

Molecular Programming

Luca Cardelli

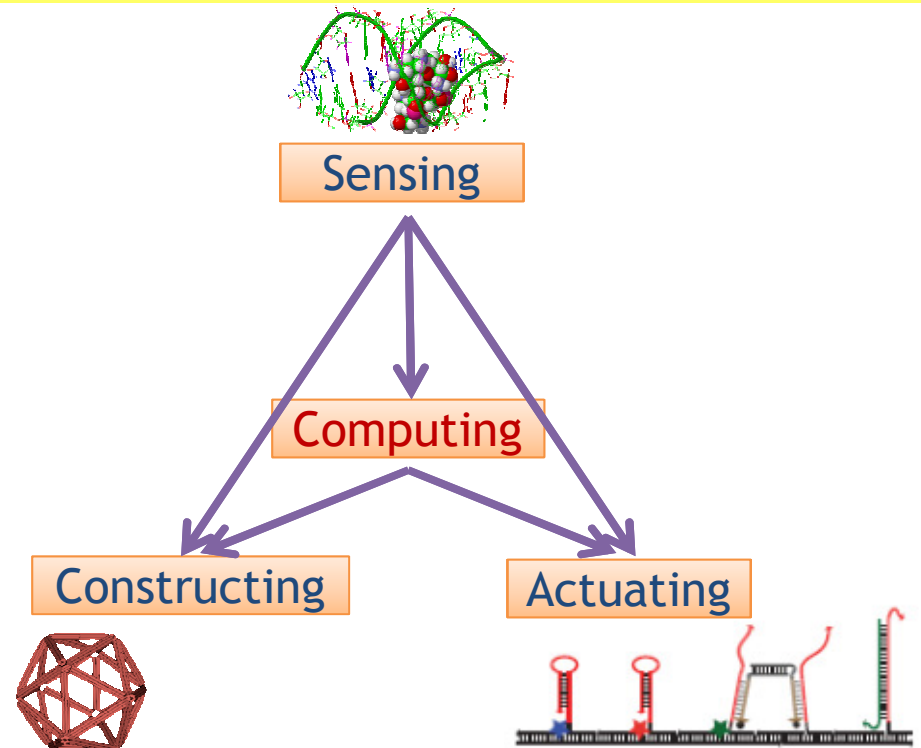
Microsoft Research

SynBioNT Sheffield, 2009-10-12

<http://lucacardelli.name>

Nano Tasks

- Sensing
 - Reacting to forces
 - Binding to molecules
- Actuating
 - Releasing molecules
 - Producing forces
- Constructing
 - By spontaneous self-assembly
 - Catalyzed by stimuli
- Computing
 - All that under 'program control'
 - Analog: Signal Filtering, Amplification
 - Digital: Logical gates
- Nucleic Acids (DNA/RNA)
 - Probably the only materials that can perform all these functions.
 - Technology relatively well developed.
 - Can interface to biological entities.



Compositionality

- Sensors and Actuators at the 'edge' of the system
 - They can use disparate kinds of inputs (sensors) and outputs (actuators)
- The 'kernel' of the system computes
 - Must use uniform inputs and outputs
- Compositionality in the kernel
 - Supporting 'arbitrary' computing complexity
 - The **output** of each computing components must be the **same kind of 'signal'** as the **input**
 - If the inputs are voltages, the outputs must be voltages
 - If the inputs are proteins, the outputs must be proteins
 - If the outputs are photons the inputs must be photons
 - If the inputs are DNA, the outputs must be DNA
 - What should our nano-signals be?

