

International Workshop on Bio-Design Automation at DAC



IWBDA 2010

June 14-15, 2010
Anaheim, California, USA



DESIGN AUTOMATION CONFERENCE



cādence™



Microsoft®
Research



The following students were provided financial support by our sponsors to attend the workshop

Michal Galdzicki, University of Washington - SynBERC/MSR Cambridge

Mario Marchisio, ETH Zurich - MSR Cambridge

Mona Yousofshahi, Tufts University - Life Technologies

Paul Bogdan, Carnegie Mellon University - SynBERC

Natasa Miskov-Zivanov, University of Pittsburgh - GenoCAD

Deepak Chandran, University of Washington - GenoCAD

Curtis Madsen, University of Utah - GenoCAD

Ehsan Ullah, Tufts University - GenoCAD

Masoud Rostami, Rice University - GenoCAD

Foreward

Welcome to the Second International Workshop on Bio-Design Automation at DAC!

IWBDA brings together researchers from the synthetic biology and design automation communities. Still in its early stages, the field of synthetic biology has been driven by experimental expertise; much of its success has been attributable to the skill of the researchers in specific domains of biology. There has been a concerted effort to assemble repositories of standardized components. However, creating and integrating synthetic components remains an ad hoc process. The field has now reached a stage where it calls for computer-aided design tools. The electronic design automation (EDA) community has unique expertise to contribute to this endeavor. This workshop offers a forum for cross-disciplinary discussion, with the aim of seeding collaboration between the research communities.

This year, the program consists of 14 talks and 18 poster presentations. These are organized into 4 sessions: Tools for Bio-Design Automation, Modeling and Standards for Bio-Design, Design of Biological Circuits and Networks and Biological Pathway and Network Optimization. In addition, we are very pleased to have four very distinguished invited speakers, Roger Brent, Pamela Silver, J. Chris Anderson and Richard Murray. Also we have two tutorial sessions: DNA Nanostructures and CAD for Genetic Circuits.

We thank all the participants for contributing to IWBDA; we thank the Program Committee for reviewing abstracts; and we thank everyone on the Executive Committee for their time and dedication. Finally, we thank Artist, Cadence, Combust, DNA 2.0, GenoCAD, Life Technologies, Microsoft Research and Synthetic Biology.net for supporting the workshop financially.

Marc Riedel, General Chair

Doug Densmore, General Secretary

Organizing Committee

General Chair - Marc Riedel (University of Minnesota)
General Secretary - Douglas Densmore (Joint BioEnergy Institute)
Program Committee Chair - Ron Weiss (MIT)
Publication Chair - Jean Peccoud (Virginia Tech)
Industry Liaison Chair - Andreas Kuehlmann (Cadence Research Labs)
Finance Chair - David Thorsley (University of Washington)
DAC Liaison & Publicity Chair - Soha Hassoun (Tufts University)

Program Committee

J. Christopher Anderson, UC Berkeley	Adam Arkin, UC Berkeley
Jacob Beal, BBN Technologies	Kevin Clancy, Life Technologies
Domitilla Del Vecchio, University of Michigan	Douglas Densmore, Joint BioEnergy Institute
Hanna El-Samad, UCSF	Drew Endy, Stanford University
Soha Hassoun, Tufts University	Alfonso Jaramillo, Ecole Polytechnique
Yannis Kaznessis, University of Minnesota	Eric Klavins, University of Washington
Heinz Koeppl, EPFL	Tanja Kortemme, UCSF
Andreas Kuehlmann, Cadence Research Labs	Vishwesh Kulkarni, IIT Bombay
Natasa Miskov-Zivanov, Univ. of Pittsburgh	Chris Myers, University of Utah
Jean Peccoud, Virginia Tech	Andrew Phillips, Microsoft Research
Marc Riedel, University of Minnesota	Howard Salis, Penn State University
Herbert Sauro, University of Washington	David Thorsley, University of Washington
Christopher Voigt, UCSF	Ron Weiss, MIT
Erik Winfree, Caltech	Chris Winstead, Utah State University

IWBDA 2010 Program

Monday, June 14

9am - 10am **Keynote Address**

Opening Remarks:

Marc Riedel, Ron Weiss

Keynote Address:

Roger Brent

Fred Hutchinson Cancer Research Center

10:30 - Noon

Tutorial #1 on DNA Nanostructures, Shawn Douglas, Harvard

Tutorial #2 on CAD for Genetic Circuits, Jean Peccoud, Virginia Tech

Noon - 2pm

Lunch and Posters Session 1

2:00pm - 4:00pm Tech. Talks Session 1 - *Tools for Bio-Design Automation*

1BDA.1 SynBioSS Designer: From DNA Sequences to Dynamic Phenotypes and Back

Emma Weeding and Yiannis Kaznessis

1BDA.2 TinkerCell: CAD Application with Support for Third-party Programs

Deepak Chandran and Herbert Sauro

1BDA.3 Automatic Compilation from High-Level Languages to Genetic Regulatory Networks

Jacob Beal, Ting Lu and Ron Weiss

1BDA.4 Designing Biological Devices in GEC

James Brown, Neil Dalchau, Michael Pedersen and Andrew Phillips

4:30pm - 6:00pm

Tech. Talks Session 2 - *Modeling and Standards for Bio-Design*

2BDA.1 A Semantic Knowledge Base of Standard Biological Parts

Michal Galdzicki, Cesar Rodriguez, Deepak Chandran, John Gennari and Herbert Sauro

2BDA.2 Modeling and Predicting the Strength of Bacterial Promoters: A Test-case with Promoters Regulated by Escherichia coli Sigma E

Virgil Rhodius, Vivek Mutalik and Carol Gross

2BDA.3 Resolving Variable Dependencies in the MPDE-SSA Algorithm

Abiezer Tejeda, Chris Winstead, Eduardo Monzon, Chris Myers and Curtis Madsen

7:00pm - 10:00pm Dinner - Sponsored by ArtistDesign and COMBEST

Naples Ristorante e Pizzeria
Downtown Disney
1550 S Disneyland Dr., Anaheim, CA 92802

Tuesday – June 15th

8:30am - 10:15am

General DAC Keynote (full conference)

10:30am - Noon

Tech. Talks Session 3 - *Design of Biological Circuits and Networks*

Open Poster Session (full conference)

3BDA.1 Automatic Design of Digital Synthetic Gene Circuits

Mario Marchisio and Joerg Stelling

3BDA.2 Design of In-vitro Synthetic Gene Circuits

Elisa Franco and Richard Murray

3BDA.3 Fan-out Considerations in Gene Regulatory Networks

Kyung Kim and Herbert Sauro

Noon - 2pm

Lunch and Posters Session 2

2:00pm - 3:30pm

Joint IWBD/AC Session

Pamela Silver, Harvard University
J. Chris Anderson, Berkeley
Richard Murray, Caltech

4:00pm - 4:50pm

Panel Session - Sponsored by ArtistDesign and COMBEST

J.Christopher Anderson (UC Berkeley)
Andreas Kuehlmann (Cadence Research Labs)
Andrew Phillips (MS Research)

5:00pm - 6:00pm

Tech. Talks Session 4 - *Biological Pathway and Network Optimization*

4BDA.1s Predictably Profitable Paths in Metabolic Networks

Ehsan Ullah, Mark Walker, Kyongbum Lee and Soha Hassoun

4BDA.2s Robust Inference of Biological Bayesian Networks

Masoud Rostami and Kartik Mohanram

4BDA.3s **Pathway Identification for Strain Engineering**

Mona Yousofshahi, Kyongbum Lee and Soha Hassoun

6:00pm - 6:15pm

Closing Remarks, Doug Densmore

6:15pm - 6:45pm

Post-workshop / Future-workshop planning (open to all)

Abstract Table of Contents

Keynote Address

Keynote Address

Roger Brent, Fred Hutchinson Cancer Research Center

Lunch and Posters Session 1

Tech. Talks Session 1 - *Tools for Bio-Design Automation*

SynBioSS Designer: From DNA Sequences to Dynamic Phenotypes and Back

Emma Weeding and Yiannis Kaznessis

TinkerCell: CAD Application with Support for Third-party Programs

Deepak Chandran and Herbert Sauro

Automatic Compilation from High-Level Languages to Genetic Regulatory Networks

Jacob Beal, Ting Lu and Ron Weiss

Designing Biological Devices in GEC

James Brown, Neil Dalchau, Michael Pedersen and Andrew Phillips

Tech. Talks Session 2 - *Modeling and Standards for Bio-Design*

A Semantic Knowledge Base of Standard Biological Parts

Michal Galdzicki, Cesar Rodriguez, Deepak Chandran, John Gennari and Herbert Sauro

Modeling and Predicting the Strength of Bacterial Promoters: A Test-case with Promoters Regulated by Escherichia coli Sigma E

Virgil Rhodius, Vivek Mutalik and Carol Gross

Resolving Variable Dependencies in the MPDE-SSA Algorithm

Abiezer Tejada, Chris Winstead, Eduardo Monzon, Chris Myers and Curtis Madsen

Dinner - Sponsored by ArtistDesign and COMBEST

Tech. Talks Session 3 - *Design of Biological Circuits and Networks*

Automatic Design of Digital Synthetic Gene Circuits

Mario Marchisio and Joerg Stelling

Design of In-vitro Synthetic Gene Circuits

Elisa Franco and Richard Murray

Fan-out Considerations in Gene Regulatory Networks

Kyung Kim and Herbert Sauro

Lunch and Posters Session 2

Joint IWBD/AC Session

Pamela Silver, Harvard University

J. Chris Anderson, Berkeley

Richard Murray, Caltech

Tech. Talks Session 4 - *Biological Pathway and Network Optimization*

Predictably Profitable Paths in Metabolic Networks

Ehsan Ullah, Mark Walker, Kyongbum Lee and Soha Hassoun

Robust Inference of Biological Bayesian Networks

Masoud Rostami and Kartik Mohanram

Pathway Identification for Strain Engineering

Mona Yousofshahi, Kyongbum Lee and Soha Hassoun

Poster Abstracts

Intracellular Disease Prevention and Tuneable Device
Behaviour in Bacteria

Sangram Bagh, Mahuya Mandal, Jordan Ang and David McMillen

Modeling Swarms of Micro-robots for Biological Applications

Paul Bogdan and Radu Marculescu

A Genetic Programming Framework for the Simulation and Design
of Self-assembling, Chemotaxis-driven Cell Aggregates

David Breen and Linge Bai

Data Model Approaches for Design, Assembly and Validation
in Synthetic Biology

Kevin Clancy

Scalable open source software framework for laboratory automation
and laboratory devices

Jonathan Cline

Towards Integrative CellML Modeling Technologies for Intracellular
Research

Mike Cooling, Ely Matos, Candice Zhou, Gary Tao and Poul Nielsen

Towards Distributed Web of Registries: Design, Implementation
and Practice of the JBEI Registry

Timothy Ham, Zinovii Dmytriv, Nathan Hillson and Jay Keasling

Towards Automated-assembly of Biological Parts

*Nathan Hillson, John Thorne, Doug Densmore,
Masood Hadi and Jay Keasling*

Digital Signal Processing with Biomolecular Reactions

Hua Jiang, Marc Riedel and Keshab Parhi

Bio-Specific HW/SW Co-Design for Multicore Systems on Chip

Iyad Al Khatib

Markov Chain Analysis of Genetic Circuits

Curtis Madsen, Chris Myers and Chris Winstead

Logical Modeling of Peripheral T Cell Differentiation

*Natasa Miskov-Zivanov, John Sekar, Michael Turner,
Lawrence Kane, Penelope Morel and James Faeder*

Synthetic gene circuits with a cell-free toolbox

Vincent Noireaux and Jonghyeon Shin

Structure-based Prediction of Residue Coevolution in Proteins

*Noah Ollikainen, Ellen Sentovich, Carlos Coelho, Andreas Kuehlmann
and Tanja Kortemme*

Architecture for Synthetic Organism Design

Matthew Peterson, Steven Fairchild and John Dileo

Sequence Refiner: Automated conversion of natural genetic
sequences into standard biological parts

Cesar Rodriguez, Adam Arkin and Andrew Endy

Using uncertainty quantification to constrain dynamic neuron
modeling parameters

Richard Schiek and Christy Warrender

Identification of Illegal States in a Discrete Transition Model of
Apoptosis Signaling

*Anupam Shrivastava, Michael Hsiao, Huy Lam, David Samuels
and Carla Finkielstein*

Keynote Speaker: Roger Brent



Roger was born in Spartanburg, South Carolina in 1955. He received a BA in Computer Science and Mathematics from the University of Southern Mississippi in 1973, where he did some work attempting to apply AI techniques to protein folding. He went on to get a Ph.D. in Biochemistry and Molecular Biology from Harvard University in 1982 for studies with Mark Ptashne. As a graduate student, he showed that the *E. coli* *lexA* gene repressed genes involved in the response to radiation damage, cloned the gene, produced and purified its protein product using and in some cases extending the newly developed recombinant DNA methods, and studied binding of the repressor to its operators, showing that its differential binding affinity for these sites affected the timing of the response. As a postdoctoral fellow, also with Mark Ptashne, he tested a number of ideas about the mechanism of transcription regulation in yeast by using the prokaryotic LexA protein and in subsequent experiments creating chimeric proteins that carried LexA fused to activators native to yeast. These "domain swap" experiments established the modular nature of eukaryotic transcription regulators.

In 1985, Roger became a Professor at Massachusetts General Hospital and Harvard Medical School Department of Genetics. He and his coworkers used yeast transcription that depended on chimeric DNA bound proteins as a genetic probe for protein function in higher organisms. This work led to the development of working two-hybrid methods (1988-1993), to the ability to scale them up via interaction mating (1992-1994), and to the eventual development of protein interaction methods as a useful way to learn more about biological function. In parallel, Roger and his coworkers developed peptide aptamers as reverse "genetic" agents to study the function of proteins and allelic protein variants (1999-2001), and, more recently, as dominant forward "genetic" reagents to identify genes and pathway linkages in organisms, such as human cells, that are intractable to classical genetic analysis. (Perhaps as important as the actual technologies is the coeval development of ideology (e.g. doctrine) for using them.) This work is described in about 80 research papers and reviews.

In parallel to his academic work, Roger is a longtime (since 1984) advisor to the biotech and pharmaceutical industries. He served on the SAB of American Home Products (Genetics Institute/Wyeth Ayerst Research), chairs scientific advisory boards for several smaller companies, and does significant ad hoc consulting work in genomics and computational biology. He is one of the founders (1987-2001) of Current Protocols, including Current Protocols in Molecular Biology, a "how to clone it" manual, which is updated every three months and has about 10,000 subscribing labs. He is founder and organizer (since 1994) of the "After the Genome" workshops. He is an inventor on 11 issued and several pending US Patents. Since the middle 1990s, he has exhorted and advised various bodies in the US and abroad on functional genomics and computational biology, including the National Institutes of Health, the Wellcome Trust, the National Science Foundation, Department of Energy, Defense Advanced Research Projects Agency, and other parts of the US Defense Department.

Roger joined the Molecular Sciences Institute in 1998 as Associate Director. He was named Director in 2000 and President and CEO in 2001. Brent joined the faculty of UCSF Department of Biopharmaceutical Sciences as an Adjunct Professor in 2000 and was named a Senior Scholar of the Ellison Medical Foundation in 2001.

In July 2009, Roger joined Fred Hutchinson Cancer Research Center as a Full Member in the Basic Sciences Division.

Tech. Talks Session 1 – Tools for Bio-Design Automation

Emma Weeding, Yiannis N. Kaznessis

Dept. of Chemical Engineering and Materials Science, University of Minnesota, Minneapolis, MN 55455

SynBioSS Designer: from DNA sequences to dynamic phenotypes and back.

As a discipline with a strong engineering character, synthetic biology can benefit from quantitative modeling. There is indeed an opportunity for the development of computer software tools that can tackle the challenges facing the synthetic biology community. Computer models of new gene networks can, in principle, shift through alternative designs and propose synthetic construction approaches before the synthetic biologist begins work in the wet lab [1,2].

Despite some progress in the development of computer models, there are gaps in the process of connecting DNA sequences to targeted phenotypes using software tools. For example, there are no universally accepted methods for representing synthetic biomolecular systems. Should all of the molecular components be included (e.g. promoters, operators, ribosome binding sites, RNAPolymerase, ribosomes, among others), or is a more reduced representation more appropriate? Is the dynamic behavior important, or are steady-state approximations? And can there be a modeling formalism that is applicable to a wide variety of synthetic constructs and amenable to automation?

In this work we will present a standardized algorithmic process for generating models of synthetic gene regulatory networks that is applicable to any synthetic construct and is suitable for automation. In particular, we present a new component of our Synthetic Biology Software Suite (SynBioSS), [3], we call Designer. Designer is a web-based tool that allows users to enter DNA components and obtain networks of reactions. Designer does this automatically, using universal principles of molecular biology. Importantly, Designer can take as input BioBricks, which are standard DNA sequences used widely by the synthetic biology community.

SynBioSS Designer takes as input molecular parts involved in gene expression and regulation (e.g. promoters, transcription factors, ribosome binding sites, etc.), and automatically generates complete networks of reactions that represent transcription, translation, regulation, induction and degradation of those parts. In this work we describe how Designer uses universal principles of molecular biology to generate models of any arbitrary synthetic biological system. These models are useful as they explain biological phenotypic complexity in mechanistic terms. In turn, such mechanistic explanations can assist in designing synthetic biological systems. We will discuss, giving practical guidance to users, how Designer interfaces with the Registry of Standard Biological Parts, the *de facto* compendium of parts used in synthetic biology applications.

Designer is freely available at www.synbioss.org/Designer. It has a tabbed interface, making the complete sequence of BioBricks, or other, custom-defined parts, visually accessible and easily manipulated. Clicking on a tab pulls up properties of that individual brick and allows the user to add, edit, and delete said properties. Properties are also easy to edit; clicking directly on an editable field causes a text input field or drop-down menu will appear, allowing the user to make appropriate changes.

A user can enter biological components, including BioBricks, in SynBioSS Designer, and receive as an output an SBML file with a reaction network that models the BioBricks. With the database of BioBricks, Designer can be used to streamline model construction. All information is now automatically, quickly, and accurately retrieved, added to the current Designer construct, and displayed, ready to be edited if necessary.

We will give practical guidance, going over three examples: a BioBrick from the Registry of biological parts, an AND-gate and a custom-made tetracycline-inducible feedback loop. We will discuss how to determine the DNA sequence of a synthetic construct so as to optimize a targeted dynamic behavior.

References

1. Kaznessis Y.N. *Computational methods in synthetic biology*. Biotechnol J. **2009**. 4(10): p. 1392-405.
2. Ramalingam, K.I., J.R. Tomshine, J.A. Maynard, et al. *Forward engineering of synthetic bio-logical AND gates*. Biochemical Engineering Journal, **2009**. 47(1-3): p. 38-47.
3. Hill, A.D., J. Tomshine, E. Wedding, et al., *SynBioSS: the synthetic biology modeling suite*. Bioinformatics, **2008**. 24(21): p. 2551-3.
4. Weeding, E., Houle, J. , Kaznessis, Y.N. SynBioSS designer: a web-based tool for the automated generation of kinetic models for synthetic biological constructs, Briefings in Bioinformatics, in press.

In what follows we present only one of the three walkthrough examples to use Designer.

WALKTHROUGH EXAMPLE: MODELING A CUSTOM AND GATE

One is not required to use BioBricks to create a system; a user can just as easily create a device using customized biological parts. For example, one can recreate the single promoter AND gates detailed by Ramalingam and coworkers (“Forward engineering of synthetic biological AND gates”, Biochemical Engineering Journal, Volume 47, Issues 1-3, 1 December 2009, Pages 38-47).

The key element of such a device is the promoter, which is constructed with elements of the tet, lac, and λ -phage promoters, and contains a particular combination of tet and lac operators. TetR and LacI proteins are constitutively expressed in the system, and thus without additional input, the promoter is repressed. In other words, GFP is produced only when both aTc and IPTG are present.

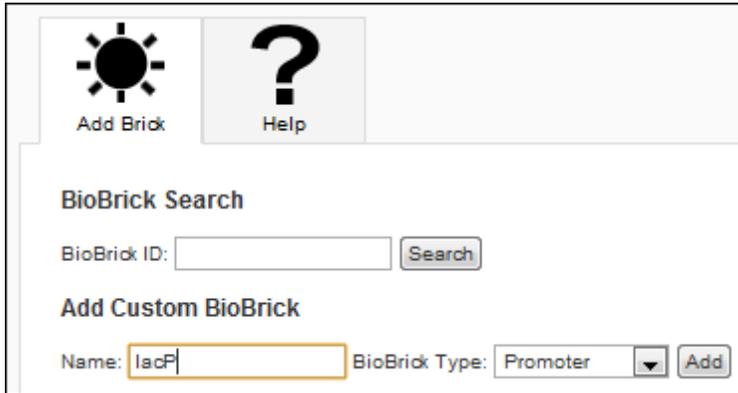
The paper by Ramalingam and coworkers details six combinations, but we will focus on the LTT configuration: one lac operator upstream of the -35 site (lacO1), one tet operator downstream of the -10 site (tetO2), and another tet operator between the -35 and -10 sites (tetO1).

Creating the Device in Designer

Throughout this process, we shall use terminology identical to that used by Ramalingam and coworkers in order to facilitate comparison between the automatically generated Designer reaction network, and the manually created reaction network in the paper.

The first step is to add the series of parts. Figure S1 illustrates how to add the hybrid promoter, which we name “lacP”. This is followed by an RBS, simply called “RBS”, then a coding region, also plainly labeled as “DNA”, and finally a terminator titled “Terminator”.

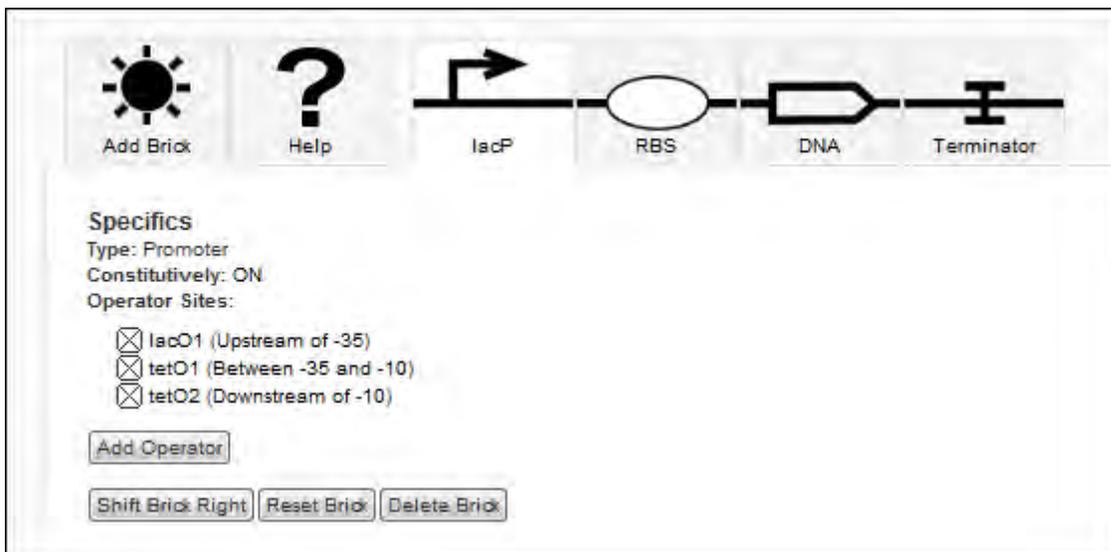
Figure S1



The screenshot shows a web interface for adding BioBrick parts. At the top, there are two buttons: "Add Brick" with a gear icon and "Help" with a question mark icon. Below these is a "BioBrick Search" section with a text input field for "BioBrick ID:" and a "Search" button. Underneath is an "Add Custom BioBrick" section with a "Name:" field containing "lacP", a "BioBrick Type:" dropdown menu set to "Promoter", and an "Add" button.

Because this device is entirely custom-made, the user must specify operator site and protein information manually. The operator site information is shown in Figure S2, while the coding region “DNA” is specified to correspond to the protein “gfp” of type “reporter” (not shown). Also note the entire sequence of parts shown in Figure S2.

Figure S2



The screenshot shows the configuration page for the "lacP" part. At the top, there are four icons: "Add Brick" (gear), "Help" (question mark), "lacP" (a bent arrow), "RBS" (an oval), "DNA" (a rectangle with a tail), and "Terminator" (a T-bar). Below these is a "Specifics" section with the following information: "Type: Promoter", "Constitutively: ON", and "Operator Sites:" with three checked checkboxes: "lacO1 (Upstream of -35)", "tetO1 (Between -35 and -10)", and "tetO2 (Downstream of -10)". At the bottom, there are three buttons: "Add Operator", "Shift Brick Right", "Reset Brick", and "Delete Brick".

The second step, inputting protein specifics, is carried out in the same manner as described in the manuscript. In this system, there are two constitutively expressed repressor proteins: LacI and TetR. The former is active as a tetramer and binds the lac operator (lacO1). The latter

repressor is active as a dimer, and binds the two tet operators (tetO1 and tetO2). The final “Current Proteins” table after inputting all of these specifics into Designer is shown in Figure S3.

Figure S3

Current Proteins			
Protein	Type	Complex	Binds
gfp	Reporter	gfp	<i>optional</i>
lacI	Repressor	lacI4	lacO1
tetR	Repressor	tetR2	tetO1 tetO2

Likewise, the third and final step, effector input, is performed in a similar manner as in the manuscript. Along with aTc, this system contains IPTG instead of HSL. The final “Current Effectors” table is shown in Figure S4.

Figure S4

Current Effectors		
Effector	Bound Complex(es)	Act in Concert
IPTG	lacI4:IPTG4	
aTc	tetR2:aTc2	

Return to 2/3 Clear All Generate NetCDF File Generate SBML File

Designer can now generate a NetCDF or SBML file containing a reaction network describing this AND gate. This series of reactions and corresponding kinetic data is shown in the following section.

