IWBDA 2014
BOSTON

6TH INTERNATIONAL WORKSHOP ON BIO-DESIGN AUTOMATION

BOSTON UNIVERSITY

JUNE 11-12, 2014

FOREWORD

Welcome to IWBDA 2014!

The IWBDA 2014 Executive Committee welcomes you to Boston, Massachusetts for the Sixth International Workshop on Bio-Design Automation (IWBDA) at Boston University. IWBDA brings together researchers from the synthetic biology, systems biology, and design automation communities. The focus is on concepts, methodologies and software tools for the computational analysis and synthesis of biological systems.

The field of synthetic biology, still in its early stages, has largely been driven by experimental expertise, and much of its success can be attributed 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. Inspired by these challenges, the field has seen a proliferation of efforts to create computer-aided design tools addressing synthetic biology's specific design needs, many drawing on prior expertise from the electronic design automation (EDA) community. IWBDA offers a forum for cross-disciplinary discussion, with the aim of seeding and fostering collaboration between the biological and the design automation research communities.

IWBDA is proudly organized by the non-profit Bio-Design Automation Consortium (BDAC). BDAC is an officially recognized 501(c)(3) tax-exempt organization.

This year, the program consists of 17 contributed talks and 9 poster presentations. Talks are organized into six sessions: Tools, Modeling I, Modeling II, Circuit Design, Automation, and Algorithms. In addition, we are very pleased to have two distinguished invited speakers: Dr. Orit Shaer from Wellesley College and Dr. Tuval Ben-Yehezkel from the Weizmann Institute of Science.

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 Synthetic Biology Engineering Research Center (synberc), National Science Foundation, Autodesk, Hudson Robotics, Twist Bioscience, ACS Synthetic Biology, Raytheon BBN Technologies, and Minres Technologies for their support.

THE FOLLOWING PARTICIPANTS WERE PROVIDED FINANCIAL SUPPORT BY OUR SPONSORS TO ATTEND IWBDA 2014

Alejandro Pelaez	Boston University
Ben Kellman	University of California, San Diego
Ben Schreck	Massachusetts Institute of Technology
Bryan Der	Massachusetts Institute of Technology
Cassandra Hoef	Wellesley College
Claudio Angione	University of Cambridge
Ehsan Ullah	Tufts University
Eric Kelsic	Harvard University
Ernst Oberotner	Boston University
Evan Appleton	Boston University
Haiyao Huang	Boston University
Jenhan Tao	University of California, San Diego
Justin Huang	University of California, San Diego
Kevin LeShane	Boston University
Linh Huynh	University of California, Davis
Mona Yousofshahi	Tufts University
Naruemon Pratanwanich	University of Cambridge
Natasa Miskov-Zivanov	Carnegie Mellon University
Neda Hassanpour	Tufts University
Nicholas Roehner	University of Utah
Philipp Boeing	University College London
Prashant Vaidyanathan	Boston University
Ruei-Yang Huang	National Taiwan University
Sara Amin	Tufts University
Sonya Iverson	Boston University
Stephanie Paige	Boston University
Swapnil Bhatia	Boston University
Tyler Wagner	Boston University
Yuan Zhao	University of California, San Diego

IWBDA 2014 SPONSORS

SPECIFICATION

Synberc
Building the Future with Biology.

DESIGN

NSF

WORKFLOW

AUTODESK

TOOL

Hudson
ROBOTICS, INC.

CLASS

TWIST
BIOSCIENCE

ALGORITHM

ACS SyntheticBiology

Raytheon
BBN Technologies

MINRES
TECHNOLOGIES

synberc

Building the Future with Biology.

Synberc is a proud supporter of IWBDA 2014 and the synthetic biology community

qb3

UCSF

VE RI TAS HARVARD

MIT

LELAND STANFORD JUNIOR UNIVERSITY DIE LUFT DER FREIHEIT WEHT 1891

synberc.org

Imagine, Design, and Automate a Better World.

Project Cyborg is a cloud-native platform of design tools, aimed to democratize, explore, and connect emerging design spaces enabled by programmable matter across domains and scales.

Whether it's synthetic biology, nanoparticle design, 3D bio-printing, DNA origami, bio-informed architecture, or self-assembling human-scale manufacturing, Project Cyborg accelerates better design across living systems and beyond.

Project Cyborg is in a restricted beta release. Apply at next@autodesk.com.
Find out more: autodeskresearch.com/groups/nano

Synthetic Biology:

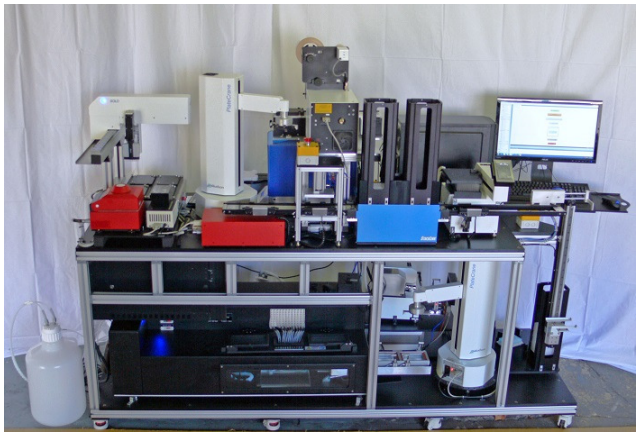
from *Start...* to *Finish.*

Complete Automation of the Entire

Synthetic Biology Process:

Gene Design and Assembly:

- Oligo Synthesis
- Deprotection and Purification
- Normalization
- Pooling
- Gene Assembly



Gene Design and Assembly Workcell

Protein Expression:

- Ligation
- Transcription
- Transformation
- Colony Picking
- Colony to qPCR
- Mini Preps

Visit us at
www.HudsonRobotics.com

ORGANIZING COMMITTEE

executive committee

General Chair – Aaron Adler (BBN Technologies)
Finance Chair: Traci Haddock (Boston University)
Program Committee Chairs - Michal Galdzicki (Arzeda Corporation) and Patrick Cai (University of Edinburgh)
Publication Chairs – Nathan Hillson (JBEI) and Jackie Quinn (Google)
Local Chairs – Traci Haddock (Boston University) and Swati Carr (Boston University)

bio-design automation consortium

Douglas Densmore (Boston University), President
Aaron Adler (BBN Technologies), Vice-President
Traci Haddock (Boston University), Treasurer
Natasa Miskov-Zivanov (Carnegie Mellon University), Clerk

founders

Douglas Densmore (Boston University)
Soha Hassoun (Tufts University)
Marc Riedel (University of Minnesota)

program committee

Michal Galdzicki, Arzeda Corp. (Co-Chair)
Patrick Cai, University of Edinburgh (Co-Chair)
Aaron Adler, Raytheon BBN Technologies
Chris Anderson, University of California, Berkeley
Shota Atsumi, University of California, Davis
Jonathan Babb, Massachusetts Institute of Technology
Jake Beal, Raytheon BBN Technologies
Swapnil Bhatia, Boston University
Kevin Clancy, Thermo Fischer Scientific
Vincent Danos, University of Edinburgh
Barbara Diventura, Heidelberg University
Traci Haddock, Boston University
Soha Hassoun, Tufts University
Eric Klavins, University of Washington
Lan Ma, University of Texas at Dallas
Natasa Miskov-Zivanov, Carnegie Mellon University
Chris Myers, University of Utah
Dimitris Papamichail, The College of New Jersey
Andrew Phillips, Microsoft Research
Cesar Rodriguez, Autodesk
Herbert Sauro, University of Washington
Guy-Bart Stan, Imperial College London
Darko Stefanovic, University of New Mexico
Ilias Tagkopoulos, University of California, Davis

IWBDA 2014 PROGRAM

Wednesday, June 11th

8:00 - 9:00 Breakfast and Registration
9:00 - 9:15 Opening Remarks

Talk Session I: Tools

9:15 - 9:40 *DIVA: More Science, Less DNA construction*
Joanna Chen, Hector Plahar, Nina Stawski, Garima Goyal, Jay Keasling and Nathan Hillson

9:40 - 10:05 *GenomeCarver: harvesting genetic parts from genomes to support biological design automation*
Emily Scher, Yisha Luo, Aaron Berliner, Jackie Quinn, Carlos Olguin and Yizhi Cai

10:05 - 10:30 *Automated Selection Finder (ASF) For Directed Evolution*
Neda Hassanpour, Brad Gaynor, Mona Yousofshahi, Nikhil Nair, and Soha Hassoun

10:30 - 11:00 Coffee Break

Discussion Session I

11:00 - 12:00 Topic : Merging experimentalists and computational efforts in BDA

Poster Session I

12:00 - 12:30 Lunch
12:30 - 13:30 Poster Session

Keynote

13:30 - 14:30 **Tuval Ben-Yehezkel** *Advanced DNA Editing Platforms*.

Talk Session II: Modeling I

14:30 - 14:55 *Metabolic Constraint-Based Refinement of Transcriptional Regulatory Networks*
Sriram Chandrasekaran and Nathan Price

14:55 - 15:20 *Integration of Circuit Design Automation and Genome-scale Modeling*
Linh Huynh, Minseung Kim and Ilias Tagkopoulos

15:20 - 15:45 *Configurable Linear Control of Biochemical Systems*
Tai-Yin Chiu, Ruei-Yang Huang, Hui-Ju Katherine Chiang, Jie-Hong Roland Jiang and Francois Fages

15:45 - 16:15 Coffee Break

Talk Session III: Modeling II

16:15 - 16:40 *A hybrid of multi-omics FBA and Bayesian factor modeling to identify pathway crosstalks*
Claudio Angione, Naruemon Pratanwanich and Pietro Lio

16:40 - 17:05 *Delay modeling in cell signaling and gene regulatory networks*
Natasa Miskov-Zivanov, Chang Sheng Clement Loh, Peter Wei and James Faeder

Evening Activities

18:00 - 19:00 Fenway Park Tour
19:00 - 21:00 Dinner

IWBDA 2014 PROGRAM

Thursday, June 12th

8:00 - 9:00 Breakfast and Registration

Talk Session IV: Circuit Design

9:00 - 9:25 *Precision Design of Expression from RNA Replicons*
Jacob Beal, Tyler Wagner, Tasuku Kitada, Andrey Krivoy, Odisse Azizgolshani, Jordan Moberg Parker, Douglas Densmore and Ron Weiss

9:25 - 9:50 *Modular design and implementation of eukaryotic synthetic gene circuits*
Mario Andrea Marchisio

9:50 - 10:15 *Automatic Enumeration and Discrimination of Model Phenotypes With Application to Synthetic Gene Oscillators*
Jason Lomnitz and Michael Savageau

10:15 - 10:45 Coffee Break

Discussion Session II

10:45 - 11:45 Topic: BDA training and curricula needs in academia and industry

Poster Session II

11:45 - 12:15 Lunch

12:15 - 13:15 Poster Session

Keynote

13:15 - 14:15 **Orit Shaer** *Reality-Based Interaction for Bio- Design*.

Talk Session V: Automation

14:15 - 14:40 *Better Scheduling Software and User Interface for Liquid-Handling Robots*
Benjamin Schreck, Jonathan Babb and Ron Weiss

14:40 - 15:05 *SMT-based Strategies for Biodesign Automation*
Boyan Yordanov and Andrew Phillips

15:05 - 15:30 *Exploration of Design Principles and an End-to-End Workflow for Synthetic Biology using a Combinational Logic Circuit Library*
Swati Carr, Calin Belta and Douglas Densmore

15:30 - 16:00 Coffee Break

Talk Session VI: Algorithms

16:00 - 16:25 *Nonlinear Biochemical Signal Processing via Noise Propagation*
Kyung Kim, Hong Qian and Hebert Sauro

16:25 - 16:50 *An Aspect Oriented Design and Modelling Framework for Synthetic Biology*
Philipp Boeing, Darren Nesbeth, Anthony Finkelstein and Chris Barnes

16:50 - 17:15 *Towards Rule-based Knowledge-Based Systems for Synthetic Biology*
Ernst Oberortner, Audrey Lewis and Douglas Densmore

Closing remarks

17:15 - 17:30 Closing remarks

KEYNOTE PRESENTATION

Tuval Ben-Yehezkel

Advanced DNA Editing Platforms

Our ability to engineer biological systems depends, to a large extent, on our ability to physically write the DNA code that programs them. In this talk I will outline recent developments in biochemistry and automation technologies that advance our ability to deliver designer DNA libraries for advanced synthetic biology projects.

Dr Tuval Ben-Yehezkel performed his PhD and post doc work at the Weizmann institute of science in Israel under the supervision of Prof. Ehud Shapiro in the field of synthetic biology. Specifically, he focused on developing methods for rapid writing of genetic material through the implementation of innovative biochemistry and automation technology and applying them to various biological problems. He is now a visiting scientist at the Weizmann institute of science and has recently founded a start-up company focused on applying synthetic biology technology for the rational design and construction of synthetic viruses aimed to function as live-attenuated vaccines.

KEYNOTE PRESENTATION

Orit Shaer

Reality-Based Interaction for Bio-Design

Synthetic biology requires a multidisciplinary, collaborative design environment in order to engineer the complex biological systems of the future. Applying advances in Human Computer Interaction to bio-design tools could potentially enhance innovation and discovery in synthetic biology.

Over the past two decades, Human-Computer Interaction (HCI) research has generated a broad range of interaction styles that move beyond the desktop into new physical and social contexts. Key areas of innovation in this respect include tabletop, tangible, and embodied user interfaces. These interaction styles leverage users' existing knowledge and skills of interaction with the real non-digital world, thus are often referred to as *Reality-Based Interfaces*. By drawing upon existing skills, reality-based interfaces offer the promise of a natural, intuitive, and often, collaborative form of interaction.

In this talk I will present a collection of software tools for bio design, which, utilize reality-based interaction techniques. We developed theses software tools to address specific technical synthetic biology challenges while simultaneously advancing the way in which users interact with computing environments. Through these case studies, I will highlight what design factors are important for developing reality-based interfaces for bio design that enhance discovery and innovation.

Orit Schaer is the Clare Boothe Luce Assistant Professor of Computer Science and Media Arts and Sciences at Wellesley College. She directs the Wellesley College HCI Lab. Her research in Human-Computer Interaction (HCI) focuses on 3D, tangible, tabletop, and mobile interaction. a user interface for genomic research, and the development of computational tools for enhancing innovation in bio-design. Dr Shaer is a recipient of several National Science Foundation and industry awards including the prestigious NSF CAREER Award, Agilent Technologies Research Award, and Google App Engine Education Award. She received her PhD and MSc in Computer Science from Tufts University. She has been a research fellow in the Design Machine Group at the University of Washington and in the University College London Interaction Center.

ABSTRACTS – TABLE OF CONTENTS

Oral Presentations

ABSTRACTS – TABLE OF CONTENTS

Poster Presentations

DIVA: More Science, Less DNA construction

Joanna Chen
Joint BioEnergy Institute
Emeryville, CA
joannachen@lbl.gov

Hector A. Plahar
Joint BioEnergy Institute
Emeryville, CA
haplahar@lbl.gov

Nina Stawski
Joint BioEnergy Institute
Emeryville, CA
me@ninastawski.com

Garima Goyal
Joint BioEnergy Institute
Emeryville, CA
ggoyal@lbl.gov

Jay D. Keasling
Joint BioEnergy Institute
Emeryville, CA
jdkeasling@lbl.gov

Nathan J. Hillson
Joint BioEnergy Institute
Emeryville, CA
njhillson@lbl.gov

ABSTRACT
DNA construction is vital to a broad range of biological research, yet it is predominantly an inefficient diversion from researchers' core research goals and expertise. We have developed a Design, Implementation, and Verification Automation (DIVA) platform to liberate researchers from building DNA, enabling them to instead focus on designing and testing their experiments of interest. DIVA's web interface allows researchers to design DNA constructs using visual biological computer-aided design tools (such as DeviceEditor [1]) and existing parts from ICE registries [2], submit their designs to a central construction queue, track DNA construction as it progresses, and then (in a few weeks) receive notice that their sequence-verified constructs have been completed. A small number of dedicated DIVA staff, who are responsible for DNA construction, use the DIVA web interface to evaluate submitted designs for feasibility, and design assembly protocols using DNA assembly design automation software (such as j5 [3]), and will be able to manage physical samples throughout the construction process with sample tracking software, and capture success and failure rates and pipeline performance metrics. With DNA construction occurring in a centralized location, DIVA staff can aggregate multiple independent construction tasks into the same multi-well plates to take advantage of laboratory automation devices and tools such as PR-PR [4], as well as leverage DNA synthesis and high-throughput next-gen sequencing capabilities. An initial version of the DIVA platform is currently being used at the Joint BioEnergy Institute (JBEI) and is under continued development. We are also working closely with the Joint Genome Institute (JGI) towards deploying DIVA to better serve their user community.

Categories and Subject Descriptors
J.6 [**Computer-Aided Engineering**]: *computer-aided design*.

General Terms
Design, Verification.

Keywords
BioCAD, DNA construction automation.

1. INTRODUCTION
A number of software tools have been developed to aid researchers design and build DNA constructs. The DIVA platform seeks to integrate these tools into a seamless and useful workflow to further save researchers time and effort.

2. THE DIVA WEB INTERFACE
Upon logging in, researchers see a dashboard displaying designs that require their attention, as well as their other recent designs and projects. The right column displays DIVA performance trends and a recent activity feed. The header provides access to all design and project listings.

