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
Molecular Programming: The Hardware Aspect

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!

Race to the Bottom

Moore’s Law is approaching the single-molecule limit

Carlson’s Curve is the new exponential growth in technology

In both cases, we are now down to *molecules*
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*
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 to each other.

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

Programmed Self-Assembly

Proteins

Membranes

DNA/RNA
Molecular Programming: The Software Aspect

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! By self-assembly.
  - Currently: only DNA/RNA.

- But DNA is an amazing *material*

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

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

G-C Base Pair
Guanine-Cytosine

T-A Base Pair
Thymine-Adenine

Sequence of Base Pairs (GACT alphabet)
Astronomical

• DNA in each human cell:
  • 3 billion base pairs
  • 2 meters long, 2nm thick
  • 750 megabytes
  • folded into a 6µm ball,
    140 exabytes (million terabytes)/mm³

• 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
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 is special about DNA?

- There are many, many nanofabrication techniques and materials
- But only DNA (and RNA) can:
  - Organize ANY other matter \([\text{caveats apply}]\)
  - Execute ANY kinetics \([\text{caveats: up to time scaling}]\)
  - Assemble 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.
Constructing...
Crosslinking
Crosslinking
Crosslinking
Crosslinking
Crosslinking

In nature, crosslinking is deadly (blocks DNA replication).

In engineering, crosslinking is the key to using DNA as a construction material.
DNA Tiling

- 4 sticky ends
- Crosslinking

Construction and manipulation of DNA tiles in free space

Prakash
2D DNA Lattices

Chengde Mao
Purdue University, USA

N-point Stars
3D DNA Structures

Ned Seeman
NYU

Andrew Tuberfield
Oxford

3D Crystal

Tetrahedron
**DNA Origami**

*Folding* long (7000bp) naturally occurring (viral) ssDNA
By lots of short ‘staple’ strands that constrain it

Paul W K Rothemund
California Institute of Technology

Paul Rothemund’s “Disc with three holes” (2006)

Black/gray: 1 long viral strand (natural)
Color: many short staple strands (synthetic)

DNA Circuit Boards

- DNA origami are arrays of uniquely-addressable locations
  - Each staple is different and binds to a unique location on the origami
  - It can be extended with a unique sequence so that something else will attach uniquely to it.

- More generally, we can bind “DNA gates” to specific locations
  - And so connect them into “DNA circuits” on a grid
  - Only neighboring gates will interact

Dalchau, Chandran, Gopalkrishnan, Reif, Phillips. 2014

Some staples are attached to “green blobs” (as part of their synthesis) Other staples aren’t
DNA Storage (Read/Write)

DNA has a data density of 140 exabytes ($1.4 \times 10^{20}$ bytes) per $mm^3$ compared to state-of-the-art storage media that reaches ~500 megabytes ($5 \times 10^8$ bytes) per $mm^3$

DNA has been shown to be stable for millions of years

We have machines that can read (sequence) and write (synthesize) DNA. The Carslon Curve of “productivity” is growing much faster than Moore’s Law.

Cost of sequencing is decreasing rapidly ($1000$ whole human genome), while cost of synthesis is decreasing very slowly.

[Rob Carlson, www.synthesis.cc]
Sensing
Aptamers

Artificially evolved DNA molecules that stick to anything you like highly selectively
Pathogen Spotlights

• DNA aptamer binds to:
  - A) a pathogen
  - B) a molecule our immune system (when allergic) hates and immediately removes (eats) along with anything attached to it!

• Result: instant immunity
  - Mice poisoned with Anthrax plus aptamer (100% survival)
  - Mice poisoned with Anthrax (not so good)

Kary Mullis (incidentally, also Nobel prize for inventing the Polymerase Chain Reaction)
Actuating
DNA Tweezers

Hybridization
Strand Displacement

DNA nanomachines
Jonathan Bath & Andrew J. Turberfield
doi:10.1038/nnano.2007.100
DNA Walkers
Polymerization Motor

Rickettsia (spotted fever)

An autonomous polymerization motor powered by DNA hybridization

Triggered amplification by hybridization chain reaction

Robert M. Dicks and Niles A. Pierce

Download Acet Polymerization Associated with Spotted Fever Group Rickettsiae Infection of Vero Cells

Robert, Dicks, Niles, and Pierce, 2020, Nature Biotechnology
Hybridization Chain Reaction

Stable mixture of two hairpins

Initiator

Triggered amplification by hybridization chain reaction

Robert M. Dirks and Niles A. Pierce

Curing
Computational Drugs

- An automaton sequentially reading the string PPAP2B, GSTP1, PIM1, HPS (known cancer indicators) and sequentially cutting the DNA hairpin until a ssDNA drug (Vitravene) is released.