Reaction Network: Modeling a Custom AND Gate

Protein Multimerization

2 lacI → lacI2	1000000000
lacI2 → 2 lacI	0
2 lacI → lacI4	1000000000
lacI4 → 2 lacI2	0
2 tetR → tetR2	1000000000
tetR2 → 2 tetR	0

Transcription

RNAP + lacP + tetO2 + tetO1 + lacO1 → RNAP:lacP:tetO2:tetO1:lacO1	0.0166
RNAP:lacP:tetO2:tetO1:lacO1 → RNAP + lacP + tetO2 + tetO1 + lacO1	0.75
RNAP:lacP:tetO2:tetO1:lacO1 → RNAP:lacP:tetO2:tetO1:lacO1*	0.3
RNAP:lacP:tetO2:tetO1:lacO1* → RNAP:DNA_gfp + lacP + tetO2 + tetO1 + lacO1	30
RNAP:DNA_gfp → RNAP + mRNA_gfp	30 nt/s, 600 nt

Translation

rib + mRNA_gfp → rib:mRNA_gfp	100000
-------------------------------	--------

rib:mRNA_gfp → rib:mRNA_gfp_1 + mRNA_gfp	33 aa/s
	33 aa/s, 220
rib:mRNA_gfp_1 → rib + gfp	aa
Regulation	
lacI4 + lacO1 → lacI4:lacO1	1000000000
lacI4:lacO1 → lacI4 + lacO1	0.005
tetR2 + tetO1 → tetR2:tetO1	1000000000
tetR2:tetO1 → tetR2 + tetO1	0.005
tetR2 + tetO2 → tetR2:tetO2	1000000000
tetR2:tetO2 → tetR2 + tetO2	0.005
Induction	
lacI4 + IPTG → lacI4:IPTG	50000000
lacI4:IPTG → lacI4 + IPTG	0.1
lacI4:IPTG + IPTG → lacI4:IPTG2	50000000
lacI4:IPTG2 → lacI4:IPTG + IPTG	0.1
lacI4:IPTG2 + IPTG → lacI4:IPTG3	50000000
lacI4:IPTG3 → lacI4:IPTG2 + IPTG	0.1
lacI4:IPTG3 + IPTG → lacI4:IPTG4	50000000
lacI4:IPTG4 → lacI4:IPTG3 + IPTG	0.1
lacI4:IPTG + lacO1 → lacI4:IPTG:lacO1	1000000000
lacI4:IPTG:lacO1 → lacI4:IPTG + lacO1	0.7
lacI4:lacO1 + IPTG → lacI4:IPTG:lacO1	1000000
lacI4:IPTG:lacO1 → lacI4:lacO1 + IPTG	0.4
lacI4:IPTG2 + lacO1 → lacI4:IPTG2:lacO1	1000000
lacI4:IPTG2:lacO1 → lacI4:IPTG2 + lacO1	0.4
lacI4:IPTG3 + lacO1 → lacI4:IPTG3:lacO1	1000000
lacI4:IPTG3:lacO1 → lacI4:IPTG3 + lacO1	0.4
lacI4:IPTG4 + lacO1 → lacI4:IPTG4:lacO1	1000000
lacI4:IPTG4:lacO1 → lacI4:IPTG4 + lacO1	0.4
lacI4:IPTG:lacO1 + IPTG → lacI4:IPTG2:lacO1	50000000
lacI4:IPTG2:lacO1 → lacI4:IPTG:lacO1 + IPTG	0.1
lacI4:IPTG2:lacO1 + IPTG → lacI4:IPTG3:lacO1	50000000
lacI4:IPTG3:lacO1 → lacI4:IPTG2:lacO1 + IPTG	0.1
lacI4:IPTG3:lacO1 + IPTG → lacI4:IPTG4:lacO1	50000000
lacI4:IPTG4:lacO1 → lacI4:IPTG3:lacO1 + IPTG	0.1
tetR2 + aTc → tetR2:aTc	50000000
tetR2:aTc → tetR2 + aTc	0.1
tetR2:aTc + aTc → tetR2:aTc2	50000000
tetR2:aTc2 → tetR2:aTc + aTc	0.1
tetR2:aTc + tetO1 → tetR2:aTc:tetO1	1000000000
tetR2:aTc:tetO1 → tetR2:aTc + tetO1	0.7
tetR2:tetO1 + aTc → tetR2:aTc:tetO1	1000000
tetR2:aTc:tetO1 → tetR2:tetO1 + aTc	0.4
tetR2:aTc2 + tetO1 → tetR2:aTc2:tetO1	1000000
tetR2:aTc2:tetO1 → tetR2:aTc2 + tetO1	0.4
tetR2:aTc:tetO1 + aTc → tetR2:aTc2:tetO1	50000000
tetR2:aTc2:tetO1 → tetR2:aTc:tetO1 + aTc	0.1
tetR2:aTc + tetO2 → tetR2:aTc:tetO2	1000000000

tetR2:aTc:tetO2 → tetR2:aTc + tetO2	0.7
tetR2:tetO2 + aTc → tetR2:aTc:tetO2	1000000
tetR2:aTc:tetO2 → tetR2:tetO2 + aTc	0.4
tetR2:aTc2 + tetO2 → tetR2:aTc2:tetO2	1000000
tetR2:aTc2:tetO2 → tetR2:aTc2 + tetO2	0.4
tetR2:aTc:tetO2 + aTc → tetR2:aTc2:tetO2	50000000
tetR2:aTc2:tetO2 → tetR2:aTc:tetO2 + aTc	0.1
Non-Specific DNA Interactions	
lacI4 + nsDNA → lacI4:nsDNA	1000
lacI4:nsDNA → lacI4 + nsDNA	1.6225
lacI4:IPTG + nsDNA → lacI4:IPTG:nsDNA	1000
lacI4:IPTG:nsDNA → lacI4:IPTG + nsDNA	1.6225
lacI4:nsDNA + IPTG → lacI4:IPTG:nsDNA	1000
lacI4:IPTG:nsDNA → lacI4:nsDNA + IPTG	1.6225
lacI4:IPTG2 + nsDNA → lacI4:IPTG2:nsDNA	1000
lacI4:IPTG2:nsDNA → lacI4:IPTG2 + nsDNA	1.6225
lacI4:IPTG3 + nsDNA → lacI4:IPTG3:nsDNA	1000
lacI4:IPTG3:nsDNA → lacI4:IPTG3 + nsDNA	1.6225
lacI4:IPTG4 + nsDNA → lacI4:IPTG4:nsDNA	1000
lacI4:IPTG4:nsDNA → lacI4:IPTG4 + nsDNA	1.6225
lacI4:IPTG:nsDNA + IPTG → lacI4:IPTG2:nsDNA	1000
lacI4:IPTG2:nsDNA → lacI4:IPTG:nsDNA + IPTG	1.6225
lacI4:IPTG2:nsDNA + IPTG → lacI4:IPTG3:nsDNA	1000
lacI4:IPTG3:nsDNA → lacI4:IPTG2:nsDNA + IPTG	1.6225
lacI4:IPTG3:nsDNA + IPTG → lacI4:IPTG4:nsDNA	1000
lacI4:IPTG4:nsDNA → lacI4:IPTG3:nsDNA + IPTG	1.6225
tetR2 + nsDNA → tetR2:nsDNA	1000
tetR2:nsDNA → tetR2 + nsDNA	1.6225
tetR2:aTc + nsDNA → tetR2:aTc:nsDNA	1000
tetR2:aTc:nsDNA → tetR2:aTc + nsDNA	1.6225
tetR2:nsDNA + aTc → tetR2:aTc:nsDNA	1000
tetR2:aTc:nsDNA → tetR2:nsDNA + aTc	1.6225
tetR2:aTc2 + nsDNA → tetR2:aTc2:nsDNA	1000
tetR2:aTc2:nsDNA → tetR2:aTc2 + nsDNA	1.6225
tetR2:aTc:nsDNA + aTc → tetR2:aTc2:nsDNA	1000
tetR2:aTc2:nsDNA → tetR2:aTc:nsDNA + aTc	1.6225
Leakiness	
RNAp + lacP + lacI4:lacO1 + tetO2 + tetO1 → RNAp:lacP:tetO2:tetO1:lacI4	0.0166
RNAp:lacP:tetO2:tetO1:lacI4 → RNAp + lacP + lacI4:lacO1 + tetO2 + tetO1	0.75
RNAp:lacP:tetO2:tetO1:lacI4 → RNAp:lacP:tetO2:tetO1:lacI4*	0.3
RNAp:lacP:tetO2:tetO1:lacI4* → RNAp:DNA_gfp + lacP + lacI4:lacO1 + tetO2 + tetO1	30
Transport	
0 → lacI4	1.00E-10
0 → tetR2	1.00E-10
Degradation	
lacI4 → 0	0.000289
lacI4:nsDNA → nsDNA	0.000193
gfp → 0	0.000289

mRNA_gfp → 0	0.0015
tetR2 → 0	0.000289
tetR2:nsDNA → nsDNA	0.000193
lacI4:IPTG → IPTG	0.000289
lacI4:IPTG:nsDNA → IPTG + nsDNA	0.000193
lacI4:IPTG2 → 2 IPTG	0.000289
lacI4:IPTG2:nsDNA → 2 IPTG + nsDNA	0.000193
lacI4:IPTG3 → 3 IPTG	0.000289
lacI4:IPTG3:nsDNA → 3 IPTG + nsDNA	0.000193
lacI4:IPTG4 → 4 IPTG	0.000289
lacI4:IPTG4:nsDNA → 4 IPTG + nsDNA	0.000193
tetR2:aTc → aTc	0.000289
tetR2:aTc:nsDNA → aTc + nsDNA	0.000193
tetR2:aTc2 → 2 aTc	0.000289
tetR2:aTc2:nsDNA → 2 aTc + nsDNA	0.000193

TinkerCell: CAD application with support for third-party programs

Deepak Chandran, Herbert M. Sauro

Bioengineering Dept, Univ. of Washington, Seattle, USA

contact: deepakc@u.washington.edu

As the field of synthetic biology advances, one of the visions is that engineers will be able to build biological circuits using an efficient pipeline that takes them from the design process to construction and testing. In order to make this vision a reality, several challenges need to be overcome [1]. One of the challenges is a computational framework that can support mathematical modeling as well as information necessary for experiments [2]. Such a framework is required in order to build models using real biological components instead than hypothetical variables and parameters. The framework would also need to support access to database(s) of biological parts as well as engineering concepts such as modularity. The software application presented here, TinkerCell, is an example of how a computer-aided design tool can partially fulfill this requirement [3].

TinkerCell is an application for visual design and analysis of synthetic biological circuits [3]. It is structured as a project to which the community can contribute. TinkerCell has a flexible plug-in architecture allowing other researchers to write code that provide new functions in TinkerCell. The code can be written in C, C++, or Python, and additional programming languages will be added: Ruby, Perl, and R. This plug-in system permits TinkerCell to act like a "host" to algorithms that other researchers have developed for analysis of biological circuits. The types of functions provided by these plug-ins can range from mathematical analysis to sequence analysis or database access. In order to support such wide range of functions, TinkerCell's has adopted a structured yet flexible model representation. TinkerCell models are composed of biological components such as proteins, RNA, promoters, or operator sites. The list of biological components is obtained from an XML file; the idea is to replace this file with a standard ontology of biological components in future. Each component in a TinkerCell model has default parameters and other attributes associated with it. For example, every promoter has a "strength" parameter that indicates how well RNA polymerase binds to the specific promoter. Similarly, reactions such as transcriptional regulation have "Kd", the dissociation constant. Each parameter is defined in reference to the component in the model that it belongs with, which is different from traditional models where parameters are generally defined independent of the variables. Storing parameters in context of the parts allows TinkerCell to import parts from a database along with their parameters, which would be difficult to do if the model was simply represented as a set of equations, variables, and parameters. Additionally, TinkerCell can store information such as uncertainties associated with parameters [4]. Experimental data and other sequence related information needed for experiments can also be stored in the model.

One main focus of TinkerCell is modular circuit design. TinkerCell allows users to encapsulate a biological circuit into a module with interfaces. The interfaces are used to connect one module to another. The circuit represented inside each module can be defined visually or using a text-based language [5]. Connections between modules can take different interpretations depending on the plug-in that is using that information. For

example, the plug-ins that perform simulation and similar mathematical analysis have a specific interpretation of connections between modules: when two components from separate modules are connected, those two components are merged into a single component. For example, when an output protein in one module is connected to an input protein in another module, then the simulation plug-in constructs a model in which the two proteins are represented by the same variable. The user interface for building models using modules allows encapsulating the internal details of a module, providing an interface similar to MATLAB's Simulink (see Figure 1).

In conclusion, TinkerCell is an example of a computer-aided design tool for synthetic biology. TinkerCell models are well structured and can hold a wide variety of information. The plug-in capability is the key feature because other researchers can use TinkerCell as a means of exposing their algorithms to the community. It will also benefit a user because more analysis methods will be available in TinkerCell. The current list of functions include deterministic and stochastic simulation, steady state analysis, sensitivity analysis, bifurcation analysis, and access to list of parts from RegulonDB. There are plans for adding functions related to sequence annotation and parts validation.

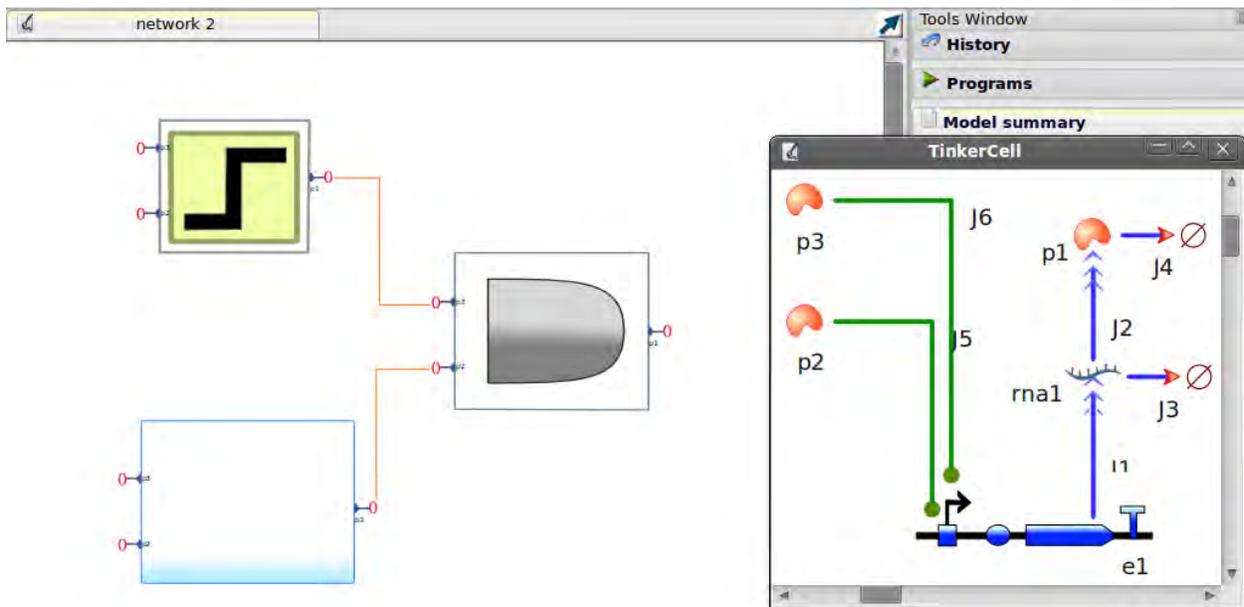


Figure 1. TinkerCell's user interface for building models using modules allows encapsulating the internal details of a module, providing an interface similar to MATLAB's Simulink.

- [1] Serrano, L. Synthetic biology: promises and challenges Nature Publishing Group, 2007
- [2] Kuznetsov, A. Synthetic Biology as a proof of Systems Biology Handbook of Research on Systems Biology Applications in Medicine, IGI Global, 2009
- [3] C. Deepak, B. Frank, and S. Herbert. TinkerCell: modular CAD tool for synthetic biology. *Journal of Biological Engineering*, 3.
- [4] S. Marino, I.B. Hogue, C.J. Ray, and D.E. Kirschner. A methodology for performing global uncertainty and sensitivity analysis in systems biology. *Journal of theoretical biology*, 254(1): 178–196, 2008.
- [5] L.P. Smith, F.T. Bergmann, D. Chandran, and H.M. Sauro. Antimony: A modular model definition language. *Bioinformatics*, 25(18), 2009.

Automatic Compilation from High-Level Languages to Genetic Regulatory Networks

Jacob Beal
BBN Technologies
Cambridge, MA 02138
Email: jakebeal@bbn.com

Ting Lu
MIT
Cambridge, MA 02139
Email: tinglu@mit.edu

Ron Weiss
MIT
Cambridge, MA 02139
Email: rweiss@mit.edu

Designing an engineered genetic regulatory network to produce a desired behavior is an extremely difficult task. Even given a combinatorial library of standard biological parts, such as BioBricks[1], there are a wide range of potential interactions between DNA, signaling molecules, expression machinery, metabolic resources, etc. As a result, a designer must solve a complicated multi-dimensional constraint and optimization problem in order to produce a working system.

Our goal is to improve the design of complicated biological systems, by finding programming abstractions and compilation techniques suitable for biological systems and then applying compiler and optimization algorithms adapted from electronic computers. A biological system designer would thus begin by expressing desired system function using a biologically-focused high-level programming language. The compiler transforms this design systematically into a genetic regulatory network, optimizing to conserve scarce biological resources (e.g. metabolic load, applicable BioBrick parts). This design can then be simulated and finally realized in cells with DNA assembled using standard protocols such as BioBricks or BglBricks[2].

A number of other projects are also attempting to address problems in biological systems design, mostly either through modelling standards, such as SBML[3] and CellML[4], or means to simplify biological model building, such as Antimony[5], little b[6], and ProMoT[7]. A few tools, like Eugene[8] and GenoCAD[9] are beginning to offer the ability to do “assembly-language” level composition of synthetic biology elements. Perhaps the most similar to this project is GEC[10], which attempts to automate the design process via iterative simulation.

We have chosen to work with designs expressed in the Proto spatial computing language[11], as it seems particularly well-matched to this goal:

- The Proto dataflow computation model matches well with the continuous parallel expression of proteins in genetic regulatory networks.

- Proto’s spatial primitives offer a path toward construction of complex multicellular systems, such as tissues and biofilms.
- We have previously shown that genetic regulatory networks generated from Proto programs are good targets for standard compiler optimization techniques[12].

At present, our compiler takes Proto programs expressed in a limited subset of the language and uses design motifs to transform them into a genetic regulatory network that uses chemical constants within the envelope of experimentally verified synthetic biological systems.

Motif-based compilation associates each high-level primitive with an abstract genetic regulatory network pattern. Using an extension to the Proto language, we associate high-level language primitives with design motifs. For example, a logical “not” operation is declared as follows:

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

The first line declares the primitive operation “not” to be a function that takes a boolean as input and returns a boolean as output. The remainder of the expression annotates this high-level function with a BioBrick motif consisting of a single regulatory region. First comes a promoter (P) annotated with the regulatory region’s constitutive expression rate of 0.193 molecules per second. This promoter is repressed (R-) by `arg0`, the first input argument to the function. Then comes a placeholder for proteins representing the function’s output, followed by a terminator T.

More complex operations can be mapped to a motif involving multiple regulatory regions, such as this two-input logical “and”:

```
(primitive and (boolean boolean) boolean
:bb-template
```

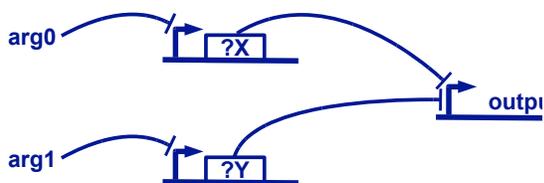


Fig. 1. Motif-based compilation associates each high-level primitive with an abstract genetic regulatory network pattern, such as an implementation of a two-input logical “and” operation. The compiler transforms each operation in the program to a motif, linking together according to the flow of data in the computation.

```
(P 0.193 R- arg0 ?X T)
(P 0.193 R- arg1 ?Y T)
(P 0.193 R- ?X R- ?Y outputs T)
```

In this case, the local variables ?X and ?Y connect connections between the three regulatory regions in this motif, implementing the logical “and” as a “nor” with inverters on each input (Figure 1).

Given a set of such mappings and a program, the compiler transforms each operation in the program to a motif, linking them together according to the flow of data in the computation. The resulting genetic regulatory network can be output in several forms, including a set of generated MATLAB files that can be used for simulation of the system. Currently this models the system with ordinary differential equations, but in future work we plan to output stochastic reactions as well.

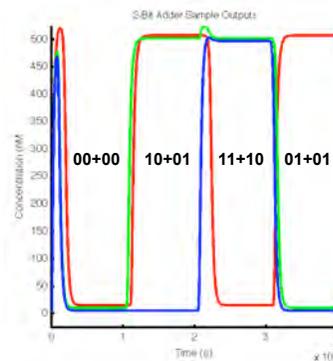
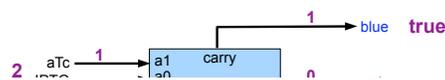
We have transformed a number of Proto programs into genetic regulatory networks using this compiler, and simulation in MATLAB verifies that the behavior of the genetic regulatory network correctly implements the original program, even for highly complex systems such as the two-bit adder shown in Figure 2, which the compiler currently transforms into an unoptimized network of 60 signal chemicals and 52 regulatory regions.

In future work, we plan to connect the compiler to existing parts libraries such that a genetic regulatory network can be compiled all the way to a DNA sequence and assembly instructions for constructing this sequence from standardized parts and to verify the behavior of compiled programs *in vivo*. We also plan to implement compiler optimizations, which we expect will improve generated program size radically, and to expand coverage to programs with extent in space and time.

REFERENCES

[1] R. P. Shetty, D. Endy, and J. Thomas F Knight, “Engineering biobrick vectors from biobrick parts,” *Journal of Biological Engineering*, vol. 2, no. 5, 2008.

[2] J. C. Anderson, J. E. Dueber, M. Leguia, G. C. Wu, J. A. Goler, A. P. Arkin, and J. D. Keasling, “Bglbricks: A flexible standard for biological part assembly,” *J. Biol. Eng.*, vol. 4, 2010.



(b) Simulated chemical implementation

Fig. 2. A two-bit adder takes two 2-bit numbers as input and produces a 2-bit number as output, plus a carry bit that is true when the sum is above 3. An example, adding 2+3 to produce 5 (1+carry) is shown in purple in (a), with inputs encoded as small-molecule signals and outputs as fluorescent proteins. ODE chemical simulations of the compiled genetic regulatory network show correct implementation, as demonstrated by the examples in (b; offset vertically for visibility).

[3] A. Finney, M. Hucka, B. J. Bornstein, S. M. Keating, B. M. Shapiro, J. Matthews, B. K. Kovitz, M. J. Schilstra, A. Funahashi, J. Doyle, and H. Kitano, “Software infrastructure for effective communication and reuse of computational models,” in *System Modeling in Cell Biology: From Concepts to Nuts and Bolts*, Z. Szallasi, J. Stelling, and V. Perival, Eds. MIT Press, 2006.

[4] A. Garny, D. Nickerson, J. Cooper, R. W. dos Santos, A. Miller, S. McKeever, P. Nielsen, and P. Hunter, “Cellml and associated tools and techniques,” *Philos. Transact. A: Math Phys Eng Sci*, vol. 366, no. 1878, pp. 3017–43, September 2008.

[5] L. P. Smith, F. T. Bergmann, D. Chandran, and H. M. Sauro, “Antimony: a modular model definition language,” *Bioinformatics*, vol. 25, no. 18, pp. 2452–54, 2009.

[6] A. Mallavarapu, M. Thomson, B. Ullian, and J. Gunawardena, “Programming with models: modularity and abstraction provide powerful capabilities for systems biology,” *Journal of The Royal Society Interface*, vol. 6, no. 32, pp. 257–270, 2009.

[7] S. Mirschel, K. Steinmetz, M. Rempel, M. Ginkel, and E. D. Gilles, “Promot: Modular modeling for systems biology,” *Bioinformatics*, vol. 25, no. 5, pp. 687–689, 2009.

[8] Berkeley Software 2009 iGem Team, “Eugene,” http://2009.igem.org/Team:Berkeley_Software/Eugene, October 2009, Retrieved May 10, 2010.

[9] M. Czar, Y. Cai, and J. Peccoud, “Writing dna with genocad,” *Nucleic Acids Research*, vol. 37, no. W40–7, 2009.

[10] M. Pedersen and A. Phillips, “Towards programming languages for genetic engineering of living cells,” *Journal of the Royal Society Interface*, 2009.

[11] J. Beal and J. Bachrach, “Infrastructure for engineered emergence in sensor/actuator networks,” *IEEE Intelligent Systems*, pp. 10–19, March/April 2006.

[12] —, “Cells are plausible targets for high-level spatial languages,” in *Spatial Computing Workshop*, 2008.

Designing biological devices in GEC

James Brown^{*†} Neil Dalchau^{*} Michael Pedersen^{*‡} Andrew Phillips^{§¶}

This paper presents a programming language for Genetic Engineering of Cells (GEC), initially described in [3] and available at <http://research.microsoft.com/gec>. The main goal of GEC is to facilitate the design, analysis and implementation of biological devices inside living cells. GEC builds on previous research in the field of synthetic biology, including a standard registry of parts (<http://partsregistry.org>) together with experimental techniques for combining these parts into higher-level devices. More recently, a range of software tools have also been developed for designing and simulating biological devices, as discussed for example in [4, 3]. The main innovation of GEC is to take the design process a step further, by allowing biological devices to be designed with little or no knowledge of the specific parts available. The user needs only a basic knowledge of the available part types, such as promoters, ribosome bindings sites, protein coding regions and terminators. These basic part types can be composed in sequence or in parallel, and the desired properties of the parts can be expressed as constraints in the GEC language. Once the biological device has been designed, the GEC compiler automatically determines the sets of actual parts that satisfy the design constraints. In most cases, multiple solutions are possible for a given design. GEC compiles a given solution to a set of chemical reactions, which can then be simulated or analyzed by the user. The solution that gives the best results can then be synthesized and put to work in living cells.

We illustrate the approach on a simple example system (Fig. 1), based on the repressilator network of transcriptional regulators [2]. The circuit is specified as a sequence of three transcriptional units, where each unit consists of a promoter (prom), a ribosome binding site (rbs), a protein coding region (pcr) and a terminator (ter). Additional constraints are specified in the form of part properties, such as the product of a protein coding region (codes) or the negative regulation of a promoter (neg), and logical variables are used to represent unknown proteins. A desired characteristic of the network is that the protein coding region of one transcriptional unit must repress the promoter of the next, in a cyclic manner. When the user compiles this design in GEC, they are presented with a set of 24 possible solutions that satisfy the design constraints. The user can then simulate each of the solutions in order to choose the best one. The design can be further refined by specifying that certain rates such as transcription, translation or transcription factor binding must lie within a specified range. This helps to reduce the initial set of possible solutions. In the case of the repressilator design, the first solution represents a condition whereby one of the promoters is much stronger than the other two. This unwanted solution can be eliminated by requiring that all three promoters are of similar strength, which reduces the number of possible solutions to six.

As a case study, GEC was used to design a significantly more complex circuit (Fig. 1), in which bacteria were engineered to exhibit predator-prey interactions [1]. Here the design also includes interactions between proteins together with transport reactions across the cell membrane. In order to map these logical variables and design constraints to physical parts, the GEC system includes a database of parts, together with a database of protein-protein interactions. Each of the parts is associated with a part identifier together with zero or more part properties. For example, the database entry (`c0051`→`pcr,codes(cIR,0.001)`) denotes a protein coding region `c0051`, which codes for the protein `cIR` at the given transcription rate. The entry (`r0051`→`prom,neg(cIR,1.0,0.5,0.00005),con(0.12)`) denotes a promoter `r0051` that is negatively regulated by `cIR` and is associated with various transcription factor binding and unbinding rates, together with the constitutive and repressed transcription rates. These rates are used to simulate the design solutions. The database of reactions simply denotes possible protein interactions, such as (`luxR+m3OC6HSL`→`{1.0} luxR-m3OC6HSL`), which denotes the formation of a complex between `luxR` and `C6`. Although such augmented databases do not yet exist on a large scale, our approaches outlines

^{*}Microsoft Research, 7 JJ Thomson Avenue, Cambridge CB3 0FB, UK

[†]Department of Plant Sciences, Downing Street, Cambridge CB2 3EA, UK

[‡]LFCS, School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, UK

[§]submitted for both oral and poster presentation

[¶]Corresponding author (andrew.phillips@microsoft.com)

a proposed solution for how these databases could be designed and implemented in future, with the ultimate goal of automating the rational design of biological devices.

