Computing with Molecules

Luca Cardelli
Microsoft Research

Development in Computational Models
Cambridge 2012-06-17
http://lucacardelli.name
A computational model

- **Molecular Soups**
  - Molecules randomly collide and can change state or composition.
    - Can we compute with that?
  - Based on the classical atomic theory of matter
    - with Brownian motion
    - nothing quantum here

- **Related to:**
  - “In small numbers” (macroscopic systems)
    - Process Algebra
    - Petri Nets
  - “In large numbers” (microscopic systems)
    - Population Protocols [Angluin et al.]
    - Amorphous Computing [Abelson et al.]
    - Swarm Intelligence – Ant Colonies
    - Epidemiology
    - Chemistry
A notion of algorithm

• Data as populations
  o Inputs and outputs are composed of uniform \textit{populations} of agents that do \textit{not} have an identity
  o Algorithms ‘emerge’ from the ‘dumb’ interactions of ‘simple’ agents

• In computing
  o Mostly explored in discrete or nondeterministic time

• In science and nature
  o Mostly explored in stochastic time
  o Stochastic because ‘interactions’ typically correspond to random collisions or chance meetings
A mathematical model

• The underlying model is Continuous–Time Markov Chains
  o Which also underlies chemistry via the Chemical Master Equation (changes of probabilities of discrete states over continuous time).

• Can be presented discretely, stochastically
  o As stochastic Petri nets, stochastic process algebras, etc.

• NOT a probabilistic model
  o Probabilities emerge from the stochastic structure (as the underlying DMC), but are not primary. We are in continuous time and we care about how long things take.
  o Non–determinism exists only in the form of ‘quantitative races’ among possibilities: who is faster is more likely to win, but there is no pure, timeless, random or probabilistic choice.
  o Interleaving rules: no two events (interactions) can ever happen at the same time in real time.
Basic Results

- The class of functions ‘over individuals’ that are computable
  - Turing machines can be encoded up to an arbitrarily small uniform error bound. “Approximately Turing-Complete”.
    - Wiedermann et al. ...

- The class of predicates ‘over collectives’ that are ‘stably computable’
  - Semi-linear predicates (first-order theory of $(\mathbb{N}, +, <)$).
Paradigm

• Stochastic chemistry is the simplest paradigm for this model
  o Finite collections of chemical reactions (with real-number rates) among a finite set of species.
  o Not necessarily preserving mass or energy (assumed to be freely provided from the ‘outside’).
  o Usually restricted to null-, uni-, and bi-molecular reactions.

  • \( x \rightarrow_r y + y \) multiply \( x \) by 2
  • \( x + y \rightarrow_r y + b \) compute the majority of populations
    \( y + x \rightarrow_r x + b \) \( x \) and \( y \) in log time [Angluin et al.]
    \( b + x \rightarrow_r x + x \)
    \( b + y \rightarrow_r y + y \)
Molecular Languages

- **Reaction-Based** \((A + B \rightarrow C + D)\) (Chemistry)
  - Limited to finite set of species (no polymerization)
  - Practically limited to small number of species (no run-away complexation)

- **Interaction-Based** \((A = !c. B)\) (Process Algebra)
  - Reduces combinatorial complexity of models by combining independent submodels connected by interactions.

- **Rule-Based** \((A{-}:B{p} \rightarrow A{p}:B{-})\) (Logic, Graph Rewriting)
  - Further reduces model complexity by describing molecular state, and by allowing one to ‘ignore the context’: a rule is a reaction in an unspecified (complexation/phosphorylation) context.
  - Similar to informal descriptions of biochemical events (“narratives”).

- **Syntactic connections**
  - The latter two can be translated (to each other and) to the first, but doing so may introduce an infinite, or anyway extremely large, number of species.
Semantic Connections

Continuous-state Semantics (Mass Action Kinetics)

<table>
<thead>
<tr>
<th>ODE</th>
<th>ODE</th>
</tr>
</thead>
</table>

Continuous Chemistry

Discrete Chemistry

Discrete-state Semantics (Chemical Master Equation)

Process Algebra

Nondeterministic Semantics

Stochastic Semantics

These diagrams commute via appropriate maps.

