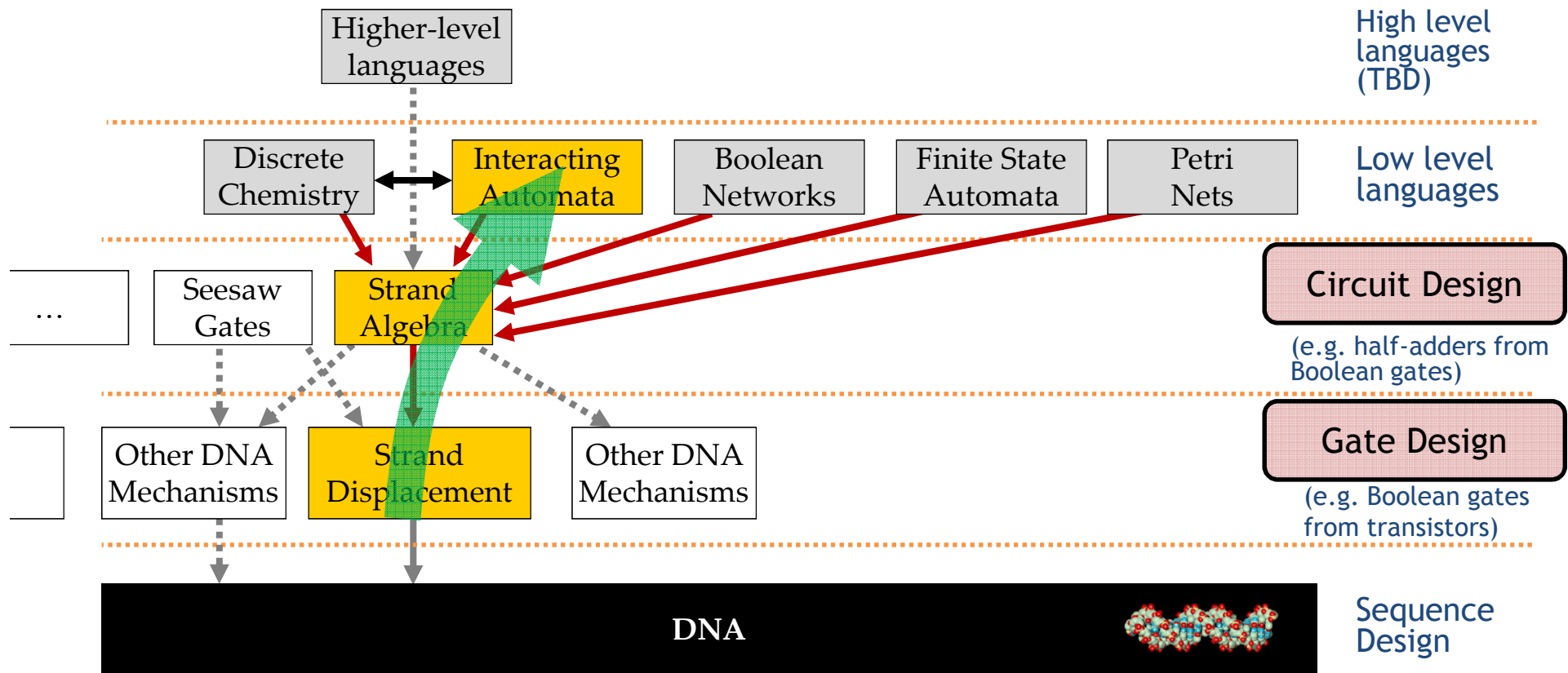
DNA Compilation

Separating Circuit Design from Gate Design



DNA Compilation

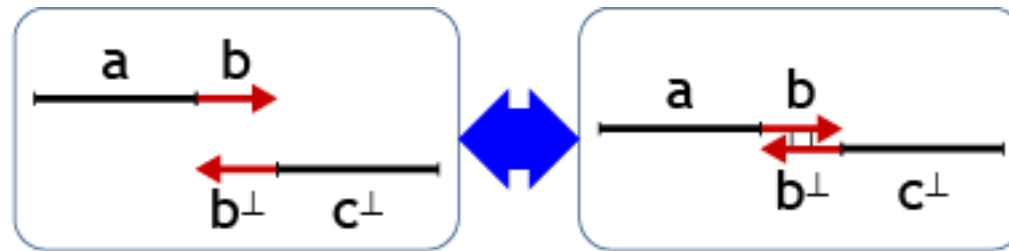
Separating Circuit Design from Gate Design



