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

t x z

x

t x y
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\textsuperscript{a,1}, Georg Seelig\textsuperscript{a,b,1}, and Erik Winfree\textsuperscript{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

“Lulu’s Trouble”

(from D.Soloveichik)
Two-Domain Architecture

- **Signals:** 1 toehold + 1 recognition region

- **Gates:** “top-nicked double strands” (or equivalently double strands with open toeholds)

Garbage collection “built into” the gates
Transducer \( x \rightarrow y \)
Transducer $x \rightarrow y$

**Input**

\[ \begin{array}{cccc}
  t & x \\
  t & a \\
  t & x & t & a & t & a \\
  x & t & y & t & a & t \\
\end{array} \]

\(ta\) is a *private* signal (a different ‘a’ for each \(xy\) pair)
Transducer $x \rightarrow y$
Transducer $x \rightarrow y$

Active waste
Transducer $x \rightarrow y$
Transducer $x \rightarrow y$

So far, a **tx signal** has produced an **at cosignal**. But we want signals as output, not cosignals.
Transducer \( x \rightarrow y \)
Transducer \( x \rightarrow y \)
Transducer $x \rightarrow y$
Transducer $x \rightarrow y$

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 $x \rightarrow y$
Transducer $x \rightarrow y$
Transducer $x \rightarrow y$
Transducer \( x \rightarrow y \)
Done.

Note the tata motif and how it helps in collection.
Fork $x \xrightarrow{\ } y + z$

(Amplifier: $x \xrightarrow{\ } x + x$)
Catalyst: $x + y \rightarrow y + z$

$yt$ is kindly provided by the left hand side.

(Autocatalyst: $x + y \rightarrow y + y$)
A + B → B + C

Experiments

Georg Seelig, Matt Olson
Autocatalytic Oscillator

\[
\begin{align*}
   x + y & \rightarrow y + y \\
   y + z & \rightarrow z + z \\
   z + x & \rightarrow x + x
\end{align*}
\]

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 $x + y \rightarrow z$
Join x+y→z
Join \( x + y \rightarrow z \)
Join $x + y \rightarrow z$
Join $x+y \rightarrow z$

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

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

$t_b$ is a *private* signal (a different ‘b’ for each $xyz$ triple)

Transducer $x \rightarrow y$
Join $x+y \rightarrow z$
Join $x + y \rightarrow z$
Join $x+y \rightarrow z$
Join $x + y \rightarrow z$
Join $x + y \rightarrow z$
Join \( x \parallel y \rightarrow z \)
Join $x + y \rightarrow z$
Join $x + y \rightarrow z$
Join $x + y \rightarrow z$
Join $x + y \rightarrow z$
Join $x+y \rightarrow z$
Join $x + y \rightarrow z$
Join $x + y \rightarrow z$
Join $x + y \rightarrow z$
Join $x + y \rightarrow z$
General $n \times m$ Join–Fork

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

Figure 9: 3-Join $J_{wxyz}$ | $tw$ | $tx$ | $ty$ → $tz$: 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
  o Reversible reactions (infinite traces)
  o Interferences (deadlocks etc.) between copies of the same gate
  o Interferences (deadlocks etc.) between copies of different gates
  o Removal of active byproducts (garbage collection) is tricky

• Populations
  o Gates come in (large) populations
  o Each population *shares private domains* (technologically unavoidable)
  o Correctness of populations means proofs with large state spaces
  o Proofs about *arbitrary* population size?

• Environment
  o The nano–environment is stochastic (noise, failures, etc.)
  o Biology is messy
  o But we should at 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

\[ S ::= t.x : x.t \]  
\[ D ::= \emptyset : t : x : t.x : x.t : x.x : D^\dagger D \]
\[ U ::= S : D : U|U : (\forall x)U \]

- **S**: single strand
- **D**: double strand
- **U**: soup

**Nick Operator**
Algebraic Equality

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

\[ D_1 \dagger (D_2 \dagger D_3) = (D_1 \dagger D_2) \dagger D_3 \]
\[ \emptyset \dagger D = D \dagger \emptyset = D \]

\[ U_1 \mid (U_2 \mid U_3) = (U_1 \mid U_2) \mid U_3 \]
\[ U_1 \mid U_2 = U_2 \mid U_1 \]
\[ \emptyset \mid U = U \mid \emptyset = U \]

\[ (\nu x)U = (\nu y)(U\{y/x}) \quad \text{if } y \notin pd(U) \]
\[ (\nu x)\emptyset = \emptyset \]
\[ (\nu x)(U_1 \mid U_2) = U_1 \mid (\nu x)U_2 \quad \text{if } x \notin pd(U_1) \]
\[ (\nu x)(\nu y)U = (\nu y)(\nu x)U \]
Reduction

\[ D_1^{\dagger}t^{\dagger}xt^{\dagger}D_2 \mid tx \leftrightarrow D_1^{\dagger}tx^{\dagger}t^{\dagger}D_2 \mid xt \] exchange

\[ D_1^{\dagger}t^{\dagger}x^{\dagger}D_2 \mid tx \rightarrow D_1^{\dagger}tx^{\dagger}D_2 \] left coverage

\[ D_1^{\dagger}x^{\dagger}t^{\dagger}D_2 \mid xt \rightarrow D_1^{\dagger}xt^{\dagger}D_2 \] right coverage

\[ D_1^{\dagger}t^{\dagger}xy^{\dagger}t^{\dagger}D_2 \mid tx \mid yt \rightarrow D_1^{\dagger}tx^{\dagger}yt^{\dagger}D_2 \] cooperation

\[ D \rightarrow \emptyset \] if \( D \) not reactive waste

\[ U_1 \rightarrow U_2 \Rightarrow U_1 \mid U \rightarrow U_2 \mid U \] dilution

\[ U_1 \rightarrow U_2 \Rightarrow (\nu x)U_1 \rightarrow (\nu x)U_2 \] isolation

\[ U_1 = U_2, \ U_2 \rightarrow U_3, \ U_3 = U_4 \Rightarrow U_1 \rightarrow U_4 \] mixing

i.e.:
Reachability

• \( U_1 \rightarrow^* U_2 \) iff \( U_1 \rightarrow \ldots \rightarrow U_2 \)
  
  o That is, \( U_1 \) may reduce to \( U_2 \).

• \( U_1 \rightarrow^\forall U_2 \) iff \( \forall U, U_1 \rightarrow^* U \Rightarrow U \rightarrow^* U_2 \)
  
  o That is, \( U_1 \) will reduce to \( U_2 \). (It cannot avoid the possibility of reducing to \( U_2 \)).
  
  o \( U \rightarrow^\forall U \) means that \( U \) is reversible.
  
  o 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

- $T_{xay} = t^t x t^t a t^a | t a | x^t y t a^t t | y t$
- $T^n_{xy} = (v a) ((T_{xay})^n)$

- $F_{xayz} = ...$
- $F^n_{xyz} = (v a) ((F_{xayz})^n)$

- $J_{xyaz} = ...$
- $J^n_{xyz} = (v a) ((J_{xyaz})^n)$
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**
  \[
  T^1_{xy} | tx \rightarrow^\forall ty
  \]
  - 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 model checking 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
  o Simple signals, simple gate structures.
  o Self-cleaning: no garbage left by operation (except inert).
  o Enabling new ways of assembling gates.
  o Some experimental evidence that it works.

• A correspondingly simple algebra
  o For verifying gate designs mechanically.

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