High-Level Languages for Synthetic Biology

Jacob Beal

November, 2010

BBN: Fusun Yaman, Aaron Adler, Richard Schantz MIT: Ron Weiss, Jonathan Babb, Noah Davidsohn BU: Doug Densmore, Swapnil Bhatia





Agenda

- What's "high-level language", and why bother?
- State of the Art
- Proto & Tool Chain

Definition

A "high-level" programming language is any language that abstracts away many of the details of how a computation will be implemented.

High-Level Language in SynBio



High-Level Language in SynBio



Why bother with HLLs?

- Accessibility: knowledge in software
- Scalability: routine work automated; higher portability
- Reliability: less human code; verification

Is SynBio there yet? Maybe...

• How will a SynBio system be described?

 How will a high-level description be transformed to *in vivo* execution?















What do we really need?

- Primitives
- Means of Combination
- Means of Abstraction

Everything else follows...

Agenda

- What's "high-level language", and why bother?
- State of the Art
- Proto & Tool Chain

Computation via Transcription Network



Computation via Transcription Network



Stablizes at *decay* = *production*

GenoCAD [Cai et al., '07]



ctcacttttgccctttagaaggggaaagctggcaagattttttacgtaataacgctaaaagttttagatgtgctttactaagtcat cgcgatggagcaaaagtacatttaggtacacggcctacagaaaaacagtatgaaactctcgaaaatcaattagcctttttatgc caacaaggtttttcactagagaatgcattatatgcactcagcgctgtggggcattttactttaggttgcgtattggaagatcaag agcatcaagtcgctaaagaaggaaaagggaaacacctactactgatagtatgccgccattattacgacaagctatcgaattatttg atcaccaaggtgcagagccagccttcttattcggcctt......

3

CFG to generate/check part sequences

Eugene [Densmore et al., '10]

Device xor(cp, lacI, tetR, rpType1, gfp, rpType2, gfp);



Circuit "parts list" w. constraints, variables

GEC [Pederson & Phillips, '09]



prom<con(RT)>; rbs; pcr<codes(PA)>; rbs; pcr<codes(PB)>; ter

Transcriptional logic programming



Proto [Beal & Bachrach, '08]

High-level primitives map to GRN design motifs



Agenda

- What's "high-level language", and why bother?
- State of the Art
- Proto & Tool Chain

Prototype Tool Chain

Stages for Engineering Cells



Abstract GRN Design Space



Abstract GRN Design Space



Major Challenge: Interference

- Effective part characterics changed by:
 - Cellular context (endogenous pathways, synthetic parts)
 - Expression noise
- Our approach: noise-rejection
 - Digital static discipline: V_{low,out} < V_{low,in} < V_{high,in} < V_{high,out}



But part variance makes a uniform standard impossible!

- Identify "standards family" parameter relation
- Create library of characterized part variants
- Adjust part choice to match on junctions

- Identify "standards family" parameter relation
- Create library of characterized part variants
- Adjust part choice to match on junctions



- Identify "standards family" parameter relation
- Create library of characterized part variants
- Adjust part choice to match on junctions



- Identify "standards family" parameter relation
- Create library of characterized part variants
- Adjust part choice to match on junctions



- Identify "standards family" parameter relation
- Create library of characterized part variants
- Adjust part choice to match on junctions



- Identify "standards family" parameter relation
- Create library of characterized part variants
- Adjust part choice to match on junctions



- Identify "standards family" parameter relation
- Create library of characterized part variants
- Adjust part choice to match on junctions



- Identify "standards family" parameter relation
- Create library of characterized part variants
- Adjust part choice to match on junctions



- Identify "standards family" parameter relation
- Create library of characterized part variants
- Adjust part choice to match on junctions



Experimental Input to Family Relation



Model constrained by characterization experiments...

Prototype Tool Chain

Stages for Engineering Cells



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

```
(primitive not (boolean) boolean
  :bb-template ((P 0.193 R- arg0 RBS outputs T)))
```



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

(primitive green (scalar) scalar :side-effect
 :bb-template ((P R+ arg0 RBS GFP outputs T)))



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



• Functional program gives dataflow computation:

(green (not (IPTG)))

• Functional program gives dataflow computation:



• Operators translated to motifs:

• Operators translated to motifs:



• Operators translated to motifs:



Optimization



Optimization







Example Complex Compilation

• 2-bit adder:

```
(def xor (a b)
 (or (and a (not b))
  (and b (not a))))
(def 2bit-adder (al a0 b1 b0)
 (green (xor a0 b0)) ; low bit
 (let ((c0 (and a0 b0))
       (x1 (xor a1 b1)))
  (red (xor x1 c0)) ; high bit
  (blue (or (and x1 c0) ; carry bit
       (and a1 b1)))))
```

(2bit-adder (aTc) (IPTG) (C4HSL) (30C12HSL))

Example Complex Compilation

• 2-bit adder:



Sample Optimization Results:

		Proteins	Functional	Promoters	Delay
			Units	(Repressed/Activated/Constitutive)	Stages
Single-Not	Unoptimized	4	4	4(2/1/1)	3
	Optimized	3	3	3(2/0/1)	2
	% Improvement	25%	25%	33%	33%
Three-Gate	Unoptimized	10	10	9(7/1/1)	5
	Optimized	4	5	4(3/0/1)	2
	% Improvement	60%	50%	55%	60%
Quad-Not	Unoptimized	7	7	7(5/1/1)	6
	Optimized	2	2	2(1/0/1)	1
	% Improvement	71%	71%	71%	83%
2-bit adder	Unoptimized	55	56	53 (37/15/1)	12
	Optimized	26	23	24(19/4/1)	7
	% Improvement	52%	59%	55%	42%

Prototype Tool Chain

Stages for Engineering Cells



Abstract Feature Mapping

• Parts Database: bipartite regulation graph



Find a non-conflicting subgraph isomorphic to design

Initial Feature Matching Results



Prototype Tool Chain

Stages for Engineering Cells



Clotho: Planning for Assembly



Automated Assembly



Toward a broadly integrated future...

Stages for Engineering Cells



Available Free Open-Source Tools

Proto

http://proto.bbn.com



High level design, simulation

Clotho

http://clothocad.org



Data management, automation

Backup material

- Prototype compiler generates GRNs that simulate correctly for a limited language subset
- Example: 2-bit adder



- Prototype compiler generates GRNs that simulate correctly for a limited language subset
- Example: 2-bit adder



- Prototype compiler generates GRNs that simulate correctly for a limited language subset
- Example: 2-bit adder



- Prototype compiler generates GRNs that simulate correctly for a limited language subset
- Example: 2-bit adder



- Prototype compiler generates GRNs that simulate correctly for a limited language subset
- Example: 2-bit adder

```
(macro xor (a b)
(muxor (muxand ,a (not ,b))
(muxand ,b (not ,a))))
(macro 2bit-adder (al a0 bl b0)
(all
(green (xor ,a0 ,b0)) ; x_0 low bit
(let ((c0 (muxand ,a0 ,b0))
(x1 (xor ,a1 ,b1)))
(red (xor x1 c0)) ; x_1 high bit
(blue (muxor (muxand x1 c0) ; carry bit
(muxand ,a1 ,b1))))))
```

(**2bit-adder** (**aTc**) (**IPTG**) (C4HSL) (30C12HSL))

Bit 0

Bit 1

Carry

- Compiled 2-bit adder (unoptimized)
 - 60 signal chemicals
- 52 regulatory regions Generated ODE simulation in MATLAB Generated ODE