Rest of the talk: bottom up

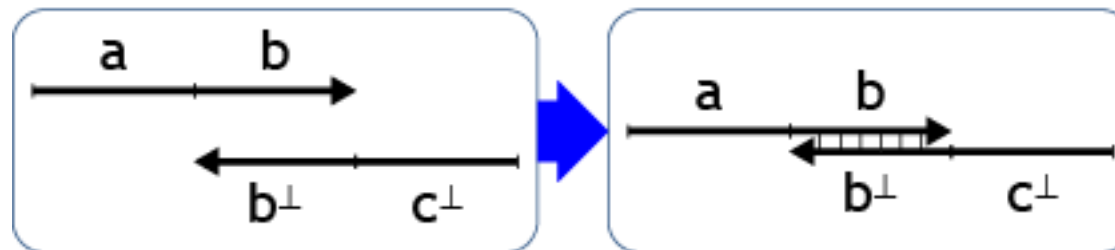
Gate Elements: Short and Long DNA Segments

Short (red)
segments



Reversible Binding

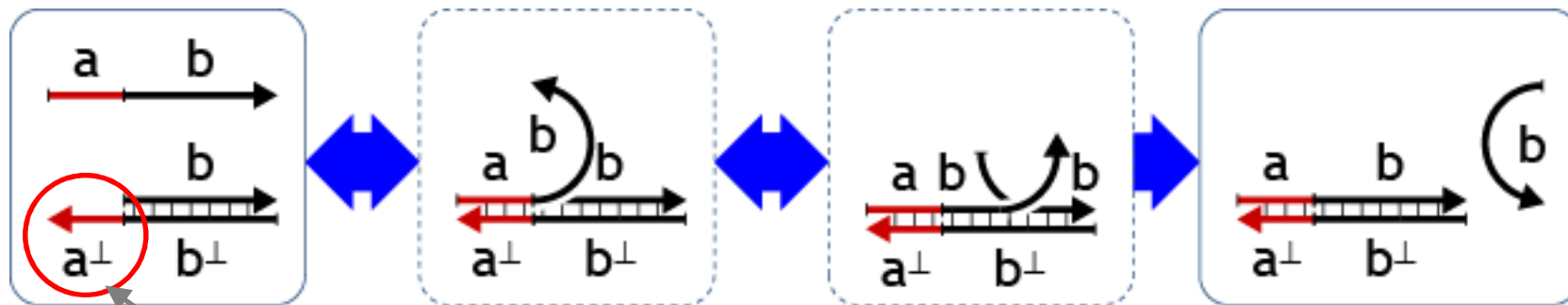
Long (black)
segments



Irreversible Binding

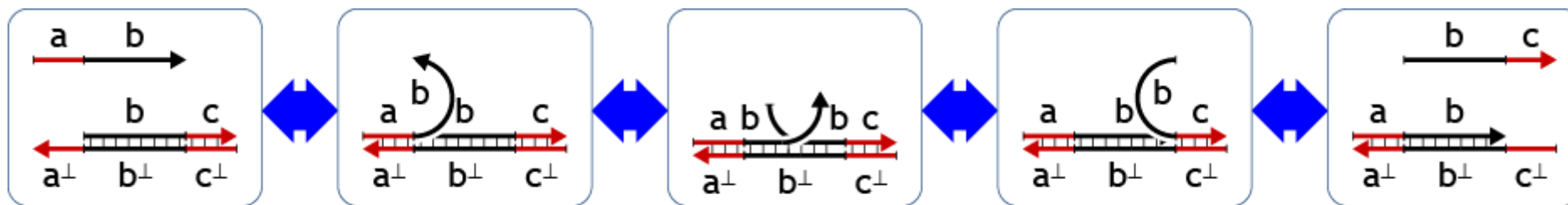
Gate Elements: Basic Mechanisms

Irreversible



Strand Displacement

Reversible



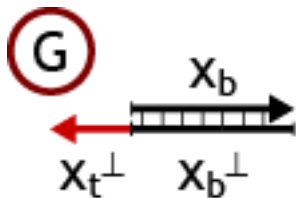
Toehold Exchange

Gate Elements: Signals and Gates

- Signals “x” are single-stranded and ‘positive’