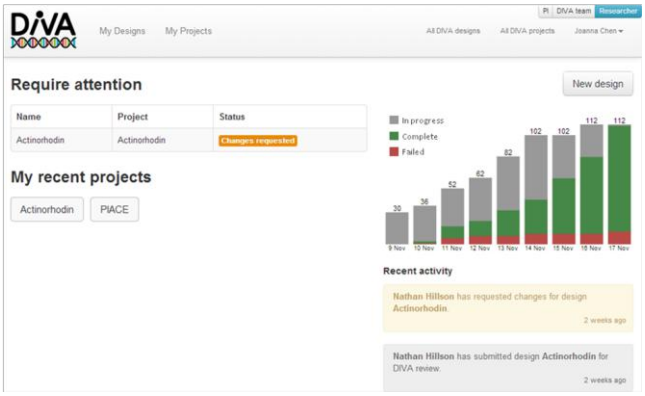


Figure 1. DIVA interface - dashboard.

Clicking the "New design" button opens DeviceEditor in DIVA, where researchers can visually design their DNA constructs. Researchers can copy annotated sequences from ICE via VectorEditor and paste them into DeviceEditor. Researchers can also run j5 from DeviceEditor to check that their designs will produce the desired construct(s). Once researchers are satisfied with their designs, they will submit them to the DIVA team for technical feasibility review.

The DIVA team then runs j5 to design the assembly protocol for constructing the DNA designs specified by the researchers. The DIVA team assesses the designs to see if they can be build subject to resource or size constraints. Designs may be approved for construction or sent back to the researchers for revision.

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

Once designs have been approved by the DIVA team, researchers can submit them into the DIVA queue for construction. Since many researchers submit designs to the same queue, the DIVA team may aggregate several designs into one construction process to take full advantage of laboratory automation. The DIVA team may use j5 to condense multiple designs into one assembly protocol and generate a PR-PR script to run the reactions using multi-well plates on a liquid handling robot. While designs are being constructed, researchers can to track the status of their constructs through the DIVA interface.

PIs have additional capabilities to manage projects, such as adding and removing project members and approving designs for construction.

3. ACKNOWLEDGMENTS

We thank Will Holtz, Gregory Linshiz, and JGI staff, including Len Pennacchio, Jan-Fang Cheng, Sam Deutsch, Sarah Richardson, Matthew Bendall, Atif Shahab, Miranda Harmon-Smith, and Christa Pennacchio, for collaborative discussions and efforts towards deploying the DIVA for the JGI DNA Synthesis Program user community.

This work conducted by the Joint BioEnergy Institute was supported by the Office of Science, Office of Biological and Environmental Research, of the U.S. Department of Energy under Contract No. DEAC02-05CH11231.

4. REFERENCES

[1] Chen J, Densmore D, Ham TS, Keasling JD, Hillson NJ. (2012) DeviceEditor visual biological CAD canvas. *J Biol Eng 6*(1).

[2] Ham TS, Dmytriv Z, Plahar H, Chen J, Hillson NJ, Keasling JD. (2012) Design, implementation and practice of JBEI-ICE: an open source biological part registry platform and tools. *Nucleic Acids Res 40*(18).

[3] Hillson NJ, Rosengarten RD, Keasling JD. (2012) j5 DNA assembly design automation software. *ACS Synthetic Biology 1*(1), 14-21.

[4] Linshiz G, Stawski N, Poust S, Bi C, Keasling JD, Hillson NJ. (2013) PaR-PaR laboratory automation platform. *ACS Synthetic Biology 2*(5), 216-222.

GenomeCarver: harvesting genetic parts from genomes to support biological design automation

Emily Scher*
School of Informatics
University of Edinburgh
Edinburgh EH9 3JR, UK

Yisha Luo*
School of Biological Sciences
University of Edinburgh
Edinburgh EH9 3JR, UK

Aaron Berliner
Autodesk Research
San Francisco, California
94111, USA

Jacqueline Quinn
Autodesk Research
San Francisco, California
94111, USA

Carlos Olguin
Autodesk Research
San Francisco, California
94111, USA

Dr. Yizhi Cai†
School of Biological Sciences
University of Edinburgh
Edinburgh EH9 3JR, UK

ABSTRACT

Concept

The advance of genome sequencing and annotation has provided a "gold mine" of genetic parts, which all synthetic biologists wish to include in their toolbox of parts with which to build synthetic biological systems. The currently available computer assisted design systems (CADs) focus heavily, if not exclusively, on composing biological systems using genetic parts [3][7][9], however, how a user obtains parts in the first place remains an open question. To make matters worse, there are a few dozen part standards being proposed and used in the synthetic biology community (104 RFCs on part standard as of today). Even though one can extract a few parts from the genome manually, there is no software to ensure the standard compatibility of parts, and it is also very difficult to scale up the design of parts.

With these problems in mind, we present GenomeCarver, a computational tool for the harvesting and packaging of biological parts from model genomes. GenomeCarver interfaces with various genomes, identifies regions of interest according to user specification (*e.g.,* promoters, open reading frames and terminators) and extraction rules (*e.g.,* a promoter is defined as 500bp upstream of the ATG start codon or last gene boundary, which comes shorter), extracts corresponding DNA sequences from the genome feature files (GFFs), checks the sequence's compatibility with the selected standard (*e.g.,* whether the given sequence includes the forbidden restriction sites of certain parts standards), and finally outputs optimized primer sequences to amplify the parts from genomic DNA, adding necessary flanking sequences to standardize the parts.

Through its compatibility with multiple genomes and multiple parts standards, GenomeCarver bridges the fields of systems biology and synthetic biology, and greatly enriches synthetic biologists' design toolbox. It complements many parts-based design tools which currently exist by supporting the Synthetic Biology Open Language standard [6].

Implementation

GenomeCarver can be accessed as an application built on Autodesk's Project Cyborg (`http://autodeskresearch.com/projects/cyborg`). Project Cyborg is a cloud-based platform for computational tools in the life sciences and programmable matter space, supporting design and engineering across domains and scales. Cyborg enables elastic computing through a node framework that natively provides support for simulation, optimization, and visualization. Being built on Cyborg, GenomeCarver is comprised of nodes for each step of the workflow connected to form a cohesive user experience that guides the user through the tool.

GenomeCarver currently supports three model organisms: yeast *Saccharomyces cerevisiae*, bacterial *Escherichia coli*, and plant *Arabidopsis thaliana*. However, GenomeCarver is flexible enough to be extended to interface a variety of organisms, which we plan to do in the near future. Similarly, GenomeCarver currently supports a finite number of mainstream parts standards such as the BioBrick 1.0[1] and yeast Golden Gate standards[2], but new standards could easily be incorporated. In a future implementation, we even plan to allow users to import their own, custom standards. While it's interfacing with multiple genomes and standards has made GenomeCarver flexible, it's being built on top of Cyborg further's the tool's flexibility, as GenomeCarver will be able to be used in conjunction with the other tools currently being developed on the same platform.

Figure 1 shows the application's workflow. First, a user chooses a genome, a category and the loci of the part. For instance, a user may choose the promoter of *Gal* loci from *Saccharomyces cerevisiae*. Optionally, the user can then define the preferred promoter and terminator lengths, or specify that they would like gene boundaries to be ignored. The default maximum promoter length is 500 base pairs, and the default maximum terminator length is 200 base pairs. If the user does not specify that gene boundaries should be ignored, then GenomeCarver will identify a gene's promoter as the upstream (5' to 3') sequence of a maximum length of 500 (or the specified maximum length) which does not overlap another gene. It will identify the terminator as the following sequence of a maximum length of 200 bases which does not overlap another gene. GenomeCarver then returns the specified sequence(s). The user can then assign the sequence(s) to a standard using another drop down menu. Once selected, the sequence will then be checked

*These authors contributed equally to this work.
†Corresponding author. E-mail:yizhi.cai@ed.ac.uk

IWBDA 2014, June 11-12, 2014, Boston, Massachusetts, USA.

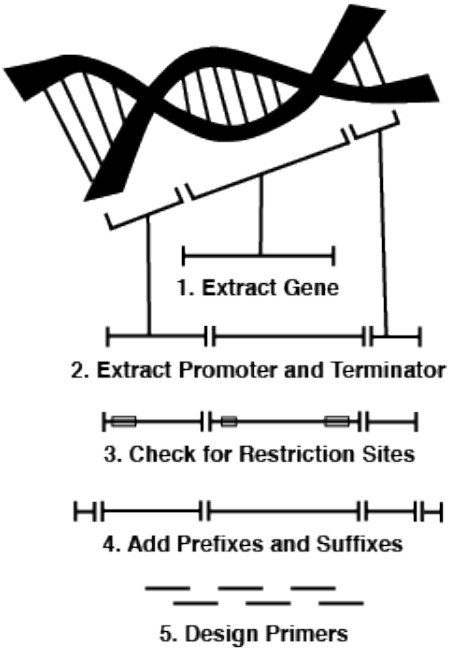


Figure 1: The workflow of GenomeCarver

for restriction sites, returning a warning if an incompatible restriction site is found. The part will then be packaged by adding the appropriate prefix and suffix. GenomeCarver then allows the packaged parts to be exported in CSV and SBOL formats [6].

Experimental verification
GenomeCarver has been used extensively in several labs in the USA, UK and China to systematically design thousands of yeast parts of each category conforming to the yeast Golden Gate standard. We used the designed primers to amplify parts from genomic DNA in a high throughput fashion, cloning the parts onto Topo vector backbone, and sequence verifying them all (data not shown in the abstract). We also demonstrated high efficient assembly of various genetic switches using these parts and standard Golden Gate reaction, and transformed these assembled switches into yeast for functional assays. Most recently, GenomeCarver has been used to design all the 6000 yeast promoters and 6000 yeast terminators, which demonstrates that we can scale up the design automation easily.

Future plan
In the next version of GenomeCarver, we are planing to include additional genomes, such as mammalian ones, as well as to support user-customized standards. Batch design functionality will also be developed to support large projects, such as BioFab (http://biofab.synberc.org/) type projects for various genomes. We will also develop better primer design strategies [8] to maximize the parts amplification success rate. We are also planning to support codon optimization for gene parts, so that a user can carve out a gene from one species and codon optimize it for another species, and GenomeCarve will output oligonucleotides for *de novo* DNA synthesis. Finally, a better integration with existing parts-based design tools will be needed for a better user design experience.

Conclusion
GenomeCarver has been built to fill a gap left by existing Synthetic Biology computational tools. It allows users to extract parts directly from genomes, and to package them into standardized formats for parts synthesis. We have used this tool to design over 12,000 parts, and constructed and verified several hundred of them. This tool, along with the parts repository we created using it, will be a useful and important addition to the synthetic biology community.

Acknowledgement
ES is supported by an Autodesk research fellowship. The project is funded by an Edinburgh Chancellor's Fellowship to YC. We thank Drs. Jef D. Boeke (New York University, USA) and Junbiao Dai (Tsinghua University, China) for helpful discussions to initiate the project.

1. REFERENCES

[1] Technical report, August 2005.
[2] Technical report, Johns Hopkins University, October 2012.
[3] Yizhi Cai, Brian Hartnett, Claes Gustafsson, and Jean Peccoud. A syntactic model to design and verify synthetic genetic constructs derived from standard biological parts. *Bioinformatics*, 2007.
[4] Carola Engler, Ramona Gruetzner, Romy Kandzia, and Sylvestre Marillonnet. Golden gate shuffling: A one-pot dna shuffling method based on type iis restriction enzymes. *PLoS ONE*, 4(5):e5553, 05 2009.
[5] Carola Engler, Romy Kandzia, and Sylvestre Marillonnet. A one pot, one step, precision cloning method with high throughput capability. *PLoS ONE*, 3(11):e3647, 11 2008.
[6] Michal Galdzicki, Cesar Rodriguez, Deepak Chandran, Herbert M. Sauro, and John H. Gennari. Standard biological parts knowledgebase. *PLoS ONE*, 6(2):e17005, 02 2011.
[7] Nathan J. Hillson, Rafael D. Rosengarten, and Jay D. Keasling. j5 dna assembly design automation software. *ACS Synthetic Biology*, 1(1):14–21, 2012.
[8] Andreas Untergasser, Ioana Cutcutache, Triinu Koressaar, Jian Ye, Brant C. Faircloth, Maido Remm, and Steven G. Rozen. Primer3âĂŤnew capabilities and interfaces. *Nucleic Acids Research*, 40(15):e115, 2012.
[9] Bing Xia, Swapnil Bhatia, Ben Bubenheim, Maisam Dadgar, Douglas Densmore, and J. Christopher Anderson. Chapter five - developer's and user's guide to clotho v2.0: A software platform for the creation of synthetic biological systems. In Christopher Voigt, editor, *Synthetic Biology, Part B Computer Aided Design and DNA Assembly*, volume 498 of *Methods in Enzymology*, pages 97 – 135. Academic Press, 2011.

Automated Selection Finder (AFS) For Directed Evolution

Neda Hassanpour
Department of Computer Science
Tufts University
Neda.hassanpour@tufts.edu

Brad Gaynor
Department of Electrical and Computer Engineering
Tufts University
bgaynor@draper.com

Mona Yousofshahi
Department of Computer Science
Tufts University
Mona.Yousofshahi@tufts.edu

Nikhil U. Nair
Department of Chemical & Biological Engineering
Tufts University
nikhil.nair@tufts.edu

Soha Hassoun
Department of Computer Science
Tufts University
soha@cs.tufts.edu

ABSTRACT
We describe in this paper a computational method, Automated Selection Finder (ASF), for constructing a selection pathway from a desired enzymatic product to a cellular host. ASF can be beneficial for identifying appropriate high throughput selections for enzyme engineering purposes.

1. INTRODUCTION
Directed evolution, sometimes described as premeditated synthetic evolution, has been used to engineer potent therapeutic agents, novel vaccines, potent antibodies, and enzymes that produce novel bioactive compounds and fine chemicals (see [1] for a recent review). Directed evolution of enzymes consists of an iterative process of creating mutant libraries and choosing desired phenotypes through screening or selection until the enzymatic activity reaches the desired goal. The biggest challenge in directed enzyme evolution is to identify an appropriate high-throughput screen or selection to isolate the variant(s) with the desired property. Traditionally, screening is accomplished through colorimetric or fluorometric assays, which have low to medium throughput. Selection, however, is more desirable because it offers far higher throughput capabilities. One example is selection for engineered xylose reductases (XR) enzyme, which leads to the overproduction of xylitol through enzymatic reduction of D-xylose. The selective pressure enriches for strains that can overproduce this chemical via an engineered a pathway in the host *E. coli* that forces xylitol to become an essential carbon source [2]. There are currently no known techniques for identifying appropriate high throughput selections for enzyme engineering purposes. That is, given an enzymatic reaction, the challenge is to automatically identify a consumption pathway, from the desired product to a metabolite within the host cell. Developing automated techniques to identify selection mechanisms can significantly expedite experimental practices in directed evolution.

We describe in this abstract a computational method, Automated Selection Finder (ASF), for constructing a selection pathway from a desired enzymatic product to a cellular host. Additionally, ASF identifies knockout targets to ensure that the added pathway becomes essential for producing cellular biomass.

2. METHODS
The ASF algorithm consists of the following four steps. First, we identify possible pathways from the desired product to a metabolite in the cellular host (blue path in Fig. 1). For each such selection pathway, we also identify supporting pathways (red path in Fig. 1) from the host to the reactant-side cofactors along the selection pathway. The supporting pathways ensure the viability of the selection pathway. To identify the selection pathways, we utilize a modified version of probabilistic pathway construction algorithm (*ProbPath*) [3], which was originally designed to identify synthesis pathways by constructing from the KEGG database a series of reactions from a metabolite within the host to the desired product. *ProbPath* was shown effective in identifying non-native synthesis pathways for a given product metabolite with production yield comparable to that of limited-in-depth exhaustive search methods [3]. Second, we identify knockout targets to maximize the consumption flux associated with the selection pathway. Third, the flux associated with the desired product is computed using Flux Balance Analysis. Finally, the candidate pathways are then scored based on yield, pathway length, and number of required knockouts. Higher yielding shorter pathways with the smallest number of knockouts are preferable.

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

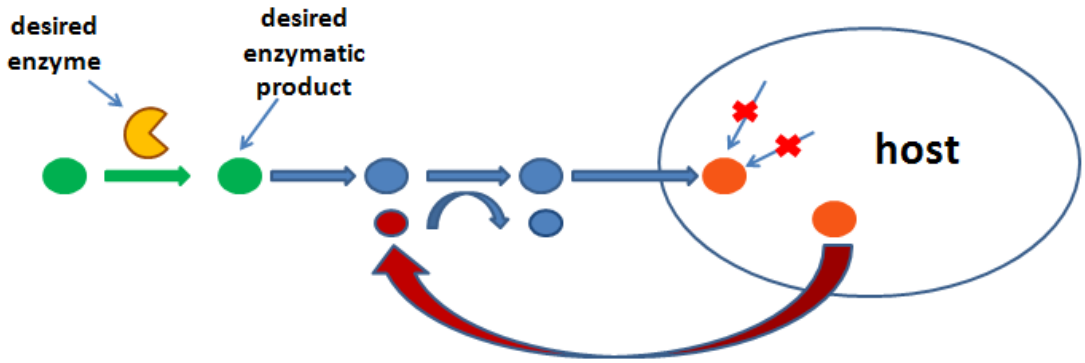


Fig. 1. Consumption selection pathway (blue) from the desired enzymatic product to a metabolite within the host, and supporting pathway (red) from the host to the cofactors on the reactant side of the reactions along the selection pathway. The "x" marks a knockdown within the host.

