Software for Programming Cells

Andrew Phillips
Biological Computation Group
Microsoft Research

Image courtesy of James Brown, Haseloff Lab, University of Cambridge
DNA: Cell Machine Code

Thymine - Adenine

Guanine - Cytosine
The Cell: The Ultimate Computer

- 1GB in a millionth of a mm$^3$
- Massively parallel
- Robust to failure
- Self-repairing
- Efficient power supply
- Can completely reproduce itself
DNA: A Multi-Platform Code

Can function across species

Glowing jellyfish

Glowing bacteria
DNA can completely transform a cell

Craig Venter and the synthetic bacteria: such cells might one day manufacture renewable fuels.

M. mycoides

M. capricolum

Extraction

Sequencing

Insertion

Transformation

...GTTTCTCCATACCCGTTTTTTTGGGCTAGC...

Synthesis
Potential for Programming Cells

• Health
  – Control viruses and cancers
  – Produce pharmaceutical ingredients

• Energy
  – Renewable energy sources

• Understand Life
  – How living things work
Challenges for Programming Cells

- Programming cells is hugely difficult
  - We still don’t fully understand the code
  - Learning how to write and compile programs to DNA
  - Issues of reliability, toxicity, strain on the host cell
  - Medical breakthroughs mostly on cell cultures

- Programs are increasingly complex
  - Can no longer be designed by trial and error
  - Computer software is needed to accelerate progress
Software for Programming Cells

Genetic Engineering of Cells (GEC) Language

Step 1: Program device design
Step 2: Compile device behaviour
Step 3: Simulate device
Step 4: Compile device to DNA
Step 5: Insert DNA into cells

Low-level DNA language

A simplified view of DNA instructions
High-level DNA language

Given a design, automatically determine the DNA
Programming a receiver device

Signal enters cell, binds to Receiver, activates GFP

```plaintext
1  cell
2  [ prom; rbs; pcr<codes(Receiver)>; ter
3    | prom<pos(Receiver-Signal)>;
4    | rbs; pcr<codes(gfp)>; ter
5    | Receiver + Signal <-> Receiver-Signal
6  ]
7  | Signal -> cell[Signal]
8  | cell[Signal] -> Signal
```
Receiver device: parts

<table>
<thead>
<tr>
<th>Database</th>
<th>GEC</th>
<th>LBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>c0040</td>
<td>pcr</td>
<td>codes(tetR, 0.01)</td>
</tr>
<tr>
<td>c0080</td>
<td>pcr</td>
<td>codes(araC, 0.01)</td>
</tr>
<tr>
<td>c0012</td>
<td>pcr</td>
<td>codes(lacI, 0.01)</td>
</tr>
<tr>
<td>c0unknow2</td>
<td>pcr</td>
<td>codes(unknown2, 0.001)</td>
</tr>
<tr>
<td>c0061</td>
<td>pcr</td>
<td>codes(luxI, 0.01)</td>
</tr>
<tr>
<td>c0062</td>
<td>pcr</td>
<td>codes(luxR, 0.01)</td>
</tr>
<tr>
<td>c0079</td>
<td>pcr</td>
<td>codes(lasR, 0.01)</td>
</tr>
<tr>
<td>c0078</td>
<td>pcr</td>
<td>codes(lasI, 0.01)</td>
</tr>
<tr>
<td>c0unknow3</td>
<td>pcr</td>
<td>codes(ccdB, 0.005)</td>
</tr>
<tr>
<td>c0unknow4</td>
<td>pcr</td>
<td>codes(ccdA, 0.1)</td>
</tr>
<tr>
<td>i723020</td>
<td>prom</td>
<td>pos(toluen-xyR, 0.001, 0.001, 1.0), con(0.0001)</td>
</tr>
<tr>
<td>r0051</td>
<td>prom</td>
<td>neg(c, 1.0, 0.5, 0.00005), con(0.12)</td>
</tr>
<tr>
<td>r0040</td>
<td>prom</td>
<td>neg(tetR, 1.0, 0.5, 0.00005), con(0.09)</td>
</tr>
<tr>
<td>r0unknow1</td>
<td>prom</td>
<td>neg(unknown1, 1.0, 0.005, 0.001), con(0.04)</td>
</tr>
<tr>
<td>r0080</td>
<td>prom</td>
<td>neg(araC, 1.0, 0.000001, 0.0001), pos(araC-arebinose, 0.001, 0.001, 1.0), con(0.1)</td>
</tr>
<tr>
<td>r0011</td>
<td>prom</td>
<td>neg(lacI, 1.0, 0.5, 0.000005), con(0.1)</td>
</tr>
<tr>
<td>r0062</td>
<td>prom</td>
<td>pos(lasR-m30C12HSL, 1.0, 0.6, 0.1), pos(luxR-m30C6HSL, 1.0, 0.6, 0.1), con(0.01)</td>
</tr>
<tr>
<td>r0080</td>
<td>prom</td>
<td>pos(lasR-m30C12HSL, 1.0, 0.8, 0.1), con(0.01)</td>
</tr>
<tr>
<td>r0059</td>
<td>prom</td>
<td>pos(cvir-m30C6HSL, 1.0, 0.8, 0.1), con(0.01)</td>
</tr>
<tr>
<td>b0034</td>
<td>rbs</td>
<td>rate(0.1)</td>
</tr>
<tr>
<td>b0015</td>
<td>ter</td>
<td></td>
</tr>
</tbody>
</table>
Receiver device: reactions
Receiver device: simulation
**Receiver device: characterisation**