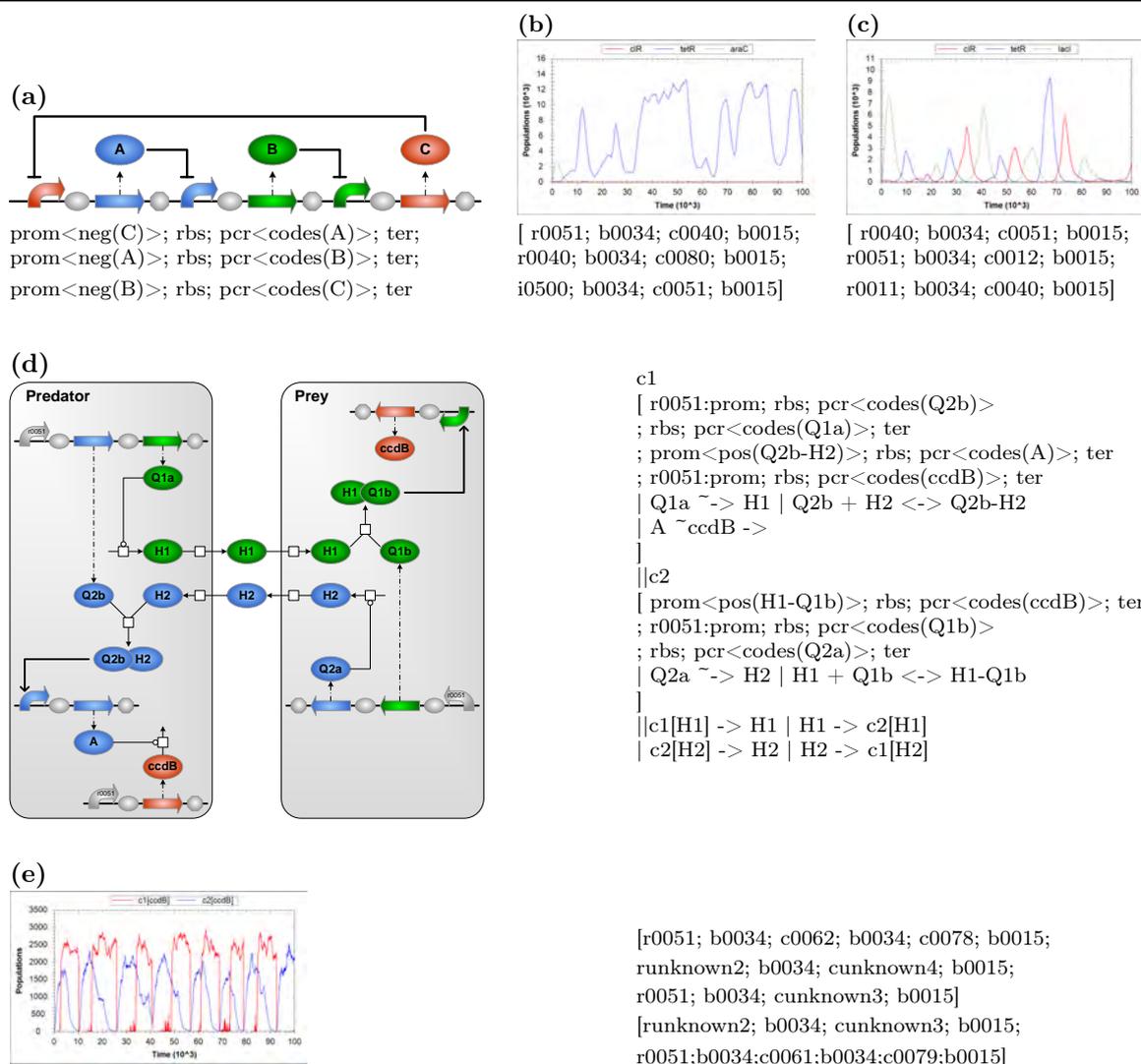


Figure 1: Designing biological devices in GEC. (a) GEC code for the repressilator network [2], together with its graphical representation, expressed in terms of part types and logical variables. Note that none of the parts are specified explicitly. The design yields a large number of possible solutions (b). Simulation of the first solution in GEC. The solution does not exhibit the desired oscillations and is not a candidate for synthesis. (c) Simulation of an alternative solution in GEC. The solution exhibits the desired oscillations and is a candidate for synthesis. (d) GEC code for a predator-prey system [1], together with its graphical representation. (e) Simulation of one of the solutions in GEC, which gives rise to oscillatory behavior.

References

- [1] Frederick K Balagaddé, Hao Song, Jun Ozaki, Cynthia H Collins, Matthew Barnet, Frances H Arnold, Stephen R Quake, and Lingchong You. A synthetic escherichia coli predator-prey ecosystem. *Mol Syst Biol*, 4:187, 2008.
- [2] M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, Jan 2000.
- [3] Michael Pedersen and Andrew Phillips. Towards programming languages for genetic engineering of living cells. *J R Soc Interface*, 6 Suppl 4:S437–S450, Aug 2009.
- [4] Priscilla E M Purnick and Ron Weiss. The second wave of synthetic biology: from modules to systems. *Nat Rev Mol Cell Biol*, 10(6):410–422, Jun 2009.

Tech. Talks Session 2 – Modeling and Standards for Bio-Design

A Semantic Knowledge Base of Standard Biological Parts

Michal Galdzicki¹, Cesar Rodriguez², Deepak Chandran³, John H. Gennari¹, Herbert M. Sauro³

¹Biomedical & Health Informatics, University of Washington, Seattle, WA 98195, USA

²BIOFAB, University of California, Berkeley, CA 94720, USA

³Bioengineering, University of Washington, Seattle, WA, 98195, USA

mgaldzic@uw.edu, carodriguez@lbl.gov, deepakc@uw.edu, gennari@uw.edu, hsauro@uw.edu

To effectively build from prior work and best practices, synthetic biology researchers need standardized descriptions of biological parts. The common approach of storing data about biological parts in a spread sheet is convenient for a small laboratory, but it is too ambiguous to establish an efficient engineering pipeline in synthetic biology. Descriptions of parts using an unambiguous language can be leveraged by software that help researchers make appropriate design decisions. To facilitate the use of such descriptions by software tools, we have created a synthetic biology knowledge base. This resource uses the Synthetic Biology Open Language Semantic (SBOL semantic) as its organizing structure. We provide information retrieval services via a simple software library (libSBOL) and a standard query language interface. Currently, we have populated the knowledge base with information from the Registry of Standard Biological Parts [1], and we have made it available for public use. In this abstract we describe the knowledge base structure, data access capabilities, and the implications of our work for automating synthetic biology design.

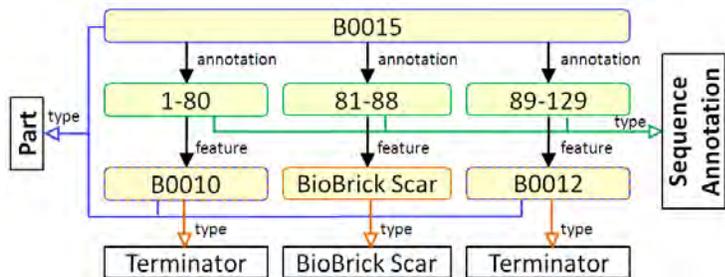
Our aim for the design of the knowledge base is to represent a rough consensus of the semantics of synthetic biology theory and practice. We use SBOL Semantic as the information model of core synthetic biology concepts and their relationships. The structure of the SBOL Semantic core module reflects the elements and layout of the Registry of Standard Biological Parts, the widely adopted public repository of parts. We have developed this model with the help of collaborators from the Synthetic Biology Data Exchange Group, a community interested in improving electronic information exchange in synthetic biology. The current iteration of the information model was developed using an open process for the evolution and standardization of data models [2]. The new work builds on the Provisional BioBrick Language (PoBoL) [3, 4]. As part of the collaboration to create the Synthetic Biology Open Language (SBOL), SBOL semantic is one of three components, along with SBOL Visual [5] and SBOL Script [6]. Each sub-language serves an orthogonal purpose in the exchange of synthetic biology information. This approach in the development of SBOL semantic aims to improve its adoption for sharing standardized biological part information across computer networks.

The main concepts represented by the knowledge base are a set of *Classes* and a *sub-class* hierarchy, as shown in Table 1. The *Class* names used in SBOL semantic are intended as the most general level terminology of distinct categories of common information objects used in the synthetic biology engineering process (Table 1). We plan to further expand the *Class* hierarchy to provide a richer vocabulary for the description of synthetic biology constructs.

Sample	Aliquot of <i>Cells</i> or <i>DNA</i> material in a container
Cell	Basic functional unit of life
Physical DNA	Continuous DNA molecule
Plasmid	Extra chromosomal DNA capable of replicating independently from chromosomal DNA
Part	A standardized building block for synthetic biology
Vector Backbone	A special <i>Part</i> into which the construct of interest is inserted to be transfected into <i>Cells</i>
Assembly Standard	Set of <i>Sequence Features</i> which designate a physical composition standard
Sequence Annotation	Position and direction describing the region for a <i>Sequence Feature</i> of a <i>Part</i>
Sequence Feature	Description of primary <i>Annotations</i> of nucleic acid sequence
BioBrick Scar	Sequence between adjacent <i>Parts</i> , created as a byproduct of Assembly Standard 10
Terminator	Transcriptional terminator sequence, example of a type of Sequence Ontology term

Table 1. Top level *Class* (bold) and example *sub-class* (regular face) SBOL semantic terminology with a simplified definition for clarity.

The SBOL semantic structure provides information about instances of each *Class*. All standard biological parts belong to the generic category of the *Class* Part and share the same kinds of properties, such as having a DNA sequence, a text description, and relationships to members of other *Classes*. For example, to create a link between information about a Part (B0015) to its DNA sequence composition, a *Sequence Annotation* relates the position and strand information to the associated *Sequence Features*, such as a transcriptional *Terminator* or *BioBrick Scar* (Figure 1). Additionally, we will extend the core SBOL semantic model to encompass the rich *Sequence Feature* categorization provided by the Sequence Ontology [7]. The defined SBOL semantic framework explicitly captures the information and the structure necessary to create a knowledgebase of *Parts* for biological engineering.



Figure

1. *Classes* (black rectangles) describe types (open faced arrows, colored by type) of *Individual* data elements (yellow rounded rectangles) and the composition relationships between them (closed faced arrows).

We have expressed SBOL semantic using RDF [8], a general-purpose language for representing information on the Semantic Web. This standard encoding allows the use of generic RDF tools to read, manipulate, and interpret SBOL data records. We relied on RDFLib [9], one such tool, to develop libSBOL, our Python software library. Then, we used libSBOL to annotate the >3,500 records we obtained from the Registry of Standard Biological Parts

with SBOL semantic structure. The SBOL encoded data, stored in Sesame database [10], is accessible by local or remote query using SPARQL, a W3C recommended query language for RDF data. For example, we used libSBOL to populate SBOLr [11], our clone of the JBEI Registry [12], though which we provide a web interface to the information contained in this knowledge base of biological parts.

Reuse of components in synthetic biology research is one of the key attributes of the engineering process which makes possible the construction of new systems with increased complexity. The RDF based SBOL framework allows us to capture the richness of semantically structured descriptions and to continue to incorporate new information needed for design in synthetic biology. Automation of design promises to make building biological machines more efficient. Finding parts that meet the specifications of designs is a critical aspect of automation of the engineering process. Leveraging Semantic Web tools to perform information retrieval can fulfill this need and offer additional benefits such as consistency checking through automated inference. Adopting these capabilities to biological system design should allow engineers to use previously created solutions and apply them to solve novel problems.

References:

1. **Registry of Standard Biological Parts** [<http://parts.mit.edu>]
2. **Draft of an RDF-based framework for the exchange and integration of Synthetic Biology data** [<http://hdl.handle.net/1721.1/45143>]
3. **Provisional BioBrick Language (PoBoL)** [<http://hdl.handle.net/1721.1/45537>]
4. Galdzicki M, Chandran, D, Sauro, HM, Cook, DL, Gennari, JH: **Bridging Synthetic Biology Models and Experiments using PoBoL**. In: *IWBDA: 2009; San Francisco, CA; 2009*.
5. Rodriguez C, Bartram S, Ramasubramanian A, Endy D: **BBF RFC 16: Synthetic Biology Open Language Visual (SBOLv) Specification**. *BBF RFC; 16* 2009.
6. **Eugene** [<http://sourceforge.net/projects/eugene/>]
7. Eilbeck K, Lewis S, Mungall C, Yandell M, Stein L, Durbin R, Ashburner M: **The Sequence Ontology: a tool for the unification of genome annotations**. *Genome biology* 2005, **6**(5):R44.
8. **RDF/XML Syntax Specification (Revised)** [<http://www.w3.org/TR/REC-rdf-syntax/>]
9. **RDFLib** [<http://www.rdfliib.net>]
10. Aduna B: **Sesame**. In. Amersfoort, Netherlands; 2009.
11. **SBOL Registry** [<http://sbolr.bhi.washington.edu/>]
12. Ham T, Dmytriv, Z, Adams, PD, Keasling, JD: **JBEI Registry: Towards a Distributed Web of Registries**. In: *IWBDA: 2009; San Francisco; 2009*.

Modeling and predicting the strength of bacterial promoters: a test-case with promoters regulated by *Escherichia coli* σ^E

Virgil A. Rhodius¹, Vivek K. Mutalik^{1,3} and Carol A. Gross^{1,2}

¹Department of Microbiology and Immunology, and ²Department of Cell and Tissue Biology, UCSF, San Francisco, CA94155. ³Current address: Joint Bioenergy Institute, Physical Biosciences Division, Lawrence Berkeley National Laboratory, Berkeley, CA94720.

Matching the output of one circuit with the input of the next circuit requires the ability to dial-in promoters with the desired output characteristics. Currently, there are no available models that reliably predict the strength of bacterial promoters from the sequence of the promoter. Using the 60 natural promoters recognized by σ^E , an alternative sigma (σ) factor in *E. coli*, as a test bed, we have made significant progress in predicting the strength of these promoters from their sequence, as well as identifying the sequence properties that distinguish between active and weak/inactive promoters. We believe our approach can be generalized to promoters recognized by other σ s.

Bacterial promoters are constructed of multiple poorly conserved motifs separated by variable length spacers (Fig. 1), making them difficult to predict accurately. In order to assess the contribution of each motif to promoter strength we divided the sequence of each promoter into 7 regions (modules), based on their sequence conservation and interactions with RNA polymerase (RNAP) (Fig. 1). We then used 3 approaches that reflected the binding requirements with RNAP to score each of these modules. (1) Position weight matrices (PWMs) that score aligned motifs and assume each nucleotide position contributes independently and additively to protein-DNA binding energy. PWMs have been well-documented in correlating protein-DNA binding energies of transcription factors with DNA sequence. Here we used PWMs to score the core promoter motifs that are predominantly recognized by the σ subunit of RNA polymerase (RNAP) (Fig. 1). (2) Upstream regions of promoters often contain tracts of As and Ts that are required to bind the α subunits of RNAP (Fig. 1). We scored these sequences using frequency counts of 3 nt A- and T-tracts to mimic the binding requirements of these subunits. (3) The variable locations of the core -35, -10 and +1 promoter motifs affect promoter strength. We employed a combined penalty term for suboptimal placement of these motifs (Fig. 1). Each promoter was then scored by summing the scores of each module to generate a total promoter score. Finally, promoter strength was measured *in vivo* using promoters fused to GFP in strains expressing basal levels of σ^E . These are stringent assay conditions with low levels of σ^E , consequently, only 18 of the 60 promoters in our library are active.

Our best model for predicting the strength of active promoters was comprised of cumulative scores of just select modules: 3 nt A- and T-tract counts in sequences upstream of the core promoter, PWMs of the -35, -10 and start motifs, and spacer penalties for suboptimal placement of the -35, -10 and start motifs. Scores of each of these modules positively correlated with promoter strength. In contrast, the remaining modules negatively correlated with promoter strength; consequently, scoring promoters by summing the scores of all modules provided no correlation with promoter strength. We optimized our best model by removing outlier promoters based on their high residuals and leverage properties; most of these outliers contained anomalously low scores in one module, suggesting unusual sequence characteristics. This optimized model generated strong correlations with promoter strength ($R = 0.88$). Finally, we tested our model using 10-fold cross-validation: this generated validated promoter scores that correlated well with promoter strength ($R = 0.82$), demonstrating good predictive utility. These results demonstrate that promoter strength can be described using PWMs of select core promoters sequences and counts of A- and T-tracts within promoter upstream regions. In addition, our best model was constructed only from sequences of promoters that were active under our stringent assay conditions, rather than using all

promoter sequences that are active under less stringent conditions. This demonstrates that only strong promoters should be used to construct accurate promoter strength models.

Many models that predict promoters make many false predictions, reflecting their poor ability to discriminate between functional and non-functional sequences. We tested the specificity of our promoter strength model by its ability to accurately distinguish between the 18 active promoters and the remaining weak/inactive promoters in our dataset. Using the model to generate a total promoter score for the inactive promoters poorly distinguished them from active promoters, since their scores overlapped with those of active promoters. However, most of the inactive promoters contained scores for at least one module that were much lower than the scores of the same module within the active promoters. Consequently, applying a threshold of lowest score for each module in the active promoters successfully identified 95% of the inactive promoters with lower scoring modules. This indicates that there is a minimum requirement of each promoter module for promoter function. Also, whilst predicting promoter strength only required scores of select modules, additional modules such as the discriminator and initial transcribed region were important for predicting promoter function.

Our findings have significant implications for accurate modeling of promoter sequences. First, promoter strength can be predicted using select combinations of PWMs and A-/T-tract counts. This suggests that it will be possible to forward-engineer promoters of specific strengths based on a common backbone sequence. Second, a two-step approach should be used for accurate promoter prediction: (1) using all promoter modules with minimum score cutoffs to predict functional promoters; and (2) using select promoter modules to score promoter strength. Finally, it is highly likely that these findings will be applicable to other promoters recognized by alternative σ s that also recognize relatively well-conserved sequences. We are currently testing this.

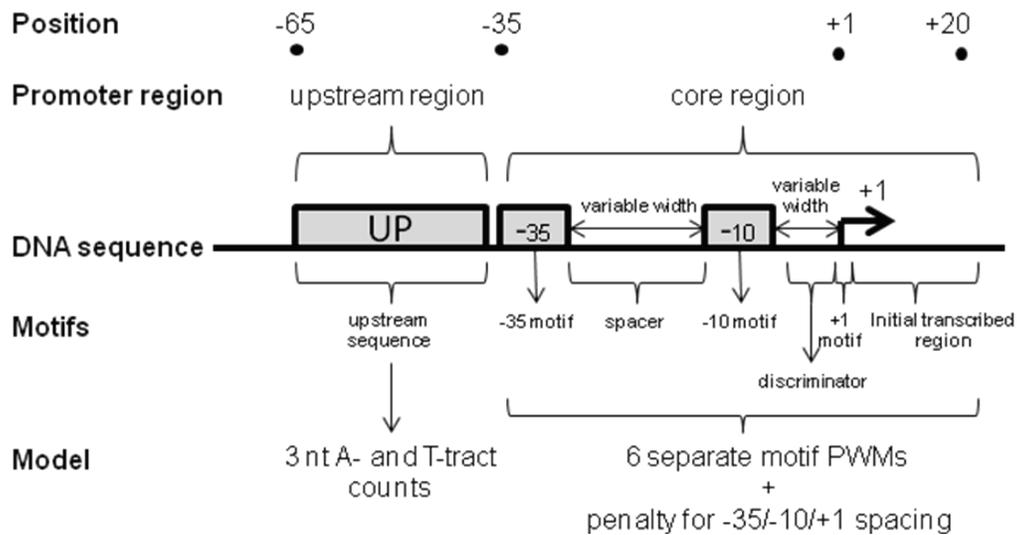


Figure 1. Structure of bacterial promoters. The figure illustrates positions of conserved motifs comprising the core and upstream regions of σ^E promoters, and the different models used to score each motif. PWM, position weight matrix; +1 is the site of transcription initiation; +1 motif is comprised of -1 and +1 position.

Resolving Variable Dependencies in the MPDE-SSA Algorithm

Abiezer Tejada, Chris Winstead, Eduardo Monzon, Chris Myers, Curtis Madsen
Utah State University

In this work, we propose some enhancements to a stochastic simulation method, called Marginal Probability Density Evolution (MPDE), that the authors previously developed [8]. The new approach introduces a Linear Gaussian Network (LGN) approximation during brief intervals of the stochastic simulation. This approach can be used to establish statistical independence among species in order to create modular models. The approach can also be used to verify independence assumptions in circuits that are synthesized from modular models. Additionally, when the LGN approximation is allowed, Principal Component Analysis (PCA) can be used to improve the accuracy of the MPDE algorithm. Further details of the proposed methods are described below.

Synthetic biology is an emerging science that intends to develop new ways to engineer biological systems. The subfield of genetic circuits consists of methods and tools for designing functional behavior in organisms by inserting exogenous genetic instructions. One major area of research for genetic circuits is to analyze and predict the behavior of synthetic gene networks by means of computational tools [1]. However, the randomness of these circuits makes *in silico* analysis cumbersome [4]. Moreover, due to complex protein interactions and stochastic events, it is difficult to establish truly modular functional models for genetic parts.

Mathematical models have been created to characterize, predict, and modify the behavior of genetically engineered networks. Chemical reaction models can be transformed into a set of first order differential equations (ODEs). Although ODE models are generally amenable to modular descriptions, they wrongly assume that a system’s chemical species vary deterministically and continuously, which often results in erroneous predictions [5]. Therefore, ODE models can make incorrect predictions when applied to highly stochastic systems, thus requiring stochastic analysis for accurate and robust designs of genetic circuits.

To arrive at a modular approach to stochastic genetic circuits, some researchers propose using probability-transfer models [2]. Probabilistic models show some promise for modular synthesis strategies in the forward design of genetic circuits. The authors recently demonstrated a modular probabilistic approach for synthesizing a “quorum trigger” circuit [6]. In this example, the probabilistic model provided three main benefits: (1) Intuitive abstract behavioral models of the circuit’s genetic components; (2) A coherent procedure for forward-design based on modular logic parts; and (3) A framework for estimating the reliability of the synthesized function.

Although abstract probabilistic models can be useful for design, there remains a gap between high-level modular models and low-level stochastic reaction-network models. Modular models implicitly assume functional independence among a circuit’s components. Current tools provide little aid for establishing that independence a-priori, or for verifying independence during simulation. To help fill this gap, the authors previously devised the MPDE method, which tracks the marginal statistical evolution of species in a reaction system. Whereas traditional Stochastic Simulation Algorithms (SSAs) generate a scalar value for each species at each time increment, the MPDE method generates the marginal probability density function for each species in the system. The goal of this approach is to represent the system’s functional behavior using an intuitive signal-plus-noise model, while staying faithful to the physics of the reaction network. Researchers frequently use some form of mean-plus-deviation representation when publishing SSA results, but there is currently no generally applicable procedure for abstracting species’ statistics in this format.

In its original presentation, the MPDE method relied on the assumption that, during a brief time-interval, all species variations are pair-wise conditionally independent, given the system’s total state. This assumption is not always accurate. In fact, some variables may be highly dependent on each other, which may completely invalidate the simulation results. Previous accounts of the MPDE method also offered no means of testing for dependencies. In this work, we identify some approximate but useful procedures for testing and resolving

variable dependencies in MPDE simulations.

Most genetic circuits contain some pairs of highly correlated species. In many cases, the most highly-dependent variables can be identified a-priori by calculating conservation constraints in the reaction-network model. For the remaining variables, it is helpful to approximate some (or all) of the system’s species as Gaussian-distributed variables. Then, during a brief time-interval, the system may be treated as a LGN. Variable dependencies appear as significant non-zero entries in an LGN’s information matrix, which is computed as the pseudo-inverse of the covariance matrix. The information matrix can be computed periodically during simulation, and can be used to spot dependencies in the reaction system (and hence to flag violations of modular independence assumptions).

In addition to these solutions, Principal Component Analysis (PCA) may be used to automatically resolve minor dependencies within the MPDE. PCA is an algorithm that performs a decorrelating transformation and dimensionality reduction on a correlated data set [7, 3]. The PCA approach corrects for simulation errors caused by small incremental dependencies in the system’s species. By using PCA together with conservation constraints, the enhanced MPDE algorithm delivers a more trustworthy signal-plus-noise picture of the system’s behavior.

Utah State University Electrical and Computer Engineering	University of Utah Electrical and Computer Engineering
Abiezer Tejada Chris Winstead Eduardo Monzon	Chris Myers Curtis Madsen

Table 1: Authors’ affiliations

References

- [1] Tal Danino, Octavio Mondragn-Palomino, Lev Tsimring, and Jeff Hasty. A synchronized quorum of genetic clocks. *Nature*, 463(7279):326–330, Jan 2010.
- [2] B. Fett, J. Bruck, and M.D. Riedel. Synthesizing stochasticity in biochemical systems. In *Design Automation Conference, 2007. DAC ’07. 44th ACM/IEEE*, pages 640 –645, june 2007.
- [3] John A. Gubner. *Probability and Random Processes for Electrical and Computer Engineers*. Cambridge University Press, 2006.
- [4] Hiroyuki Kuwahara and Ivan Mura. An efficient and exact stochastic simulation method to analyze rare events in biochemical systems. *J Chem Phys*, 129(16):165101, Oct 2008.
- [5] Chris Myers. *Engineering Genetic Circuits*. Chapman & Hall, 2009.
- [6] Nam-Phuong Nguyen, Chris Myers, Hiroyuki Kuwahara, Chris Winstead, and James Keener. Design and analysis of a robust genetic muller c-element. *J Theor Biol*, Nov 2009.
- [7] Markus Ringnr. What is principal component analysis? *Nat Biotechnol*, 26(3):303–304, Mar 2008.
- [8] Myers Chris Madsen Curtis Winstead, Chris. issa: An incremental stochastic simulation algorithm for genetic circuits. 2010.

Tech. Talks Session 3 – Design of Biological Circuits and
Networks

Automatic design of digital synthetic gene circuits

Mario Andrea Marchisio, Jörg Stelling

*Department of Biosystems Science and Engineering and Swiss Institute of Bioinformatics,
ETH Zurich, Basel, Switzerland*

E-mail: {mario.marchisio,jorg.stelling}@bsse.ethz.ch

According to the "MIT Registry of Standard Biological Parts", standard parts are DNA traits associated with well-defined functions in transcription and translation (e.g. promoters, ribosome binding sites, coding regions, terminators). As in an electrical circuit basic parts and devices can be connected to each other thanks to the exchange of the electric current, biological parts are "composable" if they share the same input/output namely one or more fluxes of molecules referred to as common signal carriers [4]. They represent the biological counterpart of the electrons and permit to assemble composable parts into biological devices—made of one or more transcription units—and devices into genetic circuits.

Following these notions, we developed a computational tool for the visual design of synthetic gene circuits by means of standard composable parts and pools of signal carriers [9].

In our model, gene expression requires at least five kinds of signal carriers: RNA polymerases, ribosomes, transcription factors, small RNAs and chemicals. The flux of each of these types of molecules is a quantifiable biological signal that allows exchange of information between parts, devices and the cell environment. Both parts and pools are modeled independently by the ordinary differential equations (ODE) formalism and generated upon specification of the corresponding kinetic parameter values and internal structure. Since our tool has been integrated into the software ProMoT (Process Modeling Tool [10]), it permits to build genetic circuits in a "drag and drop" way—as in electrical engineering—by placing biological parts and pools on the ProMoT canvas and connecting them through "wires" that enable flow of signal carriers.

