Outline

• Molecular Structures
  o Getting smaller
  o Self-assembly

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

• Molecular Compilation
  o Intermediate Languages
  o Analysis Tools and Techniques
  o Nano-programming workflow
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

Make proteins
Send signals

Protein

Machine

Aminoacids

Confound genome and regulators

Hold receptors, host reactions

Enact fusion, fission

Membrane

Machine

Phospholipids

Regulation

Confinement, Storage
Bulk Transport

Metabolism, Propulsion
Signaling, Transport

Surface and Extracellular Features
Data Structures of Biochemistry

Gene Machine

Strings

Protein Machine

Membrane Machine

Hierarchical Multisets

Trees

Records

Glycan Machine
Languages of Biochemistry

Biochemical Networks

Protein Machine

Membrane Machine

Gene Machine

Gene Regulatory Networks

Transport Networks
The Protein Machine

On/Off switches

- 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. BioCalculus [Kitano&Nagasaki], κ-calculus [Danos&Laneve]
Molecular Interaction Maps (Kohn/Kitano)

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).

Human (and mammalian) Genome Size
3Gbp (Giga base pairs) 750MB @ 4bp/Byte (CD)
Non-repetitive: 1Gbp 250MB
In genes: 320Mbp 80MB
Coding: 160Mbp 40MB
Protein-coding genes: 30,000–40,000

M. Genitalium (smallest true organism)
580,073bp 145KB (eBook)

E. Coli (bacteria): 4Mbp 1MB (floppy)
Yeast (eukarya): 12Mbp 3MB (MP3 song)
Wheat 17Gbp 4.25GB (DVD)
Function of a Regulatory Region

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.
Bitonal Diagrams

**Mito: special cases**

- **Mate**
  - P Q

- **Exo**
  - P Q

- **Endo**
  - P Q

- **Zero case**
  - P R

- **Drip**
  - P R

- **Bud**
  - P R

- **Pino**
  - Q R

- **Phago**
  - R Q
... in 3D

Controlled by surface proteins
Integration

Gene Machine

Slow

Digital

Turing Completeness

Fast

Protein Machine

Small

Analog

Big

Membrane Machine
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 \( A + B \rightarrow_r C + D \) (a program)

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

• Rich analytical techniques based on Calculus

• But prone to combinatorial explosion
  o Due to the peculiarities of protein interactions
High(er)–Level Languages

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

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

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

Says what “A” does.

\[ r: A + B \rightarrow_{k1} C + D \]
\[ s: C + D \rightarrow_{k2} A + B \]

Does A become C or D?

Says what “A” is.

A = !r_{k1} ; C
C = ?s_{k2} ; A
B = ?r_{k1} ; D
D = !s_{k2} ; B

A becomes C not D!

The same “math model”

CTMC

Reaction oriented

Interaction oriented

1 line per reaction

1 line per agent
Formal Connections

These diagrams commute via appropriate maps.

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

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

- **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**.
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

DNA wrapping into chromosomes

Andromeda Galaxy 2.5 million light years away
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 bases/second

Drew Berry
http://www.wehi.edu.au/wehi-tv
**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.*

Constructing

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

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

• MIT Registry of Standard Biological Parts

• GenoCAD
  o Meaningful sequences [Cai et al.]

• GEC
  o [Pedersen & Phillips]
“Autonomous” Computing

• 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 their 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”
Autonomous DNA Computing
Why Compute with DNA?

• 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 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

\[ t \]

\[ t \]

x y
Bad Match
Bad Match
Bad Match

Cannot proceed
Hence will undo
Four–Domain Architecture

No “garbage collection” (active waste removal)

DNA as a universal substrate for chemical kinetics

David Soloveichik, Georg Seelig, and Erik Winfree

PNAS | March 23, 2010 | vol. 107 | no. 12 | 5393–5398
Three-Domain Architecture

With garbage collection (separate pass)

a fresh; \( x_h \) generic

\[ x \mid x.y \rightarrow y \]

Strand Algebras for DNA Computing

Luca Cardelli

DNA Computing and Molecular Programming.
15th International Conference, DNA 15,
“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**

$\downarrow$

$t \ x$

$t \ a$

$y \ t$

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

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

**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
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 $x \to y$
Transducer $x \rightarrow y$

Done.