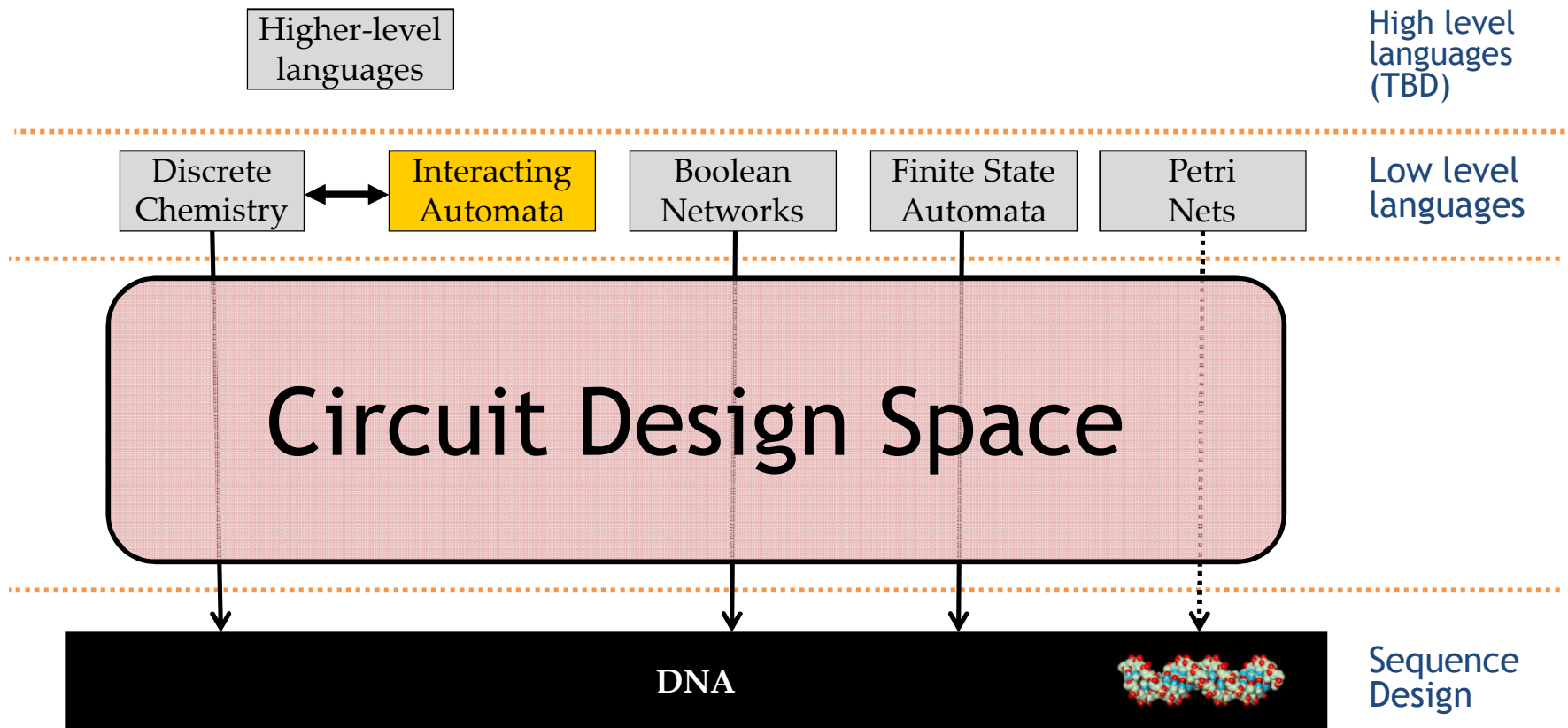
What does DNA Compute?

- 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)

