Molecular Programming
The systematic manipulation of matter

Luca Cardelli
Microsoft Research & University of Oxford

Winton Symposium, Cambridge, 2015-09-28
Objectives

• The promises of Molecular Programming:
  • In Science & Medicine
  • In Engineering
  • In Computing

• The current practice of Molecular Programming
  • DNA technology
  • Molecular languages and tools
  • Example of a molecular algorithm
The Hardware Argument

Smaller and smaller things can be built
Smaller and Smaller

First working transistor
John Bardeen and Walter Brattain, Dec. 23, 1947

First integrated circuit

50 years later

25nm NAND flash
Intel&Micron, Jan. 2010. ~50 atoms

Single molecule transistor
Observation of molecular orbital gating
*Nature*, 2009; 462 (7276): 1039

Molecules on a chip

~10 Moore’s Law cycles left!
Building the *Smallest* Things

- How do we build structures that are by definition smaller than your tools?
- Basic answer: you can’t. Structures (and tools) should build themselves!
- By *programmed self-assembly*

www.youtube.com/watch?v=Ey7Emmdf7Y
Molecular IKEA

- Nature can self-assemble. Can we?

- “Dear IKEA, please send me a chest of drawers that assembles itself.”

- We need a magical material where the pieces are pre-programmed to fit into each other.

- At the molecular scale many such materials exist...

Programmed Self-Assembly

Proteins

DNA/RNA

Membranes
The Software Argument

Smaller and smaller things can be programmed
We can program...

- Information
  - Completely!
We can program...

• Forces
  • Completely!
    (Modulo sensors/actuators)
We can program...

- Matter
  - Completely and directly!
  - Currently: only DNA/RNA.

*It's like a 3D printer without the printer!*
[Andrew Hellington]
DNA

Sequence of Base Pairs (GACT alphabet)

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

DNA wrapping into chromosomes

Andromeda Galaxy
2.5 million light years
Zipping Along

- 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 base/second

Drew Berry
http://www.wehi.edu.au/wehi-tv
What can we do with “just” DNA?

- Organize ANY matter [caveats apply]
- Execute ANY kinetics [caveats: up to time scaling]
- Build Nano-Control Devices
- Interface to Biology
Organizing Any Matter

- Use one kind of programmable matter (e.g. DNA).
- To organize (almost) ANY matter through it.

"What we are really making are tiny DNA circuit boards that will be used to assemble other components."

Greg Wallraff, IBM

6 nm grid of individually addressable DNA pixels

Executing Any Kinetics

- The kinetics of any finite network of chemical reactions, can be implemented (physically) with especially programmed DNA molecules.

- Chemical reactions as an executable programming language for dynamical systems!
Building Nano-Control Devices

- All the components of nanocontrollers can already be built entirely and solely with DNA, and interfaced to the environment
Interfacing to Biology

- A doctor in each cell
The Biological Argument

Biological systems are already ‘molecularly programmed’
Abstract Machines of Biology

Gene Machine
Nucleotides

Protein Machine
Aminoacids

Membrane Machine
Phospholipids

Regulation
Make proteins
Send signals
Direct construction
Hold receptors, host reactions
Enact fusion, fission

Confinement, Storage
Bulk Transport

Metabolism, Propulsion
Signaling, Transport

Surface and Extracellular Features

Biological Languages

Molecular Interaction Maps

Gene Machine

Protein Machine

Membrane Machine

Gene Networks

Transport Networks
But ...

- Biology is programmable, but (mostly) not by us!

- Still work in progress:
  - Gene networks are being programmed in synthetic biology, but using existing ‘parts’
  - Protein networks are a good candidate, but we cannot yet effectively design proteins
  - Transport networks are being investigated for programming microfluidic devices that manipulate vesicles
Molecular Languages

... that we can execute
Our Assembly Language: Chemistry

- A Lingua Franca between Biology, Dynamical Systems, and Concurrent Languages

- Chemical Reaction Networks
  - $A + B \rightarrow C + D$ (the program)

- Ordinary Differential Equations
  - $\frac{d[A]}{dt} = -r[A][B]$ ... (the behavior)

- Rich analytical techniques based on Calculus

- But prone to combinatorial explosion
  - E.g., due to the peculiarities of protein interactions
How do we “run” Chemistry?