### Model Simulation

1. \( C_0 + [X] \xrightarrow{k_{ex}} [X_0] \)
2. \( [G] + [X] \xrightarrow{k_{ex}} [G \cdot X_0] \)
3. \( [G \cdot X_0] \xrightarrow{k_{1}} [G \cdot X_0] + [M_{GFP}] \)
4. \( [G] \xrightarrow{k_{1}} [G] + [M_{GFP}] \)
5. \( [M_{GFP}] \xrightarrow{k_{2}} 0 \)
6. \( [M_{X}] \xrightarrow{k_{1}} [M_{X}] + [X] \)
7. \( [X] \xrightarrow{k_{2}} 0 \)

\[
\begin{align*}
\frac{d[C_0]}{dt} &= -d_{rec}[C_0] \\
\frac{d[P_{GFP}]}{dt} &= a \left( \frac{[C_0] + k_{1}k_{2}}{k + [C_0]} \right) - d_{GFP}[P_{GFP}] \\
\end{align*}
\]

### Experimental Data

**Characterisation of the luxR receiver device**

**Total GFP in Time**

**Concentration (mM)**
Spatial Receiver Device

Model Simulation

Experimental Data
Programming Turing Patterns

With Neil Dalchau, James Brown, Stephen Emmott

Modelling biophysics: thresholding

With Tim Rudge, James Brown, Jim Haseloff
Modelling biophysics: thresholding
Programming DNA interactions (A-T,G-C)
Nucleic acid circuits

Organization of Intracellular Reactions with Rationally Designed RNA Assemblies

Camille J. Delebecque, Ariel B. Lindner, Pamela A. Silver, Faisal A. Aldaye

A Logic-Gated Nanorobot for Targeted Transport of Molecular Payloads

Shawn M. Douglas, Ido Bachelet, George M. Church

SCIENCE VOL 335 17 FEBRUARY 2012

Selective cell death mediated by small conditional RNAs

Suvin Venkataraman, Robert M. Dirks, Christine T. Ueda, Niles A. Pierce

PNAS | September 28, 2010
DNA strand displacement

Computation solely in terms of nucleic acids
Interactions programmed by choice of sequence

Bernard Yurke
Software for Programming DNA/RNA

DNA Strand Displacement (DSD) Language

Step 1: Program circuit design
Step 2: Compile circuit behaviour
Step 3: Simulate circuit
Step 4: Compile circuit to DNA/RNA
Step 5: Insert DNA into cells

Lakin, Parker, Cardelli, Kwiatkowska, Phillips. Royal Society Interface, 2012
Programming a DNA Logic Circuit

Output = Input1 AND Input2
Strand displacement logic gate

Output = Input1 AND Input2

1. def Input1() = <1^ 2>
2. def Input2() = <3 4^>
3. def AND() = \{1^*\}[2 3]{4^*}
DNA Logic circuit: Reactions
DNA Logic circuit: Simulation
DNA Logic circuit: Sequences
“In addition to biochemistry laboratory techniques, computer science techniques were essential.”

“Computer simulations of seesaw gate circuitry optimized the design and correlated experimental data.”
Biological Computation Group

- DNA Computing
- Synthetic Biology
- Development
- Immunology

A common language runtime for Biological Computation

DSD, GEC, Biocharts, SPIM

Software for Programming Cells

Applications

DNA Machine Code

Systems

IF dark signal-out
ELSEIF (signal-in AND light-in)
MAKE Pigment

Devices

Inputs

Logic Gates

Outputs

Genetic Parts

Promoters

Coding Regions

Terminators

Hierarchy courtesy of James Brown Haseloff Lab, University of Cambridge
Programming a Receiver Device

```
directive sample 10000.0 1000
directive plot cell[gfp]; Signal

cell
| prom; rbs; pcr<codes(Receiver>>; ter 
| prom<pos(Receiver-Signal)>>;
| rbs; pcr<codes(gfp)>>; ter
| Receiver + Signal -> Receiver-Signal
| Receiver-Signal -> Receiver + Signal
|
| Signal -> cell[Signal]
| cell[Signal] -> Signal
| initPop Signal 100.0
```

Diagram showing the interaction between Signal, Receiver, and GFP.
Spatial receiver device

Model Simulation

Experimental Data