Automated Design of Synthetic Biology Feedback Circuits

Jacob Beal, Aaron Adler

IBE Conference
March, 2012

Work partially sponsored by DARPA; the views and conclusions contained in this document are those of the authors and not DARPA or the U.S. Government.
Vision: WYSIWYGG 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*

[DNA synthesis chart]


[Length in base pairs] 100 1,000 10,000 100,000 1,000,000

207 2,100 2,700 7,500 14,600

[Circuit size chart]


[Number of promoters] 0 1 2 3 4 5 6 7 8

Max (18 months) Moving average (18 months)

[Purnick & Weiss, ‘09]
Why a tool-chain?

This gap is too big to cross with a single method!
The TASBE architecture:

Organism Level Description

High Level Description

Abstract Genetic Regulatory Network

DNA Parts Sequence

Assembly Instructions

Cells

**High level simulator**

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

**Coarse chemical simulator**

**Detailed chemical simulator**

**Testing**

Modular architecture also open for flexible choice of organisms, protocols, methods, ...

Collaborators:
- Ron Weiss
- Douglas Densmore
A Tool-Chain Example

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

Dox → cyan → not → yellow

Dox

rtTA

CFP

B

EYFP

pHef1a

rtTA

pTre

CFP

pTre

LacI

mirff4 pHef1a-LacO1Oid

EYFP

4xff4

Organism Level Description

High Level Description

Coarse chemical simulator

Abstract Genetic Regulatory Network

Detailed chemical simulator

DNA Parts Sequence

Assembly Instructions

Testing

Cells
Today’s focus: BioCompiler

Compilation & Optimization

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

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

Signal = Concentration

Stabilizes at decay = production

Alternatives:
PoPS
RNA concentration
Motif-Based Compilation

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

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

arg0

value
Motif-Based Compilation

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

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

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

```
(primitive IPTG () boolean
 : grn-motif ((P high LacI|boolean T)
 (RXN (IPTG|boolean) represses LacI)
 (P high R- LacI value T))
```

![Diagram of IPTG, LacI, and value with arrows indicating interaction]
Motif-Based Compilation

- Functional program gives dataflow computation:

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

- Operators translated to motifs:
Optimization

Copy Propagation

Dead Code Elimination

Dead Code Elimination
GRN Optimization Library

- Copy Propagation
- Dead Code Elimination
- Double-Negative Elimination
- Constant Eliminator
- CSE: Output Consolidation
- CSE: Merge Duplicate Inputs
- NOR Compression
GRN Optimization Library

• Copy Propagation
• Dead Code Elimination
• Double-Negative Elimination
• Constant Eliminator
• CSE: Output Consolidation
• CSE: Input Merge
• NOR Compression
GRN Optimization Library

- Copy Propagation
- **Dead Code Elimination**
- Double-Negative Elimination
- Constant Eliminator
- CSE: Output Consolidation
- CSE: Input Merge
- NOR Compression
GRN Optimization Library

- Copy Propagation
- Dead Code Elimination
- **Double-Negative Elimination**
- Constant Eliminator
- CSE: Output Consolidation
- CSE: Input Merge
- NOR Compression
GRN Optimization Library

- Copy Propagation
- Dead Code Elimination
- Double-Negative Elimination
- **Constant Eliminator**
- CSE: Output Consolidation
- CSE: Input Merge
- NOR Compression
GRN Optimization Library

- Copy Propagation
- Dead Code Elimination
- Double-Negative Elimination
- Constant Eliminator
- **CSE: Output Consolidation**
- CSE: Input Merge
- NOR Compression

*GRN-specific*
GRN Optimization Library

- Copy Propagation
- Dead Code Elimination
- Double-Negative Elimination
- Constant Eliminator
- CSE: Output Consolidation
- CSE: Input Merge
- NOR Compression

GRN-specific
GRN Optimization Library

- Copy Propagation
- Dead Code Elimination
- Double-Negative Elimination
- Constant Eliminator
- CSE: Output Consolidation
- CSE: Input Merge
- NOR Compression

GRN-specific
Feedback System: SR-Latch

\[
\text{def sr-latch} \ (s \ r) \\
\quad \text{letfed}+ \ ((o \ \text{boolean} \ (\text{not} \ (\text{or} \ r \ o-bar))) \\
\quad \quad (o-bar \ \text{boolean} \ (\text{not} \ (\text{or} \ s \ o)))) \\
\quad o))
\]

\[
\text{green} \ (\text{sr-latch} \ (\text{aTc}) \ (\text{IPTG})))
\]

Diagram:
- **IPTG**
- **aTc**
- **or**
- **not**
- **green**

Connections:
- **IPTG** to **or**
- **aTc** to **or**
- **or** to **not**
- **not** to **green**
- **not** to **or**
Feedback System: SR-Latch

(def sr-latch (s r)
  (letfedom+ ((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
(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)))
Optimization of Complex Designs

(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
(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)
  (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)))

Optimization of Complex Designs

Unoptimized: 15 functional units, 13 transcription factors

Copy Propagation
Optimization of Complex Designs

(def sr-latch (s r)
  (letfед+ ((ο boolean (not (or r o-bar)))
    (o-bar boolean (not (or s o)))))
  ō))

(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

Dead Code Elimination
Optimization of Complex Designs

(def \texttt{sr-latch} (s r)
  (letf fed+ ((o boolean (\texttt{not} (or r o-bar))))
    (o-bar boolean (\texttt{not} (or s o))))
  o))

(green (\texttt{sr-latch} (aTc) (IPTG)))

\textit{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)
  (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

Dead Code Elimination
Optimization of Complex Designs

(def \textit{sr-latch} (s r)
  (letfed+ ((o boolean (not (or r o-bar)))
    (o-bar boolean (not (or s o)))))
  o)

(\textit{green} (\textit{sr-latch} (aTc) (IPTG)))

Unoptimized: 15 functional units, 13 transcription factors

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
  – Verification in ODE simulation

• Optimization competitive with human experts:
  – Test systems have 25% to 71% complexity reduction
  – Optimized systems are often homologous with hand designed networks

• Routine use within team
Onward Through the Tool-Chain…

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

Dox → cyan → not → yellow

rtTA → CFP → B → EYFP

pHef1a → rtTA → pTre → CFP → pTre → LacI → mirff4pHef1a-LacO1Oid → EYFP → 4xff4

Organism Level Description
High Level Description
Abstract Genetic Regulatory Network
DNA Parts Sequence
Assembly Instructions
Cells

Testing

High level simulator
Coarse chemical simulator
Detailed chemical simulator
Contributions

• Boolean & feedback logic circuits can be expressed in high-level language.
• GRN-specific optimizations allow dramatic optimization of circuit size.
• Automatically generated circuits are competitive with hand-designs created by experts.

Next steps: automatic verification, numerical computation, intercellular communication