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

Combinatorial Explosion
Paradigm Lost

• But chemistry *is not* a computational science!
  o Real chemical reactions just ‘happen’ between real molecules that exist in nature. We don’t control them.
  o Chemists ‘transcribe’ nature and write down ‘its’ reactions. They do not write their own chemical programs.
    • Ok, they often design new molecules, but they do not have ‘full computational control’ over what those do.

• Similarly electronics *was not* computational
  o Electron exchanges just ‘happen’ in nature.
  o Early physicists did not have the ability to program them.
  o But now we do!
Paradigm Encoded

• Find some ‘universal molecules’ that can do ‘what all the other molecules can do’
  o By ‘doing something’ here we mean ‘implementing a chemical kinetics’.
  o That is: find a universal class of molecules that can emulate the kinetics of arbitrary systems of chemical reactions among real or fictitious molecules, up to some abstraction (e.g. time dilation).

• Find a way to actually execute molecular languages, with real molecules.
Computing with DNA

- Computing with molecules was, of course, the original idea in DNA computing
  - Early examples [Adelman] encoded specific algorithms.

- But only recently people have proposed ‘universal DNA molecules’
  - Soloveichik, D., Seelig, G., Winfree, E., DNA as a Universal Substrate for Chemical kinetics.
DNA
DNA

GC Base Pair
Guanine–Cytosine

TA Base Pair
Thymine–Adenine

Interactive DNA Tutorial
(http://www.biosciences.bham.ac.uk/labs/minchin/tutorials/dna.html)

Sequence of Base Pairs (GACT alphabet)
Robust, and *Long*

- **DNA in each human cell:**
  - 3 billion base pairs
  - 2 meters long, 2nm thick
  - folded into a 6µm ball
  - 750 MegaBytes

- **A huge amount for a cell**
  - Every time a cell replicates it has to copy *2 meters of DNA* reliably.
  - To get a feeling for the scale disparity, compute:

- **DNA in human body**
  - 10 trillion cells
  - 133 Astronomical Units long
  - 7.5 OctaBytes

- **DNA in human population**
  - 20 million light years long
Natural DNA Operation

- DNA can support structural and computational complexity.

DNA replication in *real time*

- In Humans: 50 nucleotides/second
- Whole genome in a few hours (with parallel processing)
- In Bacteria: 1000 nucleotides/second (higher error rate)

DNA transcription in *real time*

- RNA polymerase II:
  - 15–30 bases/second

Drew Berry

http://www.wehi.edu.au/wehi-tv
Unnatural DNA Operation

- **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.
Sensing

**Aptamers**: natural or artificially evolved DNA molecules that stick to other molecules (highly selectively).

Adenine riboswitch aptamer

*Structural basis for discriminative regulation of gene expression by adenine- and guanine-sensing mRNAs.*


Target molecule
Constructing Sensing
Computing
Actuating

Crosslinking

Chengde Mao, Purdue
Andrew Turberfield, Oxford

Folding DNA into Twisted and Curved Nanoscale Shapes
Hendrik Dietz, Shawn M. Douglas, & William M. Shih
Actuating

DNA tweezers

DNA walkers

Bernard Yurke, Boise State
Computing

• Sensors and Actuators at the 'edge' of the system
  o They can use disparate technologies and phenomena

• Computation in the 'kernel' of the system

• Compositionality in the kernel
  o The components should use uniform inputs and outputs
  o The components should be ‘computationally complete’
“Embedded” Computing
(Synthetic Biology)

- Using bacterial machinery (e.g.) as the hardware. Using embedded gene networks as the software.

- MIT Registry of Standard Biological Parts

- GenoCAD
  - Meaningful sequences [Cai et al.]