More recently, we extended our tool to the *automatic* design of digital synthetic gene circuits. Here, standard biological parts are gathered into biological Boolean gates whose logic behavior arises from the action of proteins, small RNAs, and chemicals on their promoters and ribosome binding sites (see for instance [2, 12]). Instead of applying a brute-force approach that requires to run an optimization algorithm both on the structure and on the kinetic parameter values of the network—as pointed out by François and Hakim [5] and implemented in the computational tools OptCircuit [3] and Genetdes [11]—our tool implements the Karnaugh map method to convert a truth table, which fully specifies input–output relations, into Boolean formulas (conjunctive and disjunctive normal forms) that can be directly translated into circuits organized in three layers of gates. Hence, the circuit structure is derived without the need for any optimization procedure.

The tool utilizes a library of well-working Boolean gates whose kinetic parameters have been set to default values taken from literature and tuned via simulations. A two-input AND gate, for instance, is realized in several possible configurations, depending on the different locations it can take inside a circuit: two activators binding cooperatively to the promoter [2]; two small RNAs modifying possible hairpin structures on the RBS; two chemicals activating a tandem riboswitch (similar to the structure proposed by Win and Smolke [12]); a small RNA and a chemical acting independently on the RBS; an activator binding the promoter and a chemical or a small RNA modifying the RBS. Most of these configurations are new and their working *in vivo* still requires experimental confirmation. Other possible, more complex AND gate designs (like in [1]) have not been considered even though the tool might be easily extended to include them too.

Overall, the only required input is a truth table. For each truth table the tool automatically generates several possible circuit schemes which are ranked according to a complexity score proportional to the number of different regulatory factors present in the network.

The solution chosen by the user is finally encoded into an MDL (Model Definition Language [6]) file and can be visualized inside the ProMoT graphic user interface. Here, moreover, parameter values of every single part and pool can be modified and the whole circuit can be exported into formats suitable for both stochastic and deterministic simulations (e.g. Matlab, SBML [7]).

We have to notice that, at present, the wet-lab implementation of the most intricate networks computed by the tool is not possible. In fact, in some cases, too many different transcription factors are required whereas just a handful of them is currently used in synthetic biology. Moreover, controls at translation level like riboswitches and ribozymes [8], which drastically simplify a circuit scheme, are not yet massively engineered.

Nevertheless, with the choice of parameter values and Boolean gate configurations we made, simulations of circuits with up to four inputs show a faithful and unequivocal truth table representation. Furthermore, results of stochastic simulations highlight that the digital circuits designed by our tool are considerably robust to the intrinsic noise: this seems to confirm the validity of our electronics-based approach.

References

- [1] J. C. Anderson, C. A. Voigt, and A. P. Arkin. Environmental signal integration by a modular and gate. *Mol Syst Biol*, 3:133, 2007.
- [2] L. Bintu, N. E. Buchler, H. G. Garcia, U. Gerland, T. Hwa, J. Kondev, T. Kuhlman, and R. Phillips. Transcriptional regulation by the numbers: applications. *Curr Opin Genet Dev*, 15(2):125–135, Apr 2005.
- [3] M. S. Dasika and C. D. Maranas. Optcircuit: an optimization based method for computational design of genetic circuits. *BMC Syst Biol*, 2:24, 2008.
- [4] D. Endy. Foundations for engineering biology. *Nature*, 438(7067):449–453, Nov 2005.
- [5] P. François and V. Hakim. Design of genetic networks with specified functions by evolution in silico. *Proc Natl Acad Sci U S A*, 101(2):580–585, Jan 2004.
- [6] M. Ginkel, A. Kremling, T. Nutsch, R. Rehner, and E. D. Gilles. Modular modeling of cellular systems with ProMoT/Diva. *Bioinformatics*, 19(9):1169–1176, Jun 2003.
- [7] M. Hucka et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, Mar 2003.
- [8] F. J. Isaacs, D. J. Dwyer, and J. J. Collins. RNA synthetic biology. *Nat Biotechnol*, 24(5):545–554, May 2006.
- [9] M. A. Marchisio and J. Stelling. Computational design of synthetic gene circuits with composable parts. *Bioinformatics*, 24(17):1903–1910, Sep 2008.
- [10] S. Mirschel, K. Steinmetz, M. Rempel, M. Ginkel, and E. D. Gilles. Promot: modular modeling for systems biology. *Bioinformatics*, 25(5):687–689, Mar 2009.
- [11] G. Rodrigo, J. Carrera, and A. Jaramillo. Genetdes: automatic design of transcriptional networks. *Bioinformatics*, 23(14):1857–1858, Jul 2007.
- [12] M. N. Win and C. D. Smolke. Higher-order cellular information processing with synthetic rna devices. *Science*, 322(5900):456–460, Oct 2008.

Design of *in vitro* synthetic gene circuits

Elisa Franco and Richard M. Murray

elisaf@caltech.edu, murray@cds.caltech.edu
MC 107-81, Division of Engineering and Applied Sciences,
California Institute of Technology, Pasadena, CA 91125

The functionalities of every living organism are wired in the biochemical interactions existing among proteins, nucleic acids and all the other molecules that constitute life's building blocks. Understanding how to embed any function in this "hardware of life" via "molecular programming" is an exciting and challenging task for modern bioengineers and synthetic biologists. Programming molecules is indeed possible with very high precision when reactions are run *in vitro*, in a controlled environment with few components. The structural properties of nucleic acids make them ideal programmable molecules to perform molecular computation, construct of nano-devices and feedback systems.

Synthetic *in vitro* genetic circuits, developed at Caltech in the Winfree laboratory, are composed solely of nucleic acids and few protein species. Despite their simplicity, they can be used as a tool kit to design systems embedding important biological functionalities, such as self repressing or self activating modules, toggle switches and oscillators. We operate in this simple "molecular programming environment" to investigate bio-molecular design principles.

First of all we will demonstrate, numerically and experimentally, how the design of specific feedback interconnections can generate automatic regulation of output flow in a two-component *in vitro* genetic circuit. Flow control is a fundamental feature for the correct performance of large scale networks, of which familiar examples are the Internet, power grids and pipe networks. In the biological world, complex cellular pathways rely as heavily on a regulated flow of nucleic acids, transcription factors and other metabolites. We propose two new designs to regulate and match the RNA production of two *in vitro* transcriptional circuits. In particular, these architectures are based on self inhibition and cross activation mechanisms that can dynamically change the fraction of actively transcribing DNA strands and correctly respond to changes in the chemical environment. The second challenge we will consider is modularity. Since *in vitro* genetic circuits are a powerful benchmark to test new biochemical circuitry designs, it is fundamental to understand how to interconnect several different transcriptional modules in a large network, preserving their functionality. Insulation blocks are crucial to this purpose. Simple direct connection of different synthetic *in vitro* devices through input and output signaling molecules can result in heavy retroactivity effects that destroy the original desired behavior of each module. For example, we have observed that when a synthetic oscillator is used to drive a downstream nano-device (for instance, DNA-molecular tweezers or beacons), even a small amount of such "load" can alter the reference oscillatory signal. To overcome this problem, we designed a transcription-based insulation stage that decouples the core oscillator from its load. Experimental results show that the addition of such layer allows us to decrease the source signal deterioration when connecting larger amounts of downstream devices to the biochemical oscillator.

Fan-out Considerations in Gene Regulatory Networks

Kyung Hyuk Kim*, Herbert M. Sauro

Department of Bioengineering, University of Washington, William H. Foege Building Box 355061 Seattle, WA 98195-5061, U.S.A

*Email address: kkim@uw.edu

This abstract is for both a poster and oral presentations.

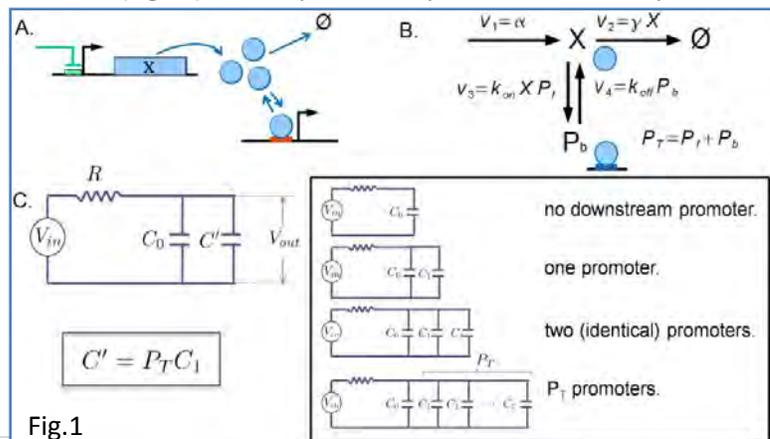
Engineering relies on modular composition that is the ability to combine functional units with the knowledge that the intrinsic property of each module is unaffected to a large degree by the composition. In biology we are less clear on the notion of a modular component, or at least biology has multiple definitions depending on application and context. Here we will define a module as a self-contained functional unit whose intrinsic properties are independent of the surrounding milieu. We will be concerned with the design of modular synthetic components and avoid the question of modularity in natural evolved systems.

In electronic engineering there exist guidelines and published constraints on how many electrical modules can be driven from a source. For example, one rule of thumb for analog circuits suggests that the impedance at the input should be ten times the impedance at the driving circuit. In digital circuits, such as TTL circuits, manufactures will quote the fan-out and fan-in for a given electrical module. The fan-out indicates how many downstream logic gates can be connected to a given output. Exceeding these limits will either cause signal distortion in analog circuits or potentially, circuit failure in digital circuits.

In synthetic biology one could imagine the development of similar criteria for connecting two biological modules together. Here we introduce the notion of fan-out for a genetic circuit. We will define the fan-out of a genetic circuit as the maximum number of downstream promoters that can be driven from an upstream circuit signal without significant time-delay or signal attenuation. We will show that the fan-out can be estimated by using gene expression noise, precisely its autocorrelation [1], and during the estimation procedure, systems retroactivity can also be measured. This fan-out/retroactivity estimation can be applied under quite general module interface conditions. Although our analysis is focused on genetic networks, the principles apply equally to protein networks.

When two synthetic gene circuits are connected, transcription factors are used to connect the two. The reaction processes involving the transcription factors (TFs) such as TF transcription, translation, degradation, and promoter-binding/unbinding, will be called module interface processes (MIPs).

We consider the case that two modules are connected (Fig.1A). The output of the upstream module is represented by X and the input of the downstream by X -specific promoters. Figure 1A represents one of the simple MIP. The dynamics of X has been known to slow down due to its downstream connection and the slow-down was defined by retroactivity [2]. We explain the slow-down by mapping the MIP (Fig.1A and B) to an RC (Resistor-Capacitor) circuit (Fig.1C). When the downstream module is not connected, the



corresponding MIP can be mapped to an RC-circuit with a capacitance C_0 . With the connection, an extra capacitance C' is shown to be connected in parallel to C_0 (Fig.1C). The total capacitance becomes the sum of the two capacitances: $C_T = C_0 + C'$. The increase in the capacitance means that the circuit responds more slowly when the downstream component is connected, specifically the response time becomes: $\tau_{P_T} = RC_T = R(C_0 + C')$, with P_T the total number of the downstream X -specific promoters (Fig.1C). We investigate how the response time is related to the total number of the downstream promoters P_T . Interestingly, C' is shown to be proportional to P_T : $C' = P_T C_1$ (Fig. 1 Black Box). This proportionality relationship becomes very useful for proposing the experimental estimation method for fan-out.

We define the gene circuit fan-out by the maximum number of promoters that the output can drive. To quantify the fan-out, we consider the frequency response of the MIP, which is shown to act as a low-pass filter with a cut-off frequency ω_c given by $\tau_{P_T}^{-1}$. Consider that the upstream module functions as a synthetic oscillator. There will be a practical upper limit in the oscillator's frequency: maximum operational frequency ω_{max} . When the cut-off frequency (Fig. 2B) is larger than ω_{max} the oscillator output will be operated in a predictable manner without any significant signal loss. As the number of the downstream promoter increases (Fig.2C) the cut-off frequency decreases. When the cut-off frequency matches with ω_{max} , the corresponding P_T is defined as the fan-out:

$$F_{\omega_{max}} = \frac{C}{C_1} \left[\frac{1/\tau_0}{\omega_{max}} - 1 \right].$$

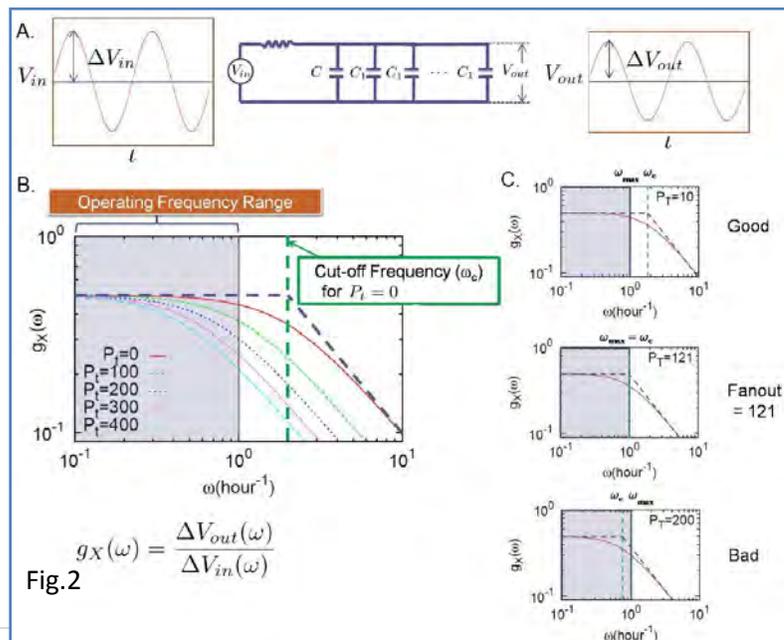
This fan-out equation has two unknown parameters: C/C_1 and τ_0 . These can be experimentally estimated by performing two independent experiments with and without any downstream X -specific promoter. In each experiment we measure the corresponding response time: τ_0 and τ_{P_T} (by using gene expression noise [1]). If the promoters are placed in plasmids, the copy number of the plasmids can be roughly known depending on what type of the origin of replication is used, and thus the copy number of the promoters P_T can be known a priori. From the known P_T and the estimated τ_0 and τ_{P_T} , we can estimate the two unknown parameters.

The above fan-out equation $F_{\omega_{max}}$ was derived from the simple MIP described in Fig.1A. The same or similar fan-out functions can be shown to be derived for the more general cases that the TFs are oligomers, are tagged for fast degradation, are auto-regulated, and regulate multiple downstream promoters having different affinities. This means that the fan-out can be estimated exactly in the same way as in the case shown in Fig.1A, by measuring the time constant with and without any downstream promoters. This measurement can be achieved from gene expression noise, specifically by estimating its autocorrelation function [1].

This fan-out study provides a way for quantifying the level of modularity in gene regulatory circuits and helps characterize and design the module interface that is required to be satisfied for the modular construction of gene circuits.

[1] Weinberger LS, Dar RD, Simpson ML (2008) Nat Genet 40:466—470.

[2] Del Vecchio D, Ninfa AJ, Sontag ED (2008) Mol Syst Biol 4:161.



Designing Biological Systems

Pamela A. Silver

Professor of Systems Biology

Harvard Medical School and the Harvard University Wyss Institute of
Biologically Inspired Engineering

Biology presents us with an array of design principles. From studies of both simple and more complex systems, we understand many of the fundamental principles of how Nature works. We are interested in using the foundations of biology to engineer cells in a logical and predictable way to perform certain functions. In doing so, we learn more about the fundamentals of biological design as well as engineer useful devices with myriad applications. For example, we are interested in building cells that can perform specific tasks, such as counting events, measuring time and remembering past events. Moreover, we design and construct proteins and cells with predictable biological properties that serve as potential therapeutics, cell-based sensors and factories for generating bio-energy, high value commodities and bio-remediation. In doing so, we progress towards the overarching goal of Synthetic Biology – to make the engineering of biology more predictable.

Programming cells as therapeutics by modular design

Chris Anderson, Berkeley

Synthetic biology is an emerging approach to genetic engineering where complex dynamic systems are built ground-up from well-characterized cellular chassis and functional parts. This approach presents solutions to some of today's most challenging problems in healthcare, chemical and materials production, bioremediation, and bioenergy.

Particularly appealing is the use of live bacteria as therapeutic agents for the treatment of cancer. We use modular design to engineer bacteria that can be safely injected into the bloodstream, localize to and invade cancer cells within solid tumors, and kill them. The challenge of constructing such agents is determining how to break down complicated, unnatural biological behaviors into simpler modular devices that can be constructed from known genetic components.

Beyond the conceptual challenge, the ability to efficiently construct large systems requires the development of new experimental and fabrication approaches including automated DNA assembly and BioCAD tools. I will describe the therapeutic bacteria system as well as the tools and experimental approaches spawned from these studies.

Feedback and Control in Biological Circuit Design

Richard M. Murray

Control & Dynamical Systems and Bioengineering California Institute of Technology

Biological systems make use of feedback in an extraordinary number of ways, on scales ranging from molecules to cells to organisms to ecosystems. In this talk I will discuss the use of concepts from control and dynamical systems in the analysis and design of biological feedback circuits at the molecular level. After a brief survey of relevant concepts from synthetic biology, I will present some recent results that combine modeling, identification, design and experimental implementation of biological feedback circuits. These results include the use of intrinsic noise for system identification in transcriptional regulatory networks, development of an *in vitro* circuit for regulating the rates of transcription of two independent genetic sequences, and design of dynamics of for an *in vivo* oscillator using transcriptional delay. Using these results as examples, I will discuss some of the open problems and research challenges in the area feedback control using biological circuits.

Tech. Talks Session 4 – Biological Pathway and Network
Optimization

PREDICTABLY PROFITABLE PATHS IN METABOLIC NETWORKS

Ehsan Ullah[□], Mark Walker^{*}, Kyongbum Lee^{*}, and Soha Hassoun[□]

[□]Department of Computer Science

^{*}Department of Chemical and Biological Engineering
Tufts University, Medford, MA

Metabolic pathway analysis is a powerful tool to identify targets for the rational redesign of complex metabolic networks. Pathway analysis is especially useful in optimizing metabolic engineering and synthetic biology objectives such as production of biofuels [1] and therapeutic proteins [2]. One approach to optimization is to improve the catalytic activity of enzymes or otherwise relieve bottlenecks in the pathway of interest.

In this context, two types of enzymatic reactions are potential engineering targets: the reaction that is the limiting step in the pathway and the reaction that is inflexible. The limiting reaction is the reaction with the lowest flux capacity in the pathway. The inflexible reaction cannot adjust its flux to support a modified flux distribution needed to meet the engineering objective. Such a reaction can be characterized by a small range of observed fluxes across different conditions. To identify pathways based on these two types of reactions, we present a new algorithm, *Predictably Profitable Path*. The algorithm considers both flux bottleneck and range to search for the most predictably profitable path in a metabolic network.

In general terms, a predictably profitable path is defined as the conservatively profitable path that is most predictable. The predictability and profitability of a path can be defined in a number of different ways, because these properties depend on the objective of pathway analysis. For analysis of flux distributions in the metabolic network, the most profitable path from a source metabolite to the destination metabolite would be the path that guarantees a minimum flux. The predictable path is defined as the path having a narrow range of flux capacity. Such a path will help in identification of inflexible reactions.

The *Predictably Profitable Path* algorithm uses graph-based searches. In the graph representation, metabolites are represented as nodes and reactions are represented as weighted edges. There are two types of edge weights: mean and range. Both types of weights are calculated from measured or estimated flux distributions. The mean weight is the nominal flux value through the corresponding reaction at a defined metabolic state. The range weight is derived from minimal and maximal flux capacity calculations for the metabolic network. The *Predictably Profitable Path* algorithm consists of two steps; removal of non-profitable edges in the network to guarantee a minimum flux from source metabolite to destination metabolite and identification of predictable path in the reduced network.

The *Predictably Profitable Path* algorithm first prunes the network by removing non-profitable edges and then identifies the most predictable path. The edge that has flux capacity below a defined threshold is non-profitable edge. The threshold value is selected such that a maximum lower value of flux can be guaranteed. This value can be calculated by assigning the lower bound of flux values as weights to the edges and then identifying the weight of best bottleneck edge in all paths from the source metabolite to destination metabolite. After pruning the network, edges are assigned weights equal to the range of flux values i.e. difference of maximum and minimum

flux for a reaction. Favored Path algorithm [4] is applied to the network favoring the minimum weights to obtain the predictably profitable path.

The algorithm is based on graph search approach having polynomial runtime. Thus, it is very efficient for large complex networks as compared to the enumeration based techniques such as elementary flux mode analysis. The algorithm is generic because by changing the weights of edges different analysis objective can be defined; for example, a weighting scheme of Gibb's free energy change will define the objective of identification of pathways that are predictably thermodynamically more feasible.

We have applied the algorithm to identify pathways in *Escherichia coli* for ethanol production from glucose [1] under anaerobic conditions. Our algorithm was able to identify the path that was engineered to maximize the production of ethanol using the flux distribution data of the network.

References

1. Trinh, C.T., P. Unrean, and F. Sreenc, *Minimal Escherichia coli cell for the most efficient production of ethanol from hexoses and pentoses*. Appl Environ Microbiol, 2008. **74**(12): p. 3634-43.
2. Ng, S.K., D.I. Wang, and M.G. Yap, *Application of destabilizing sequences on selection marker for improved recombinant protein productivity in CHO-DG44*. Metab Eng, 2007. **9**(3): p. 304-16.
3. Ullah, E., et al., *Favored Path: An Algorithm for Identifying Biochemical Pathways of Interest*. Bioinformatics, 2010, submitted.

Robust inference of biological Bayesian networks

Masoud Rostami and Kartik Mohanram
ECE Department, Rice University, Houston, TX
mrostami@rice.edu kmram@rice.edu

Rapid advancements in gene microarray technologies have led to the availability of large amounts of gene expression datasets in public repositories. By using microarrays to measure the relative expression of genes over a period of time, biologists hope to discover how different genes interact with each other in order to give rise to a specific biological phenotype. These interactions can be further exploited to predict the behavior of the network under external perturbations, which finds applications to drug discovery. However, reverse engineering of such gene regulatory networks from the time-series microarray data has remained a challenging task so far. Methods such as Bayesian networks, neural networks, and clustering have been used in literature to infer gene interactions [1]. However, the scarcity of data and inherent noise in microarray technologies has impeded the performance of these techniques. In this paper, we propose algorithms inspired by communications theory to enhance the performance of algorithms for Bayesian network (BN) inference by improving the quantization of the time-series microarray data.

Bayesian networks is one of the most widely used techniques to infer the structure of regulatory networks in practice. BN is formally defined as $\langle G, \Theta \rangle$, where G is a directed acyclic graph (DAG) that represents the conditional dependence between the gene expression variables and Θ is the set of parameters that annotates the DAG by encoding the joint probability distribution of variables as a product of conditional probabilities. Finding the best G and Θ is NP-hard. In practice, a fitness function is assigned to each network and heuristics are used to search for a network with the best fitness score [2]. The fitness scores are usually — but not necessarily — defined as $P(D|G)$, which is a measure of how likely it is that a candidate graph will generate the original data D .

BN models should be reliable to be of use in predicting system performance. The first step in BN inference is quantization of the continuous microarray data. However, there is currently no consensus on finding the exact boundaries during the quantization step. Since the microarray data already has high levels of intrinsic noise, this may render the whole process of BN inference unstable. Note that increasing the number of quantization intervals does not alleviate the problem since the required amount of data for inference has a super-exponential dependence on the number of quantization intervals, and currently available microarray technologies cannot support this huge data demand. Overall, the biological BN should be inferred in a manner such that the intrinsic noise in the microarray data does not significantly alter the structure of the graph.

Quantization of gene microarray data is usually based on either interval quantization or quantile quantization. Quantization algorithms based on clustering algorithms and threshold optimization [3] have also been proposed in literature. A common shortcoming of all these methods is that they ignore that the microarray data is inherently a time series. Every sample in a time series has a strong correlation with its neighboring samples, but the common quantization methods completely remove this information from their algorithm chain. In this paper, we investigate methods inspired by communications theory to recover and use the lost time-series information in BN inference and present results to illustrate their advantages in robust BN inference.

For intuition, assume that genes “transmit” information about their activity via the microarray “channel” by modulating the intensity of their activity. If gene activity increases, the microarray is expected to record a higher value, and vice versa. In other words, the process is analogous to pulse amplitude modulation (PAM) and demodulation via a highly noisy “channel”. In practice, modern PAM decoders use a low-pass filter at the front-end to remove unnatural spikes from the time-series data. A modern smoother, like the Loess algorithm, can be used to implement the low-pass filter. In the next step, the time-series data is quantized according to the noise model of the channel. The received samples are assessed to approximate the statistical likelihood of being “0” or “1”, i.e., $P(d = “0”|x)$ and $P(d = “1”|x)$, where d and x are the quantized transmitted data and the noisy received data, respectively. If the likelihood of being “0” is higher than likelihood of being “1”, the sample is quantized to “0” and vice versa. The likelihood term can be expanded using Bayes’ rule:

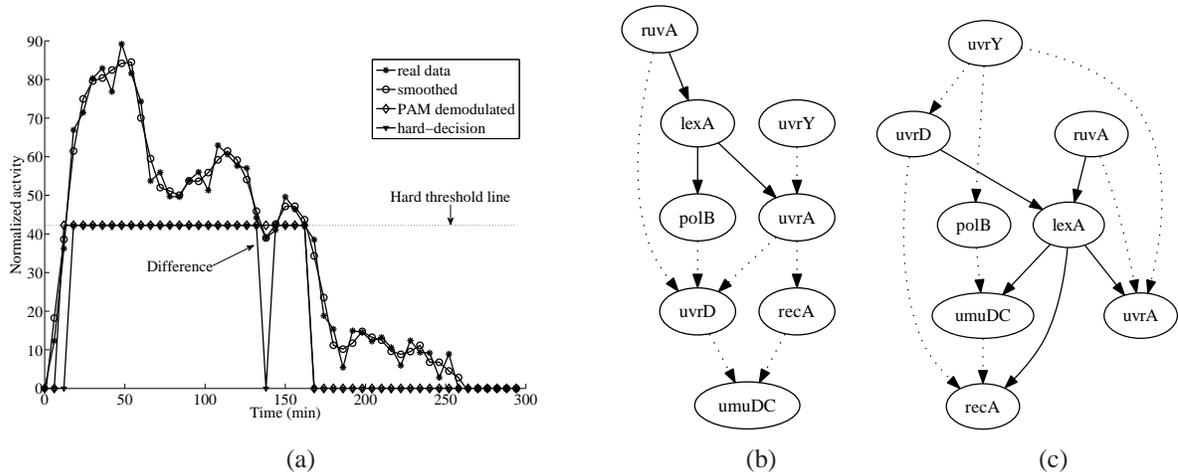


Figure 1: (a) time series from SOS DNA repair dataset, (b) network inferred by traditional quantization, and (c) network inferred by the enhanced quantization method. The true and false edges are solid and dotted, respectively.

$$P(d = "1" | x) = \frac{P(x | d = "1")}{P(x)} P(d = "1") \propto P(x | d = "1") P(d = "1") \quad (1)$$

$P(x)$ is just a scaling factor for an instance of the received time series, and can be ignored in the decision making process. The $P(x | d = "1")$ term is calculated based on the noise model that is assumed for the channel. If the decision making is solely based on this term and the channel is assumed to have additive white noise, then the decision making is equivalent to interval quantization. The second term is the probability that the transmitter has sent "1" in this sample. This probability is either known *a priori* or can be approximated by analyzing the neighborhood of this sample. If the neighborhood of a sample is clearly all "1", the probability that this sample is also "1" is also higher. So, the decoder may assign "1" to a sample, even though the value of this sample is slightly lower than the threshold.