3. RESULTS

We have applied ASF to the engineered XR with desired enzymatic product xylitol and *E. coli* as the host. Using 80 iterations of *ProbPath*, and assuming flux bounds of 0 to 1000 (mol/sec), we identified 16 pathways with varying characteristics.

Out of the identified pathways 80% provided a yield of almost 1000 (mol/sec) and 20% provided a yield ranging between 500 and 900 (mol/sec). The shortest path identified was a single reaction from xylitol to D-xylulose with path length one. On average, the selection pathway length was 5.56. The number of knockouts ranged between 1 and 4. Overall, the best pathway was from xylitol to D-xylulose as the shortest path with length 1 and maximum yield equal to 1000 (mol/sec).

4. REFERENCES

[1] Nair N. U. and Zhao H. M. 2009. *Improving Protein Function by Directed Evolution.* The Metabolic Pathway Engineering Handbook, C. D. Smolke (editor), CRC Press.

[2] Nair N. U. and Zhao H. M. 2008. *Evolution in Reverse: Engineering a D-Xylose-Specific Xylose Reductase.* ChemBioChem, 9, 1213–1215.

[3] Yousofshahi, M., Lee, K. and Hassoun S. 2011. *Probabilistic Pathway Construction.* Metab. Eng, 13, 435–444.

Metabolic Constraint-Based Refinement of Transcriptional Regulatory Networks

Sriram Chandrasekaran [1, 2, ★] & Nathan D. Price [1, 2]

[1]Institute for Systems Biology, 401 Terry Ave N, Seattle, Washington
[2]Center for Biophysics and Computational Biology, University of Illinois, Urbana-Champaign
email: chandrasekaran@fas.harvard.edu, nprice@systemsbiology.org

ABSTRACT

Cellular networks, such as metabolic and transcriptional regulatory networks (TRNs), do not operate independently but work together in unison to determine cellular phenotypes. The architecture of individual networks constrains the topology of other networks. Hence, it is critical to study network components and interactions in the context of the entire cell. Current efforts to reconstruct TRNs focus primarily on proximal data such as gene co-expression and transcription factor binding. While such approaches enable rapid reconstruction of TRNs, the overwhelming combinatorics of possible networks limits identification of mechanistic regulatory interactions. Utilizing growth phenotypes and systems-level constraints to inform regulatory network reconstruction is an unmet challenge.

We present our approach *Gene Expression and Metabolism Integrated for Network Inference* (GEMINI) that links a compendium of candidate regulatory interactions with the metabolic network to predict their systems-level effect on growth phenotypes. We then compare predictions with experimental phenotype data to select phenotype-consistent regulatory interactions.

We applied GEMINI to create an integrated metabolic-regulatory network model for *Saccharomyces cerevisiae* involving 25,000 regulatory interactions controlling 1597 metabolic reactions. The model quantitatively predicts TF knockout phenotypes and revealed condition-specific regulatory mechanisms. Understanding how the networks function together in a cell will pave the way for synthetic biology and has a wide-range of applications in biotechnology, drug discovery and diagnostics. The algorithm and associated data are available at https://sourceforge.net/projects/gemini-data/

Keywords

Systems Biology, Metabolic Networks, Transcriptional Regulatory Networks, Synthetic Biology

★ Present Address: FAS Center for Systems Biology, Harvard Society of Fellows, Harvard University

1. INTRODUCTION

The inference of transcriptional regulatory networks (TRNs) from high-throughput data is a central challenge in systems biology. The overwhelming number of possible regulatory interactions between thousands of genes and transcriptional regulators in a cell—combined with the complex and dynamic nature of these interactions—limits the success of current approaches to infer TRNs [5].

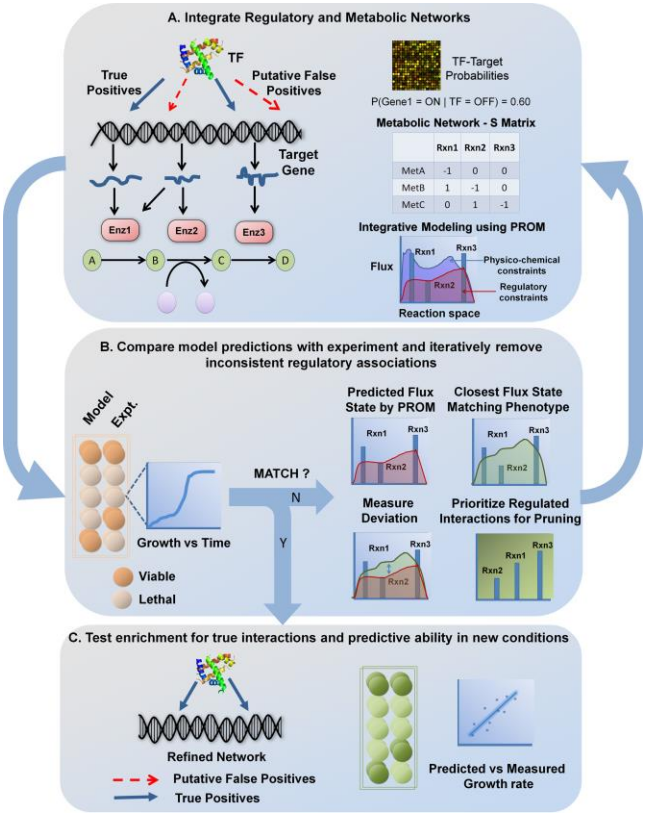


Figure 1: Process of identifying phenotype-consistent interactions using GEMINI A. High-throughput interaction data were mapped onto a metabolic network using PROM and phenotypic consequences of these interactions were predicted. **B.** Interactions that lead to inconsistencies between model predictions and experiments were identified and removed. This was achieved by comparing the flux state predicted by PROM for the TF knockout with the closest flux state that represented the measured growth phenotype. **C.** The final network that

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

matched the phenotype was tested based on its ability to retain known interactions and predict growth in new conditions.

We hypothesized that integrating regulatory interactions with metabolic networks would make it possible to more directly connect the regulatory interactions with their downstream phenotype, and thus allow us to use a broader range of data for network curation. The success of this integration would then allow the utilization of large-scale phenotypic data, which are commonly used to curate metabolic networks, to also refine regulatory interactions. To enable the concurrent analysis of transcriptional regulation and metabolism, we recently developed the Probabilistic Regulation of Metabolism (PROM) approach for integrating biochemical networks with TRNs in an automated fashion [1]. PROM takes in a genome-scale metabolic network model, a regulatory network structure consisting of TFs and their targets, and gene expression data across different conditions, as inputs to predict the phenotypic outcome of transcriptional perturbations. PROM solves the forward problem of combining disparate networks to predict phenotype (e.g., flux and growth rates).

In the work described herein, we iteratively use PROM to aid in solving the more challenging inverse problem—guiding TRN structure prediction using the metabolic network and the emergent phenotype measurements. This new approach, Gene Expression and Metabolism Integrated for Network Inference (GEMINI)[2], produces a regulatory network state that is simultaneously consistent with observed gene knockout phenotypes, gene expression data, and the corresponding metabolic network state.

2. RESULTS

The initial integrated network model was formed by compiling the yeast regulatory interactions from the Yeastract database [6] and was integrated with the yeast metabolic network (composed of 1597 reactions and 901 genes) and gene expression data [3] (consisting of 904 expression arrays in 435 conditions) using PROM. The integrated model contains 31,075 interactions between 179 TFs and 863 metabolic genes. GEMINI performed *in silico* knockouts of each TF in the model and compared the predictions (i.e., lethal or viable) to data from growth viability assays in glucose minimal media. Running GEMINI on this network eliminated over 9,000 phenotype-inconsistent interactions and results in a final network containing 22,059 phenotype-consistent regulatory interactions. In comparison to the original Yeastract network, we found the final integrated network built using GEMINI to be highly enriched (p-value = 10^{-172}, hyper-geometric test) for validated gold-standard interactions; this result suggests that GEMINI preferentially removed low-confidence interactions. These results were robust to the chosen growth conditions – glucose, galactose, glycerol and ethanol minimal media all led to significant enrichment of gold-standard interactions. Similar results were obtained using regulatory networks from different sources (binding, motif-based and expression-based inference), and using different metrics to prioritize interactions. GEMINI can also be effectively integrated with de-novo network inference approaches such as EGRIN to create integrated network models.

Further, the distribution of inconsistencies in the regulatory network was exponential suggesting that a few TFs led to most of the inconsistencies. In contrast, no such biases were found in the metabolic network architecture. Importantly, we observed that the refined network had a greater consistency with growth phenotype data in new conditions than the original network. Thus, by learning only on glucose minimal medium, the network model had greater correlation with growth rate measurements in galactose minimal medium (correlation of 0.47, p-value = 10^{-7} vs. a correlation of 0.2, p-value = 0.04 for the original unrefined Yeastract model) and in urea minimal medium (correlation of 0.62, p-value = 10^{-14} vs. a correlation of 0.22, p-value = 0.02; data from Fendt *et al*. [4]).

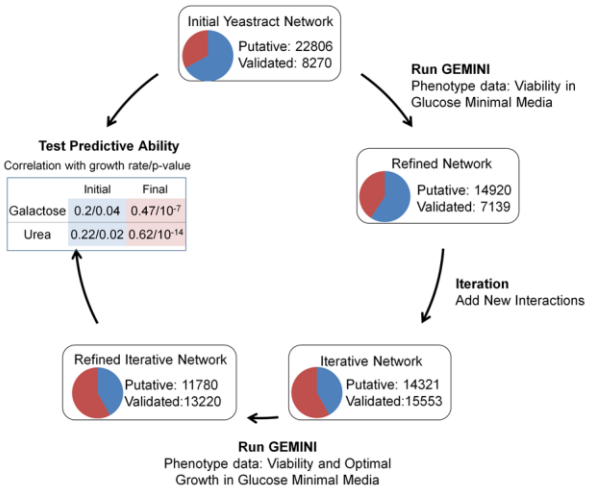


Figure 2: Iterative Approach for Network Refinement and Phenotype Prediction. By using an iterative approach, we increased the comprehensiveness of the integrated network model by adding new interactions and iteratively refining the model using GEMINI. This process enriched the fraction of validated interactions in the network (shown in red) and improved the predictive ability of the integrated network model.

3. REFERENCES

[1] Chandrasekaran, S. and Price, N.D., 2010. Probabilistic integrative modeling of genome-scale metabolic and regulatory networks in Escherichia coli and Mycobacterium tuberculosis. *Proc Natl Acad Sci U S A 107*, 41 (Oct 12), 17845-17850.

[2] Chandrasekaran, S. and Price, N.D., 2013. Metabolic Constraint-Based Refinement of Transcriptional Regulatory Networks. *PLoS computational biology 9*, 12, e1003370.

[3] Faith, J.J., et al., 2008. Many Microbe Microarrays Database: uniformly normalized Affymetrix compendia with structured experimental metadata. *Nucleic Acids Res 36*, Database issue (Jan), D866-870.

[4] Fendt, S.M., et al., 2010. Unraveling condition-dependent networks of transcription factors that control metabolic pathway activity in yeast. *Mol Syst Biol 6*(Nov 30), 432.

[5] Marbach, D., et al., 2012. Wisdom of crowds for robust gene network inference. *Nat Methods 9*, 8 (Jul 15), 796-804.

[6] Teixeira, M.C., et al., 2006. The YEASTRACT database: a tool for the analysis of transcription regulatory associations in Saccharomyces cerevisiae. *Nucleic Acids Res 34*, Database issue (Jan 1), D446-451.

Integration of Circuit Design Automation and Genome-scale Modeling

Linh Huynh, Minseung Kim and Ilias Tagkopoulos
Department of Computer Science
& UC Davis Genome Center
University of California, Davis
{huynh,msgkim,itagkopoulos}@ucdavis.edu

1. INTRODUCTION

The design of a functional component that is part of a larger, interconnected ensemble, requires the following fundamental principles: First, the availability of characterized fundamental blocks, may it be transistors and capacitors for electronic design or promoters and coding regions for biological engineering, that can be assembled together into a functional entity. Second, the development of predictive models that are accurate enough to capture the dynamic behavior of the design component and its effect at a systems-level. Third, access to optimization tools that can provide optimal solutions, given user-defined constraints and objective functions, by utilizing the former (parts and model predictions) as its inputs. In this abstract, we present our work towards an efficient circuit optimization strategy and a data-driven, genome-scale host model that can be integrated to any circuit design platform. The resulting framework accounts for host-related and secondary effects, which are generally ignored but can have substantial effect on the host and circuit behavior.

	Running time (seconds)	
Design	DP	BB
`2-cascade`	1.8e-1	2.0e-2
`3-cascade`	2.1e-1	2.0e-2
`4-cascade`	2.5e-1	4.0e-2
`band-detector`	3.4e-1	8.0e-2
`feed-forward`	6.3e-1	7.0e-2
`2-not-and`	6.4e-1	4.0e-2
`3-input-and`	2.3	1.2e-1
`3-not-and`	3.6e1	1.5e-1
`2-to-1-mux`	1.1e2	1.6e-1
`D1`	1.2e3	9.8e-1
`D2`	2.0e3	4.5e-1

Table 1: A comparison between the running time of the dynamic programming (DP) approach [1] and the branch and bound (BB) approach.

2. METHODS AND RESULTS

2.1 An efficient optimization method with optimality guaranty

The carriers of information in gene circuits are chemical molecules within the cell (broadcast) instead of electrons within an isolated wire (unicast) as it is the case in electrical circuits. This adds two more necessary constraints for gene circuit design, namely the absence of cross-talk effects and connection compatibility between sub-circuits (i.e. the output molecules of a sub-circuit match the input molecules of all connected downstream sub-circuits). These constraints make the problem of selecting parts/modules to build a circuit become very difficult. Recently, an exact approach [1] was introduced to complement previous heuristic efforts [2] for that selection problem. This approach is based on a dynamic programming paradigm to explore the solution space by enumerating all sub-solutions, albeit at a large computational cost that can be prohibiting, in very large part database and circuit sizes. To address this, we devised a branch-and-bound method that estimated the bound of the solution cost (i.e. the bound on the number of parts and modules used in a solution) by relaxing the constraints of cross-talk absence and connection compatibility. With this bound information, we repeat the search for a complete solution (i.e. a solution that satisfies both the constraints) of a cost value from the lower bound to the upper bound until such a complete solution is found. By that way, we can skip the enumeration of all sub-solutions, which results in significant computational performance improvement. In a benchmark with 11 circuits that span several functional domains and a part library with 75 parts and 271 experimentally constructed modules, this method resulted in a remarkable improvement in running time (see table 1) when compared to the original approach in [1].

2.2 A multi-layer, genome-scale model for phenotypic predictions

The simulation of a host that has been genetically engineered is an important step towards design automation. For this reason, we developed an integrated genome-scale model [3] for phenotypic predictions of natural and engineering *E. coli* strains in several laboratory environments (Figure 1). We first constructed a normalized dataset that contains the expression of 4189 genes in 2262 conditions, including data for 31 strains and over 15 different media. We then created an integrative model that contains three submodels that bridge the transcriptional, signal transduction and metabolic layers. This model covers 3704 regulatory interactions, 151 instances of signal transduction systems and 2251 metabolic reactions. Parameters in the transcriptional sub-model were determined by fitting the gene expression level of 328 transcription factors over four sets of constraints (phenomenological, capacity, environmental and

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

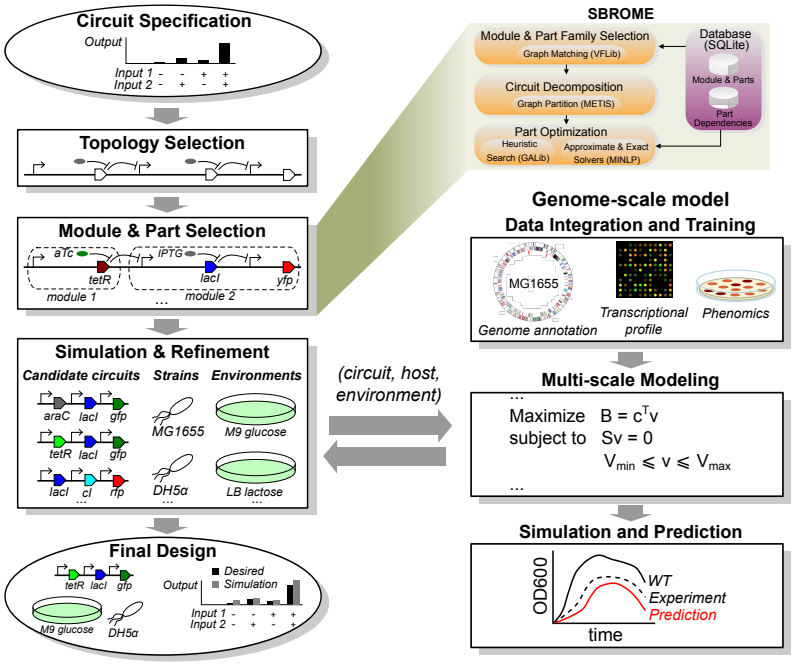


Figure 1: An overview on a computational framework to design and simulate synthetic gene circuits

genetic constraints). The integrated model was evaluated by performing cross-validation on the various datasets for growth and gene expression prediction, as well as predicting de novo experimentally measured data on the growth rate of 10 single-gene knock-outs for *E. coli* strains over different environments (28 genotype-phenotype combinations in total). Results show that our model can predict growth rates with 0.6 to 0.8 pearson correlation coefficient between the experimentally measured and computationally-derived predictions, which is significantly higher than M models and on par to other ME models so far [4]. Furthermore, the constructed model can sense environmental changes and translated them to changes in gene expression and growth, which is a significant step forward for bioengineering efforts.