Implementing "Arbitrary" Computing Functions

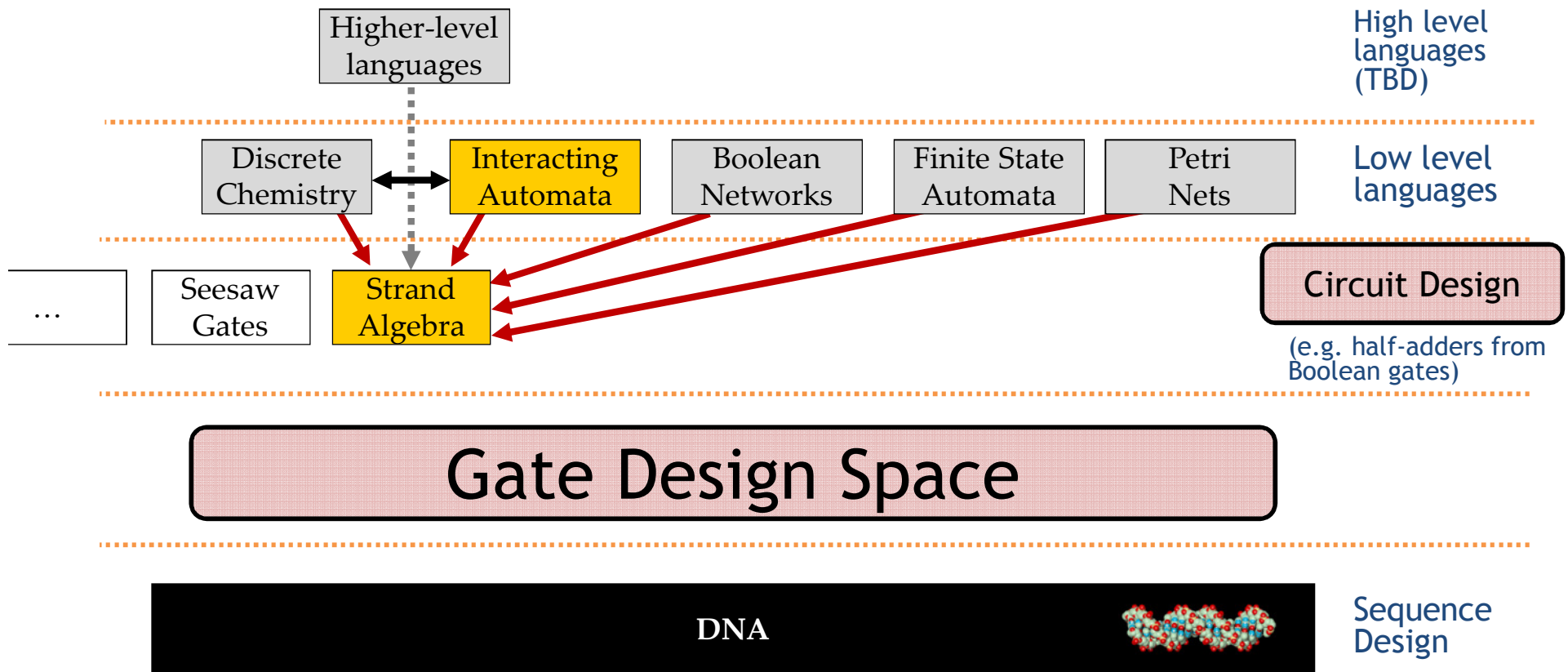
DNA Compilation

Separating **Circuit Design** from **Gate Design**



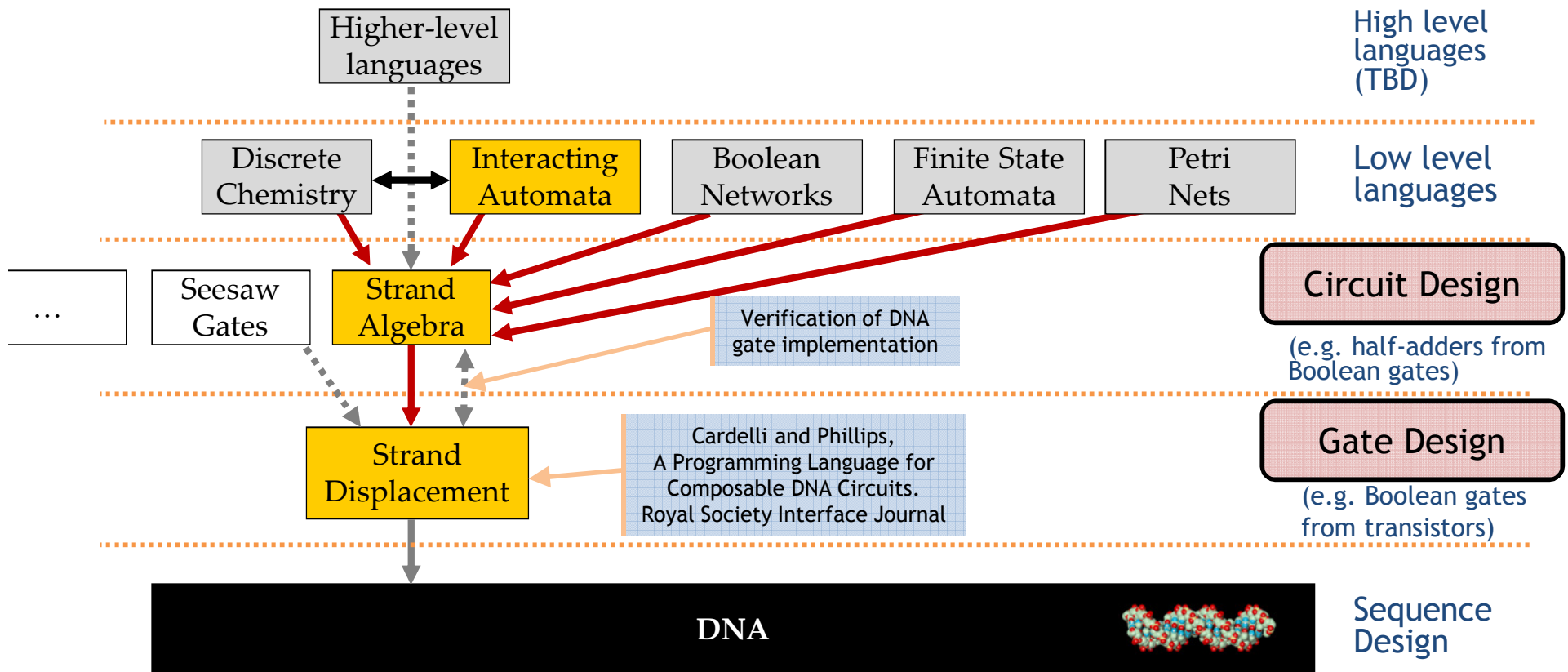
DNA Compilation

Separating Circuit Design from Gate Design



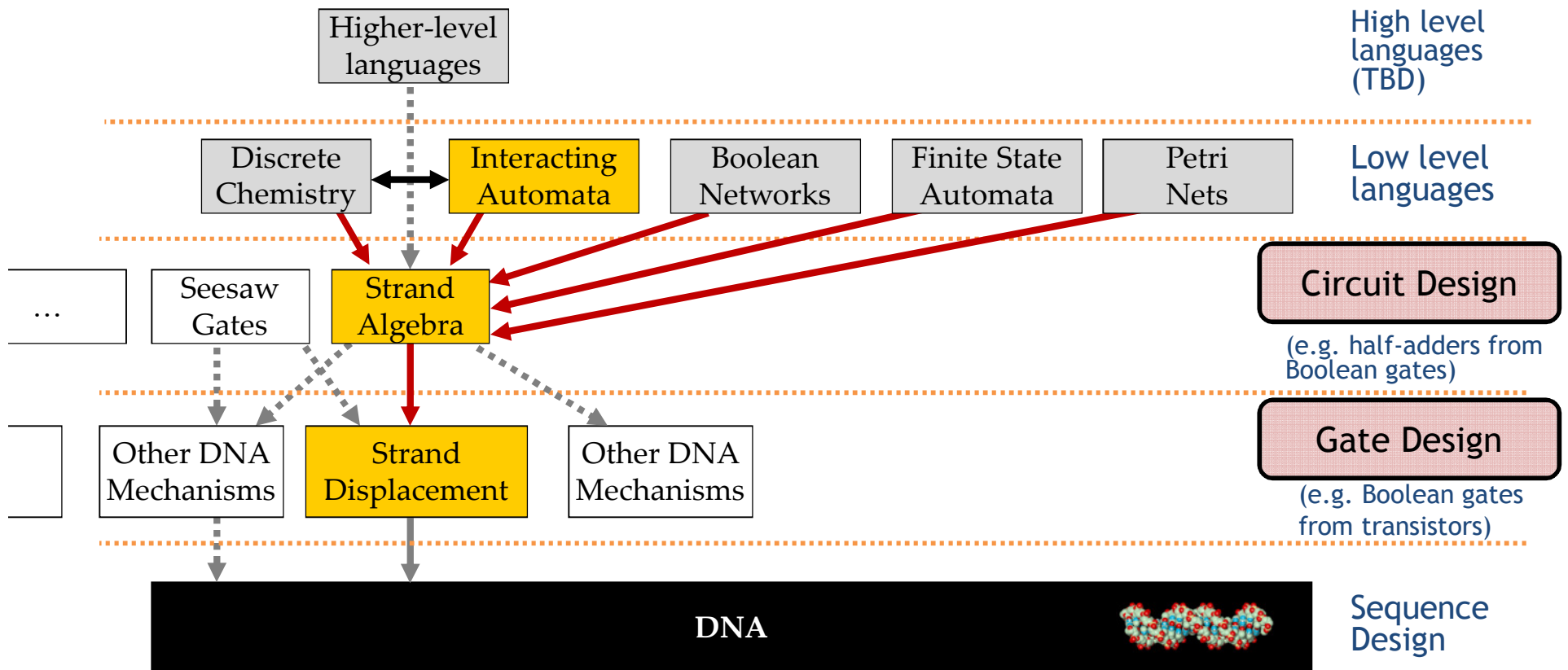
DNA Compilation

Separating Circuit Design from Gate Design



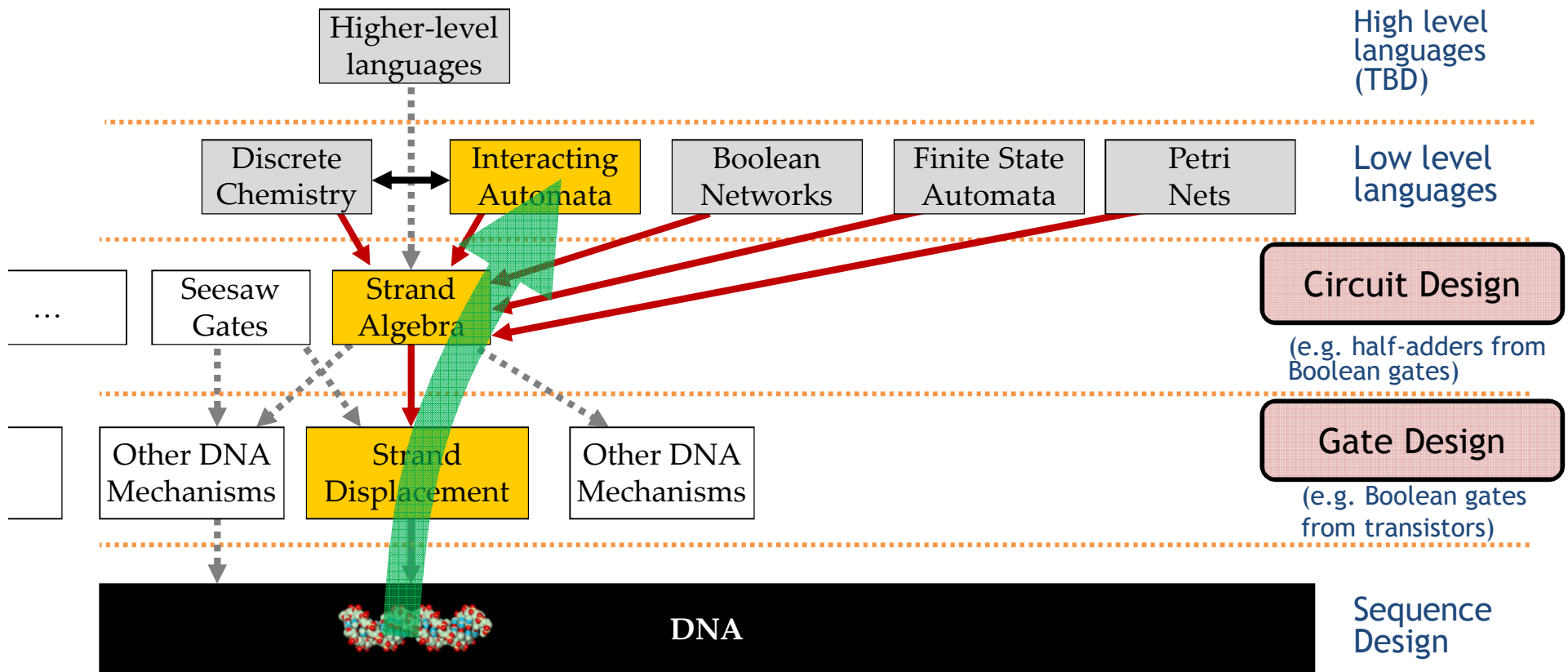
DNA Compilation