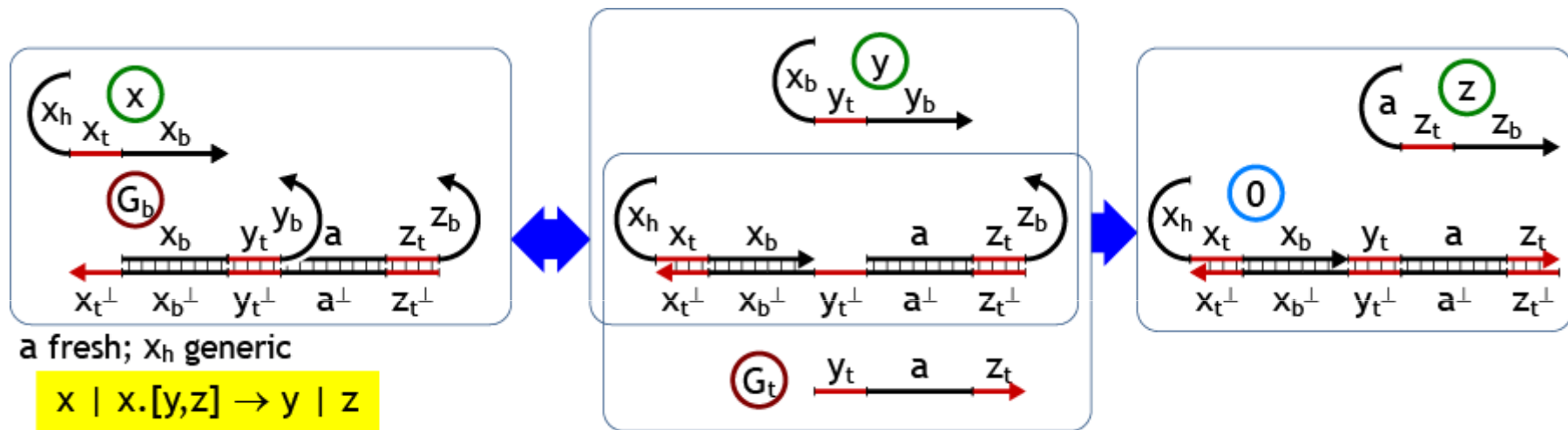
- This 3-segment signal representation is original to this work, it is based on the 4-segment signals of D. Soloveichik, G. Seelig, E. Winfree. Proc. DNA14, but leads to simpler and more regular gate structures
- Gate backbones are double-stranded, except for ‘negative’ toeholds.



- Separation of strands and gates helps the DNA realization, as one can use 3-letter alphabets (ATC/ATG) for each strand, minimizing secondary structure and entanglement.

Circuit Elements: $x.[y,z]$ Fork Gate

- A **Fork** signal-processing gate takes a signal x and produces two signals y,z according to the reaction $x \mid x.[y,z] \rightarrow y \mid z$



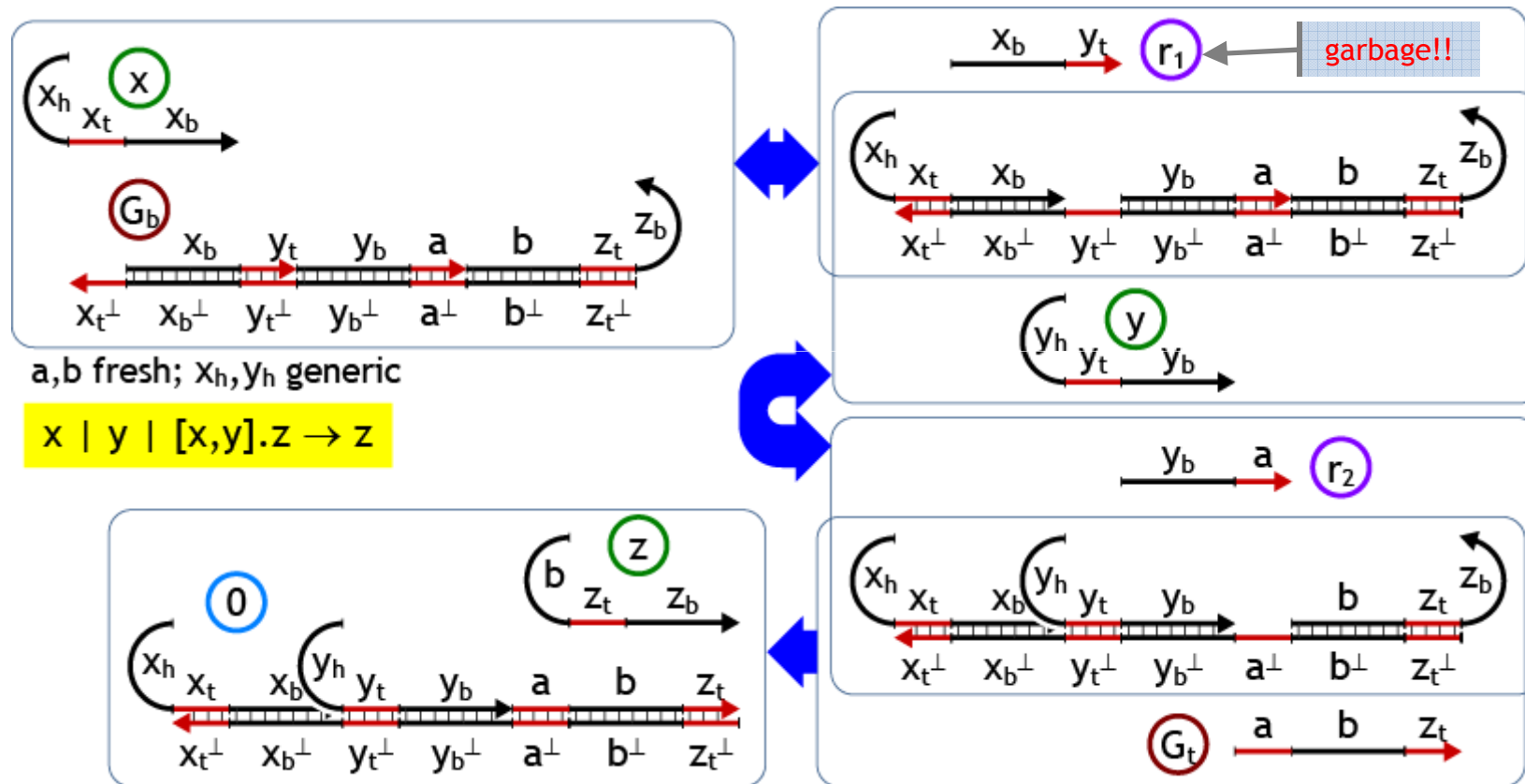
G_b, G_t (gate backbone and trigger) form the gate.