2.3 Integration of a computer-aided design optimization platform with a genome-scale simulator

As shown in Figure 1, a list of top-ranked candidate circuits from the optimization framework, act as inputs to the genome-scale simulator, in order to predict the dynamic behavior within a specific host strain and environmental conditions. All possible triplets of circuit, host strain and environmental conditions are considered and their information is used to update the transcriptional sub-models by modifying the gene regulatory network and signal transduction sub-models by adding/removing related equations. Additionally, information about the environmental conditions affects the input of signal transduction and the metabolism sub-models, by setting the bound of related fluxes and components that are implicated in signal transduction pathways. The resulting benchmark on the triplet information serves as a decision support for laboratory construction and testing.

3. DISCUSSION

In this abstract, we present our results in an optimization method for part selection and an approach to integrate a design workflow and a genome-scale simulator. Although the incorporation of a genome-scale model to a design pipeline seems straight-forward, once each of these two frameworks are in place, there are a number of points to be considered. First, genome-scale and circuit models use a very different approach to modeling, with the former relying in a very small set of parameters that usually map statistical associations and not biophysical phenomena. In contrast, circuit models try to capture, in detail, the biophysical dynamics and use the appropriate kinetic constants (e.g. dissociation constants k_D, degradation rates k_{deg}) and modeling frameworks to do so. Merging these two worlds under a unifying framework that increases the model's predictive ability is quite challenging. As both fields move forward, data availability and coordinated efforts in both disciplines will be instrumental to close this gap.

4. REFERENCES

[1] L. Huynh and I. Tagkopoulos, "Optimal part and module selection for synthetic gene circuit design automation," *ACS Synth. Biol.*, 2014.

[2] L. Huynh, A. Tsoukalas, M. Köppe, and I. Tagkopoulos, "Sbrome: A scalable optimization and module matching framework for automated biosystems design," *ACS Synth. Biol.*, vol. 2, no. 5, pp. 263–273, 2013.

[3] Carrera, J., et al, "Genome-scale models of metabolism and gene expression extend and refine growth phenotype prediction," *Mol. Syst. Biol.*, vol. 10:735, doi:10.15252/msb.145108, 2014.

[4] O'Brien, Edward J., et al, "Genome-scale models of metabolism and gene expression extend and refine growth phenotype prediction," *Mol. Syst. Biol.*, vol. 9, no. 1, 2013.

Configurable Linear Control of Biochemical Systems

Tai-Yin Chiu[1], Ruei-Yang Huang[2], Hui-Ju K. Chiang[2,4], Jie-Hong R. Jiang[2,3], and François Fages[4]
[1]Department of Physics, National Taiwan University, Taipei 10617, Taiwan
[2]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
[3]Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
[4]EPI Lifeware, Inria Paris-Rocquencourt, France
{b99202046, b97901166, d01943033, jhjiang}@ntu.edu.tw; Francois.Fages@inria.fr

1. INTRODUCTION

The advancements of synthetic biology make biochemical systems of increasing complexity realizable in living cells. Many computation and control design examples have been demonstrated either *in vivo* or *in vitro*. In principle, any polynomial ordinary differential equation can be approximated by chemical reaction networks [1]. When control systems are of concern, linear control is one of the most widely applied control methods. Any linear control system can be realized with three elementary building blocks: integration, gain, and summation. Realizing linear control with biochemical reactions has been proposed in [2], where reaction rates of the underlying reactions play a key role to achieve the desired building blocks. Essentially the reaction rates have to be matched exactly, and it imposes serious practicality restriction because in reality the reaction rates of available reactions are predetermined and can be limited. In this paper we devise a mechanism to make linear control systems configurable by adding auxiliary species as control knobs. The concentrations of the auxiliary species can be adjusted not only to compensate reaction rate mismatch, but also to reconfigure different control systems out of the same control architecture. Hence implementing linear control systems in biochemistry can be made more practical.

2. METHODS

Following [2], we represent a real variable x by the difference $(x^+ - x^-)$ between the concentrations of two molecular species x^+ and x^-. In the sequel, we shall not distinguish a species and its concentration.

2.1 Integration Block

An integration block takes an input signal $u(t)$ and outputs a signal $y(t) = \alpha \int_0^t u(\tau)\mathrm{d}\tau + y(0)$ for $\alpha \in \mathbb{R}$. The integration block for $\alpha \geq 0$ consists of a pair of catalytic reactions (one with the species of upper signs in superscript and the other with species of lower signs) in (1) and an annihilation reaction in (2).

$$x^\pm + u^\pm \overset{k_1^\pm}{\to} x^\pm + u^\pm + y^\pm \tag{1}$$

$$y^+ + y^- \overset{\eta_{\text{int}}}{\to} \phi \tag{2}$$

Auxiliary species $x^\pm$ and input species $u^\pm$ serve as catalysts in (1). With the definition that $k_1^+ x^+ = k_1^- x^- \equiv \alpha$, the kinetics of y is exactly the integration of u as shown below.

$$\dot{y}^\pm = k_1^\pm x^\pm u^\pm - \eta_{\text{int}}\ y^+ y^-$$
$$y' = \dot{y}^+ - \dot{y}^- = k_1^+ x^+ u^+ - k_1^- x^- u^- = \alpha u$$

Because the concentrations of x^+ and x^- can be controlled but not $k_1^\pm$, in theory it is always possible to design a reaction network to meet any required α. For $\alpha < 0$, the signs in the superscript of y in (1) should be swapped to $\mp$.

2.2 Gain and Weighted Summation Blocks

A weighted summation block takes a number of input signals $u_i(t)$, $i = 1, 2, ..., n$ and outputs a signal $y(t) = \sum_{i=1}^n \alpha_i u_i(t)$ for $\alpha_i \in \mathbb{R}$. A gain block is a special weighted summation block with only one input $u(t)$ and producing output $y(t) = \alpha u(t)$ for $\alpha \in \mathbb{R}$. The gain block with $\alpha \geq 0$ can be realized by two pairs of catalytic reactions of (3) and (4), where $x^\pm$ and $z^\pm$ are auxiliary species, and by an annihilation reaction of (5).

$$x^\pm + u^\pm \overset{k_1^\pm}{\to} x^\pm + u^\pm + y^\pm \tag{3}$$

$$z^\pm + y^\pm \overset{k_2^\pm}{\to} z^\pm \tag{4}$$

$$y^+ + y^- \overset{\eta_{\text{gs}}}{\to} \phi \tag{5}$$

These reactions induce the following equation.

$$\dot{y}^\pm = k_1^\pm x^\pm u^\pm - k_2^\pm z^\pm y^\pm - \eta_{\text{gs}}\ y^+ y^-$$

Let $k_u \equiv k_1^+ x^+ = k_1^- x^-$ and $k_y \equiv k_2^+ z^+ = k_2^- z^-$. The mass action of y becomes

$$y' = k_u(u^+ - u^-) - k_y(y^+ - y^-) = k_u u - k_y y$$

If k_y is large enough compared to $|s| = \omega$, Laplace transform converts the above equation to

$$G = \frac{Y}{U} = \frac{k_u}{s + k_y} \approx \frac{k_u}{k_y} \equiv \alpha \tag{6}$$

That is, with properly chosen k_y, the value of y at equilibrium equals αu, which accomplishes the implementation of the gain block. For $\alpha < 0$, the superscript of y in (3) should be swapped.

The weighted summation block can be implemented with the same reactions as the gain block, except that (3) has to be changed to

$$x_i^\pm + u_i^\pm \overset{k_{1,i}^\pm}{\to} x_i^\pm + u_i^\pm + y^\pm,\ i = 1, 2, ..., n \tag{7}$$

If the scaling factor $\alpha_j < 0$, we simply swap the signs in the superscript of y in the reaction of (7) corresponding to input u_j.

3. CASE STUDY

We perform case study on the mass-spring-damper (MSD) system as shown in Fig. 1 A. The system can be modeled by the equation

$$M\ddot{x} + b\dot{x} + kx = F.$$

With $M = 1$ kg, $b = 10$ N s/m, $k = 20$ N/m, $F = 1$ N, by Laplace transform we derive the transfer function

$$G = \frac{1}{s^2 + 10s + 20} \approx 0.2236\left(\frac{1}{s + 2.764} - \frac{1}{s + 7.236}\right)$$

The transfer function can be implemented with the block diagram shown in Fig. 1 B. The proportional-integral (PI) controller to the MSD system is shown in Fig. 1 C.

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

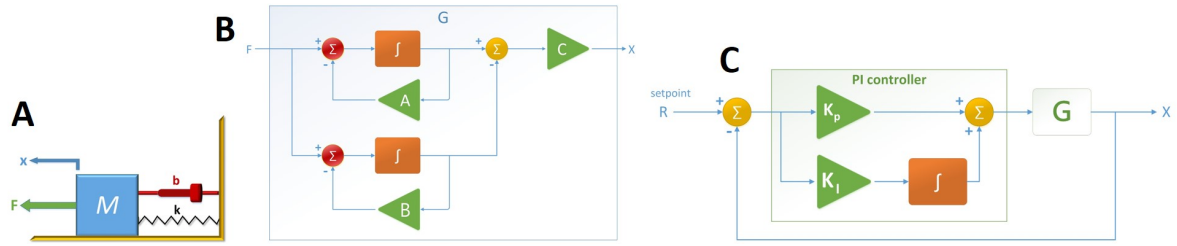
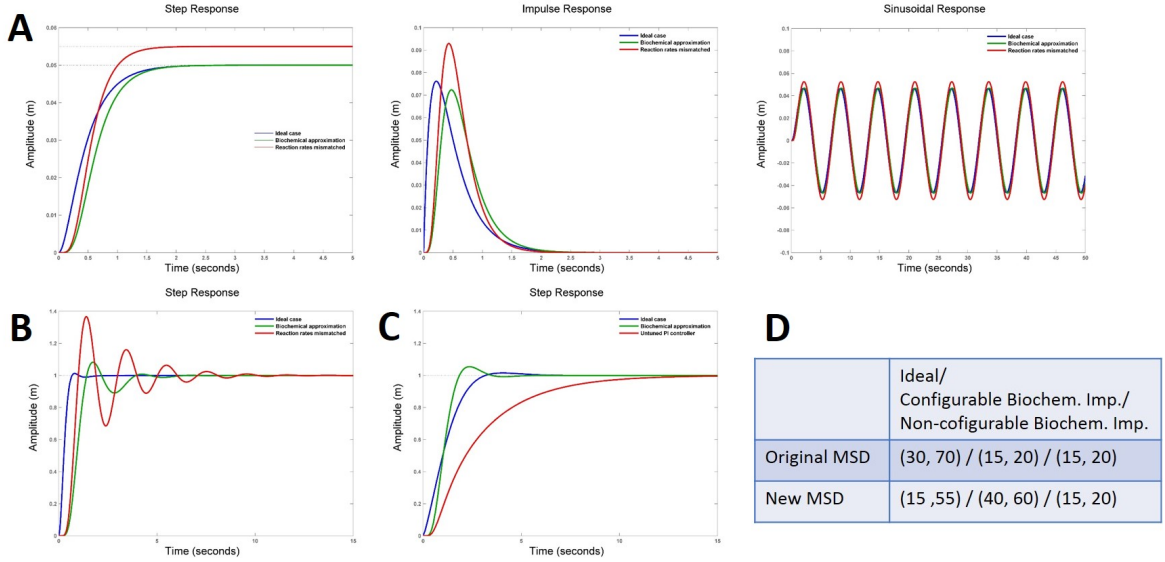


Figure 1: (A) Mass-spring-damper system. (Let $F = 1$ N, $M = 1$ kg, $b = 10$ N s/m, $k = 20$ N/m.) (B) MSD model, where triangular blocks denote gain functions with their corresponding weights, rectangular blocks denote integrators, and circle blocks denote mixers for summation and/or subtraction. (Let $A = 2.764$, $B = 7.236$ and $C = 0.2236$.) (C) PI-controlled MSD model, where G is the plant shown in (B). (Assume the values of K_P and K_I are given in Fig. 2 D.)



	Ideal/ Configurable Biochem. Imp./ Non-cofigurable Biochem. Imp.
Original MSD	(30, 70) / (15, 20) / (15, 20)
New MSD	(15 ,55) / (40, 60) / (15, 20)

Figure 2: The blue, green, and red curves represent the responses in ideal, configurable biochemical implementation, and nonconfigurable biochemical implementation cases, respectively. (A) Step, impulse, and sinusoidal (from left to right) responses of MSD. (B) Step responses of PI-controlled MSD. (Assume 10% rate mismatch in the MSD system.) (C) Step responses of PI-controlled MSD, where the MSD undergos parameter change with $b = 40$ N s/m and $k = 60$ N/m, respectively, which induces gain change of $A = 1.561$, $B = 38.44$ and $C = 0.0271$. (D) The values of (K_P, K_I) in simulation.

The block diagrams are constructed with $k_y = 10$ for all the summation and gain blocks except those summation blocks in red and gain blocks A and B with $k_y = 50$. The values of k_u are set to αk_y where the values of α equal the weights specified in the corresponding gain blocks. Also we assume k_2's have the same values as k_y's and k_1's are in 10% mismatch to k_u's.

Fig. 2 A and B show the responses of the MSD and the PI-controlled MSD systems. As can be seen, our method achieves better approximation to the ideal cases than the prior method [2]. One of the advantages of our method is that we can match the weight k_u/k_y by tuning the concentrations of $x^{\pm}$ and $z^{\pm}$, whereas in the prior method [2] no tuning is possible to avoid the inexact gain k_1/k_2 due to the mismatch of reaction rates k_1 and k_2. (Note that the biochemical implementations have their own optimal K_P and K_I values, shown in Fig. 2 D, to approximate the ideal system.)

Suppose that the spring and damper of the above MSD system are now replaced with new ones for $b = 40$ N s/m and $k = 60$ N/m. Without redesigning the PI-controller, our method can still adapt the PI-controller to the new MSD system whereas prior method has no such capability. Since we can tune the concentrations of $x^{\pm}$ and $z^{\pm}$ in biochemical implementation, it is possible for us to adapt (K_P, K_I) to optimal values $(40, 60)$ for the new PI-controlled MSD system, in contrast to the original $(15, 20)$. Fig. 2 C compares the results with and without such reconfigurability.

4. DISCUSSIONS

The aforementioned linear control systems can possibly be realized using the DNA strand-displacement technique. However the rate constants in the displacement reactions are about six orders of magnitude in 1 M^{-1} s^{-1} [3]. If we require $k_y = k_2^{\pm} z^{\pm}$ in (6) to be 100 M^{-1} s^{-1}, then the concentrations of $z^{\pm}$ will be of five orders of magnitude in nM. Such a high concentration might be impractical. To alleviate this high concentration requirement, one may try to increase the reaction rates. Zhang et al. have constructed and characterized DNA catalytic circuits driven by entropic gains [4]. Based on the entropy effects, a variant, called the tethered entropy driven catalytic circuits, has been introduced [5] to shorten the catalytic cycle and thus increase reaction rates. With these techniques, linear control systems may be effectively realized using DNA displacement reactions.

5. REFERENCES

[1] W. Klonowski. Simplifying principles for chemical and enzyme reaction kinetics. *Biophys. Chem.*, 18(2):73–87, 1983.

[2] K. Oishi and E. Klavins. Biomolecular implementation of linear I/O systems. *IET Syst Biol*, 5(4):252–260, 2011.

[3] D. Y. Zhang and G. Seelig. Dynamic DNA nanotechnology using strand-displacement reactions. *Nature Chem.*, 3(2):103–113, 2011.

[4] D. Y. Zhang, A. J. Turberfield, B. Yurke, and E. Winfree. Engineering entropy-driven reactions and networks catalyzed by DNA. *Science*, 318(5853):1121–1125, 2007.

[5] N. Gopalkrishnan, H. Chandran, S. Garg, and J. Reif. Speeding up DNA circuits using localized hybridization. In *Foundations of Nanoscience Meeting (FNANO)*, 2011.

A hybrid of multi-omics FBA and Bayesian factor modeling to identify pathway crosstalks

Claudio Angione*
Computer Laboratory
University of Cambridge - UK
ca394@cam.ac.uk

Naruemon Pratanwanich*
Computer Laboratory
University of Cambridge - UK
np394@cam.ac.uk

Pietro Lió
Computer Laboratory
University of Cambridge - UK
pl219@cam.ac.uk

1. FLUX OPTIMIZATION AND PATHWAY-BASED BAYESIAN INFERENCE

The remarkable availability of multi-omics data provides a highly comprehensive view of cellular processes at related levels of mRNA, proteins, and metabolites. Under a particular environmental condition, a bacterium may have a target of required pathway responses to achieve the desired phenotype. Therefore, one can question regarding the underlying mechanisms of those responses that enable the bacterium to obtain the desired characteristics under a compendium of different conditions.