Separating Circuit Design from Gate Design



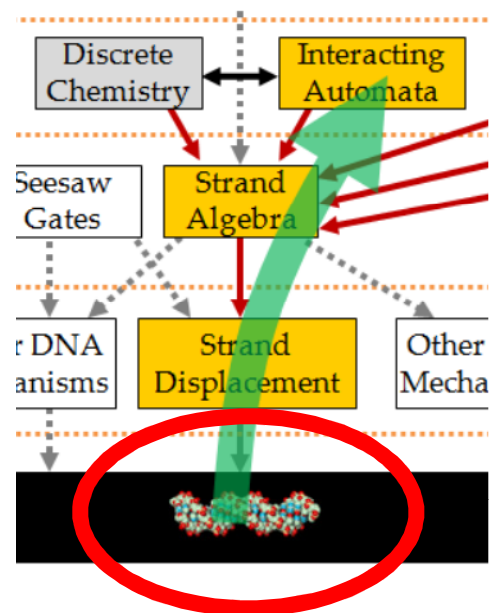
DNA Compilation

Separating Circuit Design from Gate Design

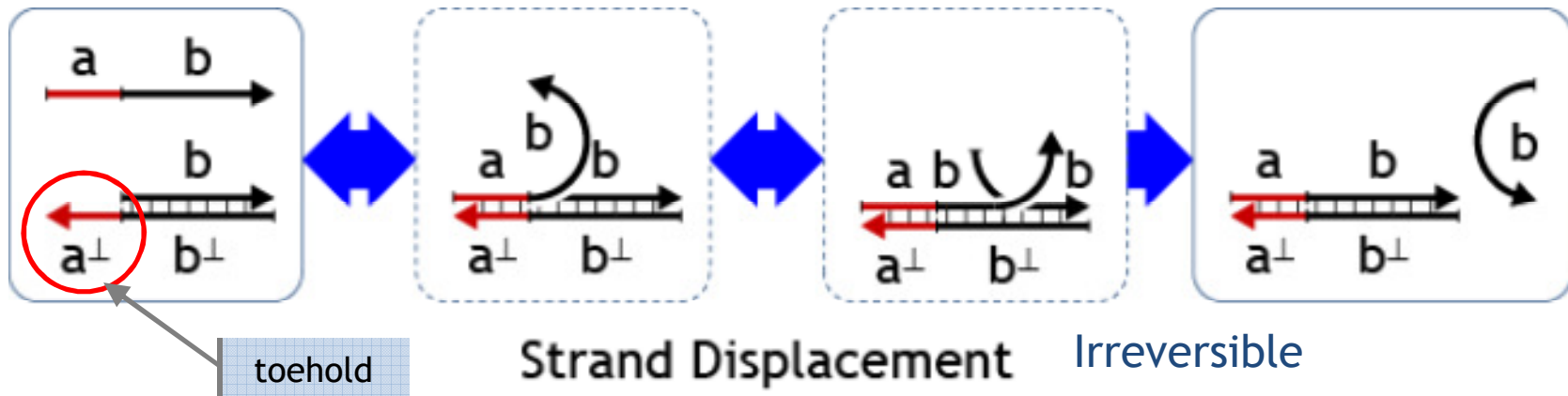


Rest of the talk: bottom up

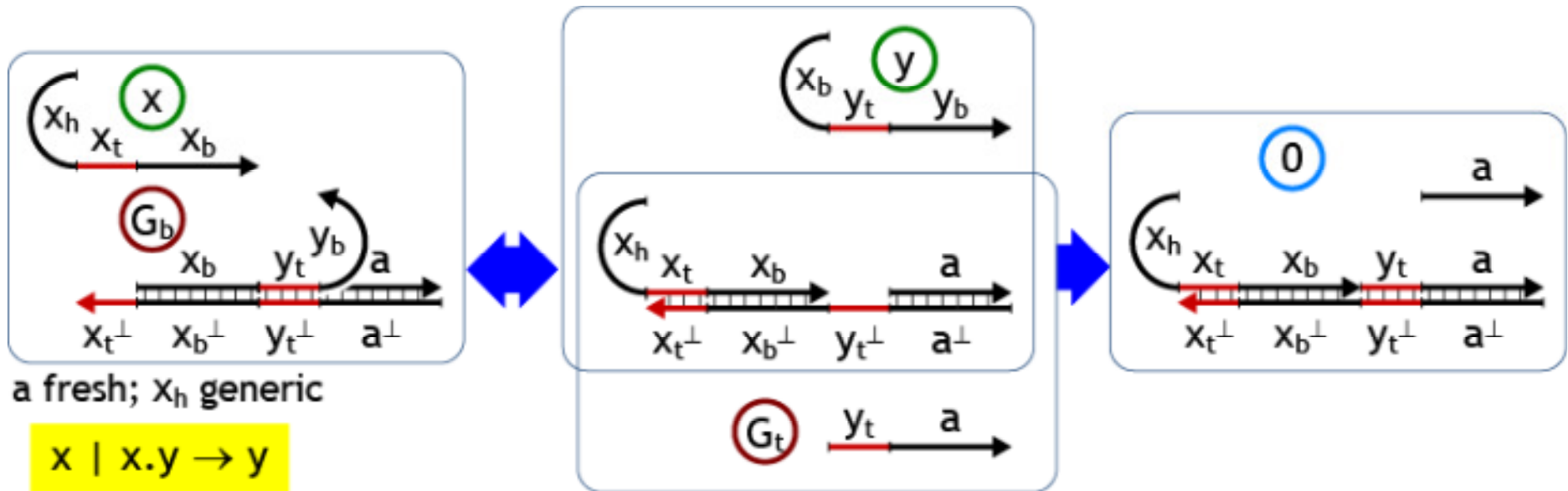
Toehold Mediated Strand Displacement



Strand Displacement Reaction



x.y Transducer Gate

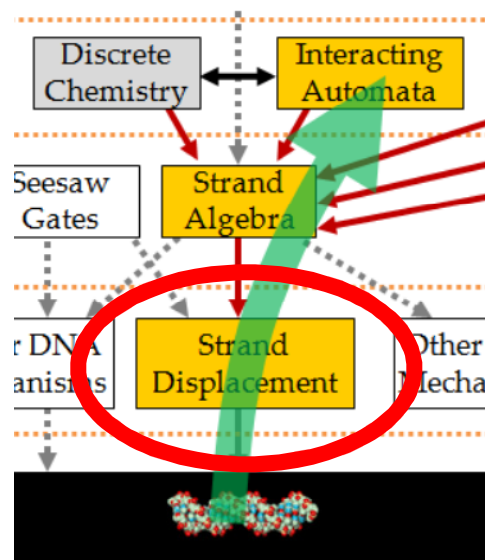


G_b, G_t (gate backbone and trigger) form the transducer.