Any history segment that is not determined by the gate structure is said to be 'generic' (can be anything).

Any gate segment that is not a non-history segment of an input or output signal is taken to be 'fresh' (globally unique for the gate), to avoid possible interferences.

Circuit Elements: $[x,y].z$ Join Gate (function)

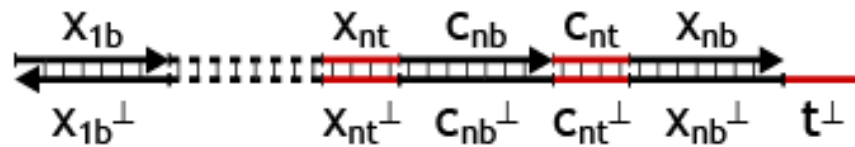
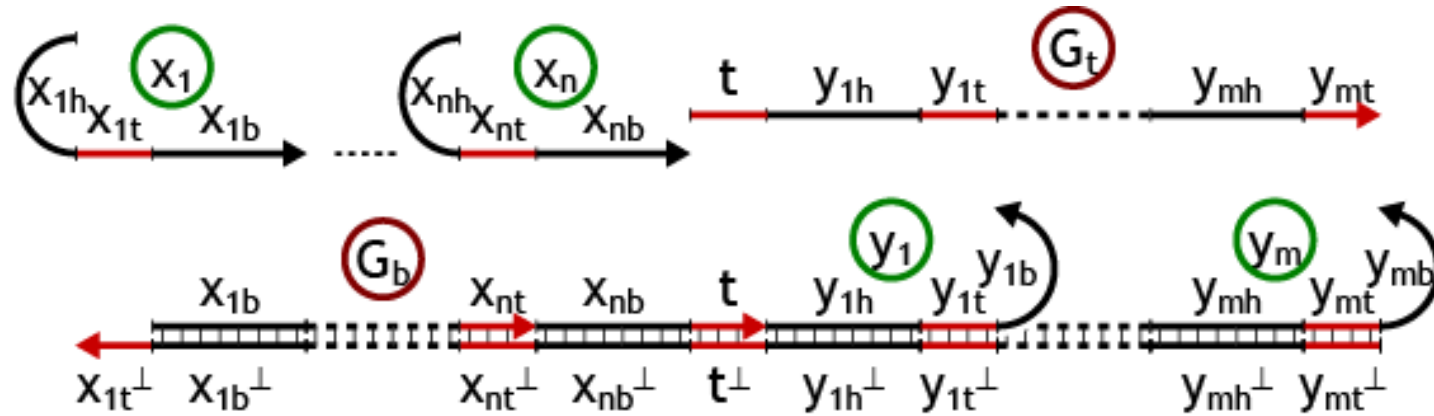
- A **Join** signal-processing gate takes *both* signals x,y and produces a signal z according to the reaction $x \mid y \mid [x,y].z \rightarrow z$



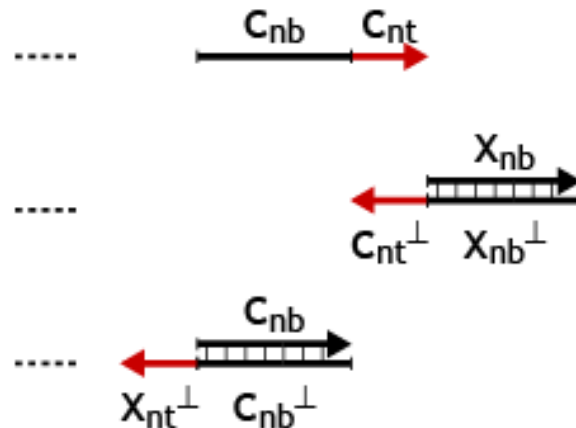
The garbage r_1 and r_2 must be collected (*after* the gate has fired) to avoid accumulation. This can be achieved by a similar scheme taking r_1, r_2 as input signals.