- GEC
  - [Pedersen & Phillips]
“Autonomous” Computing (Nano-engineering)

• Mix & go
  o All (or most) parts are synthesized
  o No manual cycling (cf. early DNA computing)
  o In some cases, all parts are made of DNA (no enzyme/proteins)

• Self-assembled and self-powered
  o Can run on its own (e.g. environmental sensing)
  o Or be embedded into organisms, but running ‘separately’
Curing

A doctor in each cell

Fig. 1 Medicine in 2050: “Doctor in a Cell”
RNA computation in dead cells

- Using RNA Hybridization Chain Reaction for imaging of mRNA expression.
  - The programmability of orthogonal RNA reactions enables spatial imaging with 5 simultaneous targets.
RNA computation in live cells

Selective cell death mediated by small conditional RNAs

Suvir Venkataraman*, Robert M. Dirks*, Christine T. Ueda*, and Niles A. Pierce*#1

PNAS | September 28, 2010
Computing with DNA Strand Displacement
DNA Computing

• Non-goals
  o Not to solve NP-complete problems.
  o Not to replace electronics.
  o Not necessarily using genes or producing proteins.

• For general ‘molecular programming’
  o To precisely control the organization and dynamics of matter and information at the molecular level.
  o To interact algorithmically with biological entities.
  o The use of DNA is “accidental”: no genes involved.
  o In fact, no material of biological origin.
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.
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

t x y
Bad Match
Bad Match
Bad Match

Cannot proceed
Hence will undo
Computation by DNA Strand Displacement
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

Strand Algebras for DNA Computing

Luca Cardelli

“Lulu’s Trouble”

(from D.Soloveichik)
DCM 2010

• Looking for a *simple* process *algebra* for strand displacement
  o For manual or automated analysis or correctness of strand displacement ‘programs’.
  o Had to be *simple* (or you could not analyze it). Hence looking for a simpler strand displacement scheme.
  o Had to be an *algebra*, hence computation could not leave garbage around, or nothing would commute.

• The technology was to be constrained by the theory
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

Two-Domain DNA Strand Displacement

Luca Cardelli

Transducer $x \rightarrow y$
Transducer $x \rightarrow y$

Input $t \ x$

$t \ a$

$y \ t$

$x \ t \ y \ t \ a \ t$

Built by self-assembly!

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

Active waste

\[
\begin{align*}
&\text{Active waste} \\
&x \quad t \\
&t \quad a \\
&t \quad x \quad t \quad a \quad t \quad a
\end{align*}
\]
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 $x \rightarrow y$
Transducer $x \rightarrow y$
Transducer $x \rightarrow y$

Done.

N.B. the gate is consumed: it is the energy source.
Join $x+y \rightarrow z$
General $n \times m$ Join–Fork

- Easily generalized to $2+\text{ inputs}$ (with $1+\text{ collectors}$).
- Easily generalized to $2+\text{ outputs}$.

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

Two-domain gate for $X + Y \rightarrow Y + B$

$X + Y \rightarrow Y + B$

$35C$

$1x = 50\text{nM}$

Yuan-Jyue Chen and Georg Seelig
U. Washington

---

### Diagram

- Output (nM) vs. hours graph

### Table

<table>
<thead>
<tr>
<th>$X + Y \rightarrow Y + B$</th>
<th>Concentration</th>
</tr>
</thead>
<tbody>
<tr>
<td>LG1</td>
<td>1.5x</td>
</tr>
<tr>
<td>LG2</td>
<td>1.5x</td>
</tr>
<tr>
<td>Input</td>
<td>1x</td>
</tr>
<tr>
<td>Catalyst</td>
<td>0x, 0.05x, 0.1x, 0.2x, 0.3x, 1x</td>
</tr>
<tr>
<td>$^\sim B$</td>
<td>2x</td>
</tr>
<tr>
<td>R1</td>
<td>2x</td>
</tr>
<tr>
<td>B readout</td>
<td>3x</td>
</tr>
</tbody>
</table>
An Accident of Simplicity

• Earlier architectures had ‘secondary structure’, which is ‘unnatural’:
  o It requires *synthetic* single-stranded DNA that is then assembled to form the desired structures.
  o Synthetic DNA has maximum length and quality problems (a fixed probability of synthesis error at each position, limiting size to about 200 bases).