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.

Strand Displacement Intermediate Language



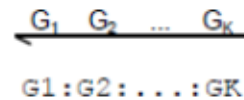
Syntax

A. Syntax of DNA molecules D

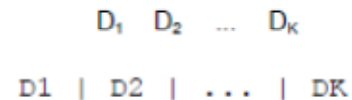
Upper strand with sequence complementary to S



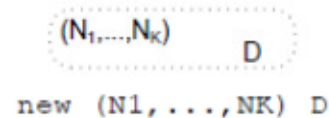
Molecule with segments G_1, \dots, G_k



Parallel molecules D_1, \dots, D_k



Molecules D with private domains N_1, \dots, N_k

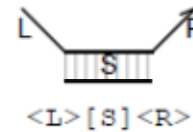


B. Syntax of DNA segments G

Lower strand with toehold N^c

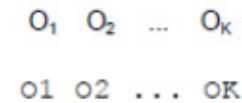


Double strand with sequence S and overhangs L, R



C. Syntax of DNA sequences S, L, R

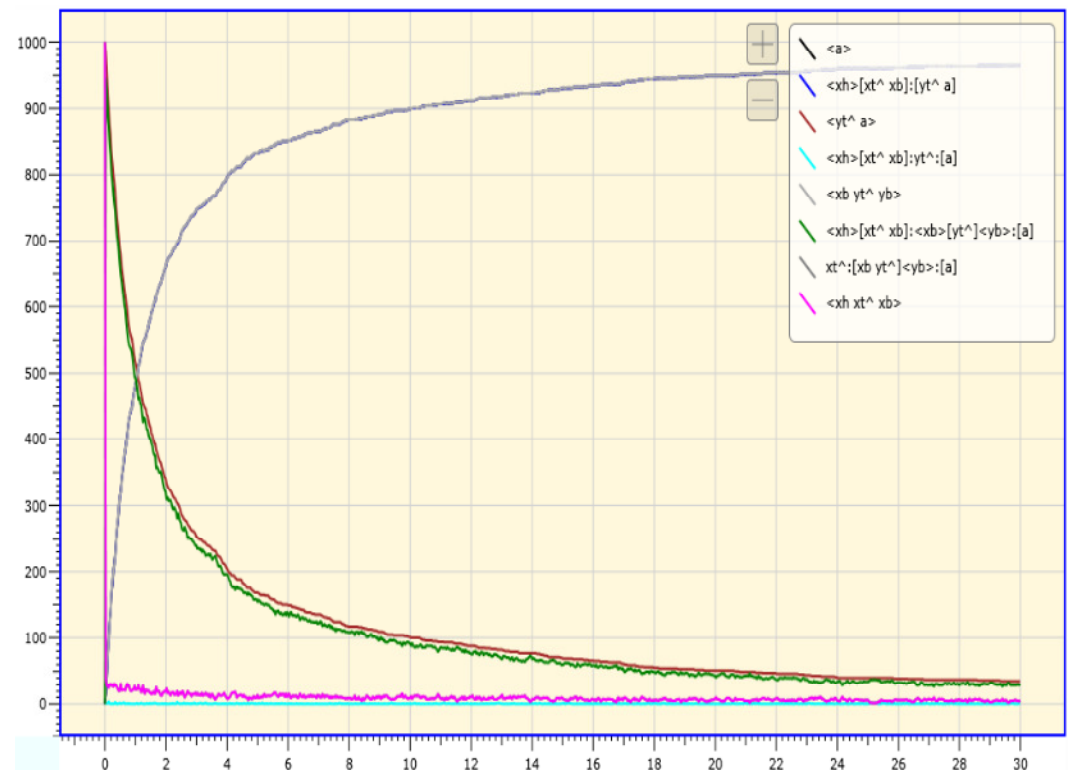
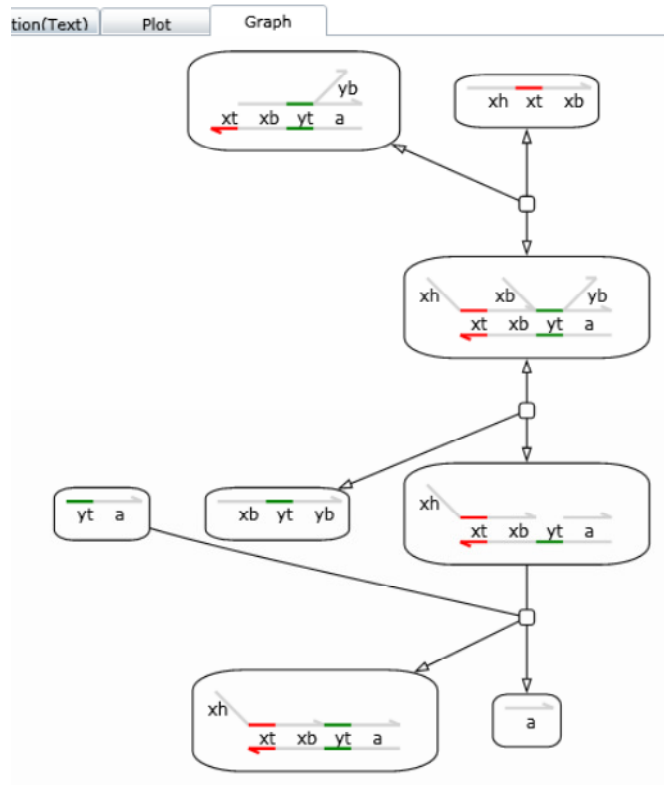
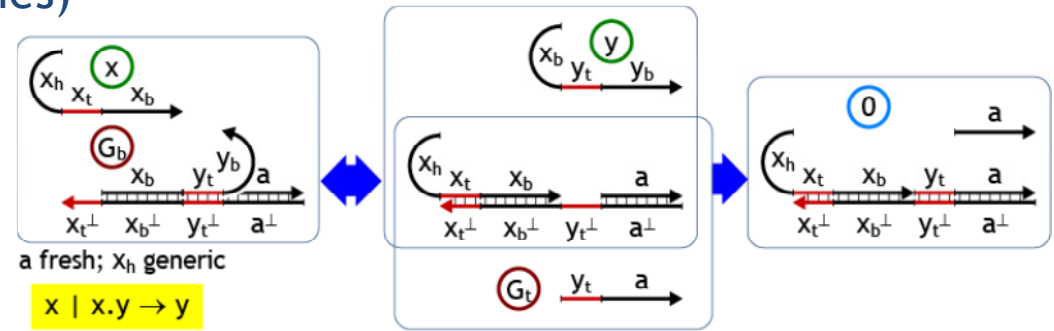
Sequence of domains O_1, \dots, O_k