$[x_1, \dots, x_n] \cdot [y_1, \dots, y_m]$ General Join/Fork Gate

$$x_1 \mid \dots \mid x_n \mid [x_1, \dots, x_n] \cdot [y_1, \dots, y_m] \rightarrow y_1 \mid \dots \mid y_m$$



Garbage collection



x_{1h}, \dots, x_{nh} generic
 t, y_{1h}, \dots, y_{nh} fresh
 $c_{2t}, c_{2b}, \dots, c_{nt}, c_{nb}$ fresh

Strand Algebra

$$P ::= x \mid [x_1, \dots, x_n] \cdot [y_1, \dots, y_m] \mid 0 \mid P \mid P \mid P^* \quad n \geq 1, m \geq 0$$

x is a *signal*
 $[x_1, \dots, x_n] \cdot [y_1, \dots, y_m]$ is a *gate*
 0 is an *inert solution*
 $P \mid P$ is *parallel composition* of signals and gates
 P^* is a *population* (multiset) of signals and gates

Reaction Rule

$$x_1 \mid \dots \mid x_n \mid [x_1, \dots, x_n] \cdot [y_1, \dots, y_m] \rightarrow y_1 \mid \dots \mid y_m$$


Auxiliary rules (axioms of diluted well-mixed solutions)

$$P \rightarrow P' \Rightarrow P \mid P'' \rightarrow P' \mid P'' \quad \text{Dilution}$$
$$P \equiv P_1, P_1 \rightarrow P_2, P_2 \equiv P' \Rightarrow P \rightarrow P' \quad \text{Well Mixing}$$

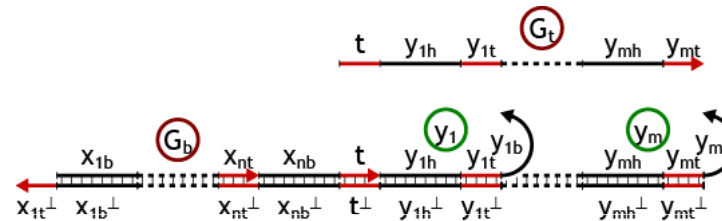
Where \equiv is a congruence relation (syntactical ‘chemical mixing’)
with $P^* \equiv P \mid P^*$ for unbounded populations.

Compiling Strand Algebra to DNA

$P ::= x \mid [x_1, \dots, x_n] \cdot [y_1, \dots, y_m] \mid 0 \mid P \mid P \mid P^* \quad n \geq 1, m \geq 0$

- $\text{compile}(x) =$ 

- $\text{compile}([x_1, \dots, x_n] \cdot [y_1, \dots, y_m]) =$



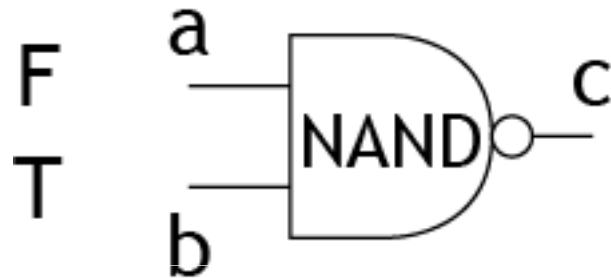
- $\text{compile}(0) =$ empty solution

- $\text{compile}(P \mid P') = \text{mix}(\text{compile}(P), \text{compile}(P'))$

- $\text{compile}(P^*) = \text{population}(\text{compile}(P))$

Boolean Networks

Boolean Networks to Strand Algebra



$$\begin{aligned} & ([a_F, b_F] \cdot c_T)^* \mid \\ & ([a_F, b_T] \cdot c_T)^* \mid \\ & ([a_T, b_F] \cdot c_T)^* \mid \\ & ([a_T, b_T] \cdot c_F)^* \mid \\ & a_F \mid b_T \end{aligned}$$

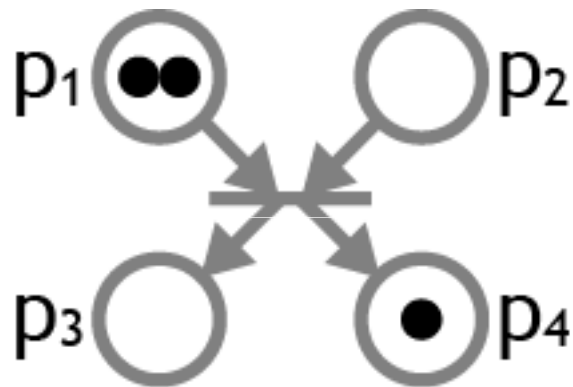
This encoding is *compositional*, and can encode *any* Boolean network:

- multi-stage networks can be assembled (*combinatorial logic*)
- network loops are allowed (*sequential logic*)

