TASBE: A Tool-Chain for Accelerating Synthetic Biology Engineering


3rd Int’l Workshop on Bio-Design Automation
June, 2011
Vision: WYSIWYG Synthetic Biology

Bioengineering should be like document preparation:
Vision: WYSIWYG Synthetic Biology

Bioengineering should be like document preparation:
Why is this important?

- Breaking the complexity barrier:
  - Multiplication of research impact
  - Reduction of barriers to entry

*Sampling of systems in publications with experimental circuits*
Why a tool-chain?

Organism Level Description

This gap is too big to cross with a single method!

Cells
The TASBE architecture:

Organism Level Description

<table>
<thead>
<tr>
<th>Description</th>
<th>Abstract Genetic Regulatory Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNA Parts Sequence</td>
<td>DNA Parts Sequence</td>
</tr>
<tr>
<td>Assembly Instructions</td>
<td>Assembly Instructions</td>
</tr>
<tr>
<td>Cells</td>
<td>Cells</td>
</tr>
</tbody>
</table>

High level simulator

Coarse chemical simulator

Detailed chemical simulator

Testing

If detect explosives:
- emit signal
- if signal > threshold: glow red

Modular architecture also open for flexible choice of organisms, protocols, methods, ...
A Tool-Chain Example

(yellow (not (cyan (Dox))))
A Tool-Chain Example

(yellow (not (cyan (Dox))))
A Tool-Chain Example

(yellow (not (cyan (Dox)))))
A Tool-Chain Example

(yellow (not (cyan (Dox)))))

Organism Level Description
High level simulator

High Level Description
Coarse chemical simulator

Abstract Genetic Regulatory Network
Detailed chemical simulator

DNA Parts Sequence

Assembly Instructions
Testing

Cells
(yellow (not (cyan (Dox)))))
A Tool-Chain Example

(yellow (not (cyan (Dox))))
Current state of the tool-chain:

- End-to-end software integration
- Automated designs match hand-generated systems verified *in vivo*
- Some protocols automated

*Next: verification of automated design & assembly*
Advances on Two Key Problems:

High level simulator

Coarse chemical simulator

Detailed chemical simulator

Testing

Compilation & Optimization

Characterization of Transfer Curves

Organism Level Description

High Level Description

Abstract Genetic Regulatory Network

DNA Parts Sequence

Assembly Instructions

Cells
Advances on Two Key Problems:

Compilation & Optimization

Organism Level Description

High Level Description

Abstract Genetic Regulatory Network

DNA Parts Sequence

Assembly Instructions

Cells

High level simulator

Coarse chemical simulator

Detailed chemical simulator

Testing

If detect explosives:
emit signal
If signal > threshold:
glow red
Transcriptional Logic

Decay

Protein

Signal = Concentration

Stabilizes at decay = production

Alternatives:

PoPS
RNA concentration
BioCompiler Overview

• BioCompiler converts high level program to abstract GRN

• Motifs map high level operators to parameterized implementation in biology.

• As with all compilers, the initial mapping can be greatly optimized.
Motif-Based Compilation

- High-level primitives map to GRN design motifs
  - e.g. logical operators:

\[
\text{(primitive not (boolean) boolean :grn-motif ((P high R- arg0 value T)))}
\]
Motif-Based Compilation

- High-level primitives map to GRN design motifs
  - e.g. logical operators, actuators:

  ```lisp
  (primitive green (boolean) boolean :side-effect :type-constraints ((= value arg0)) :grn-motif ((P R+ arg0 GFP|arg0 value T)))
  ```

  ![Diagram showing the mapping of high-level primitives to GRN motifs](image)
Motif-Based Compilation

- High-level primitives map to GRN design motifs
  - e.g. logical operators, actuators, sensors:

  \[
  \text{(primitive IPTG () boolean)} \\
  \text{:grn-motif ((P high \text{Lacl}|\text{boolean T})} \\
  \text{(RXN (IPTG|\text{boolean}) represses \text{Lacl})} \\
  \text{(P high R- \text{Lacl value T}))}
  \]
Motif-Based Compilation

- Functional program gives dataflow computation:

  (green (not (IPTG)))
Motif-Based Compilation

- Functional program gives dataflow computation:

\[(\text{green } (\text{not } (\text{IPTG})))\]
Motif-Based Compilation

- Operators translated to motifs:

```
IPTG → not → green
```
Motif-Based Compilation

- Operators translated to motifs:

```
IPTG  not  green
```

```
LacI  outputs  arg0  outputs  GFP  outputs
```

```
Motif-Based Compilation

- Operators translated to motifs:

```
IPTG → not → green
```

```
outputs
```

```
A
```

```
B
```

```
GFP
```

```
arg0
```

```
LacI
```

```
IPTG
```

```
outputs
```

```
IPTG
```

```
outputs
```

```
GFP
```

```
outputs
```

```
IPTG
```

```
LacI
```

```
A
```

```
B
```

```
GFP
```

```
outputs
```

```
IPTG
```

```
LacI
```

```
A
```

```
B
```

```
GFP
```

```
outputs
```

```
IPTG
```

```
LacI
```

```
A
```

```
B
```

```
GFP
```

```
outputs
```
Optimization

Diagram showing the flow of IPTG through the processes of Lacl, A, B, and GFP.
Optimization

Copy Propagation
Optimization

Copy Propagation

Dead Code Elimination
Optimization

Copy Propagation

Dead Code Elimination

Dead Code Elimination
Complex System: FeedbackLatch

(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
         (o-bar boolean (not (or s o))))
    o))

(green (sr-latch (aTc) (IPTG)))
Complex System: Feedback Latch

(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar))))
   (o-bar boolean (not (or s o))))
  o))

(green (sr-latch (aTc) (IPTG)))

Unoptimized: 15 functional units, 13 transcription factors
(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
    (o-bar boolean (not (or s o))))
    o))

(green (sr-latch (aTc) (IPTG)))

Unoptimized: 15 functional units, 13 transcription factors

Copy Propagation
(def sr-latch (s r)
  (letf+ ((o boolean (not (or r o-bar)))
    (o-bar boolean (not (or s o))))
    o))

(green (sr-latch (aTc) (IPTG)))

Unoptimized: 15 functional units, 13 transcription factors
Optimization of Complex Designs