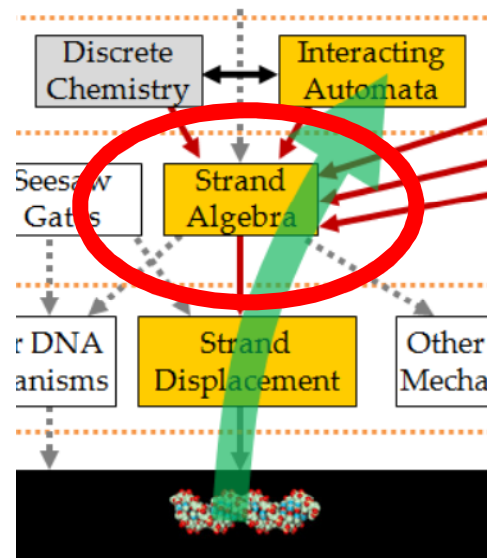
Strand Displacement Simulation Tool

1 Transducer gate $x.y$ (3 initial species)

```
directive sample 30.0 1000
new xt@1.0,1.0
new yt@1.0,1.0
( 1000 <xh xt^ xb>
| 1000 * xt^:[xb yt^]<yb>:[a]
| 1000 * <yt^ a>
)
```



Strand Algebra



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

Reaction Rule

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

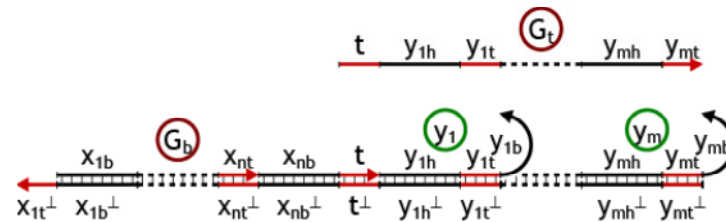
Equivalent to (stochastic) place-transition Petri Nets.