We propose a hybrid method combining multi-omics FBA and Bayesian inference, with the aim of investigating the cellular activities of a bacterium from the transcriptomic, fluxomic and pathway standpoints under different environmental conditions. More specifically, we integrate a FBA model and a Bayesian factor model to determine the degree of metabolic pathway responsiveness and to detect pathway crosstalks, starting from gene expression profiles.

To evaluate the effect of environmental changes on a bacterium, we adopt a flux balance analysis (FBA) model and vary the constraints on the metabolic fluxes accordingly. This approach decodes transcriptomic activities into metabolic fluxes changing in a cell, using the existing metabolic network as a model of reactions.

To execute dimensionality reduction but maintain informative outcomes, we conduct a Bayesian factor model on those reaction fluxes at the level of pathways (defined groups of functionally related reactions). With this approach, we indicate the degree of responsiveness of each pathway under each environmental condition. Simultaneously, we extract the information of long range interactions between fluxes in terms of crosstalks between pathways, therefore accounting for the hidden biological organization underlying the bacterial responses to different environmental conditions.

By examining comprehensive levels of mRNA, proteins and metabolites, as well as exploiting the metabolic network and the definition of pathways, our pipeline provides molecular insights extracted from bacterial responses to an ensemble of environmental conditions. For instance, extracting pathways crosstalks from environmental conditions allows to investigate the relationship of bacteria and plants in mycorrhizae, or the role played by bacteria in sepsis and health conditions (e.g., gut microbiota). Furthermore, time-series gene expression profiles combined with our approach would enable further analysis on dynamical pathway crosstalks, unveiling the temporal progression of pathway activation.

*CA and NP contributed equally to this work

2. ENVIRONMENTAL CONDITIONS MAP TO FBA FLUXES

A useful feature of our approach is the possibility to map a gene expression microarray profile to a bidimensional space of objective functions (e.g., acetate-biomass). We use a compendium of 466 *E. coli* Affymetrix Antisense2 microarray expression profiles [1], referred to different media and different conditions (e.g., pH changes, heat shock, varying glucose and oxygen concentrations). We map each condition, which corresponds to a gene expression profile, to a set of bounds for the metabolic fluxes of an *E. coli* model [3].

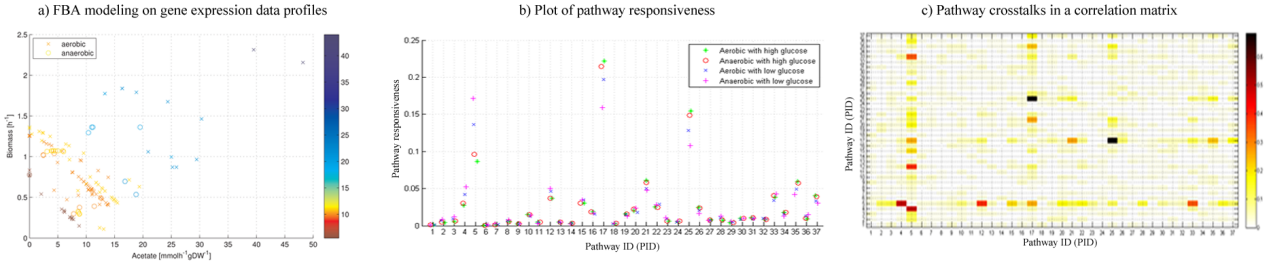
Since each reaction in the model depends on a gene set (a group of genes linked by AND and OR Boolean relations), the gene expression profiles are mapped to gene set expression profiles x by replacing AND and OR with min and max respectively. Formally, given the ith flux v_i of the metabolic network, we add the constraints $V_i^{min}\phi(x_i) \leq v_i \leq V_i^{max}\phi(x_i)$ where x_i is the expression of the ith gene set, V_i^{min} and V_i^{max} are the lower and upper bounds. Therefore, the lower and upper bounds are function of the expression of the genes responsible for the ith reaction. The function ϕ is defined as:

$$\phi(x) = \begin{cases} \gamma(1+|log(x)|)^{\mathrm{sgn}(x-1)}/\sigma_i^2 & \text{if } x \in \mathbb{R}^+ \setminus \{1\} \\ \gamma/\sigma_i^2 & \text{if } x = 1 \end{cases} \quad (1)$$

where $\mathrm{sgn}(x-1) = (x-1)/|x-1|$, σ_i^2 is the variance of the gene set responsible for the ith reaction, and γ is a weight for the variance. The variances σ_i^2 of the gene sets are computed from the variances of the genes across the conditions in the dataset, following the same rules defined to map the gene expressions to the gene set expressions. We make the assumption that the importance of a gene is inversely proportional to its variance, since a gene whose expression level is only slightly varied across conditions is assumed to be important for the bacterium [2]. Thus, we adopt the inverse of the variance as a multiplicative factor for the lower and upper bound, indicating the ability of the gene set to change the reaction fluxes for which it is responsible.

Finally, we define the synthetic and natural objectives (acetate and biomass respectively), while the oxygen and glucose uptake rate are constrained for every gene expression profile according to the oxygen and glucose of the corresponding experimental condition. We run the model once for each condition, therefore obtaining 466 flux distributions. In

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.



d) Table of pathway responsiveness

		High glucose		Low glucose	
PID	Pathway name	Aerobic	Anaerobic	Aerobic	Anaerobic
5	Nucleotide salvage	0.0865	0.0965	0.1366	**0.1714**
17	Valine, leucine, and isoleucine metabolism	**0.2219**	0.2147	0.1974	0.1590
25	Alanine and aspartate metabolism	**0.1544**	0.1487	0.1285	0.1076

*Note that pathway responsiveness is the probability, so it can range from 0 to 1.

Figure 1: Results from multi-omics FBA and Bayesian factor modeling. The 466 gene expression profiles are mapped to the acetate-biomass objective space after running FBA. Each point consists of 2583 reaction flux rates projected onto the objective space (a). The color bar shows the glucose uptake rate [mmolh^{-1}gDW^{-1}]. Interestingly, the *E. coli* strains grown in conditions with 10 mmolh^{-1}gDW^{-1} of glucose uptake rate are able to produce more biomass and acetate than those grown on higher glucose. Then, a pathway-based Bayesian analysis is performed on the flux rates in the four conditions based on the two criteria of oxygen and glucose. The average degree of responsiveness across aerobic and anaerobic conditions of high and low glucose is plotted (b), and the most responsive pathways are shown (c). PID:5 is the most responsive pathway in anaerobic conditions with low glucose, while PID:17 and PID:25 play a key role in aerobic conditions with high glucose, highlighting a pathway crosstalk between them (d).

Figure 1a, we plot the 466 flux distributions projected onto the acetate-biomass objective space.

3. BAYESIAN FACTOR MODELING

Here we perform pathway analysis on the reaction flux profiles obtained from the FBA analysis. Having regarded pathways as the latent factors underlying bacterial flux responses, we decompose the flux data matrix $\mathbf{X} \in \mathbb{R}^{R\times C}$ into a product of two matrices: $\mathbf{X} \sim \mathbf{BS}$. The first matrix $\mathbf{B} \in \mathbb{R}^{R\times P}$ denotes the membership strength of reactions in each pathway. The second matrix $\mathbf{S} \in \mathbb{R}^{P\times C}$ corresponds to the degree of pathway responsiveness specific to each condition. Note that R, C, and P are the number of reactions, conditions, and pathways respectively. We use the prior knowledge of pre-defined reaction-pathway memberships matrix $\mathbf{K} \in \{0,1\}^{R\times P}$ from [1] to guide the clustering reactions into pathways in $\mathbf{B}$.

Based on our assumption of pathway crosstalks, we model pathway dependencies by assuming a Gaussian distribution on $\mathbf{S}$ with a zero mean and a precision (inverse covariance) matrix $\mathbf{\Phi} \in \mathbb{R}^{P\times P}$ from which we compute the correlations between pathways. Finally, we make inference on $\mathbf{S}$, $\mathbf{B}$ and $\mathbf{\Phi}$, where $\mathbf{S}$ and $\mathbf{\Phi}$ indicate respectively the degree of pathway responsiveness to each condition and the crosstalks between pathways [4].

4. RESULTS AND DISCUSSION

FBA modeling was first applied to map 466 gene expression data into 2583 reaction flux rates of the *E. coli* metabolic network, subject to the maximization of acetate and biomass production. Next, Bayesian factor modeling was performed on those reaction fluxes by taking pre-defined reaction-pathway memberships as prior knowledge to infer the responsiveness of 37 pathways (covering all the 2583 reactions of the *E. coli* model) to each experimental condition (Figure 1b), and to detect crosstalks between pathways (Figure 1c).

The definitions of gene expression and the analysis of metabolic pathways in FBA models provide a pipeline to integrate and investigate data representing heterogeneous 'omic levels, with the aim of reconstructing complex multi-dimensional interactions. When the objective to maximize consists of both biomass and acetate production (Figure 1a), the *E. coli* strains grown in conditions with 10 mmolh^{-1}gDW^{-1} of glucose uptake rate produce more biomass and acetate than the strains on higher glucose. Thus, we are interested in investigating the underlying mechanisms in different oxygen conditions and glucose uptake rates (high/low glucose with the threshold of 10 mmolh^{-1}gDW^{-1}).

Table 1d shows the average of responsiveness degrees of the most responsive pathways to different oxygen and glucose conditions. PID:5 was important in anaerobic conditions on low glucose, while PID:17 and PID:25 both exhibit a key role in aerobic conditions on high glucose, highlighting a pathway crosstalk between them (1c). A recent review of amino acids and their functions shows that alanine is the primary amino acid gluconeogenesis, and valine directly synthesizes glutamine and alanine [5], suggesting that the crosstalk between them deserves closer attention.

5. REFERENCES

[1] J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J. J. Collins, and T. S. Gardner. *PLoS biology*, 5(1):e8, 2007.

[2] J. C. Mar, N. A. Matigian, A. Mackay-Sim, G. D. Mellick, C. M. Sue, P. A. Silburn, J. J. McGrath, J. Quackenbush, and C. A. Wells. *PLoS genetics*, 7(8):e1002207, 2011.

[3] J. Orth, T. Conrad, J. Na, J. Lerman, H. Nam, A. Feist, and B. Palsson. *Molecular systems biology*, 7(1), 2011.

[4] N. Pratanwanich and P. Lio'. *Molecular biosystems*, DOI:10.1039/c4mb00014e, 2014.

[5] G. Wu. Amino acids: metabolism, functions, and nutrition. *Amino acids*, 37(1):1–17, 2009.

Delay modeling in cell signaling and gene regulatory networks[·]

Natasa Miskov-Zivanov
Computer Science
Carnegie Mellon University
nmiskov@cs.cmu.edu

Chang Sheng Clement Loh
Peter Wei
Electrical and Computer Engineering
Carnegie Mellon University
{changshl, pwei}@andrew.cmu.edu

James R. Faeder
Computational and Systems Biology
School of Medicine
University of Pittsburgh
faeder@pitt.edu

ABSTRACT

In this paper, we describe methods for defining delays in discrete logical models of cell signaling and gene regulatory networks. We define here three methods and outline their main characteristics. We applied these methods in several models that we have been developing. The results that we obtained by simulating these models emphasize the fact that the selection of approach in modeling delays can have both qualitative and quantitative effects on the model's behavior and its steady state.

1. INTRODUCTION

When designing models of biological systems, natural or engineered, one often needs to take into account information about the timing in the system. In models of cell signaling and gene regulation this usually means defining rates of reactions, and either solving ODEs or simulating models using the SSA (stochastic simulation algorithm). However, it is often the case that element regulations are not well understood, and even when they are, rates of reactions are not known. Still, to better understand how the overall system works, it is critical to model the system using the available information. We present in the following the methods that can be used to implement timing information into discrete logical models.

2. BACKGROUND

Logical models do not require quantitative parameters, but rather enable the development of complex qualitative networks. When developing discrete logical models we first define key elements in the system to include in the model. The next step is to define element regulatory sets, as well as the number of values that are necessary to describe each element. In Figure 1(a)(top), we show an example of an element (ROS) and its regulators (In_1,In_2,AOX). In_1 and In_2 are implemented with Boolean variables, while ROS and AOX are implemented with discrete variables that can have three different values, 0,1, and 2. These variables are encoded with two Boolean variables each (ROS_{HI}, ROS_{LO} and AOX_{HI}, AOX_{LO}). The next step in model design is to define update rules for all model variables. We use the approach described in [1] to create model rules.

Once the model is built, we perform simulations with the BooleanNet tool [2]. Logical rules are updated in each simulation round using a random asynchronous approach, where in each step within the round, only one variable is chosen and the rule associated with that variable is evaluated to obtain a new value for the variable. The choice of rule to be evaluated next can follow different algorithms. In BooleanNet, in each update round, all rules are accessed and evaluated once, but the order in which this is done is random. Therefore, for a specified scenario (input values and initial variable states), a number of independent simulations can be conducted and an average behavior for each variable and the overall system can be computed.

3. APPROACH

Here we describe three types of delays that we account for when developing discrete logical models, and we show how these delays can be modeled. We use as an example the small regulatory network in Figure 1(a)(top) to describe different delay types.

3.1 Delay model I

We show in Figure 1(a)(middle) inputs and the output for the example network. The table in Figure 1(a)(bottom) shows combinations of inputs and the resulting output value. For example, the combination (ROS,AOX,In_1,In_2) = (0,1,0,1) results in ROS switching to value 1, but with two rounds of delay, which is accounted for by two 'd's in the table entry ('1dd'). Such table entry indicates that when the model is simulated, the input condition (ROS,AOX,In_1,In_2) = (0,1,0,1) needs to be satisfied in three consecutive rounds, and only in the third round ROS will actually switch from 0 to 1. We outline in Figure 1(b) a general implementation method and all the variables necessary for implementing these different cases of value switching with delays when the random order asynchronous simulation approach is used. To simplify the representation, the figure includes only up to three consecutive rounds and delay variables relevant for the ROS_{HI} variable rule, while longer delays can be defined and equivalent delay variables exist for the ROS_{LO} variable rule.

In this delay model, all conditions that satisfy the requirement for the same transition will be lumped into a single function and whenever the overall function is evaluated to 1 for specified number of consecutive rounds, the transition will occur. In other words, this allows for transients to occur even when the actual conditions change. Therefore, a smaller number of variables (in most cases) can be defined, since it is only necessary to check the value of the 'flags' (e.g., $ROS_{HI,D1}$, $ROS_{HI,D2}$, $ROS_{HI,D3}$, $ROS_{HI,I,D1}$, $ROS_{HI,I,D2}$, $ROS_{HI,I,D3}$) from the previous round when evaluating the function in the current round. In addition, this approach allows for minimizing logic functions of delay variables, since multiple table entries can be lumped into a single function.

3.2 Delay model II

In contrast to the previous approach, in some cases it is important to make sure that the exact condition leading to the value switch is the same while accounting for delay. If the condition changes before the required number of rounds has passed, and if the resulting output is still the same for the new conditions, the counting may need to be reset and the new condition needs to be satisfied for the specified number of consecutive rounds. For example, when switching from (ROS,AOX,In_1,In_2) = (0,1,0,1) to (ROS,AOX,In_1,In_2) = (0,1,1,0), the output is same, '1dd', but the conditions (input values) are different.

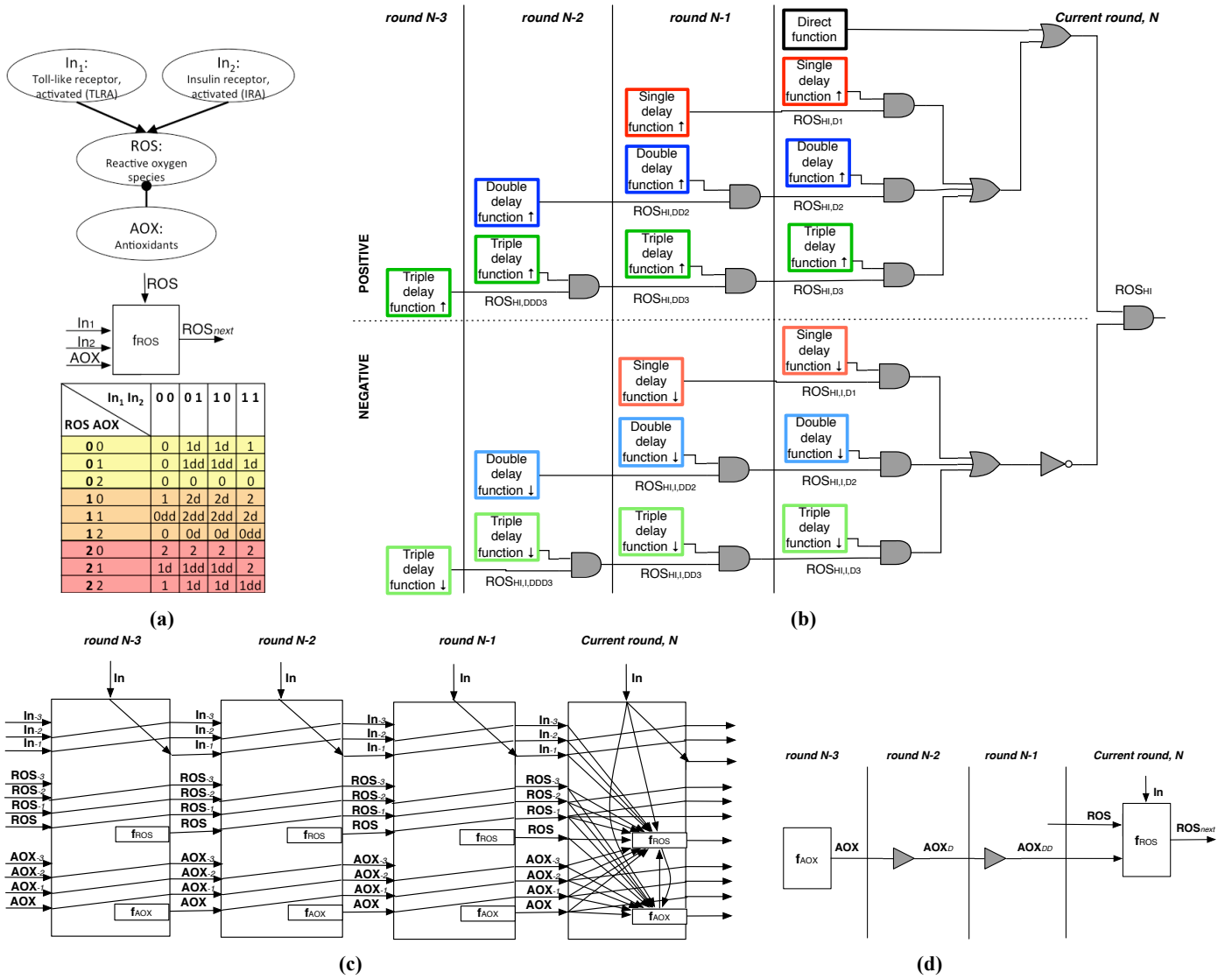
If this approach is used, one either needs to keep 'flags' for each specific condition from all previous rounds (as many rounds as delay length specified in the table), or to keep input values from previous rounds. Since the latter number of values is smaller, we use this approach, and this is shown in Figure 1(c) (**In**=(In_1,In_2), **ROS**=(ROS_{HI},ROS_{LO}), **AOX**=(AOX_{HI},AOX_{LO})). Values from one round are propagated to the next round, and variable indices (-1,-2,-3) indicate from which previous round the values are propagated.