Fig. 1(a) shows a time series for the activity of a gene from the SOS DNA repair dataset [4]. The smoothed curve is accompanied with the scaled quantized curves, with and without the $P(d = "1")$ and $P(d = "0")$ terms. The threshold is chosen to be halfway between the minimum and maximum values in the time series. The arrow shows a point where these terms make a difference in the decision making. Although that point is slightly less than the threshold, it is assigned "1" because the neighborhood of this sample makes it more probable that a "1" was transmitted by the gene. Enhancing the quantization process improves the performance of BN inference. Fig. 1(b) and 1(c) show the inferred SOS BNs with and without considering the $P(d)$ term, respectively. We use the software application Banjo for BN inference [5]. In this example, by considering the $P(d)$ term, five true edges are inferred. However, in the case of traditional quantization, only three true edges are inferred. Motivated by these results, we propose to apply our robust quantization methods to bigger networks such as the ALARM network [6]. We also believe that incorporating reported microarray noise models in the decision making process will improve its performance even further.

References

- [1] A. J. Hartemink, "Reverse engineering gene regulatory networks," *Nature Biotechnology*, vol. 23, pp. 554–555, 2005.
- [2] M. Bansal *et al.*, "How to infer gene networks from expression profiles," *Molecular Systems Biology*, vol. 3, pp. 1–10, 2009.
- [3] B. D. Camillo *et al.*, "A quantization method based on threshold optimization for microarray short time series," *BMC Bioinformatics*, vol. 6, p. S11, 2005.
- [4] M. Ronen *et al.*, "Assigning numbers to the arrows: Parameterizing a gene regulation network by using accurate expression kinetics," *Proc. National Academy of Sciences*, vol. 99, pp. 10555–10560, 2002.
- [5] "Banjo: Bayesian Network Inference with Java Objects."
- [6] I. A. Beinlich *et al.*, "The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks," in *European Conference on Artificial Intelligence in Medicine*, pp. 247–256, 1989.

Pathway Identification for Strain Engineering

Mona Yousofshahi^Δ, Kyongbum Lee^{*}, and Soha Hassoun^Δ

^ΔDepartment of Computer Science

^{*}Department of Chemical and Biological Engineering
Tufts University, Medford, MA

Introduction

Engineering synthetic metabolic pathways has shown promise in the production or overproduction of commercially useful biomolecules including isoprenoids (Pitera, Paddon et al. 2007) (Martin, Pitera et al. 2003)(Watts, Mijts et al. 2005), polyketides (Pfeifer, Admiraal et al. 2001)(Peiru, Menzella et al. 2005), non-ribosomal peptides (Watts, Mijts et al. 2005), bioplastics (Aldor, Keasling 2003) and polymer building blocks (Nakamura, Whited 2003). Current strain engineering practices however rely heavily on domain knowledge and laboratory experimentation. Bio design automation tools promise to facilitate such practices and to reduce experimental efforts.

When synthesizing new biochemical pathways, two issues must be addressed. The first is identifying a desirable pathway, i.e. set of enzyme-catalyzed reactions, from a large database such as KEGG, or MetaCyc. The second is ensuring that the pathway integrates with the host to achieve maximal yield while minimally affecting the growth and other essential functions of the host. We focus in this paper on the first problem, *pathway identification*.

A method for exploring the biochemical state space using a heuristic search based on minimizing the cost of transformation is presented in PathMiner (McShan, Rao et al. 2003). The resulting pathways are chemically parsimonious and do not favor pathways that involve the addition of large functional groups to compounds. Other methods to identify pathways are based on atom mapping, and they suffer from large computational times (Blum, Kohlbacher 2008)(Pitkänen, Jouhten et al. 2009) (Hatzimanikatis, Li et al. 2005). As another approach to the problem, Optstrain (Pharkya, Burgard et al. August 2004) builds an integrated computational framework for identifying stoichiometrically balanced pathways while maximizing product yield and evaluating different microbial hosts. This method requires the curation of a database, a non-trivial task, and uses reaction addition and deletion for path identification. The success of this method substantially depends on the curated database.

Unlike the aforementioned approaches, we utilize a graph-based search to identify potential synthetic pathways. First, we explore possible pathways to produce a given target metabolite, regardless of whether the resulting paths are viable. In the next step, the pathway is added to the host, and viability is verified by applying flux balance analysis (FBA). The objective of FBA is to maximize target product formation subject to the constraint that the modified host produces at least 80% of the wild type's biomass yield. Viable pathways are then ranked based on maximum product yield.

Methods

We investigated the use of graph-based search algorithms to identify synthetic pathways using a specific host, *E. coli*. Two approaches are developed. In both approaches, the number of reactions used in constructing the pathway is used as a cost function and bound to the maximum number of reactions that can be reasonably added to the host (Peiru, Menzella et al. 2005).

In the first approach, a tree is constructed recursively, starting from the target metabolite as the root of the tree. A reaction that contains the target metabolite as one of its products is added to the tree and represented by an edge. This edge stems from the root and links to child nodes representing the reaction's main and cofactor constituent metabolites. To control the multiplicity of the paths and preserve memory, a cost function technique is employed and only paths which meet the minimum cost requirement are retained during tree construction. A simple cost function is the number of added reactions along the path. FBA is then performed on all resulting paths to obtain the maximal yield and the reaction flux distribution associated with the production of the target metabolite. The candidate pathways are ranked by the maximal yield of the target metabolite.

In the second approach, the graph-based search algorithm focuses on metabolite connectivity within the KEGG database. Metabolite connectivity reflects the number of reactions in which that metabolite participates. Analysis of the KEGG database shows that metabolite connectivity follows a power law distribution similar to other evolved, scale-free networks (Barabasi 2009). In a scale-free network, high degree nodes with large connectivity act as connector hubs. Assuming A, B and C are 3 metabolites with A having a high node connectivity and B and C having low node connectivity, a path from B to C is likely to proceed through A rather than directly. Our search algorithm uses metabolite connectivity as a guide in the path construction. During tree construction, instead of exploring all reactions leading to a particular target (a node within the tree), an edge is added to the tree based on a weighted-random selection among all reactions that can lead to the particular target. The weighting is based on the metabolite connectivity. If the number of all reactions in a pathway exceeds the maximum number of reactions that can be genetically added to the host (e.g. 25 for *E. coli*), the algorithm backtracks along the tree edge. Another reaction is then selected. This process continues until a pathway that satisfies the bound condition is found. After the pathway is constructed, FBA is performed to calculate the yield. Because of its stochasticity, this algorithm is run many times to find the highly probable pathways.

Results

To evaluate our algorithms, we used a genome-scale model of *E. coli* metabolism (iAF1260), searching for the paths leading to more efficient production of isopentenyl diphosphate (IPP), a precursor for the isoprenoid class of natural products. Using the first approach, our algorithm finds 44 distinct pathways. FBA with maximizing IPP production as the objective and producing 80% of maximum biomass as a constraint identified 16 pathways, including a pathway involving mevalonate synthesis. The second approach identifies only a single pathway, involving mevalonate synthesis. The mevalonate pathway has been experimentally selected as the preferred route of IPP synthesis in *E. coli* (Pitera, Paddon et al. 2007).

Conclusion

We designed two algorithms to identify synthetic pathways which lead to potentially higher rates of production of a target metabolite. Our first algorithm finds all possible pathways leading to the target metabolite production in a higher rate than the wild-type host without ignoring any cofactors in the process. The limitations of this method are slow run times and large memory requirements due to combinatorial explosion. In our second approach, we perform a probabilistic search for pathways based on the degree of metabolite connectivity determined from the KEGG database. With both methods, we were able to reproduce experimentally obtained results reported in the literature. Our approach does not take into the consideration potential undesirable side effects when integrating with the host. We intend to address this issue in our future work.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. 0829899.

References

- ALDOR, I.S. and KEASLING, J.D., 2003. Process design for microbial plastic factories: metabolic engineering of polyhydroxyalkanoates. *Current opinion in biotechnology*, **14**(5), 475-483.
- BARABASI, A., 2009. Scale-Free Networks: A Decade and Beyond. *Science*, **325**(5939), 412-413.
- BLUM, T. and KOHLBACHER, O., 2008. MetaRoute: fast search for relevant metabolic routes for interactive network navigation and visualization. *Bioinformatics*, **24**(18), 2108-2109.
- HATZIMANIKATIS, V., LI, C., IONITA, J.A., HENRY, C.S., JANKOWSKI, M.D. and BROADBELT, L.J., 2005. Exploring the diversity of complex metabolic networks. *Bioinformatics*, **21**(8), 1603-1609.
- MARTIN, C.H., NIELSEN, D.R., SOLOMON, K.V. and PRATHER, K.L.J., 2009. Synthetic Metabolism: Engineering Biology at the Protein and Pathway Scales. *Chemistry & biology*, **16**(3), 277-286.
- MARTIN, V.J.J., PITERAL, D.J., WITHERS, S.T., NEWMAN, J.D. and KEASLING, J.D., 2003. Engineering a mevalonate pathway in Escherichia coli for production of terpenoids. *Nature biotechnology*, **21**(7), 796-802.
- MCSHAN, D.C., RAO, S. and SHAH, I., 2003. PathMiner: predicting metabolic pathways by heuristic search. *Bioinformatics*, **19**(13), 1692-1698.
- NAKAMURA, C.E. and WHITED, G.M., 2003. Metabolic engineering for the microbial production of 1,3-propanediol. *Current opinion in biotechnology*, **14**(5), 454-459.
- PEIRU, S., MENZELLA, H.G., RODRIGUEZ, E., CARNEY, J. and GRAMAJO, H., 2005. Production of the Potent Antibacterial Polyketide Erythromycin C in Escherichia coli. *Applied and Environmental Microbiology*, **71**(5), 2539-2547.
- PFEIFER, B.A., ADMIRAAL, S.J., GRAMAJO, H., CANE, D.E. and KHOSLA, C., 2001. Biosynthesis of complex polyketides in a metabolically engineered strain of E. coli. *Science*, **291**(5509), 1790.
- PHARKYA, P., BURGARD, A.P. and MARANAS, C.D., August 2004. OptStrain: A computational framework for redesign of microbial production systems.
- PITERA, D.J., PADDON, C.J., NEWMAN, J.D. and KEASLING, J.D., 2007. Balancing a heterologous mevalonate pathway for improved isoprenoid production in Escherichia coli. *Metabolic engineering*, **9**(2), 193-207.
- PITKÄNEN, E., JOUHTEN, P. and ROUSU, J., 2009. Inferring branching pathways in genome-scale metabolic networks. *BMC Systems Biology*, **3**(1), 103.
- WATTS, K.T., MIJTS, B.N. and SCHMIDT-DANNERT, C., JUN 2005. Current and Emerging Approaches for Natural Product Biosynthesis in Microbial Cells.

Poster Abstracts

Intracellular disease prevention and tuneable device behaviour in bacteria
Sangram Bagh, Mahuya Mandal, Jordan Ang, David McMillen

Synthetic biology includes an effort to control cellular behavior. One long-term goal is to implement medical interventions inside living cells, creating intracellular “disease fighters”; one may imagine a system to detect oncogenesis and respond by inducing apoptosis, or one that detects viral infection and responds to halt the spread of the virus. Although significant challenges lie ahead before human disease prevention is practical, progress in modulating cellular states and understanding the molecular underpinnings of viral and other diseases have been rapid, and it is timely to explore the features that any intracellular disease-prevention device requires. Such a device should: lie dormant in the absence of the disease state; detect the onset of a lethal disease pathway; respond to halt or mitigate the disease’s effects; and be subject to external deactivation when required. We have created a device that displays these properties, in the simplified case of a bacterial disease. Our system detects the onset of the lytic phase of bacteriophage lambda in *Escherichia coli*, responds by preventing this lethal pathway from being followed, and is deactivated by a temperature shift. By providing a practical demonstration of intracellular disease prevention, our system has implications in programmed therapeutics and cellular engineering.

We will also present recent results on the construction of a simple genetic device whose behaviour can be tuned to display a variety of input-output curves and to implement a logical AND response to chemical input signals.

Modeling Swarms of Micro-robots for Biological Applications

Paul Bogdan and Radu Marculescu
 Carnegie Mellon University
 {pbogdan, radum}@ece.cmu.edu

Extended Abstract

Key Words: Systems biology, stochastic behavior, random walks, micro-robots, swarms.

One of the main challenges of modern medicine is the silent progression and migration of various diseases through the human body. For instance, cancer is the second leading cause of death in United States because in most situations tumors appear and develop undetected by many of the current screening tests or the immune system itself. Nevertheless, a large body of research invested over the last 50 years in the analysis of tumor angiogenesis suggests that tumors silent progression is most of the time accompanied by an increased demand of nutrients and oxygen [1][2]. Most of the time these events are not easily detected by the immune system and the cancer cells can corrupt the neighboring and remote organs up to the point where even surgery cannot guarantee a cure.

Despite these challenges, we argue that we can turn our attention to mother nature to find solutions that complement the traditional approaches for diagnosis and treatment. More precisely, a possible approach can rely on exploiting the affinity of certain bacteria to swim towards locations with increased consumption of nutrients and oxygen representing tumor angiogenesis hotspots. Eventually, the motility of such bacteria can be harnessed to engineer micro-robotic swarms capable of monitoring, detection and targeted drug delivery within the human body [3]. Although, fabrication of single micro-robots is well underway [3], characterizing the *collective dynamics* of swarms of such micro-robots is not an easy task mainly because there is no computational framework proposed to date for modeling, analyzing and designing such swarms for specific applications like targeted diagnostic and drug delivery.

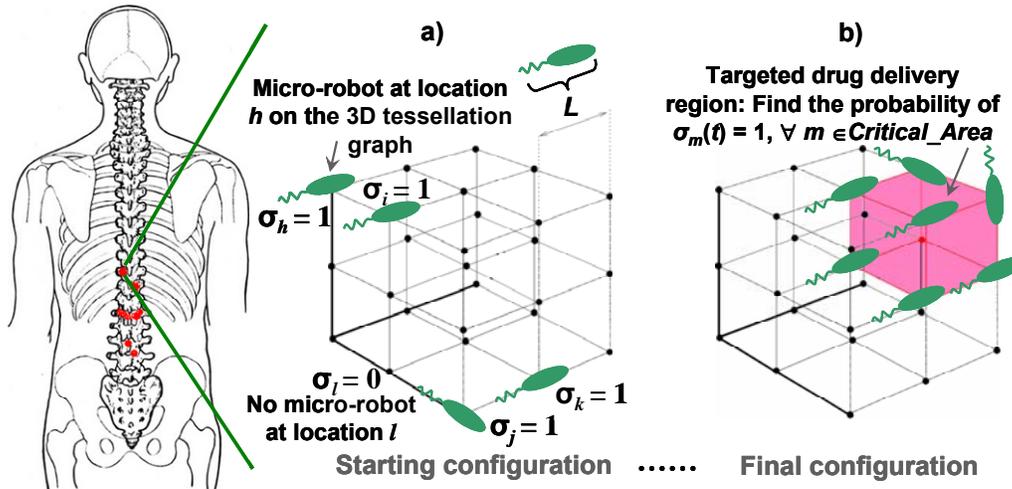


Fig. 1. Detection, monitoring and drug delivery problem: Dense network of micro-robotic swarms are released in hard to access regions of the human body. The micro-robots dynamics is modeled as the collective behavior of multiple interacting random walks in a 3D graph tessellation of space. Each node in the graph has associated a binary random variable (σ) representing whether or not this location is occupied by at least one micro-robot. For detection and health monitoring purposes the goal is to find the time-dependent coverage of the swarm. The goal of the drug delivery problem is to find the time-dependent probability for the nodes in the disease area (see Fig.1(b)) to be occupied by micro-robots.

Towards this end, we propose a mathematical model for capturing the dynamics of a large number (or teams) of self-driven micro-robots (*i.e.*, bacteria propelled capsules) able to swim and access infinitely small spaces inside human body; this voyage takes place in a completely non-invasive manner due to their dimensions at micro- or nano-scale of such devices. Such engineered micro- or nano-robots are capable of performing massively parallel and distributed tasks which may be used for diagnostic and drug delivery purposes. Based on the attractiveness of chemotactic bacteria (*e.g.*, *Serratia Marcescens*) to increased oxygen consumption around tumor locations, we envision the possibility of constructing a dense network of such micro-robots which can sense and move through the interstitial spaces of the human body due to bacteria swimming mechanisms. For instance, the micro-robotic swarms can be deployed in the spinal cord as shown in Fig. 1 (a) and let to freely swim and sense the environment in order

to detect potential cancer risks. To model their dynamics in space and time, we tessellate the 3D space into a graph as shown in Fig. 1 (a) and estimate the probability for a micro-robot to visit a particular node in the human body.

In order to advance the state-of-the-art and enable active networks of micro-robotic swarms that are more effective in combating various critical diseases with minimal impact on the human body, we develop a statistical physics approach aimed at characterizing the dynamics of *multiple interacting random walks* in a 3D space. In other words, we map the movement of T micro-robots swarming within a confined 3D region to the problem of tracking the trajectories of T simultaneous random walks which can interact at various points in time and space. Getting into more specifics, we study the evolution of T random walks contained within a certain region via a master equation which determines the probability $P(\sigma_1, \dots, \sigma_M; t)$ of having a random walker (*i.e.*, micro-robot) at a certain location on a lattice graph (*i.e.*, which represents a tessellation of a certain 3D space). As can be seen from Fig. 1 (a), the binary random variable σ_j represents whether or not there exists a micro-robot at location j (*i.e.*, $\sigma_j=1$ if the micro-robot is present and $\sigma_j=0$ if not). For detection and monitoring purposes, our focus is on determining the coverage and frequency of visiting certain nodes in the 3D space by at least one random walker. In contrast, for solving the drug delivery problem, the goal is to have as many micro-robots as possible reaching a critical area as shown in Fig. 1 (b) (see target region highlighted in red) which is equivalent to having $\sigma_m=1, \forall m \in \text{Critical_Area}$. Note that classical diffusion theory *cannot* be applied to such a scenario since various hydro-dynamical and chemical interactions are crucial to be considered in order to capture the true dynamics of the T random walks. Our proposed statistical physics approach is meant to capture the collective and competitive behavior of particles and predict the evolution of the swarm as a function of the density of walkers and the strength of their interactions. More precisely, we try to estimate the number of nodes in the graph visited by at least one random walker by time t , the frequency of visits and the hitting time for a collection of random walkers to reach a targeted region.

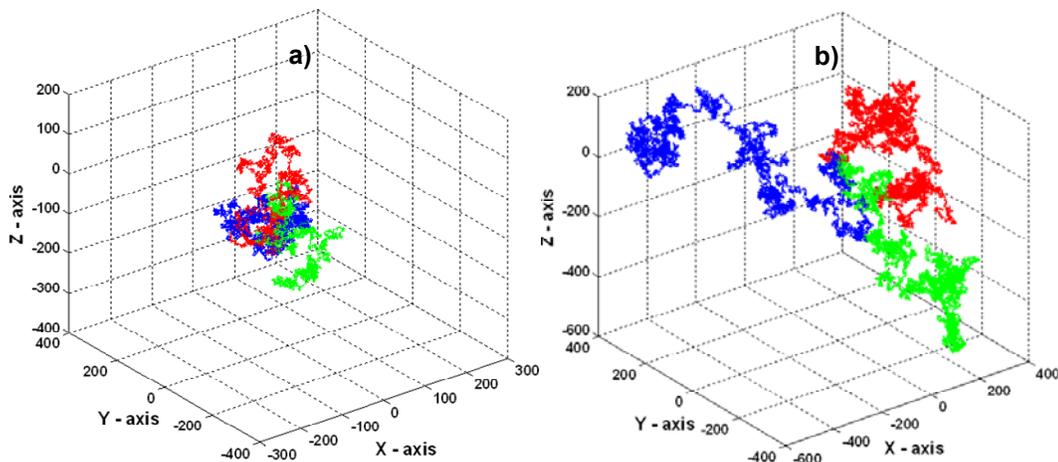


Fig. 2. Preliminary modeling results: (a) Trajectories of three micro-robots with a random walk behavior without interaction and (b) with repellent interaction. The interaction is defined as the change in the direction of motion of a micro-robot when two of them intersect their trajectories.

To stress the importance of considering the interactions among multiple random walks, we simulate the 3D trajectories of three random walkers without interaction (in Fig. 2(a)) and *with* interaction (in Fig. 2(b)) for the case of having a 0.5 probability of stepping to a neighboring location at each iteration. In these experiments, the simulation length consists of 50000 iterations. To model the hydrodynamic interactions in the case of interacting random walks, we consider that when two random walks meet, they repel each other and change directions. As shown in Fig. 2(b), the maximum distance traveled by the interacting random walks is approximately 500 distance units, while for the case of non-interacting random walkers it is approximately 140 distance units. The difference in distance between the two types of random walks can be significant when considering more advanced chemical and hydro-dynamical interactions among bacteria or micro-robots. Accurate modeling of trajectories and distances travelled by teams of such micro-robots is of crucial importance for solving the diagnostic and drug delivery problems in biological systems.

References

- [1] D. Hanahan and R.A. Weinberg, "The Hallmarks of Cancer," Cell, Vol. 100, Issue 1, pp. 57-70, January 2000.
- [2] Y. Lazebnik, What are the hallmarks of cancer? Nature Reviews Cancer, No.4, pp. 232-233, March 2010.
- [3] M. Sitti, "Miniature Devices: Voyage of the microrobots," Nature, vol. 458, pp. 1121-1122, 30 April 2009.

A Genetic Programming Framework for the Simulation and Design of Self-assembling, Chemotaxis-driven Cell Aggregates

Dr. David Breen

Ms. Linge Bai

Department of Computer Science, Drexel University, 3141 Chestnut Street, Philadelphia, PA 19104
{david,lb353}@cs.drexel.edu 215-895-1626

We have conducted a number of research projects related to the simulation of 2D chemotaxis-based cell aggregation and its application to cell-biology-inspired spatial self-organization algorithms. This work began with the development of a software system that is capable of simulating the chemotaxis-based cell aggregation behaviors of PC12 cells [1,2]. It was extended to model an initially mixed two-cell-type population and the local interactions that lead to the formation of a sorted, two-layered structure [3,4]. The cell aggregation simulation system provided a foundation for an approach to self-organizing geometric primitives that implements cell-based automated shape composition [5,6,7]. We now intend to extend and apply our approach for spatial self-organization to the problem of designing chemotaxis-driven, self-assembling cell aggregates. The approach provides a genetic programming framework for designing the chemotactic properties/behaviors of individual living cells that, when allowed to move via chemotaxis, will form into cell aggregates having a user-specified shape. Employing the framework will identify the cell behaviors, i.e. chemical emission, chemotactic response, motility and adhesivity, that are needed to produce living cells that aggregate into specific shapes.

Our approach for automated shape composition provides the overall paradigm for simulating and designing self-assembling cell aggregates. The automated shape composition process begins by randomly placing virtual cells in a 2D environment. The virtual cells interact with each other via chemotaxis. This interaction produces movement that leads them to aggregate into a single user-specified shape. We call these self-organizing virtual cells *Morphogenetic Primitives* (MPs). Each MP is represented by a small disk existing in a toroidal, i.e. periodic, 2D computational environment. Each MP emits a “chemical” into the environment. An MP detects the cumulative field at eight receptors on its surface, and calculates the field gradient from this input. MPs move in the direction of the field gradient with a speed proportional to the magnitude of the gradient.

While MPs' fundamental interaction is based on a chemotaxis paradigm, for now we do not limit their behaviors / properties to be physically realistic or completely consistent with biology. Instead, developmental biology provides a motivating starting point for MPs. In order to customize the interactions of MPs for shape composition, we alter the chemical concentration fields around individual cells. Instead of the chemical concentration dropping off as a function of distance r ($1/r$ is the physically accurate description), we define the concentration field with a mathematical function of cell-cell distance d , one cell's angular location θ in another cell's local coordinate system and simulation time t . Since at this time there is no prescriptive way to specify a particular local field function that will direct MPs to form a specific macroscopic shape, we employ genetic programming [8] to produce the mathematical expression that explicitly specifies the field function. The fitness measure associated with each individual field function is based on the shape that emerges from the chemical-field-driven aggregation simulation, and determines which functions will be passed along to later generations. The genetic process stops once an individual (i.e. a mathematical expression) in the population provides the desired shape, or after a certain number of generations have been produced and evaluated.

The genetic programming framework that produces local interactions leading to automated shape composition consists of three major components and is presented in Figure 1(a). The components are: a genetic programming (GP) engine (the Open Beagle System [9]), a cell aggregation simulation system [1,2] and a shape comparison module. The local interaction rules that direct MPs to aggregate into shapes and patterns are explicitly represented as a “chemical” field function. In the context of GP, the individuals to be evolved are these field functions. We start with a population of functions, which is initially randomly generated. Each function is compiled into a chemotaxis-based cell aggregation simulation program, and defines the chemical field that surrounds all of the individual cells for a particular aggregation simulation. A simulation program that uses each field function is executed on a node in our cluster, usually producing some kind of aggregated structure. The resulting MP aggregate is compared to the user-desired shape, and a scalar fitness value is calculated that quantifies how well the computed shape matches the desired shape. A subset of the top candidates are then used to create the next generation of field functions via mutation and cross-over (sexual reproduction). The process continues until a field function produces the desired shape or the maximum number of generations is reached. Figure 1(b) contains a gear-like shape produced from the shape composition approach. Several steps of the computational aggregation are included.

The framework will be modified and extended in two ways in order to design living cells that self-assemble into user-defined shapes. The “modified” cell aggregation simulator will be replaced by one that accurately models cell properties and behaviors, rather than one that utilizes artificial chemical field functions. The internal workings of the simulator will be implemented with a data flow language, a type of programming language that is easily merged, mutated and manipulated. Secondly, motivated by the work of Sims [10], the GP engine will be modified to evolve the dataflow structure of the simulator, rather than just the chemical field functions of individual cells. This will change the internal program run for the cells during simulation and the general behavior of the virtual cells. The challenge here is to evolve the actions and behaviors of the simulated cells while ensuring that the resulting simulations remain biologically accurate. Given that these goals can be achieved, the framework will identify the properties of real cells that lead to the aggregation of specific shapes.

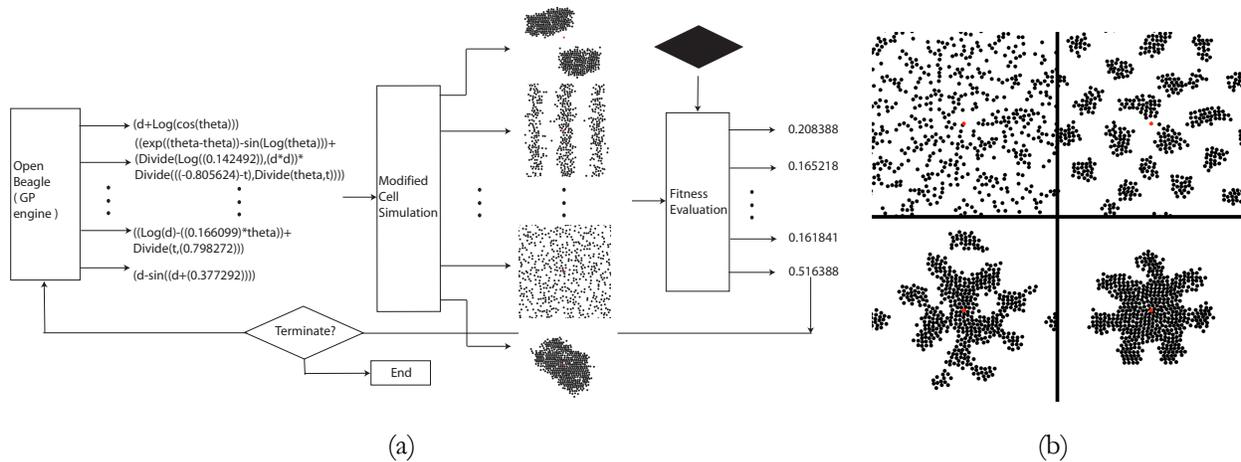


Figure 1. (a) Genetic programming framework for cell-based automated shape composition. (b) Morphogenetic primitives self-organizing into a gear-like shape.

