Outline

• Molecular Structures
  o In Science and Engineering
  o Self-assembly

• Molecular Languages
  o Natural languages: proteins, genes, membranes
  o Modeling languages (systems biology)
  o Executable languages (nano-engineering)

• Molecular Computation
  o Molecular Programming
  o Molecular Compilation
Molecular Structures
Smaller and Smaller

First working transistor

First integrated circuit

50 years later

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

Single molecule transistor

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=Ey7Emmddf7Y
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
Molecular Languages
- natural languages -
Abstract Machines of Biochemistry

- **Gene Machine (Nucleotides)**
  - Regulation
  - Confront genome and regulators
  - Hold receptors, host reactions
  - Make proteins
  - Send signals
  - Direct construction

- **Protein Machine (Aminoacids)**
  - Metabolism, Propulsion
  - Signaling, Transport
  - Enact fusion, fission

- **Membrane Machine (Phospholipids)**
  - Confinement, Storage
  - Bulk Transport
  - Surface and Extracellular Features

Systems Biology (Networks)

Gene Machine

Gene Regulatory Networks

Biochemical Networks

Protein Machine

Transport Networks

Membrane Machine
Computation (Languages)
**The Protein Machine**

On/Off switches

Binding Sites

Switching accessible switches
- May cause other switches and binding sites to become (in)accessible.

Binding accessible sites
- May cause other switches and binding sites to become (in)accessible.

*cf. BioCalculs* [Kitano&Nagasaki], *κ*-calculus [Danos&Laneve]
Molecular Interaction Maps

Epidermal Growth Factor Receptor Pathway Map

- degradation
- recycling
- endocytosis
- small GTPase
- MAPK cascade
- transcription
- cell cycle
- Ca signaling
- GPCR-mediated transactivation

CellDesigner

The Gene Machine

Regulation of a gene influences transcription. The regulatory region has precise DNA sequences meant for binding regulators.

Transcription produces molecules (RNA or, through RNA, proteins) that bind to regulatory region of other genes (or that are end-products).
The Membrane Machine

Molecular transport and transformation through dynamic compartment fusion and fission.

Voet, Voet & Pratt
Fundamentals of Biochemistry
Wiley 1999. Ch10 Fig 10-22.
Molecular Languages
- modeling languages -
From Instructions to Programs

• We have seen the instruction sets:
  o Proteins – complexation, phosphorilation
  o Genes – activation, inhibition
  o Membranes – fusion, fission

• How do we combine them into programs?
  o I.e., into models (quantitative programs)

• How do we study their semantics?
  o I.e., their kinetics (quantitative semantics)
Chemistry

• Chemical reactions
  o \( A + B \xrightarrow{r} C + D \) (a program)

• Ordinary Differential Equations
  o \( \frac{d[A]}{dt} = -r[A][B] \ldots \) (a semantics)

• Rich analytical techniques based on Calculus

• But prone to combinatorial explosion
  o E.g., due to the peculiarities of protein interactions
High(er)-Level Languages

• Gene Networks
  o Synchronous Boolean networks
    • Stewart Kauffman, etc.
  o Asynchronous Boolean networks
    • René Thomas, etc.

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

• Membrane Networks
  o Membrane Computing
    • Gheorghe Păun, etc.
  o Brane Calculi
    • Luca Cardelli, etc.
Reactions vs. Reagents

Says what “A” *does*.  
\[ r: A + B \rightarrow_{k_1} C + D \]
\[ s: C + D \rightarrow_{k_2} A + B \]

Does A become C or D?

Says what “A” *is*.

```
A = !r_{k_1} \cdot C  
C = ?s_{k_2} \cdot A  
B = ?r_{k_1} \cdot D  
D = !s_{k_2} \cdot B  
```

The same “math model”

CTMC

Reaction oriented

1 line per reaction

Interaction oriented

1 line per agent

A becomes C not D!
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** \((A = !r; C)\) (Process Algebra)
  - Reduces combinatorial complexity of models by combining independent submodels connected by interactions.

- **Rule-Based** \((A{\{\ldots\}}:B{\{p\}} \rightarrow A{\{p\}}:B{\{\ldots\}})\) (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)

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)
But what about Execution?

• Chemistry is not easily executable
  o Please Mr Chemist, execute me these reactions that I just made up

• Similarly, the molecular languages seen so far are descriptive (modeling) languages

• How can we actually execute molecular languages? With real molecules?
Molecular Languages
- executable languages -
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**


**Target molecule**
Constructing

Crosslinking

Chengde Mao, Purdue

Andrew Turberfield, Oxford
Actuating

DNA tweezers

DNA walkers

Bernard Yurke, Boise State
Computing

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

• Computation in the 'kernel' of the system

• Compositionality in the kernel
  - The components should use uniform inputs and outputs
  - 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]

```
<table>
<thead>
<tr>
<th>Part</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0040:prom</td>
<td>promoter</td>
</tr>
<tr>
<td>b0034:rbs</td>
<td>ribosome</td>
</tr>
<tr>
<td>c0040:pcr</td>
<td>PCR</td>
</tr>
<tr>
<td>b0015:ter</td>
<td>terminator</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>Part</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>prom&lt;neg(C)&gt;</td>
<td>promoter</td>
</tr>
<tr>
<td>rbs</td>
<td>ribosome</td>
</tr>
<tr>
<td>pcr&lt;codes(A)&gt;</td>
<td>PCR</td>
</tr>
<tr>
<td>ter</td>
<td>terminator</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>prom&lt;neg(A)&gt;</td>
<td>promoter</td>
</tr>
<tr>
<td>rbs</td>
<td>ribosome</td>
</tr>
<tr>
<td>pcr&lt;codes(B)&gt;</td>
<td>PCR</td>
</tr>
<tr>
<td>ter</td>
<td>terminator</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>prom&lt;neg(B)&gt;</td>
<td>promoter</td>
</tr>
<tr>
<td>rbs</td>
<td>ribosome</td>
</tr>
<tr>
<td>pcr&lt;codes(C)&gt;</td>
<td>PCR</td>
</tr>
<tr>
<td>ter</td>
<td>terminator</td>
</tr>
</tbody>
</table>
```
“Autonomous” Computing