Petri Nets

Petri Nets to Strand Algebra

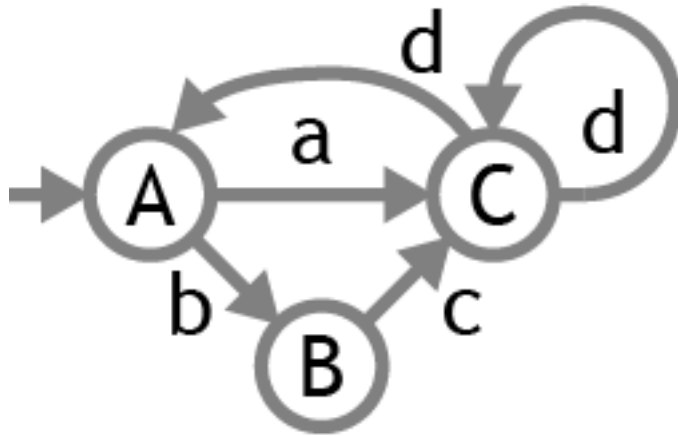
Transitions as Gates
Place markings as Signals



$$([p_1, p_2] \cdot [p_3, p_4])^* \mid p_1 \mid p_1 \mid p_4$$

Finite State Automata

FSA to Strand Algebra



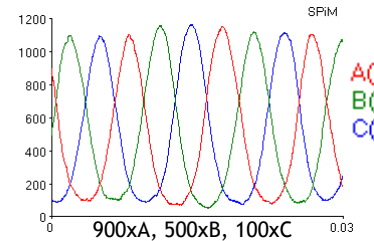
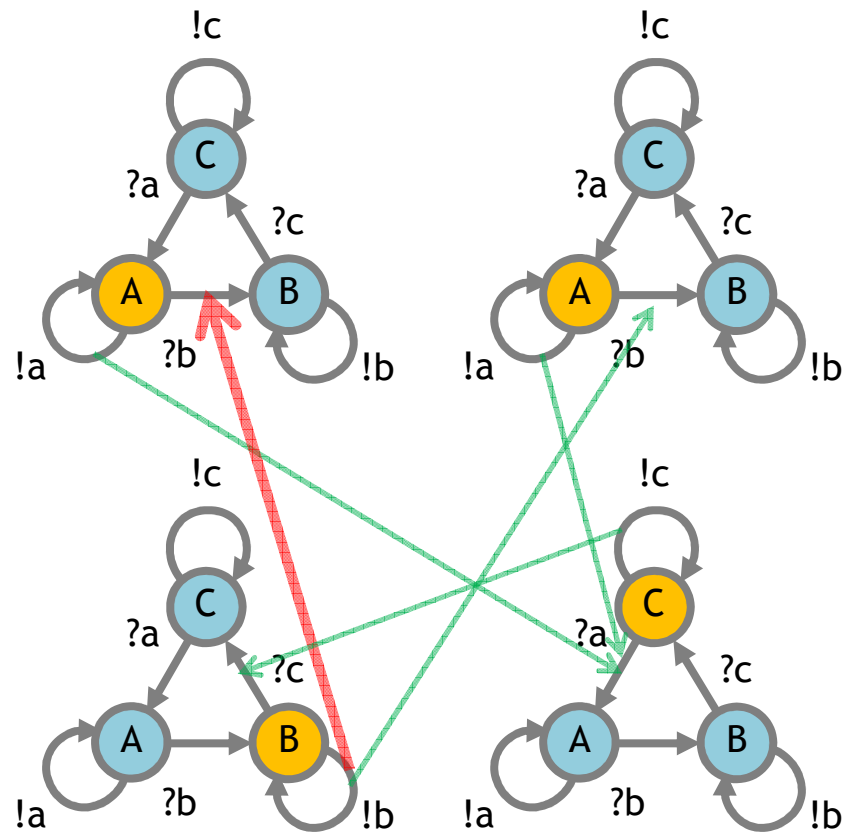
$$\begin{aligned}
 & ([A, a]. [C, \tau])^* \mid \\
 & ([A, b]. [B, \tau])^* \mid \\
 & ([B, c]. [C, \tau])^* \mid \\
 & ([C, d]. [C, \tau])^* \mid \\
 & ([C, d]. [A, \tau])^* \mid \\
 & A \mid \tau
 \end{aligned}$$

Input strings

a, b, c, d

$$\begin{aligned}
 & \tau . [a, \sigma_1] \mid \\
 & [\sigma_1, \tau] . [b, \sigma_2] \mid \\
 & [\sigma_2, \tau] . [c, \sigma_3] \mid \\
 & [\sigma_3, \tau] . d
 \end{aligned}$$