References.

- [1] M. Eyiurekli, P. Lelkes and D. Breen, “A Computational System for Investigating Chemotaxis-Based Cell Aggregation,” *Proc. 9th European Conference on Artificial Life*, Sept. 2007, pp. 1034-1049.
- [2] M. Eyiurekli, P. Manley, P. Lelkes and D. Breen, “A Computational Model of Chemotaxis-based Cell Aggregation,” *BioSystems*, Vol. 93, No. 3, pp. 226-239, Sept. 2008.
- [3] M. Eyiurekli, P. Lelkes and D. Breen, “Simulation of Chemotaxis-Based Sorting of Heterotypic Cell Populations,” *Proc. IEEE / NIH BISTI Life Science Systems & Applications Workshop*, Nov. 2007, pp. 47-50.
- [4] M. Eyiurekli, L. Bai, P. Lelkes and D. Breen, “Chemotaxis-based Sorting of Self-Organizing Heterotypic Agents,” *Proc. 25th ACM Symposium on Applied Computing*, March 2010, pp. 1315-1322.
- [5] L. Bai, M. Eyiurekli and D. Breen, “Self-Organizing Primitives for Automated Shape Composition,” *Proc. International Conference on Shape Modeling and Applications*, June 2008, pp. 147-154.
- [6] L. Bai, M. Eyiurekli and D. Breen, “Automated Shape Composition Based on Cell Biology and Distributed Genetic Programming,” *Proc. Genetic and Evolutionary Computation Conference*, July 2008, pp. 1179-1186.
- [7] L. Bai, M. Eyiurekli and D. Breen, “An Emergent System for Self-Aligning and Self-Organizing Shape Primitives,” *Proc. 2nd IEEE Int. Conference on Self-Adaptive and Self-Organizing Systems*, Oct. 2008, pp. 445-454.
- [8] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.
- [9] C. Gagné and M. Parizeau, “Genericity in evolutionary computation software tools: Principles and case-study,” *International Journal on Artificial Intelligence Tools*, vol. 15, no. 2, pp. 173–194, 2006.
- [10] K. Sims, “Evolving virtual creatures,” *Proc. SIGGRAPH*, 1994, pp. 15–22.

Data Model Approaches for Design, Assembly and Validation in Synthetic Biology

Kevin Clancy
Genomics Technologies R&D
Life Technologies
Carlsbad, California

Abstract

The application of synthetic biology techniques to the reengineering or novel creation of parts, devices and systems needs a high level of skill, knowledge, experimental competence and time to produce working components. As the field progresses from an artisan culture to that of an engineering discipline, software and associated systems will need to emulate and learn from much of the current experience and lessons taught at the bench.

An important aspect of design is the support for creating assemblies using multiple experimental strategies. The current parts/device/assembly approach was based on restriction enzyme-based approaches and requires a certain level of engineering of biological sequences to fit in with defined assembly strategies, resulting in limitations for simultaneous assembly and in the introduction of scars within assembled sequences. However, as the cost of oligonucleotides drops and there is software support for other assembly methods, it becomes important to support developing assemblies with other types of approaches. A second important part of the design of parts, devices and assemblies is the inclusion of validation and verification methodologies to simulate expected behavior and assist with troubleshooting problems with the design. Validation and verification tests can occur both at the level of software and experimental design and appropriate design of such experiments is linked to the method of assembly as much as the inherent nature of the parts used.

We present an alternative data model that explicitly takes into account novel experimental strategies for parts assembly. We show how this data model can be used to incorporate data on the types of validation and verification strategies that can be used for a given assembly methodology. As an example, we use this data model to show how to implement a new cloning methodology that will incorporate oligo based assembly of multiple blocks of DNA, thus allowing users to use or move beyond conventional restriction enzyme-based cloning. We demonstrate how validation and verification strategies can be developed within the data model to support and optimize different types of assembly technologies. The ultimate intention of this data model is to provide a means of bettering current design practices through guided experimental planning.

Scalable open source software framework for laboratory automation and laboratory devices
Jonathan Cline

Biology laboratories contain computerized equipment as independent units controlled via front-panel user interfaces or via computer software/graphical applications. A single biology experiment may employ a combination of equipment (pumps, valves, robotic arms, shakers, liquid handlers, thermocyclers, detectors, robotic systems), many of which require programming independently, with differing programming limitations. A scalable software framework has been developed which provides a unified approach to controlling varied biology laboratory equipment and robotics systems over an internet network. The software provides engineers with plug-in Perl modules for adding new device command-compilers and probing for connected devices, while providing the laboratory user the ability to write higher-level programming abstracted from individual device commands. The placement of this software package in a large open source repository (CPAN, as Perl Robotics) invites wide scale development to support a range of new and legacy laboratory devices. Current hardware support includes Tecan Genesis/EVO, FIALab MicroSIA system, and custom USB peripherals.

Towards Integrative CellML Modeling Technologies for Intracellular Research

Cooling M. T.¹(m.cooling@auckland.ac.nz), Matos E. E.², Zhou C.¹, Tao G.¹, Nielsen P. F.¹

1. Auckland Bioengineering Institute, The University of Auckland, New Zealand
2. Organizational Knowledge Management Center, Federal University of Juiz de Fora, Juiz de Fora, Minas Gerais, Brazil

Modularity is crucial for the *in silico* design and testing of biological systems [1]. Recently, we developed an online library of modular mathematical model components for synthetic biology [2] using the modular model exchange format CellML [3]. In addition to synthetic biology, where new biological constructs are being created, this library is now being extended for general intracellular modeling in the biomedical context [4,5] as researchers seek to understand the wealth of systems already existing in the natural world.

To deliver the most benefit to researchers from the expanding repository contents, we seek to provide module classification, searching, composition, and visualization services to enable integrative modeling and analysis. These integrative technologies will be supported by annotation schemes adding biological and model structure semantics. Two prototype ontologies have already been developed, one supporting electrophysiological model composition [6], and the other supporting model visualization [7,8]. We are extending and integrating these ontologies with international standards [9,10,11] where appropriate, and have recently extended the visualization ontology to cater for synthetic biology modules from our new Repository (see Fig 1.).

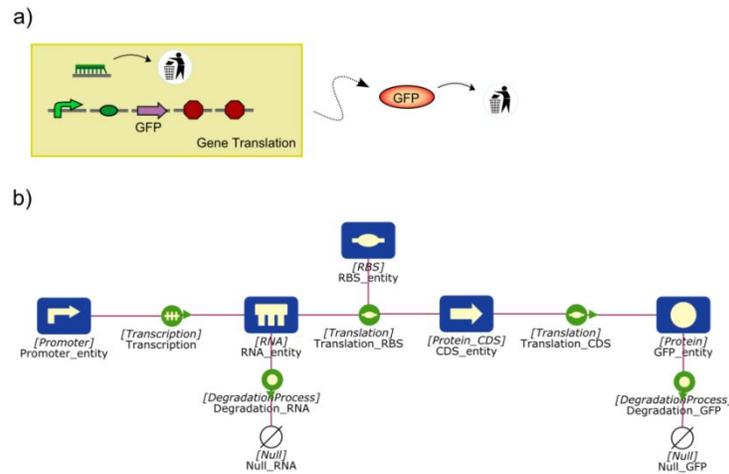


Fig 1: a) A schematic of a GFP producing system, which encompasses translation, transcript and RNA and product degradation. **b)** A computer-assisted visualization using standardized symbols of the same system, implemented as an annotated model composed from modular CellML components.

Once annotated, tools will be required to deliver core integrative functionality. Our provisional architecture for providing core services is shown in Fig 2.

Modular, annotated model components will be stored in the CellML Repository [12], accessible either via web interfaces [13] or through client-side applications. Search, composition, and visualization functionality will be implemented by extensions to the CellML API [14], supporting the augmented construction of intracellular models covering signal transduction, gene regulation, metabolism, and electrophysiological processes. Analysis services for these models will include simulation, sensitivity analysis, and parameter fitting. Due to the computation-intensive

nature of these functions, distributed parallel computing technology will be employed, with a prototype service currently under development with 'BestGRID' [15] in New Zealand and 'Nimrod' [16] in Australia.

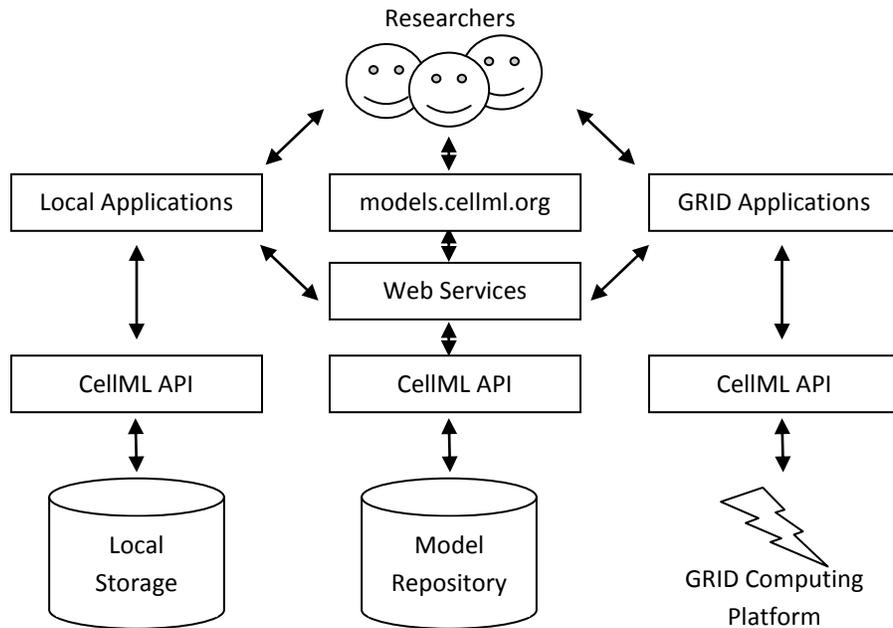


Fig 2: The proposed architecture for integrative model services. The CellML API supports the development of services which can be executed on local machines, online via web services, and/or in GRID architectures.

The development of this technology will make the power of modular CellML available to all researchers, and simultaneously make it possible to compose models encompassing an increased proportion of cellular processes. This will lead to more realistic investigation of emergent functions and greatly facilitate the pursuit of solutions to scientific questions within the complex intracellular environment.

References

- [1] Knight TF (2005) Engineering novel life, *Molecular Systems Biology*, doi: 10.1038/msb4100028.
- [2] Cooling MT et al. (2010) *Bioinformatics*, doi: 10.1093/bioinformatics/btq063.
- [3] Cueller AA et al. (2003) *Simulation*, 79, 740-747.
- [4] Ho H et al. (2010) 6th World Congress on Biomechanics, *submitted*.
- [5] Cooling MT and Hunter PJ (2010) 5th international symposium on Biomechanics in Vascular Biology and Cardiovascular Disease.
- [6] Matos EE et al. (2010) *Journal of Biomedical Informatics*, 43, 125-136.
- [7] Wimalaratne SM et al. (2009) *Bioinformatics*, 25, 2263-2270.
- [8] Wimalaratne SM et al. (2009) *Bioinformatics*, 25, 3012-3019.
- [9] Ashburner M et al. (2000) *Nature Genetics*, 25, 25-29.
- [10] Le Novere N (2006) *BMC Neuroscience*, 7, S11.
- [11] Le Novere N et al. (2005) *Nature Biotechnology*, 23, 1509-1515.
- [12] Lloyd CM et al. (2008) *Bioinformatics*, 24, 2122-2123.
- [13] <http://models.cellml.org>
- [14] <http://www.cellml.org/tools/api>
- [15] <https://www.bestgrid.org/>
- [16] Abramson D et al. (1994) *Parallel Computing and Transputers Conference*, Wollongong, 17-27.

Towards Distributed Web of Registries: Design, Implementation and Practice of the JBEI Registry

Timothy Ham, Zinovii Dmytriv, Nathan Hillson, Jay Keasling

The tremendous growth of Synthetic Biology and the increasing number of new software tools have highlighted the need for a robust, freely available and distributed parts database software (a registry of parts). Even as the number of standard biological parts have grown, the way the parts are stored and managed have not advanced at the same pace. Several automation tools now exist, but they are hampered by lack of a registry that can be used programmatically to pull from and push parts into. Additionally, it is desirable for any design software to keep track of local intermediates as they are created and different designs are tried and discarded, without polluting a public registry. An open source, distributed registry that anyone can use via on-line or directly as a software library stack would solve many of these issues and aid the advancement of other automation tools by unifying and simplifying how parts are stored and managed.

The Joint BioEnergy Institute (JBEI) Registry team aims to provide the community with an open source implementation of a Registry of Biological Parts. The JBEI Registry (JBEIR) is a software platform and a web application that provides tools to organize, categorize, search and manage biological parts in a web accessible way. It is designed from the ground up to bridge the gap between legacy biological constructs and the new BioBricks paradigm by transparent support between plasmids, strains, and "parts". Additionally, JBEIR has been designed for a distributed installation, so that anyone can run their own registries, and yet provide mechanisms for simple information exchange and data synchronization. It is built upon open source software, and licensed in the most liberal way to encourage adoption and participation. Or users can simply start using it from our web site.

With the new version 2 release, we provide a new synthetic biology inventory platform (ICE) as well as a new graphical sequence annotator (VectorEditor). ICE (Inventory of Composable Elements) is the software stack that manages the parts, with a friendly web user interface. Version 2 now has per user workspaces and permission control on parts. Additionally, it provides a SOAP interface for web services as well as Java API service layer. As ICE implements a clean service layer, it is possible to use it's API as a data abstraction layer to build other automation tools. This means that ICE can be used as a library to provide registry functions without running the web interface. VectorEditor is a cross platform sequence viewer and annotator that works within a web browser on multiple computing platforms. These tools are released under the BSD license for anyone to use, modify and extend.

JBEI ICE and VectorEditor has been chosen by the newly formed International Open Facility Advancing Biotechnology (BIOFAB) as their registry and platform to advance techniques and tools for synthetic biology. The open, modular, and extensible architecture we have developed represents a starting point for many useful bio-design automation software.

Towards Automated-assembly of Biological Parts

N.J. Hillson, J.W. Thorne, D. Densmore, M.Z. Hadi, and J.D. Keasling
Fuels Synthesis and Technology Divisions, Joint BioEnergy Institute

The production of clean renewable biofuels from cellulosic starting material requires concerted feedstock engineering, deconstruction of plant matter into simple sugars, and microbial fermentation of the sugars into biofuel. These three efforts share significant molecular biological challenges, including the construction of large enzymatic libraries (e.g. vast collections of glycosyl transferases, cellulases, and efflux pumps), the generation of combinatorial libraries (e.g. multi-functional enzyme domain fusions; variations in copy number, promoter and ribosomal binding site strength), and the concurrent assembly of multiple biological parts (e.g. the incorporation of an entire metabolic pathway into a single target vector). With these challenges in mind, we are developing hybrid multi-part assembly methodologies and translating them to robotics-driven protocols. Given a target library to construct, our vision is that the high-throughput methodology will provide automated oligo and optimal assembly process design, and robotic control of the PCR and multi-part assembly reactions. The beneficial output of this work will include reagents and resources for, and collaborations with, members of the larger life sciences communities, reducing the time, effort and cost of large scale cloning and assembly tasks, as well as enabling research scales otherwise not feasible without the assistance of computer-aided design tools and robotics.

DIGITAL SIGNAL PROCESSING WITH BIOMOLECULAR REACTIONS

Hua Jiang, Marc D. Riedel, and Keshab K. Parhi

Department of Electrical and Computer Engineering
University of Minnesota
Email: {hua, mriedel, parhi}@umn.edu

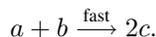
1 Summary

We present a design methodology for performing digital signal processing (DSP) operations such as filtering with biomolecular reactions. From a DSP specification, we demonstrate how to synthesize reactions that produce time-varying output quantities of molecules as a function of time-varying input quantities. We synchronize the computation with a three-phase “handshaking” protocol that transfers quantities between molecular types based on the absence of other types. Our method produces computation that is accurate and independent of the reaction rates, assuming only that some reactions are faster than others. The scheme is efficient: both the number of molecular types and the number of reactions grow linearly with the size of the DSP specification. We illustrate our method with the design of a moving-average filter. The method is generally applicable to the design of finite-impulse response (FIR) and infinite-impulse (IIR) filters. These can perform a variety of useful operations such as high, low and band-pass filtering.

2 Background

Interesting biochemistry typically involves complex molecules such as proteins and enzymes. Within the confines of a cell, the *quantities* of such molecules are often surprisingly small: on the order of tens, hundreds, or thousands of molecules of each type. At this scale, individual reactions matter and the problem must be modeled discretely.

In our view of biomolecular computation, the quantities of molecules are whole numbers (i.e., non-negative integers). Reactions fire and alter these quantities by discrete, integer amounts. Consider the reaction



When this reaction fires, one molecule of a is consumed, one of b is consumed, and two of c are produced. (Accordingly, a and b are called the *reactants* and c the *product*.) Each reaction has an associated *rate* (listed above the arrow

in our notation). Given several reactions, the probability of each firing is proportional both to its rate and to the quantities of its reactants present. Although we refer to rates in relative and qualitative terms, e.g., “fast” vs. “slow,” these are, in fact, quantitative values that are either deduced from biochemical principles or measured experimentally.

3 Objectives

A typical signal processing operation produces an output signal by *filtering* or *transforming* an input signal: examples are smoothing a signal with a moving-average filter and performing a Fast Fourier Transform (FFT). Digital signal processing with integrated circuits for applications such as audio and video is a mature, sophisticated domain. We aim to apply and extend such concepts to the new domain of biomolecular computation.

We will examine the abstraction of signal processing from a design perspective: how can we synthesize biomolecular reactions that produce specific output quantities of molecules as a function of input quantities, performing filtering operations? The challenge with biomolecular computation is that the reactions fire asynchronously at variable rates, dependent on factors such as temperature. In spite of this, we aim to implement *computation* that does not depend on the rates.

Although conceptual for the time being, our method has potential applications in domains of synthetic biology such as biochemical sensing and drug delivery. We are exploring DNA-based computation via strand displacement as a possible experimental chassis.

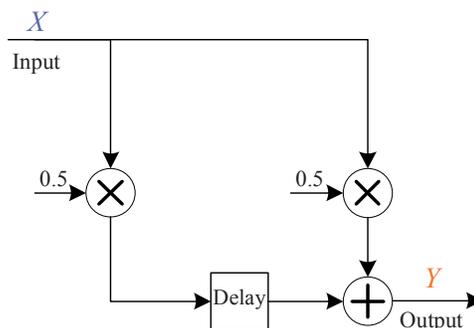


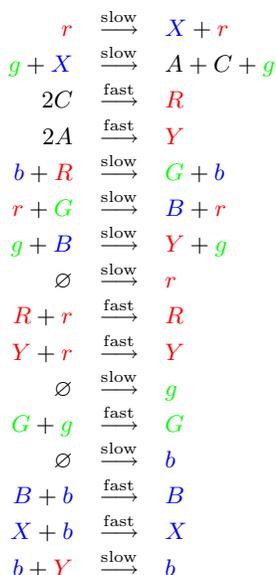
Fig. 1. Moving average filter.

This work is supported by NSF EAGER Grant CCF0946601 and by the Biomedical Informatics and Computational Biology program at the University of Minnesota.

4 Method

DSP systems are generally specified in terms of four basic modules: *splitting*, *scalar multiplication*, *addition*, and *delay elements*. An example of such a specification is shown in Figure 1. The input signal is split; both signals are multiplied by one-half; one signal is delayed; and the two signals are added to form the output. This system computes a *moving average*: given a time-varying input signal X , the output signal Y is a smoother version of it. More precisely, the output is one-half the current input value plus one-half the previous value.

In the context of biomolecular computation, the signals are *quantities* of molecular types. We implement a biomolecular moving-average filter with the following reactions. These molecular types are labeled in Figure 2.



Quantities are transferred between categories of molecular types, coded by three colors: red, green and blue. Quantities are transferred between two types in the absence of the third type: red goes to green in the absence of blue; green goes to blue in the absence of red; and blue goes to red in the absence of green. To synchronize on the absence, we continually generate types r , g , and b . (The symbol \emptyset indicates “no reactants” meaning the products are generated from a large or replenishable source.) These types only persist in the absence of the corresponding color-coded signals: r in the absence of R and Y ; g in the absence of G ; and b in the absence of X and B . Note that the input X is sampled in the blue phase. The output Y is produced in the red phase.

Splitting is implemented by choosing a reaction producing several different product types: X goes to both A and C . Scalar multiplication by 0.5 is implemented by stoichiometry: two molecules of C go to one of R and two of A go to one of Y . Addition is implemented by choosing several reactions producing the same product type: both A and B go to Y .

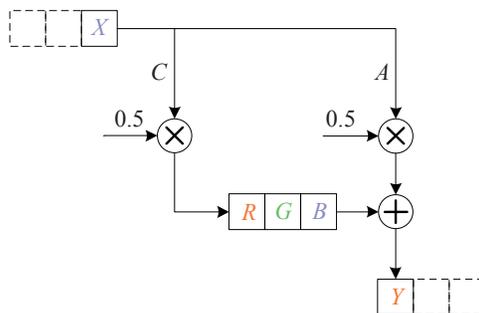


Fig. 2. Moving average filter with a three-compartment delay.

The reactions implement the requisite computation: the output quantity of Y produced in the red phase is one-half the current value and one-half the previous value of the input quantity of X sampled in blue phases. The computation require only two rate categories: “fast” and “slow”. Given reaction rates that broadly fit this categorization, the computation is exact; the moving-average function does not depend on the specific reaction rates.¹

5 Simulation Results

We used Gillespie’s stochastic simulation algorithm (SSA) to simulate these reactions. We used a rate of 1 for “slow” and a rate of 1000 for “fast”. We plot the results in Figure 3. The figure shows the quantity of the input and output – types X and Y respectively – as a function of “sampling phases”. The time for each phase is set by the reactions; it is generally several magnitudes longer than it takes for slow reactions to fire to completion. We set the input quantity to new random values at the beginning of each sampling phase. The figure shows the theoretical output, i.e., an exact calculation of a moving average of the input, in green. It shows the simulated output in red. We see that the output is a smoother version of the input. The simulated output has nearly perfect agreement with the theoretical output.

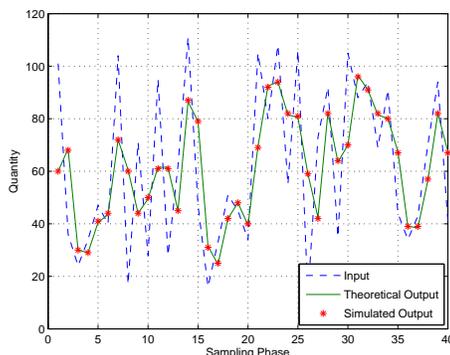


Fig. 3. Simulation results for the moving average filter (1000 trajectories in SSA).

¹The assumption that we make about the reactions is that those in the “fast” rate category are much faster than those in the “slow” rate category; if a “fast” reaction can fire, it does so repeatedly, until its reactants are consumed and it can fire no more – this before a “slow” reaction fires at all.

Bio-Specific HW/SW Co-Design for Multicore Systems on Chip

Iyad Al Khatib (iyadek@kth.se)

Royal Institute of Technology (KTH), ICT, ECS, Stockholm, Sweden

Bio-system-level Design, Synthesis and Optimization

Submitted to both: Poster and Oral presentations

I. ABSTRACT

Exploiting chip technologies in the nanometer scale paves the way towards adding more and more processing-cores on the same chip. Multi-core processing increases the computational power for systems-on-chips (SoC), thus leveraging chip abilities to run more complex and data-intense applications [1]. Many biological and medical applications require high computational power, and the use of Multi-Processor SoC (MPSoC) technologies for biochips can offer a solution. However a gap still exists between the biological field and the Information Technology field since there is a lack of methodologies bridging between the two. The success of such multicore biochips depends not only on reliable chip fabrication but also on mapping relevant applications (with a market demand) to these chips. Thus, it is not the hardware (HW) alone nor the software (SW) alone that will increase the possibilities of success, but it is the reliable combination of a HW, SW, and biological algorithms that function together to serve other disciplines (e.g. synthetic biology, medicine, biomedical engineering, etc), whose application areas are largely demanded. The most relevant applications are those requiring high computational capabilities while still respecting critical requirements in time and power consumption. Current trends in MPSoC must deal with HW/SW Codesign methodology since it has a significant effect on the efficiency of the final system as well as the time to design, develop, test, and produce. When designing both the HW and SW for a bio-related application, the biological (application) layer imposes many parameters. Therefore, a design flow is important in such a multidisciplinary work in order to ensure that system performance meets expectations.

We investigate several aspects of a HW/SW codesign methodology as well as design approaches that can meet the requirements of three different (but goal-related) disciplines, namely: bio-related applications, SW, and HW (See Figure 1). In using this methodology we are able to propose several application-aware and realtime approaches that optimize the MPSoC design for rendering high SW computations within the needed components neither more nor less depending on the application-level requirement). Our treatment of the first design shows that there are optimal numbers of resources for each application requirement, which- in turn- minimize costs and better performance. Our experimental results demonstrate that different accuracy

demands from the biological-layer may result in communication limited architecture rather than computation limited architecture. Hence, the methodology aids in investigating the choice of onchip communications, busses, and general architecture [2]. The intention is to use a methodology that looks at the biological application, SW design, and HW design at the same time in order to evolve to new systems on chip that may make a difference. Technically, we aim at rendering more computations in less time, on a biochip with smaller size, and with less expense. The performance demand and the vision of having a market success, i.e. contributing to lower costs, pose many challenges on the HW/SW codesign to meet these goals. This calls upon the development of new integrated circuits for biological applications featuring increased energy efficiency while providing higher computation capabilities, i.e. better performance. One of the applications we work on is the realtime 12-lead ECG analyses (using 12 input biological signals at high data rates), which is an ideal target for a bio-specific MPSoC implementation. In the first iteration of the design, we need to develop flexible SW algorithms that can fit within the dynamic design flow. In this respect, we explore the bio-specific design space by analyzing different HW and SW architectures. As a result, we realize a design with twelve processors that can compute 3.5 million arithmetic computations and respect the real time hard deadline for our application (3.5-4 secs), and that can deploy the bio-specific algorithms. Then, we investigate the configuration space looking for the most effective solution, performance/energy-wise. Consequently, we present three interconnect architectures (Single Bus, Full Crossbar, and Partial Crossbar) and compare them with existing solutions. Contrary to other domains (e.g. multimedia and entertainment) we need accurate system component models already in the early design stages to limit the degrees of execution unpredictability to the minimum. Therefore, we need cycle-accuracy [3] in the design, modelling, and simulation. We demonstrate a practical case study for our application-specific MPSoC HW/SW codesign flow, where we exploit industrial IP cores of STMicroelectronics DSPs (ST220 and STBus) [1], and we look at the challenging 12-lead ECG application, with different input frequencies in the KHz ranges. We create robust algorithms that can fit the SW-level parallelization, HW level bottlenecks (computation and communication), and real-time diagnosis. Our Bio-

specific design methodology envisions full system modeling accuracy, high HW/SW parallelism exploitation, and computation vs. communication parallelism [2]. Moreover, we illustrate the potential advantages that MPSoC technology can bring to realtime biological analysis in the following points: (i) larger time margin to run diagnosis algorithms, (ii) energy efficiency, (iii) and improved scalability to challenging higher sampling frequencies and to more accurate analysis algorithms.