3.3 Delay model III

The simplest delay model is shown in Figure 1(d). In this case, the delay is implemented in the form of 'buffers' that add steps to the pathway, thus delaying the propagation of the new value of an element (e.g., AOX) to some or all of its downstream elements (e.g., ROS). This approach can be used when modeling pathway sections without crosstalk or when only indirect causal relationships are known, while the overall timing of the pathway still needs to match the timing of

[·] This work was supported by NIH grant UL1-RR024153 and National Science Foundation Expeditions in Computing grant 0926181.

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.



In1 In2 / ROS AOX	0 0	0 1	1 0	1 1
0 0	0	1d	1d	1
0 1	0	1dd	1dd	1d
0 2	0	0	0	0
1 0	1	2d	2d	2
1 1	0dd	2dd	2dd	2d
1 2	0	0d	0d	0dd
2 0	2	2	2	2
2 1	1d	1dd	1dd	2
2 2	1	1d	1d	1dd

Figure 1. Delay models. **(a)** Example from the Malaria signaling network model [4]: positive regulation of reactive oxygen species (ROS) by Toll-like (In_1) and Insulin receptor (In_2) signals, and negative regulation by antioxidant (AOX) (top); black-box representation of inputs (including current value of ROS) and outputs of the f_{ROS} circuit (middle); input combinations and output values, which include indicators of the number of delay steps used in the first delay model shown in (b) (bottom). **(b)** Delay modeling method I. **(c)** Delay modeling method II. **(d)** Delay modeling method III.

other pathways in the network. In the example in Figure 1(d), $\mathbf{ROS}_{next}$ is a function of $\mathbf{AOX}_{DD}$. Depending on the simulator setup, this delay may be fixed such that it always takes exactly two additional rounds for the new AOX value to propagate to ROS, or two rounds is the upper bound to what this delay adds to propagating the signal from AOX to ROS. In the former case, the rules for variables $\mathbf{AOX}_D$ and $\mathbf{AOX}_{DD}$ have lower ranks than rules for **AOX** and $\mathbf{ROS}_{next}$ such that $\mathbf{AOX}_{DD}$ is always updated before $\mathbf{AOX}_D$, taking the value of $\mathbf{AOX}_D$ from the previous round, and similarly, $\mathbf{AOX}_D$ always takes the value of **AOX** from the previous round.

4. RESULTS

We used the described modeling approaches during the development of two different models, T cell differentiation model [3] and immune crosstalk in malaria infection in mosquitoes [4,5]. We analyzed the application of the delay modeling approach III in [3], and showed that different delays can affect transient behavior of elements by shifting their response curves left or right and by increasing the magnitude of transients. However, this doesn't seem to affect the qualitative behavior and steady state values. On the other hand, applying the delay modeling approach I can affect the results qualitatively as well as quantitatively, which we observed in the model of immune crosstalk in malaria [5].

5. DISCUSSION

We described here the delay modeling approaches that can be used in developing discrete models of cell signaling and gene regulation. In addition, we have recently created a simulator that can account for a variety of synchronous and asynchronous, as well as stochastic and deterministic rule combinations. This new simulator can handle the rules that are defined with the delay modeling approach II. Our next step is to implement and simulate our models using this new modeling and simulation approach, and to study qualitative and quantitative differences arising from different delay models.

6. REFERENCES

[1] N. Miskov-Zivanov *et al*. in Proc. of DAC, 2013.
[2] I. Albert *et al*. in Source Code for Biology and Medicine, 2008.
[3] N. Miskov-Zivanov et al. *Sci. Signal*, 2013.
[4] Y. Vodovotz *et al*. in Complex Systems and Computational Biology, 2013.
[5] N. Miskov-Zivanov *et al*. in preparation.

Precision Design of Expression from RNA Replicons

Jacob Beal
Raytheon BBN Technologies
Cambridge, MA, USA
jakebeal@bbn.com

Tyler E. Wagner
Center for Synthetic Biology
Boston University
Boston, MA, USA
wagnert@bu.edu

Tasuku Kitada
Biological Engineering
MIT
Cambridge, MA, USA
kitada@mit.edu

1. MOTIVATION

RNA replicons are an emerging platform of increasing interest, particularly for vaccination and therapeutic applications [3]. A replicon is based on a virus, but replaces the infective capsid proteins with engineered "payload" genes [5]. Here we focus on replicons derived from alphavirus, a positive-strand RNA virus, with architecture and lifecycle shown in Figure 1: the replicon RNA begins with a complex of non-structural proteins (NSPs) that create "viral factories" where it replicates [4]. A subgenomic promoter next induces production of shorter mRNAs containing engineered payload genes, which are translated to produce the proteins encoded by the payload sequences. Finally, both mRNA and proteins are removed by normal processes of dilution and decay.

Replicons provide distinct advantages as a platform for synthetic biology: as they are based on RNA, there is no direct path to affect cellular DNA, which reduces safety issues for medical applications. Unlike ordinary mRNA, they self-amplify, producing much stronger gene expression and over a longer time period, but can trigger a cell's immune response, thereby curtailing expression. For a full and referenced discussion of the background of this work, see [1].

In order to enable rapid engineering of replicon-based systems, we have characterized the expression dynamics of Sindbis replicons in baby hamster kidney BHK-21 cells with constitutively expressed payloads. From this characterization, we construct a quantitative model that predicts expression in three-replicon systems with better than 2-fold accuracy. This predictive model can then be inverted to produce an algorithm for designing transfection mixtures to produce either desired expression ratios or absolute expression levels.

2. QUANTITATIVE EXPRESSION MODEL

To characterize the expression dynamics for Sindbis replicons in BHK-21 cells, we carried out two experiments: a logarithmic dose-response test of a single species of replicon over a 50-hour timespan, and a linear titration of pairs of co-transfected replicons expressing different fluorescent proteins. Both experiments use the TASBE characterization method [2] to obtain calibrated flow cytometry data in absolute units; in particular, Molecules of Equivalent FLuorescein (MEFL). Full details of this characterization and modeling are given in [1]; for this abstract we merely summarize the key results and the model derived.

IWBDA 2014 Boston, Massachusetts, USA
This work was sponsored by DARPA DSO under grant W911NF-11-054; the views and conclusions contained in this document are those of the authors and not DARPA or the U.S. Government..

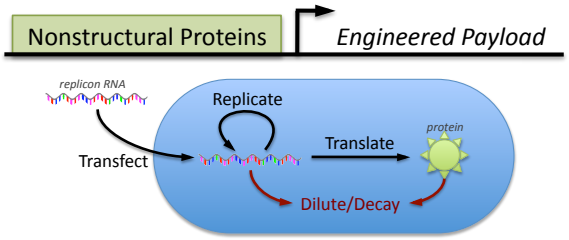


Figure 1: General architecture of RNA replicon design and expression dynamics.

Based on these experiments, we construct the following model: following a short initial delay of $\delta_E = 4.02$ hours as the replicons amplify, fluorescence converges exponentially with half-life $\lambda_E = 5.86$ hours towards a dose-independent translation-limited mean saturation level of $S = 5.44e7$ MEFL. This saturation level is modulated by a log-normal distribution of cell variance $V = 10^{\mathcal{N}(0,\sigma)}$, where the standard deviation $\sigma = 0.365$. The initial transient also appears dose-dependent, but cannot be quantified from these experiments.

Not all cells are transfected, and those that are transfected receive varying initial dosages of plasmids, which affects the final ratio of fluorescence. Cells are transfected with a Poisson distribution dependent on cell variance, receiving an initial "founder population" of each replicon of $f_i = Pois(\alpha \cdot V \cdot d_i)$, where $\alpha = 0.0127$ and d_i is the initial dose of replicon i. This initial dose is then amplified by a Polya Urn process (modeling the transition, as the number of replicons grows, from high sensitivity to stochastic effects to a stable ratio), giving an expected ith replicon proportion

$$p_i = (1 - \sum_{j<i} p_j) \cdot Beta(f_i, \sum_{j>i} f_j)$$

where $Beta(x, y)$ is the beta distribution (the ratio limit of a Polya Urn process). Transfected cells are thus predicted to have a distribution of expressions for the ith replicon at time t of:

$$E(t, i) = V \cdot p_i \cdot max(0, S(1 - 2^{\frac{\delta_E - t}{\lambda_E}}))$$

Finally, as replicon expression rises, the growth of transfected cells drops relative to untransfected. The fraction of expressing cells for total dose d at time t is thus:

$$F(d, 0) = \tau \cdot P(Pois(\alpha \cdot d) > 0)$$

$$F(d, t) = \frac{F(d, 0)}{F(d, 0) + (1 - F(d, 0)) \cdot max(1, 2^{\frac{\delta_F - t}{\lambda_F}})}$$

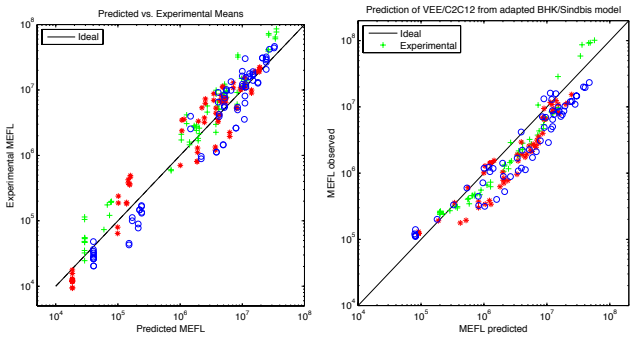


Figure 2: The quantitative expression model gives high precision predictions across a wide range of expression levels both for Sindbis/BHK (left) and VEE/C2C12 (right).

Where $\tau = 0.977$ is the maximum transfection efficiency, $\delta_F = 21.5$ hours is the time before significant impairment of transfected cells, and $\lambda_F = 8.89$ hours is the relative doubling time for untransfected cells after that point.

We have validated this model with a collection of three-replicon mixtures, chosen to test the model with dosage ratios ranging across two orders of magnitude and expression levels ranging across three. Measured at various time points up to 50 hours post-transfection, we find this model provides precise predictions of the mean expression (Figure 2), with a mean prediction error of only 1.7-fold. Model generality was tested by transfer to VEE replicon in C2C12 myoblasts, deriving parameters from a new set of experiments and adding an immune response term, to produce similar accuracy.

3. EXPRESSION DESIGN ALGORITHM

Given this forward model of expression, we can derive an algorithm for designing replicon mixtures with a desired combination of expression levels. In particular, we will design for the peak level of mean expression and peak fraction of cells expressing all elements in the mixture. Precise per-cell control of stoichiometry is, unfortunately, not an option given the degree of cell-to-cell variation, but as we will see the peak mean expression can be accurately designed.

We begin with an assumption that δ_E, λ_E, δ_F, and λ_F are such that cells reach near-peak expression before there is an significant shift in the ratio of expressing to non-expressing cells. This is the time we target, designing for a relative expression of R_i for the ith replicon and for a yield of Y fraction of cells expressing all replicons. The yield for the specified ratio, at a given minimum dosage d_m is:

$$Y = \tau \cdot \prod_i (1 - \int e^{-\alpha d_m \frac{R_i}{\min\{R_i\}} 10^{\sigma x}} \phi_{0,\sigma}(x) dx)$$

where the integral computes the expected fraction of cells not receiving the ith replicon ($e^{-\lambda}$ for a Poisson of parameter λ) with respect to the log-normal distribution of per-cell variation ($\phi_{0,\sigma}$ being the probability density function for a normal distribution with mean 0 and standard deviation σ). Although complicated, this function is monotonic in d_m, so it is easy to solve numerically for a dosage d_m to produce the specified yield Y. The full set of dosages then proceeds directly from the ratio: $d_i = d_m \cdot \frac{R_i}{\min\{R_i\}}$.

With this algorithm for designing ratios, we can also derive an algorithm for designing absolute expression levels.

Specification		Designed	Simulation	
Peak Expression	Yield	Dosages (ng)	Peak Mean MEFL	Yield
1:1 Ratio	0.95	766, 766	2.21e7, 2.21e7	0.950
3:1 Ratio	0.50	212, 71	4.18e7, 1.60e7	0.537
5:2:1 Ratio	0.75	892, 357, 178	3.34e7, 1.21e7, 5.86e6	0.775
3e6 MEFL	0.80	171, 2920	2.50e6	0.799
1e6, 1e7 MEFL	0.50	56, 557, 2420	1.29e6, 1.11e7	0.520
2e7, 1e7 MEFL	0.95	1250, 627, 1530	1.66e7, 7.73e6	0.953

Figure 3: Examples of applying design algorithm, comparing specification with simulated behavior for design (ballast dose last in MEFL designs).

Given a set of desired peak mean expressions E_i, we compute a required "ballast" expression:

$$E_b = S - \sum_i E_i$$

and design for the ratios of the set of E_i and E_b.

Figure 3 shows examples of applying this design algorithm, comparing the specification to a simulation of the designed dosages on a population of 100,000 cells at 20 hours post-transfection using the stochastic model in the prior section. The simulation matches the specification closely, with a maximum expression error of 1.3-fold and a maximum yield error of 4%. Obviously, this could be further improved with additional tuning of the design algorithm, but the need is not urgent since these accuracies are significantly better than the accuracy of simulation for predicting experimental observations.

4. CONTRIBUTIONS AND FUTURE WORK

We have developed a replicon expression model that accurately predicts the behavior of multiple replicon/cell-line combinations, and from this model an algorithm for forward design of expression ratios or levels. Experimental results indicate that this model may be broadly applicable. Future work aims to extend to therapeutically relevant cell lines and *in vivo* systems, as well as regulatory interactions, thus enabling rapid precision engineering of medical applications.

5. ADDITIONAL AUTHORS

Additional authors: Andrey Krivoy (MIT), Odisse Azizgolshani (UCLA), Jordan Moberg Parker (UCLA), Douglas Densmore (BU), Ron Weiss (MIT)

6. REFERENCES

[1] J. Beal, T. E. Wagner, T. Kitada, A. Krivoy, O. Azizgolshani, J. M. Parker, D. Densmore, and R. Weiss. Model-driven engineering of gene expression from RNA replicons. *submitted*.

[2] J. Beal, R. Weiss, F. Yaman, N. Davidsohn, and A. Adler. A method for fast, high-precision characterization of synthetic biology devices. Technical Report MIT-CSAIL-TR-2012-008, MIT, April 2012.

[3] K. Lundstrom. Alphaviruses in gene therapy. *Viruses*, 1:13–25, 2009.

[4] D. Paul and R. Bartenschlager. Architecture and biogenesis of plus-strand rna virus replication factories. *World J. Virol.*, 2:32–48, 2013.

[5] C. Xiong, R. Levis, P. Shen, S. Schlesinger, C. M. Rice, and H. V. Huang. Sindbis virus: an efficient, broad host range vector for gene expression in animal cells. *Science*, 243:1188–1191, 1989.

Modular design and implementation of eukaryotic synthetic gene circuits

Mario Andrea Marchisio
School of Life Science and Technology, Harbin Institute of Technology (HIT)
2 Yikuang Street, 150080 Harbin, P.R. China
marchisio@hit.edu.cn

Keywords
Circuit design, Parts, Pools, yeast

Bacterial synthetic gene circuits are designed in a modular, drag-and-drop fashion within the framework of composable Parts and Pools [5]. DNA Parts exchange fluxes of common signal carriers, such as RNA polymerases and ribosomes [2], that are stored into Pools. Prokaryotic Parts do not show a high degree of complexity and a detailed model based on mass-action kinetics can be derived rather easily. Eukaryotic Parts, in contrast, gather a considerable number of species and reactions. Promoters, for instance, can host many protein binding sites; RNA interference (RNAi), a widespread translation regulation mechanism, involves several proteins such as the Dicer and the RISC complex; mRNA get spliced in the nucleus and, then, it is transported into the cytoplasm to be translated by the ribosomes. Therefore, cell compartments have to be taken into account in order to model eukaryotic cells properly (see Figure 1).