- Chemistry is not easily executable
  - “Please Mr Chemist, execute me this bunch of reactions that I just made up”

- Most molecular languages are not executable
  - They are descriptive (modeling) languages

- How can we execute molecular languages?
  - With real molecules?
  - That we can design ourselves?
  - And that we can buy on the web?
Molecular Programming with DNA

Building the cores of programmable molecular controllers
The role of DNA Computing

- **Non-goals**
  - Not to solve NP-complete problems with large vats of DNA
  - Not to replace silicon

- **Bootstrapping a carbon-based technology**
  - To precisely control the organization and dynamics of matter and information at the molecular level
  - DNA is our engineering material
    - Its biological origin is “accidental” (but convenient)
    - It is an information-bearing programmable material
    - Other such materials will be (are being) developed
Domains

- Subsequences on a DNA strand are called **domains**
  - *provided* they are “independent” of each other

- 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.
Short Domains

Reversible Hybridization

DNA double strand
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

\[ t \quad x \quad z \]

\[ t \quad x \quad y \]
Bad Match
Bad Match
Bad Match

Cannot proceed
Hence will undo
Two-Domain Architecture

• Signals: 1 toehold + 1 recognition region

• Gates: “top-nicked double strands” with open toeholds

Garbage collection “built into” the gate operation

Two-Domain DNA Strand Displacement

Luca Cardelli

Plasmidic Gate Technology

- Synthetic DNA is length-limited
  - Finite error probability at each nucleotide addition, hence ~ 200nt max
- Bacteria can replicate plasmids for us
  - Loops of DNA 1000’s nt, with extremely high fidelity
  - Practically no structural limitations on gate fan-in/fan-out

Only possible with two-domain architecture
Transducer
Transducer $x \rightarrow y$

Input

$tx$
Transducer $x \rightarrow y$

Ta is a private signal (a different ‘a’ for each $xy$ pair)

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
Transducer $x \rightarrow y$
So far, a $\textbf{tx}$ signal has produced an $\textbf{at}$ 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
(no proteins, no enzymes, no heat-cycling, etc.; just DNA in salty water)
Transducer $x \rightarrow y$
Join \( x + y \rightarrow z \)
Tools and Techniques

A software pipeline for Molecular Programming
Development Tools

MSRC Biological Computation Group

Visual DSD
A Development Environment for DNA Strand Displacement

A programming language for composable DNA circuits
Andrew Phillips and Luca Cardelli
Execution

A wetlab pipeline for Molecular Programming
Output of Design Process

- Domain structures
  - (DNA sequences to be determined)

“Ok, how do I run this for real”
From Structures to Sequences

DSD Structure → “Dot-Paren” representation

Output Sequences

“Ok, where do I buy these?”
“DNA Synthesis”
From Sequences to Molecules

- Copy&Paste from nupack
Molecules by FedEx

“Ok, how do I run these?”
Add Water
Execute (finally!)

- Fluorescence is your one-bit ‘print’ statement
Debugging

- A core dump

DNA strand length

Various processing stages

Calibration scale
Delivery!

Engineering Entropy-Driven Reactions and Networks Catalyzed by DNA
David Yu Zhang, et al.
Science 318, 1121 (2007);
DOI: 10.1126/science.1148532
A Molecular Algorithm

Running something interesting with DNA
Approximate Majority Algorithm

- Given two populations of agents (or molecules)
  - Randomly communicating by radio (or by collisions)
  - Reach an agreement about which population is in majority
  - By converting all the minority to the majority
  [Angluin et al., Distributed Computing, 2007]

- 3 rules of agent (or molecule) interaction
  - $X + Y \rightarrow B + B$
  - $B + X \rightarrow X + X$
  - $B + Y \rightarrow Y + Y$

“our program”
DNA Implementation, at U.W.

- Programmable chemical controllers made from DNA
  [Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik and Georg Seelig]
Final Remarks
A Brief History of DNA

Turing Machine, 1936

DNA, 3,800,000,000

Systematic manipulation of information

20th century

Computer programming

DNA Algorithm, 1994

Systematic manipulation of matter

21st century

Molecular programming

Transistor, 1947

Structural DNA Nonotech, 1982
Acknowledgments

- Microsoft Research
  - Andrew Phillips, Biological Computation Group

- Caltech
  - Winfree Lab

- U.Washington
  - Seelig Lab