Vitravene (GGGTTGGCCTCTCTCTGG)

Based on restriction enzymes
Interfacing to Biology

• A doctor in each cell

*Fig. 1 Medicine in 2050: “Doctor in a Cell”*
Molecular Programming: The Biological Aspect

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

- **Gene Machine**
  - Nucleotides
  - Generate proteins
  - Send signals
  - Direct construction
  - Consume genome and regulators

- **Protein Machine**
  - Aminoacids
  - Metabolism and propulsion
  - Signaling and transport
  - Hold receptors, host reactions

- **Membrane Machine**
  - Phospholipids
  - Enact fusion and fission
  - Confinement and storage
  - Bulk transport

- **Glycan Machine**
  - Surface and extracellular features

---

Biological Languages

Gene Machine

Protein Machine

Membrane Machine

Molecular Interaction Maps

Gene Networks

Transport Networks

A

B

C

x

y
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
### Chemical Programming Examples

<table>
<thead>
<tr>
<th><strong>specification</strong></th>
<th><strong>program</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y := \min(X_1, X_2)$</td>
<td>$X_1 + X_2 \rightarrow Y$</td>
</tr>
<tr>
<td>$Y := \max(X_1, X_2)$</td>
<td>$X_1 \rightarrow L_1 + Y$</td>
</tr>
<tr>
<td></td>
<td>$X_2 \rightarrow L_2 + Y$</td>
</tr>
<tr>
<td></td>
<td>$L_1 + L_2 \rightarrow K$</td>
</tr>
<tr>
<td></td>
<td>$Y + K \rightarrow 0$</td>
</tr>
</tbody>
</table>

$max(X_1,X_2)= (X_1+X_2)-\min(X_1,X_2)$

(but is not computed “sequentially”: it is a form of concurrent computation)

*chemical reaction network*
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?
Towards High(er)-Level Languages

- **Gene Networks**
  - Synchronous Boolean networks
    - Stewart Kauffman, etc.
  - Asynchronous Boolean networks
    - René Thomas, etc.

- **Protein Networks**
  - Process Algebra (stochastic $\pi$-calculus etc.)
    - Priami, Regev-Shapiro, etc.
  - Graph Rewriting (kappa, BioNetGen etc.)
    - Danos-Laneve, Fontana & al., etc.

- **Membrane Networks**
  - Membrane Computing
    - Gheorghe Păun, etc.
  - Brane Calculi
    - Luca Cardelli, etc.

- Waiting for an architecture to run on...
Action Plan

- Building a full software/hardware pipeline for a new fundamental technology
  - Mathematical Foundations  [~ concurrency theory in the 80’s]
  - Programming Languages  [~ software engineering in the 70’s]
  - Analytical Methods and Tools  [~ formal methods in the 90’s]
  - Device Architecture and Manufacturing  [~ electronics in the 60’s]

- To realize the potential of Molecular Programming

- “With no alien technology” [David Soloveichik]

- This is largely a ‘software problem’ even when working on device design
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
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 $x \rightarrow y$
Transducer $x \rightarrow y$

Input $t \ x$

$\begin{array}{c}
\text{ta} \\
t \\
\end{array}$

$\begin{array}{c}
t \\
x \\
t \\
t \\
\end{array}$

$\begin{array}{c}
t \\
x \\
t \\
t \\
\end{array}$

$\begin{array}{c}
y \\
t \\
\end{array}$

$\begin{array}{c}
x \\
t \\
y \\
t \\
\end{array}$

$\begin{array}{c}
x \\
t \\
y \\
t \\
\end{array}$

Built by self-assembly!

**ta** is a *private* signal (a different ‘a’ for each xy pair)
Transducer $x \rightarrow y$
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
(no proteins, no enzymes, no heat-cycling, etc.; just DNA in salty water)
Transducer $x \rightarrow y$
Join $x + y \rightarrow z$
Large-scale Circuits (so far...)

Scaling Up Digital Circuit Computation with DNA Strand Displacement Cascades

Lulu Qian¹ and Erik Winfree²,³,⁴

3 JUNE 2011 VOL 332 SCIENCE

Neural network computation with DNA strand displacement cascades

Lulu Qian¹, Erik Winfree²,³,⁴ & Joshua Bruck³,⁴

368 | NATURE | VOL 475 | 21 JULY 2011
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
A Language for DNA Structures

- Describe the initial *structures* (not behavior)
Compute Species and Reactions

- Recursively computed from the initial structures
Reaction Graph and Export
Simulation

- Deterministic
- Stochastic (Gillespie)
- Probabilistic (CME)
- Linear Noise Approximation
- "JIT"
State Space Analysis

INITIAL STATE:

TERMINAL STATE:
Modelchecking

- Export to PRISM probabilistic modelchecker
Verification

- Quantitative theories of system equivalence and approximation.
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

Thermodynamic Synthesis

“Ok, where do I buy these?”

www.nupack.org
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
Output
Debugging

- A core dump

DNA strand length

Various processing stages

Calibration scale
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”
Surprisingly good (in fact, optimal)

- Fast: reaches agreement in $O(\log n)$ time w.h.p.
  - $O(n \log n)$ communications/collisions
  - Even when initially $#X = #Y!$ (stochastic symmetry breaking)

- Robust: true majority wins w.h.p.
  - If initial majority exceeds minority by $\omega(\sqrt{n \log n})$
  - Hence the agreement state is stable

Stochastic simulation of worst-case scenario with initially $#X = #Y$
Circuit component $X + Y \rightarrow 2B$
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

Transistor, 1947

Systematic manipulation of information

Computer programming

DNA Algorithm, 1994

Systematic manipulation of matter

Structural DNA Nonotech, 1982

Molecular programming

20th century

21st century
Acknowledgments

- Microsoft Research
  - Andrew Phillips, Biological Computation Group
- Caltech
  - Winfree Lab
- U.Washington
  - Seelig Lab
Questions?
Resources

- Biological Computation Group at MSR
  https://www.microsoft.com/en-us/research/group/biological-computation/

- Molecular Programming Project at Caltech
  http://molecular-programming.org/

- Georg Seelig’s DNA Nanotech Lab at U.W. CS&E
  http://homes.cs.washington.edu/~seelig/

- “DNA Computing and Molecular Programming” Conference Proceedings
  http://www.dna-computing.org/