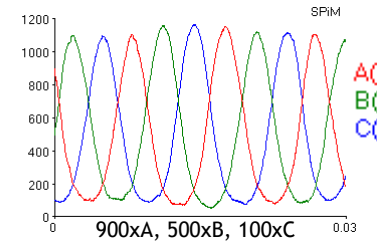
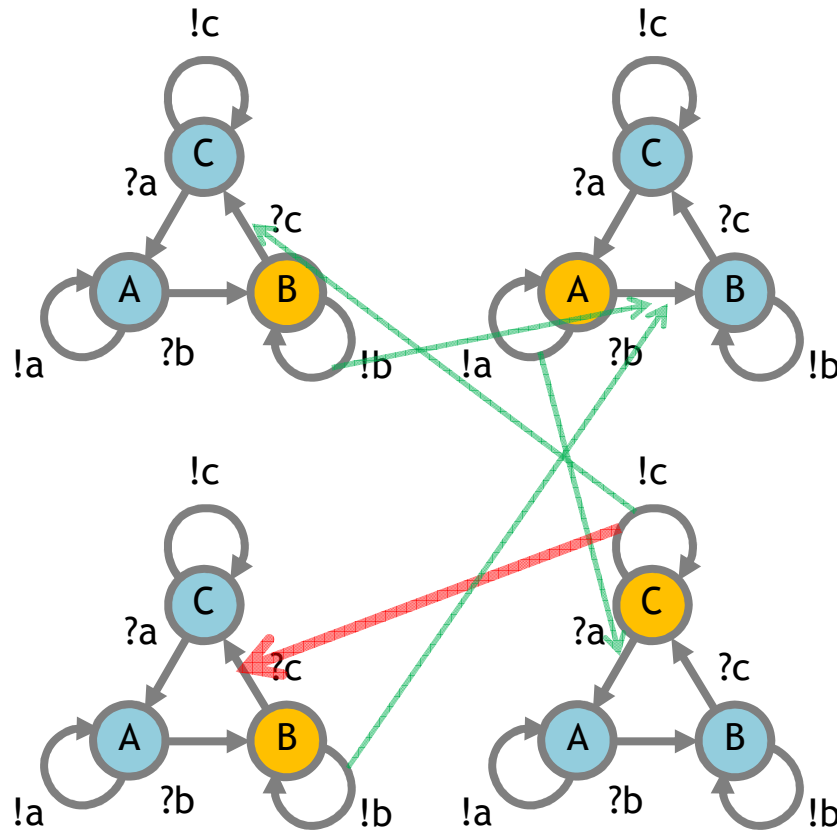
Interacting Automata



$([A, B]. [B, B])^* \mid$
 $([B, C]. [C, C])^* \mid$
 $([C, A]. [A, A])^* \mid$
 $A \mid A \mid B \mid C$

This is a uniform population of identical automata,
 but heterogeneous populations of interacting automata can be similarly handled.

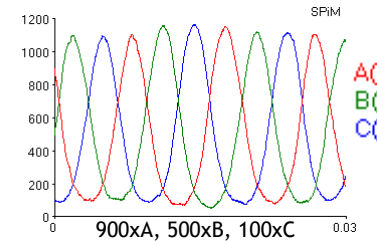
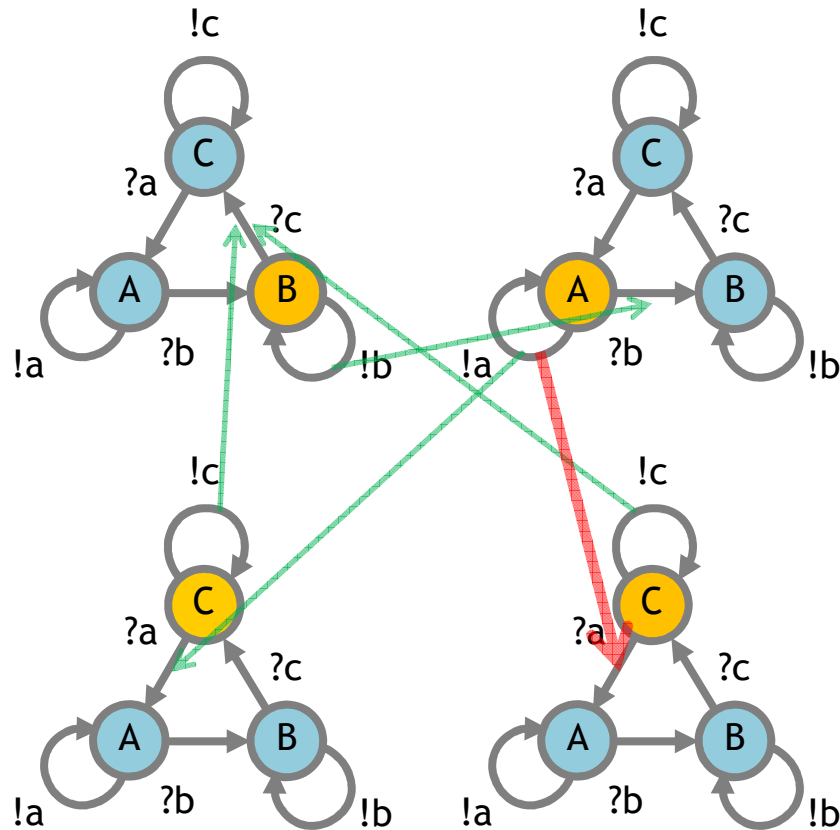
Interacting Automata



$([A, B]. [B, B])^*$ |
 $([B, C]. [C, C])^*$ |
 $([C, A]. [A, A])^*$ |
A | B | B | C

This is a uniform population of identical automata,
 but heterogeneous populations of interacting automata can be similarly handled.