Compiling Strand Algebra to DNA

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

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

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

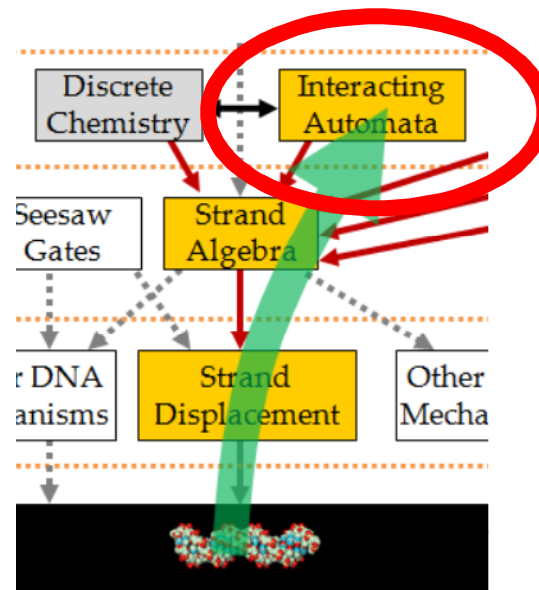


- $\text{compile}(0) =$ empty solution

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

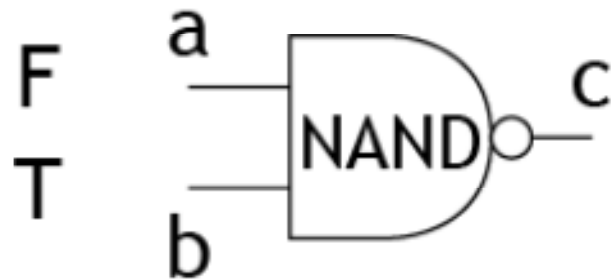
- $\text{compile}(P^*) = \text{population}(\text{compile}(P))$

Computational Abstractions



Boolean Networks

Boolean Networks to Strand Algebra



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

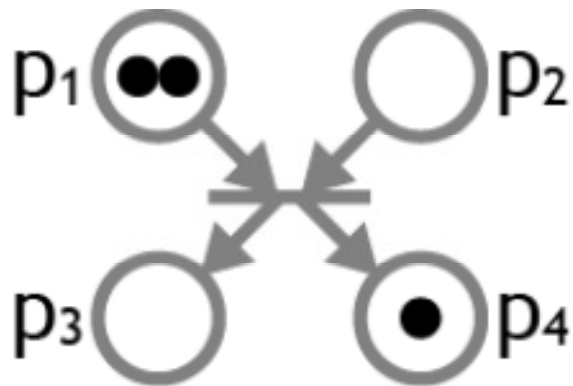
This encoding is *compositional*, and can encode *any* Boolean network:

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

Petri Nets

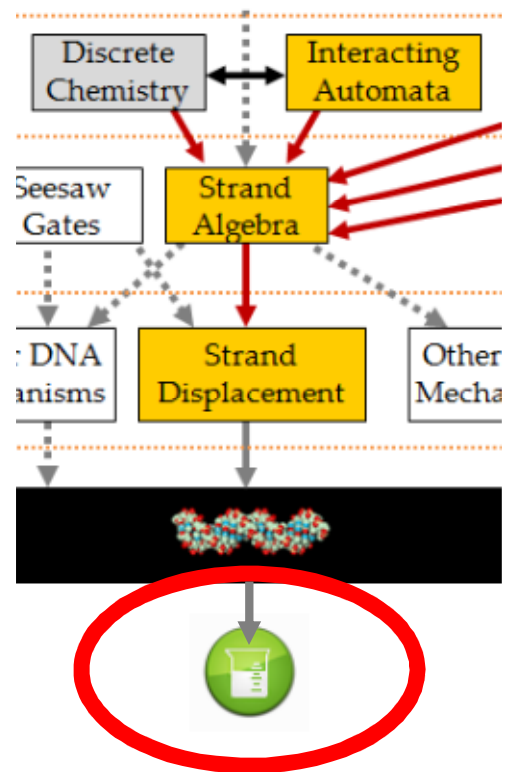
Petri Nets to Strand Algebra

Transitions as Gates
Place markings as Signals



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

Experiments



Sequence Design

NUPACK BETA
nucleic acid package

Analysis Design Downloads

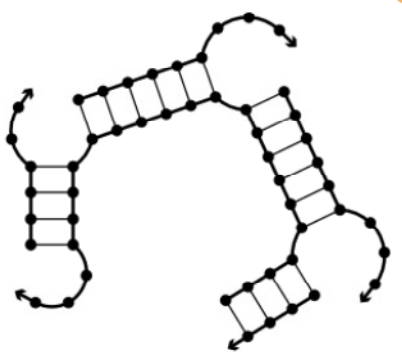
Input References Demos Help

Nucleic acid type: RNA DNA

Number of designs: 1

Target structure: (((...+(((...+(((...+(((+)))))))))))))...

Preview:



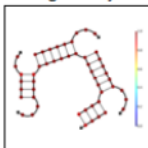
Input

NUPACK BETA
nucleic acid package

Analysis Design Downloads

Input Results References Demos Help

Designability summary



Sequence designs

Average percentage of correct nucleotides	Average number of incorrect nucleotides	GC content	Sequence
99.1%	0.475	74.5%	GGCCUC+GCAAGCACC+GCC AGCUUG+GCTC+GAGCGTUG GCGCUUGC GCGCGUG

Analyze

Output

Copyright © 2007-2009 Caltech. All rights reserved. | [Contact](#) | [Funding](#) | [Terms of use](#)

Conclusions

Conclusion

- Nucleic Acids
 - Programmable matter
- DNA Strand Displacement
 - A computational mechanism at the molecular level
- DNA Compilation
 - High-level languages (Boolean Networks, Petri Nets, Interacting Automata)
 - Intermediate languages (Strand Algebra, Strand Displacement Language).
 - Sequence generation.
- Tools
 - Thermodynamic analysis.
 - Simulation.
 - Verification/Optimization (not yet).



Q?

<http://lucacardelli.name>