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

- Easily generalized to 2+ inputs (with 1+ collectors).
- Easily generalized to 2+ outputs.

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

• An intermediate language for molecular computing
  o Signals: $x$
  o Gates: $[x_1, \ldots, x_n].[y_1, \ldots, y_m]$
  o Parallel composition: $|$  
  o Populations: $(\ldots)^*$

$\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad
Petri Net Transitions

• Computing power equivalent to Petri Nets
  - Not Turing complete, but as good as chemistry itself.
  - The correspondence is not completely trivial: gates are consumed by activation, hence a persistent Petri net transition requires a stable population of gates.

• Hence, many other mechanisms are expressible
  - E.g. Boolean networks
Compilation Issues

Monolithic Compilers

Language Design #1

Boolean Networks

Language Implementation #1

Language Design #2

Petri Nets

Language Implementation #2

Language Design #3

Language Implementation #3

...
Compilation Issues

Circuit Design

Intermediate Language

Gate Design

Structural Language

Device Design

Booleans

Petri Nets

Strand Algebra

4-domain Signals

3-domain Signals

2-domain Signals

Intermediate Language #2
Optimization Issues

• Reduce number of species

• Optimize kinetics

• Etc.
Verification Issues

- **Environment**
  - The nano-environment is messy (stochastic noise, failures, etc.)
  - But we should at least ensure our designs are *logically correct*

- **Verifying Components**
  - Reversible reactions (infinite traces)
  - Interferences (deadlocks etc.) between copies of the same gate
  - Interferences (deadlocks etc.) between copies of different gates
  - Removal of active byproducts (garbage collection) is tricky

- **Verifying Populations**
  - Gates come in (large) populations
  - Each population *shares private domains* (technologically unavoidable)
  - Correctness of populations means proofs with large state spaces
Correctness

- The spec of a transducer:

\[ x.y \mid x \rightarrow y \]

- Is it true at all?
- Is it true _possibly, necessarily, or probabilistically_?
- Is it true in the context of a _population of identical transducers_?
- Is it true _in all possible contexts_?
- If false, does it become true for _infinite populations_?
Interfering Transducers

• Let a be the private transducer domain, but let’s share it between x.y and y.x

• Interference: \( x.a.y \mid y.a.x \mid x \not\leftrightarrow \forall x \)

• But still: \( x.a.y \mid y.a.x \mid x \mid y \rightarrow \forall x \mid y \)

• A large population of such gates in practice does not deadlock easily.

• The wisdom of crowds: individuals can be wrong, but the population is all right.
Model checking DNA Systems

- Using the PRISM stochastic model checker
  - Termination probability of interfering transducers
    \[ x \mid x.a \] \[ y \mid y.a \]

Design and Analysis of DNA Circuits using Probabilistic Model Checking.
http://qav.comlab.ox.ac.uk/papers/dna-pmc.pdf. September 2010
Molecular Programming Workflow
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]

3-domain gate

3-domain input

```
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
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

Input Structure

Output Sequences

Ok, I want these

www.nupack.org
DNA Synthesis
Sequences to Molecules
Molecules by Mail

Custom Oligonucleotide Synthesis

Innovation and Precision in Nucleic Acid Synthesis
Add Water
Execution

- Fluorescence is your ‘print’ statement
Output
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
Conclusions
Summary

• Molecular Structures
  o Hard to build... but they can build themselves!

• Molecular Languages
  o Concurrent, quantitative

• Molecular Compilation
  o Molecular architectures, verification, optimization

• Molecular Programming
  o In silico, in vitro, in vivo...
Acknowledgments

• Microsoft Research
  o Andrew Phillips

• Caltech
  o Winfree Lab

• U.Washington
  o Seelig Lab