• The two-domain architecture is (almost) ordinary biological DNA
  o Just double-stranded (with nicks), hence it can be produced *biologically*.
  o Biological DNA has much better quality and practically no length restriction: bacteria are so much better than we are at making it.

• Makes a new manufacturing technology possible
  o Gate-laden plasmids (circular DNA) are inserted into bacteria, who kindly produce large quantities of them overnight.
  o We then chop them up into gates and introduce the nicks via enzymes.

Yuan-Jyue Chen, Neil Dalchau, Cezanne Camacho, Matt Olson, David Soloveichik, Andrew Phillips, Luca Cardelli, and Georg Seelig.
DNA Programming
Strand Displacement Language
Compiling Chemistry to DNA (X→Y)

def R1x1(N,x,y) =
    new a
    ( N* <t^ a> |
    | N* <y t^>
    | N* t^*:x t^]:a t^]:[a]
    | N* [x]:[t^ y]:[t^ a]:t^* )

Input X

Output Y

def Species(N,x) =
    N*<t^ x>
Model-Checking Compilation ($X \rightarrow Y$)

Transducer State Space ($\text{Species}(1,x) \mid R1x1(1,x,y)$)
Stochastic Model Checking

PRISM results for sequential transducers
In addition to biochemistry laboratory techniques, computer science techniques were essential.

“Computer simulations of seesaw gate circuitry optimized the design and correlated experimental data.”
Turing–Powerful DNA Computers

Encoding a Stack

Encoding state transitions

Model–Checking a DNA Ripple Carry Adder

<table>
<thead>
<tr>
<th>Input A</th>
<th>Input B</th>
<th>Output X</th>
<th>Output C</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0 1 0 0</td>
<td>1 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>1 0 0 0</td>
<td>1 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>1 0 0 0</td>
<td>1 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>0 1 0 0</td>
<td>0 1 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>1 0 0 0</td>
<td>1 1 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>1 0 0 0</td>
<td>1 1 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>0 0 0 0</td>
<td>0 1 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>0 1 1 0</td>
<td>0 0 1 1</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>0 1 0 0</td>
<td>0 1 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>0 1 1 0</td>
<td>0 1 1 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

Lakin & Phillips, DNA17 2011
Localised circuits

Hairpins tethered to origami

- Increased speed
- Reduced interference

Chandran, Gopalkrishnan, Phillips, Reif. DNA Computing, 2011
Conclusions
A Brief History of DNA

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1936</td>
<td>Turing Machine</td>
</tr>
<tr>
<td>1947</td>
<td>Transistor</td>
</tr>
<tr>
<td>1982</td>
<td>Structural DNA</td>
</tr>
<tr>
<td>1994</td>
<td>DNA Algorithm</td>
</tr>
</tbody>
</table>

- **Digital Computers**
  - Computer programming
  - Systematic manipulation of information
    - 20th century

- **Matterware??**
  - Systematic manipulation of matter
    - 21st century

- **DNA Computers**
  - Molecular programming
Acknowledgments

- **Microsoft Research**
  - Andrew Phillips
    - Languages and tools for DNA strand displacement.

- **Bologna**
  - Pierluigi Zavattaro
    - Computational power of ‘chemical’ process algebras.
  - Cosimo Laneve
    - Reversibility in population models.

- **Aalborg**
  - Radu Mardare
    - Stochastic process algebra and logic.

- **Caltech**
  - Erik Winfree & Winfree Lab
    - DNA strand displacement as a computational method and technology.
  - David Soloveichik
    - The Programming Language of Chemical Kinetics.

- **U.Washington**
  - Georg Seelig, Yuan-Jyue Chen
    - Manufacturing two-domain gates.