```
(def sr-latch (s r)
  (letféd+ ((o boolean (not (or r o-bar))))
    (o-bar boolean (not (or s o))))
  o))

(green (sr-latch (aTc) (IPTG)))
```

Unoptimized: 15 functional units, 13 transcription factors
(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar))))
    (o-bar boolean (not (or s o))))
)
(green (sr-latch (aTc) (IPTG)))

Unoptimized: 15 functional units, 13 transcription factors
Optimization of Complex Designs

(def sr-latch (s r)
  (letf ed+ ((o boolean (not (or r o-bar))))
    (o-bar boolean (not (or s o)))))

(green (sr-latch (aTc) (IPTG)))

Unoptimized: 15 functional units, 13 transcription factors
Optimization of Complex Designs

(def sr-latch (s r)
  (letf ed+ ((o boolean (not (or r o-bar)))))
  (o-bar boolean (not (or s o))))

(green (sr-latch (aTc) (IPTG)))

Unoptimized: 15 functional units, 13 transcription factors

Common Subexp. Elim.
(def sr-latch (s r)
  (letfedi ((o boolean (not (or r o-bar))))
    (o-bar boolean (not (or s o)))))

(green (sr-latch (aTc) (IPTG)))

Unoptimized: 15 functional units, 13 transcription factors

Dead Code Elimination
Optimization of Complex Designs

(def sr-latch (s r)
  (letfedge ((o boolean (not (or r o-bar)))))
  (o-bar boolean (not (or s o)))))

(green (sr-latch (aTc) (IPTG)))

Unoptimized: 15 functional units, 13 transcription factors
Optimization of Complex Designs

```
(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar))))
    (o-bar boolean (not (or s o))))
  o))

(green (sr-latch (aTc) (IPTG)))
```

Unoptimized: 15 functional units, 13 transcription factors
(def sr-latch (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
    (o-bar boolean (not (or s o))))
    o))

(green (sr-latch (aTc) (IPTG)))

Unoptimized: 15 functional units, 13 transcription factors
Optimization of Complex Designs

(def sr-latch (s r)
  (letfeder+ ((o boolean (not (or r o-bar))))
    (o-bar boolean (not (or s o))))
  o))

(green (sr-latch (aTc) (IPTG)))

Final Optimized:
5 functional units
4 transcription factors

Unoptimized: 15 functional units, 13 transcription factors
Compilation & Optimization Results:

• Automated GRN design for arbitrary boolean logic and feedback systems

• Optimization competitive with human experts:
  – Test systems have 25% to 71% complexity reduction
  – Optimized systems homologous with hand design
Advances on Two Key Problems:

Organism Level Description

High Level Description

Abstract Genetic Regulatory Network

DNA Parts Sequence

Assembly Instructions

Cells

High level simulator

Coarse chemical simulator

Detailed chemical simulator

Testing

If detect explosives:
emit signal
If signal > threshold:
glow red

Characterization of Transfer Curves
Key Problem: Device Characterization

Goal: quantify single-cell I/O concentration relation

Three phases of lab work:
- Multi-plasmid (qualitative test)
- Single-plasmid (rough quantitative)
- Chromosomal integration (fine quantitative)
Example System: C7C7 repressor

- Pon

- EYFP

+ Pon

- mKate

- AmCyan
Example System: VP16Gal4 activator

Dox

pHef1a → rtTa → TRE → AmCyan → TRE → VP16Gal4 → pUASgal4 → EYFP → pHef1a → mKate

- pHef1a
- rtTa
- TRE
- AmCyan
- TRE
- VP16Gal4
- pUASgal4
- EYFP
- pHef1a
- mKate

Graphs:

- AmCyan (A.U.) vs. Dox (nM)
- EYFP (A.U.) vs. Dox (nM)
From Fluorescence to Static Discipline

Fluorescence of proxy proteins at N hours

Color-corrected Fluorescence

Model-Compensated Transfer Curve

Inflection Points

Fluorescence of proxy proteins at N hours

Model-Compensated Transfer Curve

Color-corrected Fluorescence

Repressor→output transfer curve w. varying plasmid count
Color Correction: Bleed-over Matrix

- Selecting an optimal combination of proteins:
Color Correction: Piecewise Models

EYFP impact on AmCyan measurements

- Data
- Linear fit (midrange)
- Linear fit (all)

EYFP fluorescence vs. AmCyan fluorescence graph
Color Correction: Piecewise Models

EYFP impact on AmCyan compensation models

- Piecewise fit
- 2 std. dev.
- Linear fit (all)

log₁₀(EYFP fluorescence)

log₁₀(AmCyan fluorescence)
Characterization Contributions

- Multi-stage characterization protocol for transcriptional devices
- Model mapping multi-plasmid behavior to predictions of single copy behavior
- Improved FACS color correction
- Preliminary characterization results
Contributions:

- TASBE: open tool-chain architecture
- Demonstration of end-to-end automated design
- Advances on key sub-problems:
  - Compilation and Optimization
  - DNA Part Selection [Next talk]
  - Flexible Protocol Automation [Following talk]
  - Characterization of Transfer Curves
Toward a community platform…

- Free, open source core
  - Proto, Clotho available now, others by arrangement
- Work on interchange standards (SBOL, CHRIS)