A precise description of even fairly simple eukaryotic gene circuits demands a high number of species. They give rise to a dense network of reactions. Rule-based modeling is a way to cope with the *combinatorial explosion* in the interaction number. Software such as BioNetGen [1] calculates all the species and reactions involved in a biological system upon specification of *seed* species (the molecules present before that any interaction takes place), their initial concentrations, and *rules* that explain how different species interact. However, rule-based modelling is non-modular, thus not suitable *per se* for composable Parts and Pools. The Model Definition Language (MDL)–utilized by the software ProMoT [7]–is, in contrast, highly modular. Biological circuit components are represented by specifying the species and the reactions they contain together with an interface where fluxes of molecules (biological currents such as PoPS–Polymerase Per Second) are calculated and exchanged. In graphic terms, an interface is made of *terminals* on the component's icon: corresponding terminals on different Parts and Pools are connected with wires on the ProMoT graphic user interface.

The software we developed [4] gives a faithful description of eukaryotic Parts and Pools by using both BNGL (BioNetGen Language) and MDL. The user has to specify, into an input text file, the Part (or Pool) structure together with the kinetic parameters values associated with the reactions that take place inside the Part. For instance, a promoter requires to indicate the number and the kind of transcription factors acting on it, their possible activation or inhibition by chemicals, the number of operators corresponding to each repressor and activator, and the presence of cooperativity behavior. Moreover, numerical values such as the transcription factor and RNA polymerase binding and unbinding rates, the transcription initiation rate, and the promoter leakage have to be provided. The knowledge of many parameter values, not always easy to measure, is the main drawback of a precise model based on mass-action kinetics. The input information is converted into an BNGL file. After processing it, BioNetGen computes a list of Part species and reactions that is returned to our software. Our software calculates the fluxes exchanged by the Part and writes them, together with the species and reactions received by BioNetGen, into an MDL file that describes the Part as an independent module (see Figure 2). MDL files are loaded into ProMoT and the corresponding Parts and Pools are wired into synthetic gene circuits.

Initially, we tested our software on the RNAi logic evaluator [8] (12 genes, 197 species, and 474 reactions) and on an alternative circuit–based on transcription regulation–that gathered 1165 reactions. Results were in good agreement with published experimental data. Later, we used the software to re-design and improve basic Boolean gates (YES, NOT, AND) implemented in yeast [3]. They were based on bipartite promoters made by joining a fragment of the *VPH1* promoter (up to the TATA box) with the minimal *CYC1* promoter (containing the TSS-Transcription Start Site). In between we placed one or two binding sites for bacterial (orthogonal) repressors. A couple of these gates did not work in their original configuration. According to our previous work on gene digital circuits [6], such a problem can be solved just by re-engineering the bipartite promoter. With our software we showed, quantitatively, that a multiple integration of the transcription unit containing the bipartite promoter had the same effect. Experimental measurements (via flow cytometry) confirmed that the two strategies were equivalent (see

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

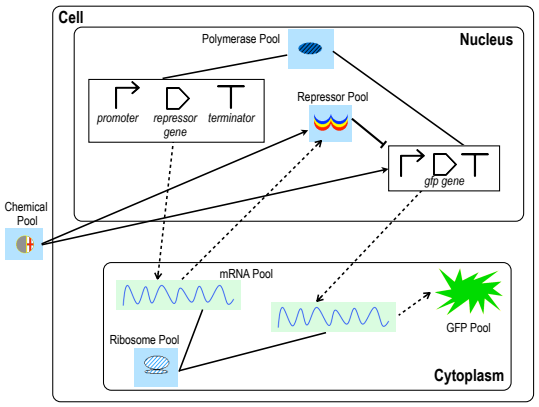


Figure 1: Eukaryotic circuit with Parts and Pools. Green fluorescent protein (GFP) expression is regulated by a repressor that can be inhibited by a chemical. Straight lines indicate an exchange of molecules, dashed arrows transcription or translation, straight arrows transcription activation, the bar-ending line transcription repression.

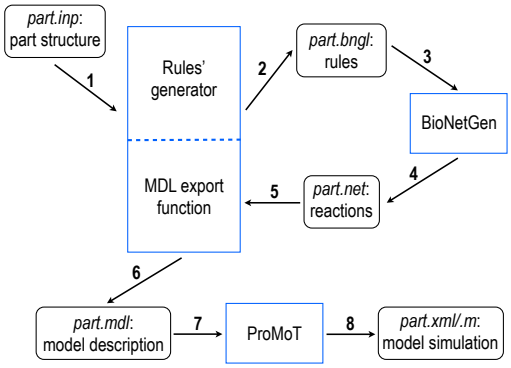


Figure 2: Eukaryotic Part generation. Our software is made of two main components: the rule generator (to call BioNetGen) and the MDL export function. Part.inp is a text file where the user specifies both Part structure and parameter values. This is converted into an BNGL file (part.bngl) on which BioNetGen computes Part's species and reactions. They are written into a new file, part.net. Our software creates an MDL file (part.mdl) by merging the content of part.net with an appropriate Part interface. Part.mdl is loaded into ProMoT and used to design synthetic gene circuits.

Figure 3). Therefore, our computational tool proved to be a useful instrument for circuit scheme comparison and performance assessment.

1. REFERENCES

[1] M. L. Blinov, J. R. Faeder, B. Goldstein, and W. S. Hlavacek. BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics (Oxford, England)*, 20(17):3289–3291, Nov. 2004.

[2] D. Endy. Foundations for engineering biology. *Nature*, 438(7067):449–453, Nov. 2005.

[3] M. A. Marchisio. In silico design and in vivo implementation of yeast gene Boolean gates. *J Biol Eng*, 8(1):6, Feb. 2014.

[4] M. A. Marchisio, M. Colaiacovo, E. Whitehead, and J. Stelling. Modular, rule-based modeling for the design of eukaryotic synthetic gene circuits. *BMC systems biology*, 7(1):42, 2013.

[5] M. A. Marchisio and J. Stelling. Computational design of synthetic gene circuits with composable parts. *Bioinformatics (Oxford, England)*, 24(17):1903–1910, Sept. 2008.

[6] M. A. Marchisio and J. Stelling. Automatic design of digital synthetic gene circuits. *PLoS computational biology*, 7(2):e1001083, Feb. 2011.

[7] S. Mirschel, K. Steinmetz, M. Rempel, M. Ginkel, and E. D. Gilles. PROMOT: modular modeling for systems biology. *Bioinformatics (Oxford, England)*, 25(5):687–689, Mar. 2009.

[8] K. Rinaudo, L. Bleris, R. Maddamsetti, S. Subramanian, R. Weiss, and Y. Benenson. A universal RNAi-based logic evaluator that operates in mammalian cells. *Nat Biotechnol*, 25(7):795–801, July 2007.

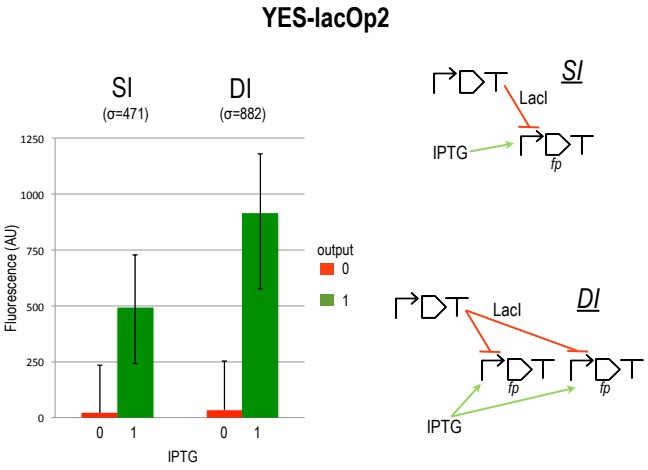


Figure 3: YES gates sensing IPTG. The YES gate returns 1 in presence of its single input signal and 0 in its absence. A clear improvement in the gate signal separation σ (the distance between high and low output at steady state) is obtained by only modifying the transcription initiation rate of the promoter leading the fluorescent protein (fp) synthesis [6]. This is carried out through a double integration (DI) of the transcription unit containing the fp gene. The original YES gate (two *lac* operators, single integration–SI) did not show a clear distinction between 0 and 1 output. This was achieved with the double integration strategy that permitted to set a unequivocal 0/1 threshold at 500 arbitrary units (AU) of fluorescence.

Automatic Enumeration and Discrimination of Model Phenotypes With Application to Synthetic Gene Oscillators

Jason G. Lomnitz
University of California at Davis
One Shields Ave.
Davis, CA 95616
+1 (530) 754 6682
jlomn@ucdavis.edu

Michael A. Savageau
University of California at Davis
One Shields Ave.
Davis, CA 95616
+1 (530) 754 6682
masavageau@ucdavis.edu

ABSTRACT
Synthetic biology integrates multiple disciplines with the aim of designing and engineering new systems that carry out specific functions. However, the task of integrating systems of biological components involves interactions of nonlinear processes with emergent behaviors that we do not fully understand. There is a need for mathematical and computational tools that aid in the automatic design of new biological systems. Here, we show a design strategy that provides a global perspective of the phenotypic potential of mechanistic models of biological phenomena. This strategy is largely independent of specific parameter values, and allows for designs to be discriminated based on their potential to exhibit specific phenotypes and ensembles of phenotypes. As an example of our strategy, we apply it to a general class of circuit designs exhibiting oscillatory behavior and bi-stability.

Categories and Subject Descriptors
J.3 [**Computer Applications**]: Biology and genetics;
I.6.5 [**Computing Methodologies**] Model Development–*modeling methodologies*

General Terms
Design, Theory

Keywords
System design space, automated design, dynamic phenotypes, circuit architecture, mode of transcription control.

1. INTRODUCTION
Biological systems are proving useful as platforms where well-known biological components may be combined into integrated systems that perform specific functions. Some examples include constructs that can compute binary logic [2, 4], produce metabolites useful in industry and medicine [8, 10], or provide insight into the operation of more complex natural circuitry [1, 5, 9, 12]. However, it is a challenge to identify the repertoire of potential behaviors that are latent in any particular circuit design and thus, the design of such systems and its analysis is typically treated in an ad hoc fashion. We have previously developed a system design space methodology for deconstructing intractable nonlinear models into a series of simpler models that can be readily analyzed and the results efficiently reassembled to characterize the global repertoire of the original system [3, 6, 11]. We have also applied this method to elucidate the design of a number of natural systems.

In this work we present a new strategy to assist in the design of novel synthetic systems, as well as to elucidate the design principles of natural biological systems. It enables the automatic enumeration of qualitatively distinct phenotypes of nonlinear models in a manner that is independent of parameter values. Once enumerated, we are able to obtain parameter sets for any given phenotype of the system. The same can be applied to identify parameter sets for desired ensembles of phenotypes. The local behaviors can be analyzed in detail, and computational efforts can be directed to test specific predictions. Taken together, this strategy can be applied in the design cycle to guide the realization of systems with desired behaviors.

2. METHODS AND RESULTS
We describe a strategy that provides a global perspective on system behavior. It involves first formulating mechanistic models consisting of nonlinear ordinary differential equations that capture the architectural features of a system without specifying numerical values for the parameters. The system design space methodology is then used to enumerate the set of qualitatively distinct phenotypes by deconstructing the complex system into nonlinear sub-systems, based on mathematically defined phenotypes [6, 11]. The result is the automatic enumeration of the complete set of valid phenotypes that defines the phenotypic repertoire of the system.

Assuming the complete phenotypic repertoire of a system has been enumerated, how does one obtain meaningful sets of parameters for the phenotypes of a system? A naive approach would be to sample parameter space until the desired phenotype is found. However, there are severe limitations to this approach: First, the number and nature of the behaviors is highly dependent on the sample size. If we were to look for qualitatively distinct phenotypes by sampling 4 points for each parameter in a model with n parameters, we would need to sample 4^n points. Even with a small number of parameters the number of samples is large. Second, there is no guarantee that all phenotypes will have been sampled, and the likelihood of missing behaviors altogether is high.

Our design tool provides a powerful means to obtain unique parameter sets for any valid phenotype of the system. By virtue of the mathematically defined boundaries of a phenotype, we may efficiently obtain a unique parameter set within the feasible region

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

of any qualitatively distinct phenotype by means of linear programming techniques. We are able to extend this approach and identify ensembles of phenotypes that are co-localized within some lower-dimensional slice of parameter space.

The ability to automatically identify parameters for phenotypes that are co-located within a slice of parameter space allows us to determine the maximum number of qualitatively distinct phenotypes that are co-located within a slice of parameter space that is of particular interest.

This has applications to synthetic biology, where a particular design is engineered to exhibit specific behaviors under specific operating conditions. In complex nonlinear systems, these behaviors are typically generated from ensembles of distinct phenotypes that are observed as the input signals change. The task of designing a biological system that behaves according to the desired ensemble of phenotypes can be challenging, again due to the complex nonlinearity of biological mechanisms. Our strategy enables us to identify groups of qualitatively distinct phenotypes with desired behaviors that are co-located within some slice of parameter space, and the parameter values associated with the slice can be obtained.

As an application of our strategy, we revisit a class of seven two-gene circuits involving an activator and a repressor that are able to exhibit oscillatory behaviors. We have previously compared the members of this class with respect to the robustness of their oscillatory potential using mathematically controlled comparisons [7]. Here, we show that new insight is gained by applying our automated strategy to obtain a global perspective of their phenotypic potential. We also extend the analysis to a larger class of two-gene circuits that includes 9 additional designs that had not been found to oscillate under the controlled comparisons. Our automated approach is able to identify two new designs that exhibit sustained oscillations, one with an architecture that can be reduced to one of the pervious 7 and one that is entirely new.

3. CONCLUSIONS

The integration of phenotypes into a system design space allows the qualitatively distinct phenotypes to be identified, enumerated, characterized and compared. The system design space provides an efficient means to characterize system behavior over a broad range of parameter values, and the landmarks in this space represent particular constellations of parameters that define relevant design principles [7].

Here, we present a new strategy that can be used as a tool to assist in the design of novel biological circuits by allowing global properties of system behavior to be identified through the automatic enumeration of phenotypes. Once enumerated, we can very efficiently obtain parameter sets that correspond to each of the phenotypes in the complete phenotypic repertoire of the system. We have also shown that this strategy allows specific ensembles of behaviors to be found within a desired slice through parameter space. Again, parameter sets for such ensembles can efficiently be obtained, if they exist.

In the current application to synthetic oscillators, we identify new designs within a general class of two-gene circuits that are capable of exhibiting oscillatory behaviors. For many of the designs we have identified multiple phenotypes that are capable of exhibiting oscillatory behaviors. Our tool allows us to find a slice through parameter space where multiple distinct oscillatory phenotypes are co-located.

It should be stressed that the parameter sets we obtain may not correspond to the most biologically relevant values. However, parameter values can be bounded and even fixed by specific constraints, and once constrained, the analysis can be repeated to guarantee that relevant requirements are met. This is particularly useful in the context of synthetic biology, where a catalog of parts with characterized parameters and specific ranges of operation can inform the search for phenotypes and ensembles of phenotypes.

4. ACKNOWLEDGMENTS

This work was supported in part by a grant from the US Public Health Service (RO1-GM30054). We thank Alberto Marin-Sanguino and Christiana Sehr for offering suggestions.

5. REFERENCES

[1] Benner, S.A. and Sismour, A.M. 2005. Synthetic biology. *Nature reviews. Genetics*. 6, 7 (Jul. 2005), 533–543.

[2] Bonnet, J. et al. 2013. Amplifying Genetic Logic Gates. *Science*. 340, 6132 (May 2013), 599–603.

[3] Fasani, R.A. and Savageau, M.A. 2010. Automated construction and analysis of the design space for biochemical systems. *Bioinformatics*. 26, 20 (Oct. 2010), 2601–2609.

[4] Guet, C.C. et al. 2002. Combinatorial synthesis of genetic networks. *Science*. 296, 5572 (May 2002), 1466–1470.

[5] Kim, J.K. and Forger, D.B. 2012. A mechanism for robust circadian timekeeping via stoichiometric balance. *Molecular systems biology*. 8, (2012), 630.

[6] Lomnitz, J.G. and Savageau, M.A. 2013. Phenotypic deconstruction of gene circuitry. *Chaos*. 23, 2 (Jun. 2013), 025108.

[7] Lomnitz, J.G. and Savageau, M.A. Strategy Revealing Phenotypic Differences Among Synthetic Oscillator Designs. (*Submitted*).

[8] Martin, V.J.J. et al. 2003. Engineering a mevalonate pathway in Escherichia coli for production of terpenoids. *Nature Biotechnology*. 21, 7 (Jul. 2003), 796–802.

[9] Mukherji, S. and van Oudenaarden, A. 2009. Synthetic biology: understanding biological design from synthetic circuits. *Nature reviews. Genetics*. 10, 12 (Dec. 2009), 859–871.

[10] Ro, D.-K. et al. 2006. Production of the antimalarial drug precursor artemisinic acid in engineered yeast. *Nature*. 440, 7086 (Apr. 2006), 940–943.