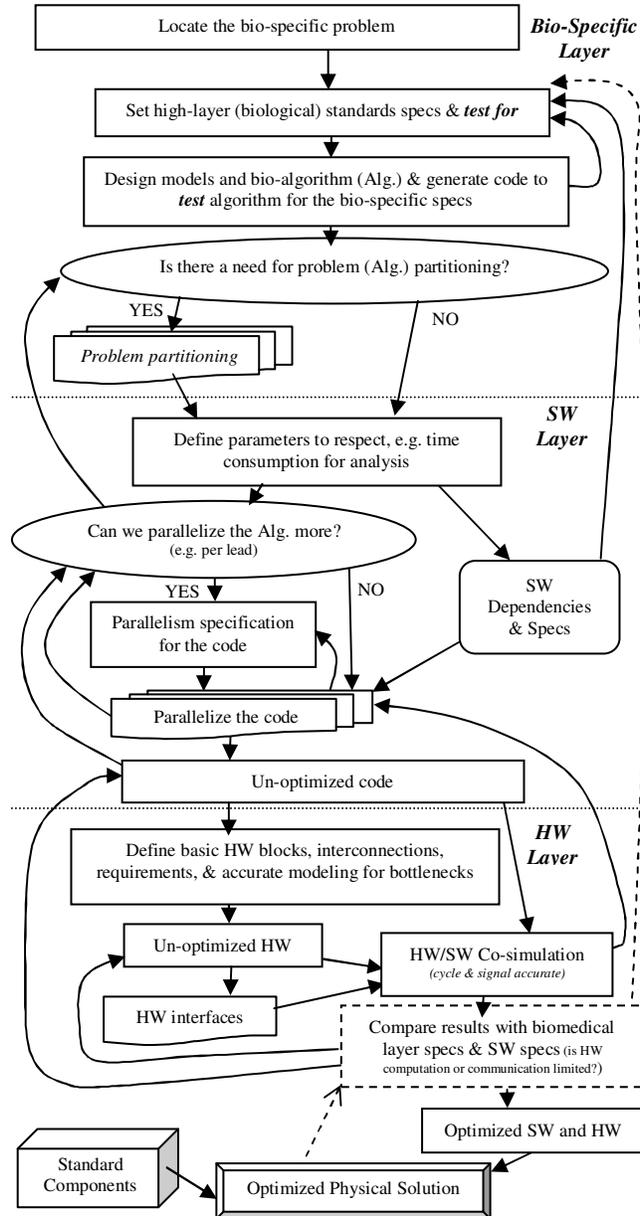


Figure 1. Methodology for bio-specific HW/SW co-design for Multicore biochips [4].

II. REFERENCES

- [1] I. Al Khatib, F. Poletti, D. Bertozzi, L. Benini, M. Bechara, H. Khalifeh, A. Jantsch, and R. Nabiev, "A Multiprocessor System-on-Chip for Real-Time Biomedical Monitoring and Analysis: ECG Prototype Architectural Design Space Exploration," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Vol. 13, Issue 2, Article No. 31, April 2008.
- [2] Loghi, M., Angiolini, F., Bertozzi, D., Benini, L., and Zafalon, R. Analyzing On-Chip Communication in a {MPSoC} Environment, In *Proceedings of Design and Test in Europe Conference (DATE)*, February 2004, 752-757.
- [3] M. Loghi, M. Poncino, and L. Benini. Cycle-Accurate Power Analysis for Multiprocessor Systems-on-a-Chip, *GLSVLSI04: Great Lake Symposium on VLSI*, April 2004, 401-406.

Markov Chain Analysis of Genetic Circuits

Curtis Madsen and Chris Myers

Dept. of Electrical and Computer Engineering
University of Utah
Salt Lake City, UT 84112

Email: curtis.madsen@utah.edu, myers@ece.utah.edu

Chris Winstead

Dept. of Electrical and Computer Engineering
Utah State University
Logan, Utah 84322

Email: winstead@ece.usu.edu

When designing and analyzing genetic circuits, researchers are often interested in the likelihood of the system being in a given state. Usually, this involves simulating the system to produce some time series data and analyzing this data to discern the state probabilities. However, as the complexity of models of genetic networks grow, it becomes more difficult for researchers to reason about the different states by looking only at time series simulation results of the models. To address this problem, this paper introduces a methodology for converting a *Genetic Circuit Model* (GCM) into a *labeled Petri net* (LPN). The state space of the LPN is then computed and the resulting state graph is analyzed using *Markov chain analysis* to determine the probabilities of the circuit being in any of these states. This paper illustrates a use of this methodology to determine the likelihood of failure in three implementations of a genetic Muller C-element.

Applying Petri nets to the analysis of biological systems is not a new concept. Goss and Peccoud used *stochastic Petri nets* (SPNs) to model systems by assigning the number of tokens in each place to be the amount of each species and the transitions to be reactions in the network [1]. In addition, *hybrid Petri nets* (HPNs) which are composed of a discrete part for modeling discrete quantities and a continuous part for modeling continuous quantities have been used to analyze biological networks [2]. Our LPNs, however, model discrete variables as the species amounts, the places as important species thresholds, and the transitions as abstractions of many reactions firing to move between these thresholds. The conversion process of a GCM into an LPN consists of creating a new variable in the LPN for each species in the GCM, creating a place for each threshold provided by the user, linking these places together with transitions to and from the place with the closest greater than threshold and the place with the closest less than threshold, and adding variable assignments to the places. Finally, transition rates are calculated by formulating equations based on the activation and repression influences.

Once the conversion process is complete, the LPN's state graph can be found using a simple iterative algorithm that walks through the possible markings of the places in the LPN starting with the initial marking. From here, an iterative Markov chain analysis algorithm called the power method is used to calculate the probability of the system being in each state of the state graph. The general idea behind this method is to initialize a vector to a random initial distribution where

all of the entries sum to one. This vector is then continually multiplied by the transition matrix until the values in the vector converge [3]. Figure 1 shows an example of converting a GCM consisting of two species that repress each other into an LPN using thresholds at 0, 20, and 40 and performing Markov chain analysis on the resulting LPN to determine the system's state probabilities. As seen in the figure, the GCM may grow in complexity when converted into an LPN; however, the LPN is used as an internal logical representation of the circuit and is not normally viewed or modified by the user.

The methodology presented here has been implemented within `iBioSim` [4], and it has been applied to several implementations of a genetic Muller C-element. C-elements are state holding gates where the output either matches the inputs when all of the inputs agree, or the output retains its value from the last time the inputs are the same and does not change until all of the inputs are equal again. These circuits are useful when designing decision circuits for systems such as quorum sensing networks because they can be used for a system to come to a consensus on a signal and then can help maintain that decision with their state holding properties [5].

The first analyzed circuit is implemented by a majority gate where the two inputs to the circuit and a feedback signal from the circuit's output are fed through three NAND gates in varying combinations. The outputs from these NAND gates are then compared with each other and the signal that has the majority of the votes is selected as the new output. The next circuit is an implementation of a speed-independent C-element [6]. The idea behind this circuit is that no matter how fast or slow the inputs change, the output maintains its state or changes accordingly. The final circuit constructs the C-element from a genetic toggle switch [7]. This circuit works by taking an inverted NAND gate signal of the two inputs as the switch part of the circuit and an inverted NAND gate signal of the inverted inputs as the reset part of the circuit. These circuits are described in more detail in [5] and [8].

When performing the Markovian analysis on these circuits, special conditions were added that would keep track of the probability of changing from one state to another state instead of just keeping track of the probability of being in each individual state. This way, the probabilities of the C-elements losing their states could be plotted as shown in Figure 2.

In conclusion, the logical representation of a genetic network combined with a Markov chain analysis engine will

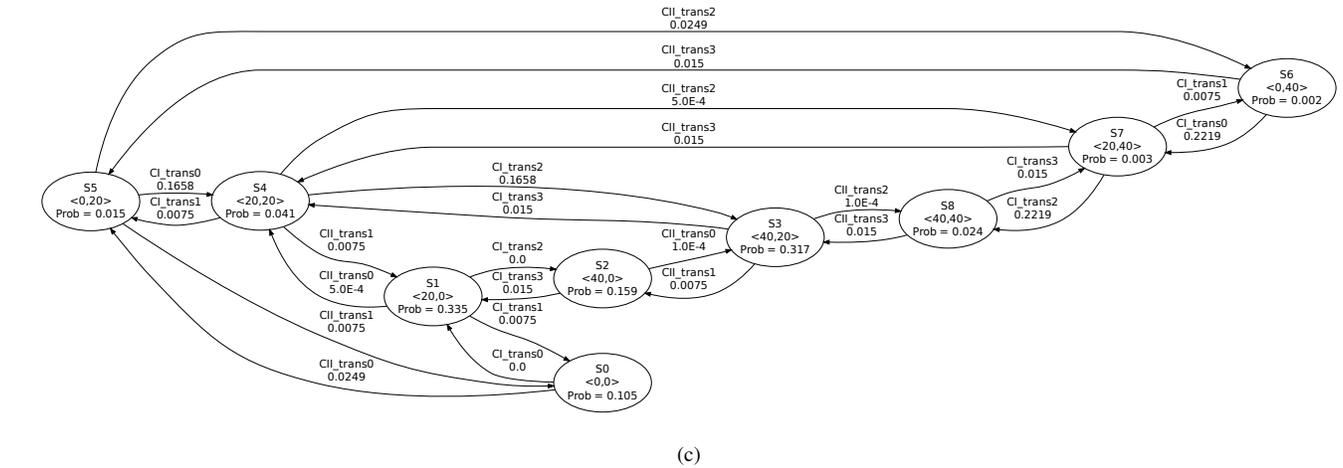
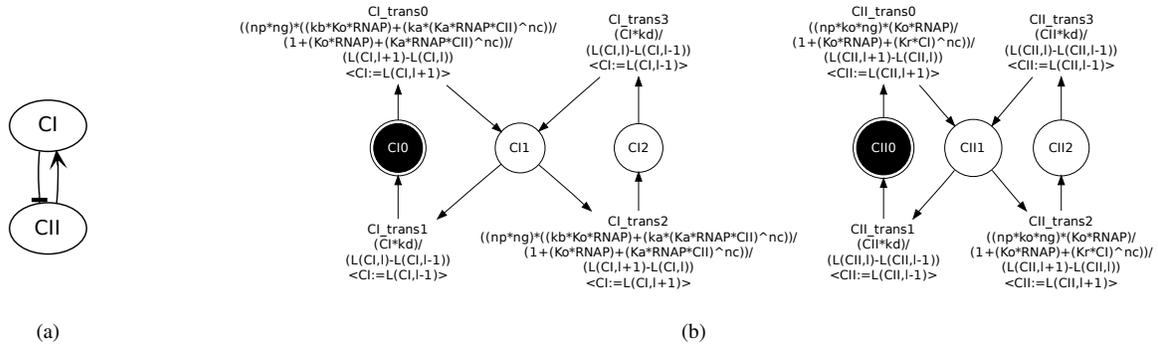


Fig. 1. (a) A simple GCM where CI represses CII and CII activates CI. (b) The LPN that is generated from the GCM in (a). This LPN was generated by selecting levels at 0, 20, and 40 for CI and CII. The subnet on the left represents the concentration of species CI, and the subnet on the right represents the concentration of species CII. (c) The resulting state graph annotated with probabilities after Markovian analysis is performed on the LPN in (b).

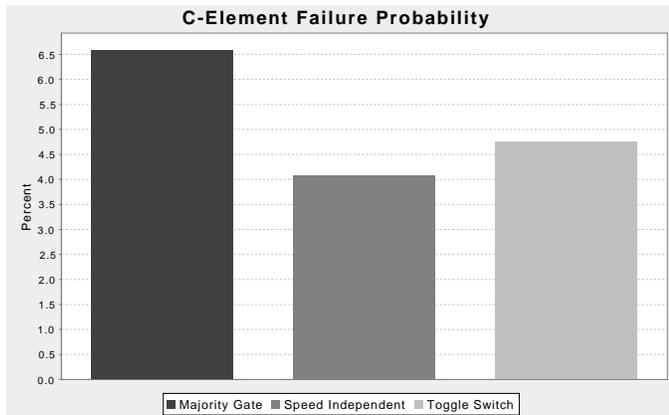


Fig. 2. Results of performing Markov chain analysis on three genetic Muller C-element circuits. These results show that the majority gate C-element fails about 6.60 percent of the time while the speed-independent and toggle switch C-elements perform better only failing about 4.09 percent and 4.75 percent of the time, respectively. All of these genetic circuits are analyzed with levels at 0 and 25.

help researchers greatly because it will allow them to obtain the probability of each state of the system occurring. This will ultimately lead to researchers making better design

decisions because they will be able to see how robust their models are. The methods discussed here have been implemented in a tool called *iBioSim* which is freely available at <http://www.async.ece.utah.edu/iBioSim/>.

REFERENCES

- [1] P. J. Goss and J. Peccoud, "Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets," *Proceedings of the National Academy of Sciences USA*, vol. 95, no. 12, pp. 6750–6755, 1998.
- [2] H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano, "Hybrid petri net representation of gene regulatory network," in *Pacific Symposium on Biocomputing*, R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, Eds., vol. 5. Singapore: World Scientific Press, 2000, pp. 341–352.
- [3] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*. 41 William Street, Princeton, NJ, 08540: Princeton University Press, 1994.
- [4] C. J. Myers, N. Barker, K. Jones, H. Kuwahara, C. Madsen, and N.-P. D. Nguyen, "iBioSim: a tool for the analysis and design of genetic circuits," *Bioinformatics*, vol. 25, no. 21, pp. 2848–2849, 2009.
- [5] N. phuong Nguyen, C. Myers, H. Kuwahara, C. Winstead, and J. Keener, "Design and analysis of a robust genetic muller c-element," *Journal of Theoretical Biology*, vol. In Press, Corrected Proof, pp. –, 2009. [Online]. Available: <http://198.81.200.2/science/article/B6WMD-4XP383G-1/2/fc20a38ddc3de22ed188297b64396c6>
- [6] O. Mayevsky, V. Varshavsky, V. Marahovsky, and Y. Mamrukoy, USSR patent certificate n1081801, the inventions bulletin n 13, 1983.
- [7] T. S. Gardner, C. R. Cantor, and J. J. Collins, "Construction of a genetic toggle switch in *escherichia coli*," *Nature*, vol. 403, pp. 339–342, 2000.
- [8] N. Nguyen, "The design of a genetic muller c-element," Ph.D. dissertation, University of Utah, 2008.

Logical Modeling of Peripheral T Cell Differentiation

Natasa Miskov-Zivanov¹, John A. P. Sekar¹, Michael S. Turner², Lawrence P. Kane²,
Penelope A. Morel² and James R. Faeder¹

Short Abstract — Peripheral naïve T-cells can differentiate into several types of effector cells and the relative numbers produced of each cell type are critical for many immune-related pathologies. To study this system, we have constructed a logical model in which each molecule type is treated as a discrete variable. We first validated that the model reproduced several important experimental observations: the cell fate dependence on antigen dose and on the Akt/mTOR pathway, the inverse correlation between Foxp3 and pS6, and the changes in Foxp3 expression in time for high antigen dose. Construction and validation of the model helped to clarify the logical relationships among molecular inputs at several key control points in the process. Predictions of the model have also led to the design of new experiments to probe the behavior of key elements such as PTEN, SMAD3, and IL-2 under different initial conditions.

I. MOTIVATION

MECHANISMS involved in DC-mediated expansion of regulatory T (Treg) cells, as opposed to helper T (Th) cells, are still not well understood, although recent data have demonstrated roles for antigen dose, costimulatory molecules and specific cytokines [5][7]. More specifically, experimental data suggest that there is a T cell-intrinsic mechanism such that low T-cell receptor (TCR) signaling levels favor Treg induction [7]. Recent studies have also emphasized the role of the Akt/mTOR signaling pathway in inhibiting the induction of Treg cells [5]. Therefore, it is essential to understand the pathways leading to DC-mediated differentiation of naïve T-cells. Preventing Treg cell induction at the level of DC-T cell interactions might be one way to eliminate antigen-specific Treg cells and thus decrease or even reverse immune suppression in cancer.

II. LOGICAL MODELING APPROACH

Logical and, in particular, Boolean modeling approaches have been playing an increasingly important role in the analysis of complex biological systems. Modeling regulatory networks by means of Boolean networks was introduced in the late sixties [3], but has received more attention in recent years [2][4]. We created a Boolean model, in which each element of the network is represented as a Boolean variable. These variables can take two values: ‘1’ and ‘0’, representing active (present) and inactive (absent) state of the element, respectively. Interactions between elements in a logical model are represented using Boolean functions, *i.e.*, combinations of the basic operators ‘and’, ‘or’, and ‘not’. The first step in creating a logical model is to decide which elements and pathways need to be included in the model. For the purpose of studying the differentiation of naïve T cells, we have identified a set of elements that are found to have important roles in this process [5][7]. Next, the most difficult step in the formulation of the model is determination of the best logical representation of variable update rules when there are multiple input variables. The decision whether to represent these relationships using ‘and’ or ‘or’ is not always straightforward. However, in most cases it is possible to infer model relationships from verbal descriptions. Furthermore, we have found examples where only one of several possible formulations of the regulatory interactions for a particular node produces realistic model behavior.

III. RESULTS

We created a logical model for the T cell differentiation following the steps described in Section II and ran simulations of the model using BooleanNet tool [1]. Each simulation run uses random asynchronous updates of values of elements in the model, with overall 50 simulation steps. Selected results are presented in Fig. 1, together with their corresponding experimental results [5][7]. For each scenario, 300 independent simulations were conducted in order to find the average variable values. Foxp3 is used as a marker for Treg cells, and thus, high percentage of Foxp3 represents higher amounts of Treg cells.

Experimental results from [7] presented in Fig. 1(a)(top) show that high antigen dose (**TCR high**) produces mostly Th cells and low antigen dose (**TCR low**) case generates a high fraction of Foxp3-expressing Treg cells. Simulation results in Fig. 1(a)(bottom) show that **TCR high** trajectories all result with Foxp3 expressed, whereas **TCR low** trajectories all end with Foxp3 absent. The **TCR high** simulations also match the experimental observation of transient Foxp3 expression, as can be seen from the red curve in Fig. 1(a)(bottom) and for the 18 hour time points in Fig. 1(a)(top). Fig. 1(b)(top) presents simulations in which **TCR high** is set for four time steps and then turned off, corresponding experiments in which the antigen is removed after 18 hours. Again, simulations agree qualitatively with the experimental finding that a substantial fraction of the resulting cell population expresses Foxp3 (data not shown). Results presented in Fig. 1(b)(bottom) illustrate the consequences of blocking Akt or mTORC1 activation following **TCR high** after four time steps, corresponding

Acknowledgements: This work was funded by NIH grant CA73743 (PAM) and MST was supported by T32 CA82984

¹Department of Computational and Systems Biology, School of Medicine, University of Pittsburgh, E-mail: {nam66, jas237, faeder}@pitt.edu ²Department of Immunology, School of Medicine, University of Pittsburgh, E-mail: {mstst22, morel, lkane}@pitt.edu.

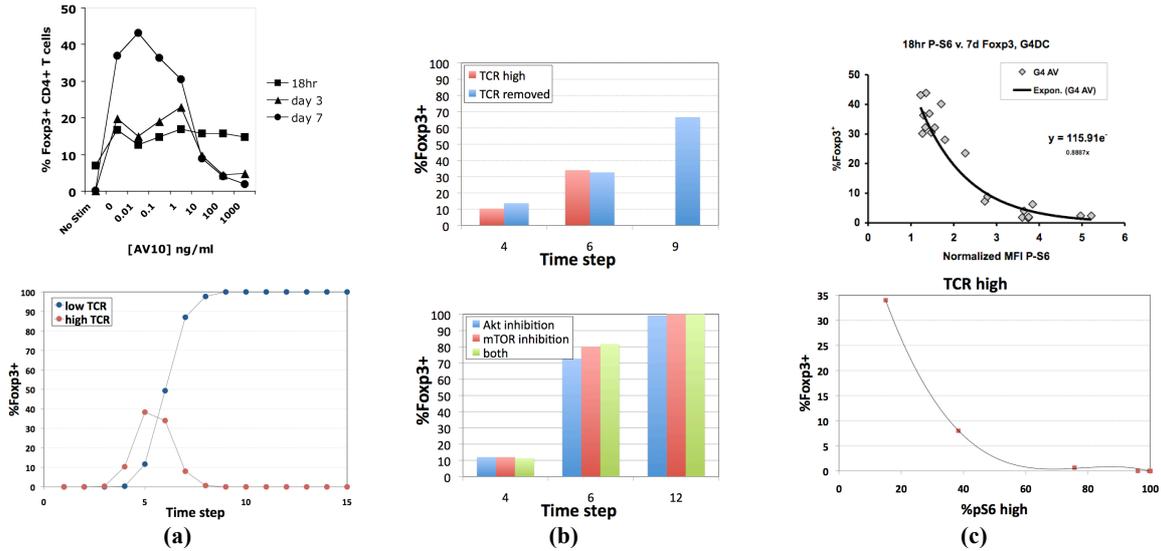


Fig. 1. Logical model reproduces experimental results. (a) Dependence of Foxp3 expression on time and antigen dose (TCR low vs. TCR high): results from experiments [7] (top) and logical model (bottom) (b) Logical model results for the dependence of Foxp3 expression on: antigen removal after four steps (top) and the addition of Akt and mTORC1 inhibitors after four steps (bottom). (c) The inverse correlation of pS6 and Foxp3: results from experiments [7] (top) and logical model (bottom).

to experiments done with inhibitors that are added 18 hours after strong TCR stimulation. Inhibition is modeled by fixing the value of either Akt or mTORC1 at ‘0’ after four time steps. Again, simulation results agree with the experimental finding that a substantial Foxp3 positive population is generated under these conditions. Finally, Fig. 1(c) shows results obtained from experiments [7] (top) and logical model simulations (bottom) that indicate an inverse correlation between pS6 levels at early time points (18 hours) and Foxp3 expression later on (7 days).

IV. DISCUSSION

The logical model that we have constructed for T cell differentiation reproduces many of the important experimental results. The analysis of the results obtained using the logical model has also initiated further experimentation. Analysis of transient Foxp3 expression for the high-antigen dose case (Fig. 1(a)) indicates that the timing of initial Foxp3 expression relative to mTORC1 activation determines whether transient expression occurs. In our simulations early STAT5 activation of the Foxp3 promoter occurs in all trajectories with transient Foxp3 expression, which points to the need for further experiments to determine whether interfering with STAT5 activation is sufficient to eliminate transient expression or other factors are involved that should be considered in the model. Factors influencing transient expression may also affect the cell fate decision under other conditions. Simulation results also emphasize the need for better experimental characterization of PTEN, particularly its differential activity in Treg vs. Th.

Furthermore, the simulation results have also suggested the rationale behind oscillations in the Foxp3 and IL-2 values after logical relationships are changed from ‘and’ to ‘or’ or vice versa. These oscillations either indicate which of the alternative relationships is the one closest to the realistic behavior, or they indicate that there is an important impact of the negative feedback loop that could only be captured when the variables representing element states are allowed to have more than two different values.

Thus, by using the logical model that we constructed, we have identified the elements that may play a critical role in the system and we are conducting new experiments to obtain further measurements of these elements (e.g., PTEN, SMAD3, IL-2), as well as to gain more insight into element relationships (e.g., pS6 kinetics with respect to the existence of IL-2). We anticipate that these measurements will further advance our understanding of the determinants of the peripheral T cell fate and thus, result in creating mechanism for controlling T cell differentiation and for preventing Treg induction.

REFERENCES

- [1] Albert, I. *et al.* (2008), *Source Code Biol Med*, 3-16.
- [2] de Jong, H. (2002), *J Comput Biol* 9, 67-103.
- [3] Kauffman, S. A. (1969), *J Th Biol* 22, 437-467.
- [4] Saez-Rodriguez *et al.* *PLoS Comput Biol* 3, e163.
- [5] Sauer, S *et al.* (2008), *PNAS* 105, 7797-7802.
- [6] Thomas, R and D’Ari, R. (1973), *Biological Feedback*, CRC Press.
- [7] Turner, M. S. *et al.* (2009), *J. Immun* 183, 4895-4903.

Synthetic gene circuits with a cell-free toolbox
Vincent Noireaux, Jonghyeon Shin

In the past 10 years, advances in molecular biology have led to the emergence of systems biology and synthetic biology. These new research areas aim at understanding the information structure and dynamics of living systems such as gene regulatory networks. Many innovative approaches have been developed to study the flow of information in biological systems. While most of the studies are performed *in vivo* or *in silico*, only a few *in vitro* approaches have been proposed.

Cell-free protein synthesis is increasingly used to produce large amounts of proteins *in vitro*. Cell-free systems combine a powerful bacteriophage transcription, in most cases the T7 RNA polymerase, to a cytoplasmic extract from an organism, such as *E. coli*, that provides the translation machinery. These systems have been prepared for many types of applications, mostly in biotechnology. Recently, cell-free protein synthesis was used to reconstitute information processes outside a living organism such as elementary gene circuits and pattern formation. These studies were limited, however, by the current available cell-free systems which have not been optimized for synthetic biology purposes. In particular, transcription is restricted to a few promoters and only a few synthetic bacteriophage promoters regulated by operators have been described. Moreover, the high protein synthesis rate of cell-free systems needs to be balanced with a high degradation rate of both transcripts and gene products to ensure economical bookkeeping of the information processing. No mechanisms of messenger RNA inactivation and protein degradation have been described to adjust these parameters. These limitations reduce considerably the potential of cell-free protein synthesis as a system to engineer and to run gene circuits *in vitro*.

Our laboratory has developed a cell-free expression toolbox specifically adapted for the synthesis of gene circuits *in vitro*. Transcription/translation is carried out in a *E. coli* extract which works with seven *E. coli* sigma factors and two bacteriophage RNA polymerases. The system includes mechanisms to tune the mRNA inactivation rate and the protein degradation rate. Protein synthesis is controlled by adjusting gene concentrations, promoter strengths, synthesized messengers and proteins lifetime. The toolbox provides unique transcription modularity and a large set of adjustable parameters to engineer synthetic networks *in vitro*.

This cell-free toolbox is used for two purposes: (1) the construction and the study of elementary gene circuits and (2) a constructive approach to synthetic cell. Multiple stage transcription cascades, AND gates and negative feedback loops have been engineered. These circuits have revealed underlying mechanisms such as the competition of sigma factors for the *E. coli* core RNA polymerase that can result in transcription auto-regulation or inhibition. Output signals of these circuits can be tuned in a wide dynamic range depending on mRNA and protein degradation rates. The toolbox is also used to synthesize an artificial cell using a constructive bottom-up approach. The cell-free extract is encapsulated into cell-sized phospholipids vesicles. Properties of the synthetic vesicles are developed from the internal gene expression. The main question asked by this work is: how far can we go in the reconstitution of gene networks *in vitro* with a constructive, deterministic approach? The perspectives and the limitations of this approach will be discussed.