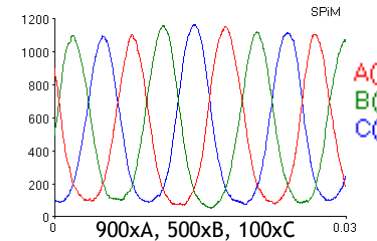
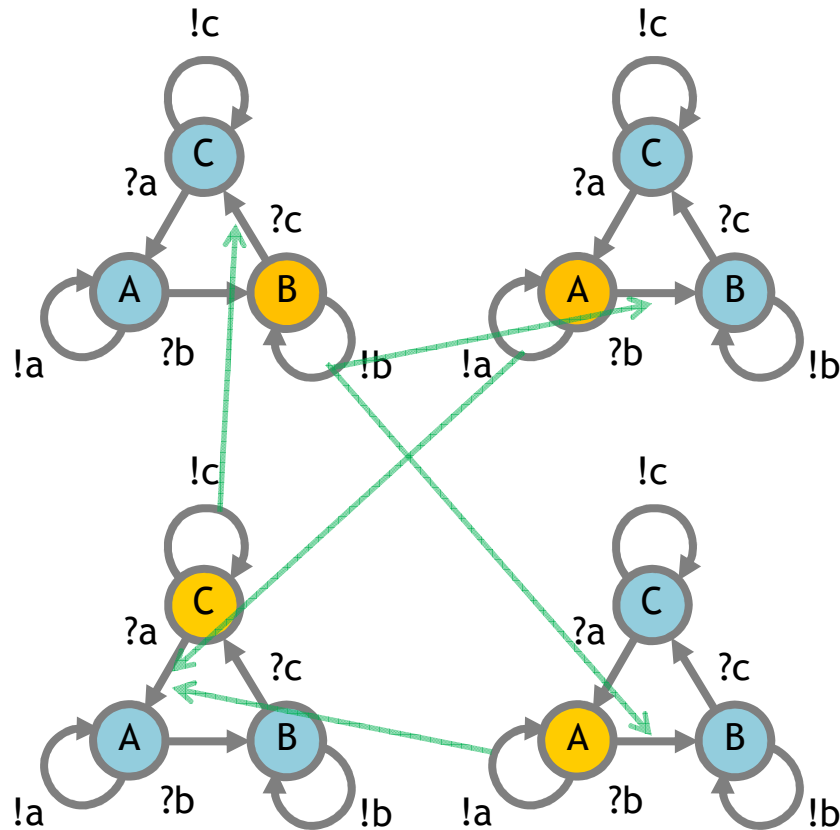
Interacting Automata



$([A, B]. [B, B])^*$ |
 $([B, C]. [C, C])^*$ |
 $([C, A]. [A, A])^*$ |
A | **B** | **C** | **C**

This is a uniform population of identical automata,
 but heterogeneous populations of interacting automata can be similarly handled.

Interacting Automata



$([A, B]. [B, B])^* \mid$
 $([B, C]. [C, C])^* \mid$
 $([C, A]. [A, A])^* \mid$
A \mid **A** \mid **B** \mid **C**

This is a uniform population of identical automata,
 but heterogeneous populations of interacting automata can be similarly handled.

More in the Paper

- Stochastic strand algebra
 - Matches the stochastic semantics of interacting automata
 - Uses a technique for implementing constant buffered populations, to replace P^* with finite populations
- Nested strand algebra
 - An higher-level language (with nested expressions)
 - A compilation algorithm into the basic strand algebra

Conclusions

- **Strand algebra** is an intermediate language for DNA compilation
 - It has its own abstract kinetics
 - Supports the compilation of multiple higher-level languages
 - Boolean Networks
 - Finite State Automata
 - Petri Nets
 - **Interacting Automata**
 - Etc.?
 - Into (possibly) multiple lower-level DNA architectures
 - **Strand displacement**
 - Etc.?
- **Acknowledgments**
 - The DNA 14 organizers for inviting me and providing stimulus for this work.
 - Extensive discussions with Lulu Qian, David Soloveichik and Erik Winfree on possible gate designs (about 15 of them!).
 - John Reif and Urmi Majumder for working on the problem.