(Nano-engineering)

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

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

A doctor in each cell

Fig. 1 Medicine in 2050: “Doctor in a Cell”

Molecules and computation

Ehud Shapiro
Rivka Adar
Kobi Benenson
Gregory Linshitz
Aviv Regev
William Silverman
RNA operation 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.
Molecular Computation
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.
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

Diagram showing vectors with labels t, x, y, and z.
Bad Match
Bad Match

Cannot proceed
Hence will undo
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**

$\text{ta}$ is a *private* signal (a different ‘a’ for each $xy$ pair)

Built by self-assembly!
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 $x \rightarrow y$
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+ inputs (with 1+ collectors).
• Easily generalized to 2+ outputs.

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

```python
def bind = k*exp(-DeltaG_over_RT) /*/s*/
def unbind = k*exp(DeltaG_over_RT) /*/s*/
new tbind, unbind
new unbind, unbind
newRD 0,0,0,0

def onex = 50.0

(* x + y -> y + z *)
def Cat(N, x, y, z) =
  new a
  [(1.5%) * t ^={N}[(x t) := y[=][a]]
   [(1.5%) * x := [(y t) := z[=][a]]
   [(2.0%) * u := a]
   [(2.0%) * v := a]

  def Rep([a, x, y] =)
    (3.0%) * t ^={x}[(x)[=]<0]]

  onex = Cat(N, x, y, z)
  Cat(onex, x, y, z)
  Rep(onex, y, z)
  onex = <t x>
  onex = <t y>
```

Diagram showing DNA reactions and their corresponding chemical structures.
Experiments

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

$X + Y \rightarrow Y + B$

35°C

$1x = 50\text{nM}$

Yuan-Jyue Chen and Georg Seelig
U.Washington.

<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>R2</td>
<td>3x</td>
</tr>
</tbody>
</table>

Output (nM)
Molecular Programming Workflow
Molecular Compilation

Circuit Design

Intermediate Language

Gate Design

Structural Language

Device Design

- Boolean Networks
- Petri Nets
- Strand Algebra

- 4-domain Signals
- 3-domain Signals
- 2-domain Signals

Intermediate Language #2
Circuits to Signals and Gates

- E.g., a simple Petri Net fork transition

- In Strand Algebra:  \( x \mid ([x].[y,z])^* \)
**Signals and Gates to Structures**

- Visual DSD [Andrew Phillips]

![Diagram](image)

```
[object]
```

Actual script:

directive sample 5000.0 1000
directive plot sum(<_ T^ xb>); sum(<_ T^ yb>); sum(<_ T^ zb>)
def scaling = 1000
def bind = 0.0003/(float_of_int scaling) (* /nM/s *) (* =3*10^5 /M/s *)
def unbind = 0.1126 (* /s *)
new T@bind,unbind

def F1x2(N,Xb,Yb,Zb) =
    new a
    ( N * T^ [Xb T^] <Yb> ; [a T^] <Zb>
    | N * <T^ a T^>
    )
    ( F1x2(10^scaling,xb,yb,zb)
    | (1^scaling)^* <xh T^ xb>
    )
```
Signals and Gates to Structures

• Fork gate: the reactions

[Diagram showing fork gate with x input, y output, and z output]
Signals and Gates to Structures

- Fork gate: the reaction graph
Signals and Gates to Structures

- Fork gate: the behavior
Signals and Gates to Structures

- Fork gate: check

Ok, I want this
Structures to Sequences

NUPACK BETA
nucleic acid package

www.nupack.org

Input Structure

Output Sequences

Ok, I want these
DNA Synthesis

DNA synthesis commonly refers to: DNA replication - DNA biosynthesis (in vivo DNA amplification); Polymerase chain reaction - enzymatic DNA synthesis (in vivo).
Sequences to Molecules
Molecules by Mail

Custom Oligonucleotide Synthesis

Innovation and Precision in Nucleic Acid Synthesis

www.idtdna.com
Add Water
Execution

- Fluorescence is your ‘print’ statement
Debugging
Publishing!

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

Diagram A and B depict the reaction pathways and catalytic inputs.

Diagram C shows a gel electrophoresis image with bands labeled.

Diagram D illustrates the flow of the reaction components.

Diagram E displays a graph of fluorescence over time with data and simulation lines.
Conclusions
A Brief History of DNA

Turing Machine, 1936
Transistor, 1947
DNA, 33,800,000,000
Software
systematic manipulation of information
20th century
Matterware?
systematic manipulation of matter
21st century
DNA Computers
Molecular programming
DNA Algorithm, 1994
Structural DNA, 1982

Digital Computers
Computer programming
Acknowledgments

• Microsoft Research
  o Andrew Phillips

• Caltech
  o Winfree Lab

• U.Washington
  o Seelig Lab