Structure-based Prediction of Residue Coevolution in Proteins

Noah Ollikainen¹, Ellen Sentovich, Carlos Coelho², Andreas Kuehlmann², Tanja Kortemme¹

¹ University of California, San Francisco, CA

² Cadence Research Laboratories, Berkeley, CA

Abstract

Discovering pairs of amino acid residues in proteins that have undergone correlated evolution (coevolution) provides valuable information for both understanding protein evolution and predicting the effects of mutations during protein design. Previous efforts have been aimed at detecting residue coevolution through statistical analyses of protein sequence alignments¹. These studies have identified networks of coevolving residues in a number of protein families, and these networks may play functional roles such as mediating allosteric communication between distant sites in a protein. However, in many cases it is not clear whether the coevolution of a pair of residues is structurally or functionally significant. Moreover, these sequence-based methods require a large number of diverse protein sequences belonging to a given protein family in order to identify statistically significant pairs of coevolving residues. This prevents these methods from being applied to proteins without numerous extant sequences, for example, novel proteins that have been designed to carry out a particular function. Here, we present a novel method to predict the extent of coevolution between all pairs of residues using the three-dimensional structure of a given protein. This method uses a deterministic, SAT-based protein design algorithm² to identify the global minimum energy conformation for all possible single mutations. A pair of residues is “coupled” if the mutation of one residue perturbs the side-chain conformation of another residue from its global minimum conformation, and the extent of coupling between two residues is quantified using a metric based on mutual information. This approach is evaluated using a dataset of known coevolving residues in the SH3³, PDZ, PAS, SH2 and S1A serine protease protein families⁴. A comparison of coevolving residues identified with a sequence-based approach and coupled residues predicted with our structure-based approach for SH3 domains is shown in Fig. 1, and a cluster of coupled residues that constitutes the hydrophobic core is shown in Fig. 2. We demonstrate that our approach is more effective at predicting residue coevolution than a similar method that uses Monte Carlo optimization, emphasizing the importance of deterministic protein design algorithms. Our approach can be used to understand how mutations affect distant sites on a protein, which is of great importance for protein design applications that require optimizing the catalytic efficiency of an enzyme or tuning the specificity of a protein-protein interaction.

- 1) Lockless, S.W. and R. Ranganathan. Evolutionarily conserved pathways of energetic connectivity in protein families. *Science* **286**, 295–299 (1999).
- 2) Ollikainen, N., E. Sentovich, C. Coelho, A. Kuehlmann and T. Kortemme. SAT-based Protein Design. International Conference on Computer-Aided Design. 128-135 (2009).
- 3) Larson, S. M., A. A. Di Nardo and A. R. Davidson. Analysis of covariation in an SH3 domain sequence alignment: applications in tertiary contact prediction and the design of compensating hydrophobic core substitutions. *J. Mol. Biol.* **303**:433-446 (2000).
- 4) Halabi, N., O. Rivoire, S. Leibler and R. Ranganathan. Protein sectors: evolutionary units of three-dimensional structure. *Cell* **138**:774–786 (2009).

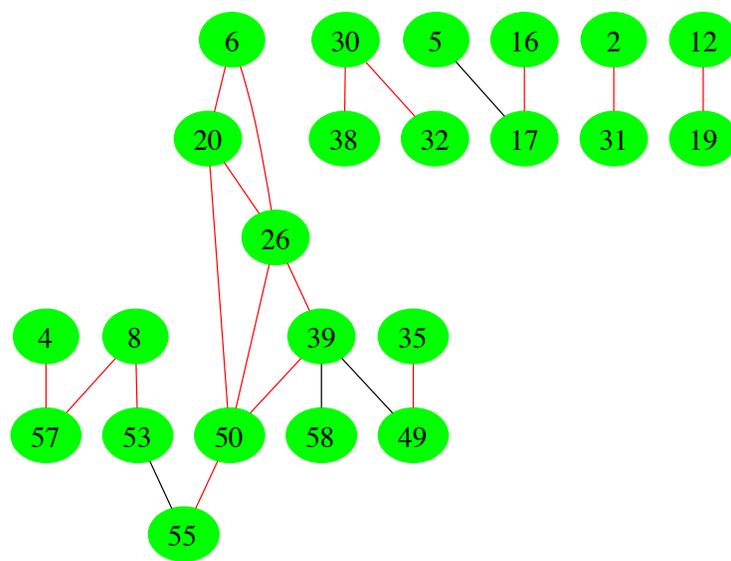


Figure 1. Coevolving residue networks in the SH3 domain protein family. Nodes represent residues and edges represent pairs of coevolving residues identified using a sequence-based approach. Red edges denote pairs of coevolving residues that were successfully predicted with our structure-based approach.

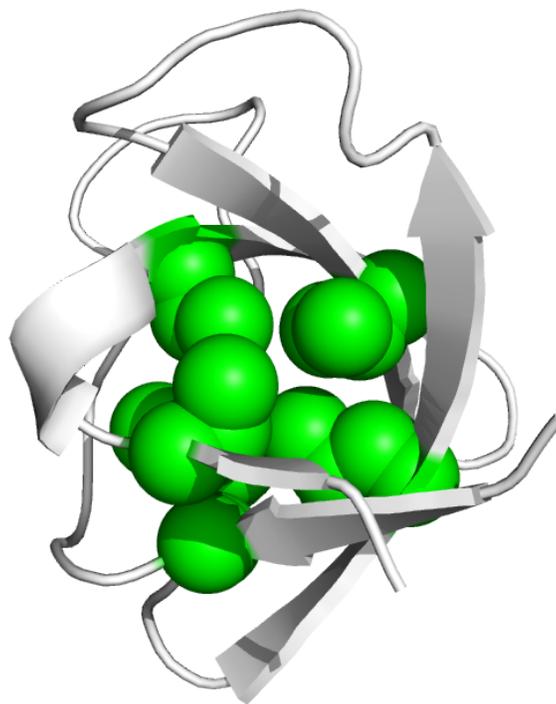


Figure 2. Structure of an SH3 domain showing a cluster of coevolving residues (6, 20, 26, 39, 50 and 55) with side-chains illustrated as green spheres.

Architecture for Synthetic Organism Design Matthew Peterson, Steven Fairchild, John Dileo

Computational tools that enable automatic and accurate synthetic organism design would significantly enhance researchers' abilities to rapidly generate organisms with novel capabilities and functions. Realization of this goal is non-trivial given the complexity of living systems and the difficulty of predicting synthetic part behavior in the context of complete cellular systems. Nonetheless, progress has been made towards the development of such tools, and improvements are expected to continue as the understanding of biological design principles improve. This poster presents a hypothetical synthetic organism design architecture (SODA) for end-to-end engineering of synthetic systems. The purpose of this poster is to evaluate what system-level capabilities are required for synthetic organism design, analyze the state of the art for existing capabilities, and evaluate key areas for future research. In analyzing the proposed architecture, the poster also discusses research conducted by the authors in developing tools for automatic protein engineering.

The proposed synthetic organism design architecture (SODA) is illustrated in Figure 1. As this figure indicates, the architecture takes as input the desired system function and proposed components. Here, input functionalities can range from chemical synthesis, to sensing, to information processing, to bioremediation and generation of bioelectricity. The inputs also include a set of existing synthetic parts, expected connectivity between these parts, and the desired organism chassis. Using these inputs, the SODA ultimately returns a set of synthetic parts which achieve the desired functionality within the given cellular system. Finding this set of parts requires the various modules which are depicted in Figure 1. These modules are responsible for part selection, part generation, pathway modeling, and parameter optimization. The design process also requires information from various biological and parts databases. It is expected that the system will be iterative and fully autonomous. Thus the SODA is expected to be able to automatically determine which functionalities are needed for completing system design, find and generate these parts based on existing databases, and evaluate if the selected set of parts will have the desired activity. If the parts do not have the desired activity, then the SODA is expected to determine which parameters should be modified to give the desired activity based on previous design concepts and then repeat the part selection, generation, and evaluation process based on these parameters. Figure 1 also reflects the expected reality that computationally designed synthetic systems will ultimately require experimental verification.

Along with describing an idealized concept for synthetic organism design, the poster also evaluates the current state of the art for the various databases and design modules. For the design modules, specific computational tools are analyzed that could potentially be used for completing each task. This includes computational methods for automatically selecting parts, engineering biomolecules with specific capabilities, analyzing the function of cellular pathways, and evaluating how a set of synthetic parts will alter other pathways within the chassis. Through analyzing such capabilities,

conclusions can be made regarding areas where major capability gaps exist in computational tools for synthetic system design. While some of these capabilities are very challenging areas (e.g., de novo protein design for a specific function), others are likely achievable in the near term (e.g. re-engineering a protein for a novel but related function). Research being conducted at MITRE to automatically engineer enzymes with specific functions is highlighted to show how some of these gaps are being addressed. Ultimately, the set of capability gaps provides a detailed roadmap for key areas where future research should be conducted for enabling automatic synthetic organism design.

Altogether, the goals of the proposed synthetic organism design architecture are to facilitate further discussion, understanding, and development of methods for automatically engineering synthetic organisms and to foster integration and interoperability between technologies developed by the synthetic biology community.

Sequence Refiner: Automated conversion of natural genetic sequences into standard biological parts

Cesar A. Rodriguez¹, Adam Arkin^{1,2}, Drew Endy^{1,3}

Affiliations: ¹BIOFAB, ²Department of Biological Engineering, University of California at Berkeley ³Department of Biological Engineering, Stanford University

Purpose

We are working to make the engineering of biology easier. One aspect of our work is to develop an increased capacity to refine and standardize large numbers of natural genetic sequences. The primary goals of such refinements are to improve the physical and functional composition of the resulting objects, as per Canton et al. [PMID: 18612302]. We also want to enable many people to more readily and accurately produce high quality standard biological parts; such parts collections are needed to make real the designs produced using higher-level synthetic biology software design tools. Thus, we developed open-source software that supports the automated conversion of non-standard natural genetic sequences into simpler standard biological parts.

Methods

The software, which we named Sequence Refiner, is comprised of client and web service components; the Sequence Refiner Client and the Sequence Refiner Service, respectively. The client can read in DNA sequences encoded in Genbank format. The sequences are transmitted to the Sequence Refiner Service via an HTTP POST request from the client to the server. The server modifies the incoming sequences in accordance with one or more refinement standards. Presently, BioBricks Foundation Physical Assembly Standards #10, 12, 21, and 25 are supported. Most existing standards require refinements that are synonymous changes of codons in a DNA sequence. The selection of alternate codons can be customized to be consistent with a particular codon preference scheme. For example, codon preference based on tRNA abundance [Ikemura; PMID: 6175758] is implemented; random silent substitutions may also be selected. The Sequence Refiner client is an Adobe Flex application that executes via the Flash Player that is installed in all major web browsers. The Sequence Refiner service is a RESTful web service written in Java and uses J2EE and BioJava packages. The Sequence Refiner service is hosted at Google App Engine. The source code for Sequence Refiner is open and freely available (please see <http://www.biofab.org/software> for distribution information).

Results

Preliminary testing of the Sequence Refiner demonstrates sequence refinements in the millisecond to second timescale. The resulting sequences are fully compliant with the BBF physical assembly standards noted above. The Sequence Refiner can thus

replace the manual editing of sequences that requires tens of minutes and is prone to human error.

Conclusions

Sequence refinement has the potential to increase the quantity and quality of standards compliant DNA sequences. The Sequence Refiner supports the goal of making biology easier to engineer.

Using uncertainty quantification to constrain dynamic neuron modeling parameters

Richard Schiek¹, Christy Warrender²

¹Electrical and Microsystems Modeling, ²Cognitive Science Applications
Sandia National Laboratories, PO Box 5800, MS 0316
Albuquerque, NM, 87122-0316, USA

¹rlschie@sandia.gov

Presentation or Poster

Abstract

The goal of this work is to develop computational and statistical tools to enable uncertainty quantification of neural simulations. We extend fourier-based techniques to use experimental data in the refinement of neuron simulation parameters and topology.

Computer simulations of neural activity are valid constructs within a restricted operating regime. This work calculates limits or bounds within which one can be confident that a simulation is dynamically behaving in the same manner as an experiment. To that end we have implemented common neurological ion-channel models (e.g. Hodgkin-Huxley, Connor-Stevens) in a dynamic cable-equation format within a circuit simulator, Xyce (xyce.sandia.gov); see simulation outline below. This allows one to use a netlist style syntax to describe a collection of neurons for simulation. As with any circuit simulation, the model parameters for the circuit components are critical in determining the circuit's performance.

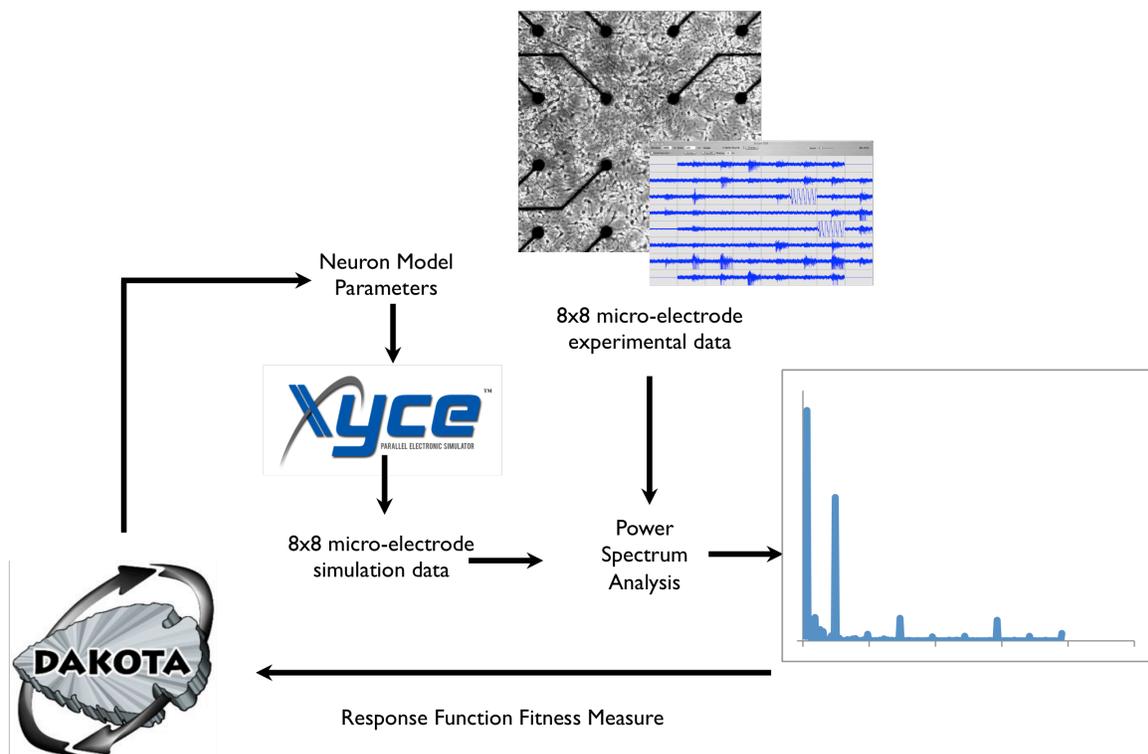
Experimental data from micro-electrode array recordings on hippocampus cell cultures (data courtesy of B. Wheeler & D. Khatami, U. of Florida) were used to bound the simulation parameters. Specifically, transient data from the simulations is compared to micro-electrode array data. However, direct comparison cannot be made between the experimental and simulation data in the transient domain because of the unknown initial condition state of the experiments and the unknown topology of the experimental system. Two approaches are taken to mitigate these problems. First, by transferring the results to a frequency domain and constructing a power-spectra, one can compare the two sets of data and infer important properties. Differences between the simulated and experimental power-spectra allow one to both optimize the fit of the simulation to the experiments and calculate the uncertainty allowed in the simulation's model parameters. Second, since the underlying cellular topology is unknown for the experimental system, random topologies (and some directed topologies for verification) are generated and used in the simulation. Thus, both the model parameter space is searched and the circuit topology space is searched for systems that dynamically mimic the experiments.

This allows one to quantify what is unknown or unrepresented in both the experiments and simulations leading to a better understanding of both results. As

expected, low-fidelity data can be matched with a low fidelity, or simple model. When the experimental system becomes more complex (i.e. exhibits long term potentiation) more complex models are needed to fully describe the data.

Finally, because the computational process can be automated, tens to thousands of simulation parameters and or topologies can be tested for their sensitivity on the results. To confront the geometric scaling for simulation needs, we utilize a hierarchial parallel simulator. Xyce can run on multiple processors in a distributed or shared memory system and the uncertainty quantification controller, Dakota, can control and dispatch multiple jobs in parallel. Thus, one can efficiently utilize a computing cluster where each compute node has multiple cores and there are multiple compute nodes.

We will present results comparing simulated and experimental systems under both un-stimulated and post-stimulation conditions. Uncertainty quantification for the Hodgkin-Huxley and more complex Connor-Stevens neuron models indicates that sodium and potassium ion conductance terms are critical for this experimental system. We will include bounds within which the simulations are relevant and discussion of scalability to larger systems.



Simulation and Uncertainty Quantification Loop: Experimental system and typical transient data show at top. Sample power spectra shown on lower right

Sandia National Laboratories is a multi-program laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin company, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Identification of Illegal States in a Discrete Transition Model of Apoptosis Signaling

Anupam Shrivastava, Huy Lam, Michael S. Hsiao, David Samuels*, Carla Finkielstein

{anupams, lamhuy, mhsiao, finkielc}@vt.edu and *david.c.samuels@vanderbilt.edu

EXTENDED ABSTRACT

Recent research efforts in bioinformatics and computational biology focus on gaining a deeper insight into cell death processes, allowing researchers to study the relation between the malfunction of apoptosis and development of human diseases such as cancer. The apoptosis process involves a complex system of pathways that includes different proteins and signals [2][3][4][5][6]. Constructing the pathways as a finite state transition model (FSM) provides a unique level of abstraction allowing analytical algorithms to be applied to the study of this model. One such recent work is the modeling of the mitochondrially mediated apoptotic signaling pathways as a discrete transition system model [1]. This framework can model healthy as well as slightly altered, or faulty, signaling pathways. Subsequently, digital system verification and testing techniques developed in the EDA domain can thus be used to aggressively analyze the model.

The inputs to the apoptosis FSM consist of all external substrates that assist the reaction. These include: Granzyme B, Caspase 8, NMT1, BCL_XL, etc. Mitochondria and Cytoplasm, acting as assembly sites for many of the pro-apoptotic and anti-apoptotic factors have also been taken into consideration in the buildup of this full system FSM. Eight state elements (flip-flops) have been used for each protein to represent its state, or in other words, its concentration in the system. The rate constants and concentrations are modeled in such a way that a unit increment in the state value denotes an increment of 0.1 nano-molar in protein concentration. Hence, concentration value modeled is 0 to 25.5 nano-molar corresponding to 0 to 255 state range.

An initial attempt using a simple combination of guided logic simulation, Bounded Model Checking (BMC) [7], and simple SAT-based induction [8] has been used to find a subset of reachable states in the apoptosis Model. SAT-based Induction has also been used to identify a subset of illegal states in the FSM. However, the results of such a simple, straight-forward analysis on this model, as discussed in [1], reveal the inadequacy of the approach. While these techniques do well in classifying the states to a certain extent, many states of different proteins could neither be reached nor proven illegal.

Therefore, in this work, we attempt to gain insight into the apoptosis model by finding out the nature of these unknown states. We propose an image computation based methodology, in which both the reached and proven unreachable states of various proteins will be used in our formulation. Our contributions can be summarized as follows:

- Categorization of 93% of previously unknown individual protein states as illegal states with our new formulation.
- Extension of our framework to efficiently study the interaction of any two proteins in the apoptosis system.

PROPOSED METHODOLOGY

Circuit abstraction has shown to be a promising approach for Model Checking [9]. We first tackle the problem of proving those unknown states as illegal by proposing a new methodology based on SAT based image computation of the abstract circuit. We refer to the

core procedure of our framework as *extract_illegal()*. We methodically constrain the one time-frame apoptosis circuit as follows:

- Constrain the target protein at the Pseudo Primary Inputs (PPI) to be in the current set of reachable states as obtained from the simulation trace of the concrete model.
- Constrain the target protein at the Pseudo Primary Output (PPO) to be outside the current reachable state set. This ensures that if the SAT Solver returns a satisfying solution, a new reachable state in the abstract circuit is obtained and it can be added to the reachable state set to constrain the search in the next iteration.
- Use the illegal states learned till now for all proteins to constrain both the PPI and PPO.

Figure 1 shows the progress of learning illegal states of different proteins with iterative calls to *extract_illegal()*. A single iteration consists of learning illegal states for all the proteins. We follow an order from protein1 (BCL2) to protein8 (tBID_cyto) regarding the calls to this function. The graph also shows the influence of different proteins on one another in terms of defining their illegal states.

Table 1 is a summary of results regarding the learning of new illegal states in this apoptosis model. These encouraging results further help us to analyze protein combinations as discussed in the next section.

STUDY OF PROTEIN COMBINATIONS

Analyzing two-protein interactions with a better knowledge of the illegal state space becomes viable now. Explicit enumeration of two protein states will simply be infeasible because of the large number of SAT solver calls involved. Instead, we branch on the most-significant bits (MSB) to reduce the search. Let us consider a state 1101xxxx1100xxxx. The first 8 bits belong to protein A and last 8 bits belong to protein B. Obtaining the state 1101xxxx1100xxxx as an illegal state specifies that any state between 208 to 223 of protein A together with any state between 192 to 207 of protein B would form an illegal state combination. The number of MSBs is investigated to study the trade-offs between performance and quality of result.

Here, we give an example which specifies the importance of the MSB learning approach. BCL2 and tBID_mito have 12876 states as possibly legal. When we first perform the 4 MSB learning, we obtain a graph as shown in Figure 2, which indicates that there is scope of learning more illegal states in such cases where crude blocks of illegal states are obtained.

More illegal states are learned with 5 MSB learning as shown in Figure 3. The illegal states learned earlier in 4 MSB learning are used as constraints to reach results faster. Table 2 shows a summary of results obtained with the analysis of two protein combinations. These results demonstrate the feasibility and potential of the proposed approach.

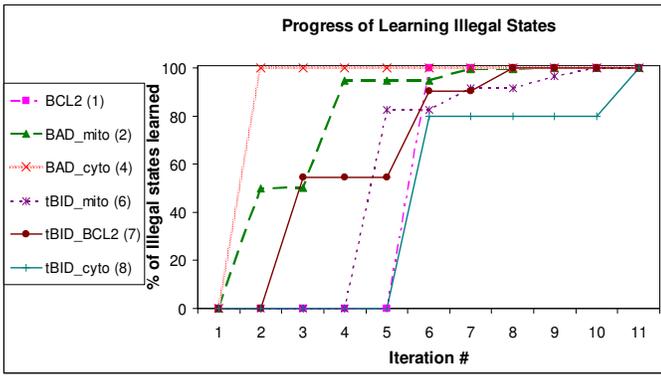


Figure 1: Progress of learning illegal states

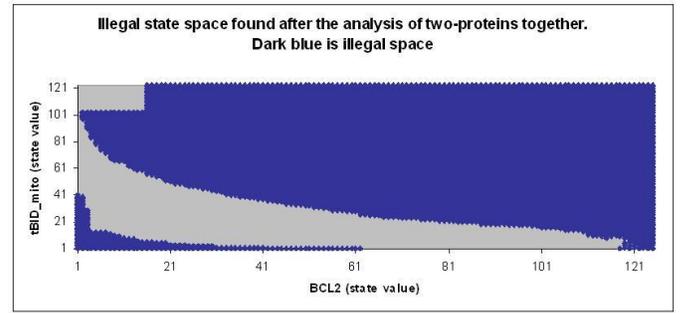


Figure 3: 5 MSB learning refines and adds to the illegal states obtained from 4 MSB learning for proteins BCL2 and tBID_mito.

Table 1: Results obtained on individual proteins

Proteins	Reachable States	Illegal States	Unknown States
BCL2	1 to 125	0, 126 to 255	NA
BAD_mito	0 to 10	17 to 255	11 to 16
BAD_p14	0 to 250	251 to 255	NA
BAD_cyto	0 to 238	239 to 255	NA
BAD_BCL2	0 to 235	?	236 to 255
tBID_mito	1 to 103	0, 124 to 255	104 to 123
tBID_BCL2	139 to 252	0 to 127, 253 to 255	128 to 138
tBID_cyto	0 to 11	12 to 255	NA

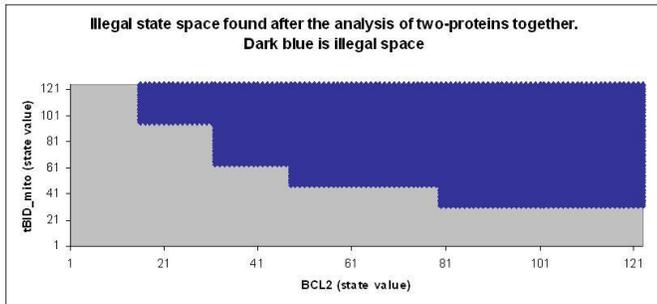


Figure 2: 4 MSB learning for BCL2 and tBID_mito showing the reduced resolution of illegal states learned (dark color).

Table 2: Protein Combination analysis

Protein Combination	Illegal States
BCL2 with tBID_cyto	658 out of 1500
BCL2 with tBID_mito	9233 out of 12876
BAD_mito with BAD_cyto	524 out of 2618
tBID_mito with tBID_cyto	728 out of 1236

REFERENCES

- [1] Lam, H. H. (2007). *Discrete transition system model and verification for mitochondrially mediated apoptotic signaling pathways*. Blacksburg, Va: University Libraries, Virginia Polytechnic Institute and State University. <http://scholar.lib.vt.edu/theses/available/etd-06282007-191549>.
- [2] G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, G.R. Gopinath, G. R. Wu, L. Matthews, S. Lewis, E. Birney, and L. Stein. Reactome: a knowledge base of biological pathways. *Nucleic Acids Research*, 33:D428-D432, 2005.
- [3] K.A. Janes, J.G. Albeck, S.Gaudet, P.K. Sorger, D.A. Lauffenburger, M.B.Yaffe. Systems Model of signaling identifies a molecular basis set for cytokine-induced apoptosis. *Science*, 310(5754): 1646–1653, 2005.
- [4] Marie-Veronique Clement David Hsu P.S. Thiagarajan Geoffrey Koh, Huey Fern, Carol Teong. A decompositional approach to parameter estimation in pathway modeling: A case study of the akt and mapk pathways and their crosstalk. *Bioinformatics*, 00(00):1–10, 2006.
- [5] K. H. Cho, S. Y. Shin, W. Kolch, and O. Wolkenhauer. Experimental design in systems biology, based on parameter sensitivity analysis using a monte carlo method: A case study for the tnf alpha-mediated nf-kappa b signal transduction pathway. *Simulation-Transactions of the Society for Modeling And Simulation International*, 79(12):726–739, 2003.
- [6] L. J. Miller and J. Marx. Apoptosis. *Science*, 281(5381):1301–, 1998.
- [7] E. Clarke, Armin Biere, R. Raimi, Y. Zhu. Bounded Model Checking using Satisfiability Solving.
- [8] M. Sheeran, S. Singh, G. Stalmarck. Checking Safety Properties using induction and a SAT Solver.. In *Proc. of FMCAD*, Lecture notes in Computer Science, pp. 108-125, 2000.
- [9] P. Chauhan, E. Clarke, J. Kukula, S. Sapra, H. Veith, D. Wang. Automated abstraction refinement for model checking large state spaces using sat based conflict analysis. In *FMCAD' 02: Proceedings of the 4th international conference on Formal Methods in Computer-Aided Design*. Pages 33-51, London UK 2002 Springer-Verlag.