[11] Savageau, M.A. et al. 2009. Phenotypes and tolerances in the design space of biochemical systems. *Proceedings of the National Academy of Sciences of the United States of America*. 106, 16 (Apr. 2009), 6435–6440.

[12] Tigges, M. et al. 2010. A synthetic low-frequency mammalian oscillator. *Nucleic acids research*. 38, 8 (May 2010), 2702–2711.

Better Scheduling Software and User Interface for Liquid-Handling Robots

Ben Schreck
Weiss Lab
MIT Bldg NE-47
500 Technology Square
Cambridge, Massachusetts
bschreck@mit.edu

Jonathan Babb
Weiss Lab
MIT Bldg NE-47
500 Technology Square
Cambridge, Massachusetts
jbabb@mit.edu

Ron Weiss
Weiss Lab
MIT Bldg NE-47
500 Technology Square
Cambridge, Massachusetts
rweiss@mit.edu

ABSTRACT

This paper outlines the design of a software system that enhances the workflow efficiency and usability of a liquid-handling robot for executing wet lab protocols, with a focus on synthetic biology. The system we propose builds on the BioCAD software suite[1] that includes an object-oriented suite of Python tools to interact with a liquid-handling robot through its API. Our extension to this suite breaks up protocols into distinct, atomic steps, and schedules them by interleaving these steps to reduce waiting time where the robot would ordinarily do nothing while, for instance, a reaction occurs. A remote web interface has been implemented that allows researchers to easily define and submit jobs to be performed on the robot. When fully implemented, this system promises to remove much of the repetitive, manual labor so common in life science research by lowering the barrier to entry to robotic automation. Even though it is designed for a particular robot in a particular synthetic biology lab, it generalizes to many flavors of robot and fields of wet-lab research.

Categories and Subject Descriptors

D.4.1 [**Operating Systems**]: Process Management; I.2.9 [**Artificial Intelligence**]: Robotics; J.3 [**Computer Applications**]: Life and Medical Sciences

Keywords

automation, synthetic biology, scheduling, user interface

1. PREVIOUS WORK

The BioCAD software suite was originally designed as a sub-component of the larger TASBE system[1], which had the goal of automating the entire synthetic biology workflow, from circuit specification to in-vivo assembly. TASBE[1] demonstrated that it was possible to automatically build a functional genetic circuit only specifying the highest-level abstraction of its operation (such as a NOT-gate with a set of specific gene-promoter sequence inputs and outputs). BioCAD is in charge of the final stage of converting a set of synthetic biology protocols that would ordinarily be performed by hand, into a format that is readable by a liquid-handling robot. For a concise description of the synthetic biology workflow and the challenges facing synthetic biology researchers, see Douglas Densmore and Soha Hassoun's article on bio-design automation[3].

2. ARCHITECTURE

Our proposed system consists of a user-facing front-end web interface, a back-end server to handle requests from this web interface and convert them into jobs to be sent to a queue, a scheduler that maps job requests from the queue into defined robot protocols, and the modified BioCAD software suite[1]. The web interface is a Ruby on Rails site, located on a Heroku server in the cloud. This remote server also acts as a client to a server running on the computer connected to the robot, in that it can send job requests to the robot computer to be scheduled. Everything on the robot computer is written in Python as an extension to BioCAD[1].

2.1 Web Interface

The basic functionality of the Ruby on Rails web interface has been implemented. An administrative user can define protocols on the site that correspond to the same protocols implemented in BioCAD. These protocols are all presented to a user (synthetic biology researcher), who can tweak the specific parameters associated with them to fit his or her current experiment. For instance, a parameter might be the type or quantity of reagent or cell line to be used. The user can then save this modification in the database for future use. What remains to be implemented is a way to submit jobs to be scheduled, an interface for presenting the current state of a scheduled job (e.g. what sub-task is currently being run or how much longer will the protocol take to finish), debugging tools, and a collaboration/reward system.

2.2 Scheduling Algorithm

Overview.

Each job consists of n individual ordered tasks, each of which can be thought of as an atomic instruction, and demand the full attention of the robot. A task might be "Transfer $5\mu L$ of sample X to the working wells and shake for 5 minutes". A job also contains $n-1$ individual waiting periods, which are interleaved between each of the tasks. These waiting periods may contain a minimum and a maximum wait time, and correspond to a wait time for a reaction for instance, or cell incubation. However, the system also defines wait periods between tasks, which may be scheduled with any arbitrary space between them. These waiting periods have their minimum time set to zero and their maximum time set to infinity. In general, this is a hard problem because even though the first task of a job may fit into a wait

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

```
queue = SchedulerQueue()
function Schedule(queue):
  scheduler = Scheduler()
  # select() finds the job with the highest score
  # and removes it from the queue
  # score depends on complexity (number of constraints)
  # and length. Eventually will take into account
  # resource allocation
  job = queue.select()
  serialSchedule = scheduler.makeSchedule(job)
  scheduler.setScheduler(serialSchedule)
  waitStartIndex = 0
  while (not queue.isEmpty()):
      # findNextWait() finds the next wait
      # period in the job with a minimum time
      wait = scheduler.findNextWait(waitStartIndex)
      if wait != None:
        waitStartIndex = wait + 1
      removedJobs = list()
      while (not queue.isEmpty()):
          jobToInterleave = queue.select()
          # check if job can be scheduled
          # on top of existing schedule
          valid, schedule = scheduler.interleave(wait,
                      jobToInterleave)
          if valid == True:
              scheduler.setSchedule(schedule)
              queue.add(removed_jobs)
              wait=scheduler.findNextWait(waitStartIndex)
          else:
              # keep record of jobs removed
              # from the queue
              removedJobs.append(jobToInterleave)
      queue.addJobs(removedJobs)
      return scheduler.schedule
```

Figure 1: Scheduler Pseudocode

period of another, they may still be subsequently impossible to schedule. A brute force optimal solution is combinatorial in nature, requiring something along the lines of a backtracking search strategy.

We have implemented a simple scheduler, with the intention of improving it in the future, potentially using an iterative repair technique like that used in the Gerry scheduling system[4]. An initial algorithm has been designed in the pseudocode in Figure 1.

Performance.

This algorithm has complexity $O(W * N)$, where W is the total number of wait periods in all the jobs, and N is the total number of discrete tasks that need to be scheduled. While not necessarily optimal, it will at least find a sequential scheduling. Figure 2 shows the percent total time reduction of the scheduler for different numbers of scheduler protocols.

3. FUTURE WORK

In the next few weeks, we plan on adding resource constraints to the scheduler (e.g. which protocols are using which plates at what times), testing it out on the robot, finishing the implementation of the entire system, and testing it out with researchers for usability and efficiency. We also plan on testing it against the scheduling software that comes with the Tecan Freedom Evo robot: Tecan EvoWare Plus, and explore ways of integrating our approach with the current vendor scheduling software from Tecan. We can compare the schedulers simply by timing how long it takes each one to schedule some set of protocols. After establishing a working scheduler in the complete system, we will implement an iterative repair scheduler like that used in Gerry[4] to improve our throughput even further.

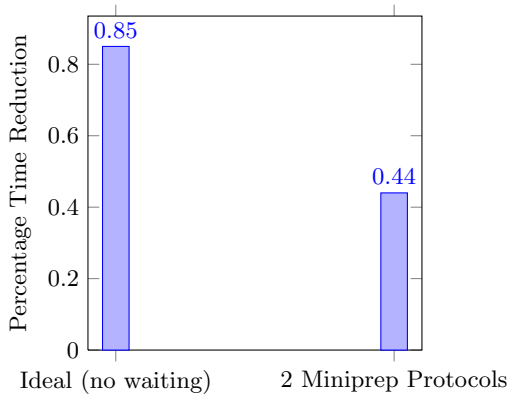


Figure 2: Percentage total time reduced by scheduling various numbers of Miniprep Protocols in parallel

4. CONCLUSIONS

By providing an extensible, remote, and simple user interface, this system aims to empower biologists to automate the time-intensive, mechanical aspects of their experiments, which should increase their own research efficiency, reduce the amount of tedious low-level tasks required to execute an experiment, and enhance the repeatability of their experiments. The scheduling software on the robot allows for real-time queuing of jobs, so that researchers can issue a job to be run, and keep track of its status remotely until notified that it is done or that a manual step is required. The design proposed here and partially implemented in the lab promises to increase workflow efficiency of standard synthetic biology protocols here in the Weiss Lab, and in the future any other synthetic biology lab with access to a liquid-handling robot.

5. REFERENCES

[1] J. Beal, R. Weiss, D. Densmore, A. Adler, E. Appleton, J. Babb, S. Bhatia, N. Davidsohn, T. Haddock, J. Loyall, R. Schantz, V. Vasilev, and F. Yaman. An end-to-end workflow for engineering of biological networks from high-level specifications. *ACS Synth. Biol.*, 1(8):317–331, July 2012.

[2] G. Bradski. Opencv. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] D. Densmore and S. Hassoun. Design automation for synthetic biological systems. *IEEE Design and Test of Computers*, 29(3):7–20, June 2012.

[4] M. Zweben and M. S. Fox. *Intelligent Scheduling.* p241-255. Morgan Kaufmann, San Francisco, California, 1994.

SMT-based Strategies for Biodesign Automation

[Extended Abstract]

Boyan Yordanov
Biological Computation Group
Microsoft Research
yordanov@microsoft.com

Andrew Phillips
Biological Computation Group
Microsoft Research
Andrew.Phillips@microsoft.com

1. INTRODUCTION

Synthetic biology holds the promise of addressing many societal problems through the application of engineering principles to the design of biological systems. However, despite recent theoretical and experimental advances, the construction of biological devices that behave "as required" remains challenging. Biodesign automation (BDA) tools [3, 2] aim to decrease the cost and improve the robustness of biological circuits but the development of scalable methods allowing the high-level specification of circuits and their intended dynamical behavior is still an active research area.

Here, we present a new direction towards the development of biodesign automation tools through Satisfiability Modulo Theories (SMT) based methods. We consider the problem of automatically generating designs realizable from a database of biological parts for circuits that satisfy specifications of dynamical and logical component interaction properties. Instead of developing dedicated technology-mapping algorithms, we focus on a framework that is extensible and can handle more general instances of the computationally challenging problems arising as part of circuit design [6]. To ensure the scalability of the approach while supporting expressive specification languages we formalize and encode the design of circuits as an SMT problem, and use powerful solvers such as Z3 [1], which have been successfully applied to hardware and software design and verification.

Our method builds on techniques from [9] where SMT-based methods were used for the construction of systems from devices but allows reasoning about biological parts and the functions they implement. While our approach is quite general and potentially useful for other BDA tools, it is currently implemented in `GEC` [4], where the existing functionality is extended by allowing the design of circuits with abstracted topology from large parts databases using logical and dynamical specifications. In the following, we briefly introduce the formalization of the approach and illustrate it using preliminary results from several case studies.

2. SMT-BASED ENCODING

We consider a database as the tuple (S, P, R) of species S, parts P and reactions R. Various types of parts and the properties they allow are described in [4]. We represent a biological circuit constructed from this database as ($S'\subseteq S$, $P'\subseteq P$, $R'\subseteq R$) (*i.e.* only species, parts and reactions from the database are used). We assume that the database contains a finite number of parts represented by a unique numerical identifier and encode such a database using the SMT theory of bit-vectors, which is handled efficiently by Z3 [1]. S, P and R are represented by the set of bit-vectors of appropriate size and each species $s \in S$, part $p \in P$ and reaction $r \in R$ is a unique bit-vector from that set (circuits are encoded similarly). We use the maps $\hat{s} : S' \to S$, $\hat{p} : P' \to P$ and $\hat{r} : R' \to R$ to relate the components of a circuit to their corresponding database entries and, in the following, use $\hat{s}$, $\hat{p}$, $\hat{r}$ to represent the database entry corresponding to circuit species s, part p or reaction r. We use $\texttt{nextTo} : P' \to P'$ to capture the sequence of parts in a circuit (*i.e.* $\texttt{nextTo}(p, p')$ signifies that part p' follows p in the DNA sequence implementing the circuit). The $\hat{s}$, $\hat{p}$, $\hat{r}$ and `nextTo` maps are initially unknown but are constructed automatically as part of the circuit design generation in a way that guarantees the satisfaction of certain additional constraints described below.

Database Properties. We encode the properties of parts using $f : P' \to \mathbb{B}$ for $f \in \{\texttt{prom}, \texttt{pcr}, \texttt{rbs}, \texttt{ter}\}$, which hold true only for promoters, protein coding regions, ribosome binding sites, and terminators parts (*i.e.* $\texttt{prom}(p)$ iff $\hat{p}$ is a promoter in the database). Similarly, we use $f' : P' \times S' \to \mathbb{B}$ for $f' \in \{\texttt{pos}, \texttt{neg}, \texttt{codes}\}$ to capture the regulation and expression interactions between parts and species (*e.g.* $\texttt{pos}(p, s)$ iff $\hat{s}$ activates $\hat{p}$, while $\texttt{codes}(p, s)$ iff $\hat{p}$ codes for $\hat{s}$). Finally, we encode reactions using functions $f'' : R' \times S' \to \mathbb{B}$ for $f'' \in \{\texttt{prod}, \texttt{reac}, \texttt{cat}\}$, where $\texttt{prod}(r, s)$, $\texttt{reac}(r, s)$ and $\texttt{cat}(r, s)$ iff $\hat{s}$ is a product, reactant or catalyst for $\hat{r}$.

Topology Abstraction. While a number of constraints can be defined using the basic properties outlined above, in the following we focus only on a small subset sufficient to illustrate the approach. We define

$$\texttt{expresses}(p, s) \leftrightarrow \exists p_0, p_1, p_2 \,.\, [\texttt{codes}(p_1, s) \wedge \texttt{nextTo}(p, p_0) \wedge \texttt{nextTo}(p_0, p_1) \wedge \texttt{nextTo}(p_1, p_2) \wedge \texttt{prom}(p) \wedge \texttt{rbs}(p_0) \wedge \texttt{pcr}(p_1) \wedge \texttt{ter}(p_2)]$$

to capture the expression of proteins (*i.e.* a protein species $s \in S'$ is expressed when a transcriptional cassette is formed that codes for s). To reason about species' regulation interactions, we define $\texttt{pos}_s(s, s')$ iff $\exists p.\, \texttt{pos}(s, p) \wedge \texttt{expresses}(p, s')$ and $\texttt{neg}_s(s, s')$ iff $\exists p.\, \texttt{neg}(s, p) \wedge \texttt{expresses}(p, s')$. We extend these properties to allow more complex regulation topologies involving intermediate species: $\texttt{neg}_{1s}(s, s')$ iff $\texttt{neg}_s(s, s') \vee \exists s''.([\texttt{pos}_s(s, s'') \wedge \texttt{neg}_s(s'', s)] \vee [\texttt{neg}_s(s, s'') \wedge \texttt{pos}_s(s'', s)])$ and $\texttt{pos}_{1s}(s, s')$ iff $\texttt{pos}_s(s, s') \vee \exists s''.([\texttt{pos}_s(s, s'') \wedge \texttt{pos}_s(s'', s)] \vee [\texttt{neg}_s(s, s'') \wedge \texttt{neg}_s(s'', s)])$ for one intermediate.

Dynamical constraints. Currently, we capture the dynamics of circuits using a Boolean encoding where we rep-

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

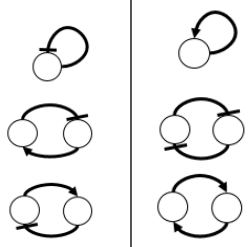
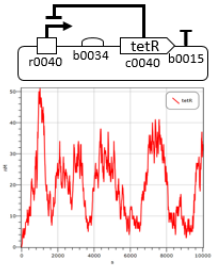
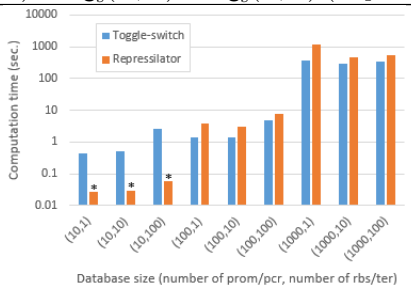
(A) Topology abstraction		(B) Dynamical constraints	(C) Scalability
$\texttt{neg}_{1s}(A,A)$	$\texttt{pos}_{1s}(A,A)$	$A[0] \wedge \neg A[1] \wedge A[2]$	$\texttt{neg}_s(A,B) \wedge \texttt{neg}_s(B,A)$ ("toggle switch") $\texttt{neg}_s(A,B) \wedge \texttt{neg}_s(B,C) \wedge \texttt{neg}_s(C,A)$ ("repressilator")
			

Figure 1: (A) Some of the circuit topologies generated using different logical specifications. (B) A circuit design generated using dynamical specifications capturing a cycle of oscillations (*i.e.* species A is enabled at steps 0 and 2 but is disabled at step 1) and a stochastic simulation of the circuit from GEC. (C) Scalability of the method on the design of two types of circuits from random databases (* indicate that no solutions were found). For each experiment, the specifications used are listed in the second row of the table.

resent a state of the system as a bit-vector $x \in S'$, which signifies whether each species is available or not. Transitions between states x and x' are captured by requiring that, in the next state x', a species s is available iff there exists a reaction producing s that is enabled (all its reactants and catalysts are available in x) or there exists a part that is enabled (all its positive and none of its negative regulators are available in x) and expresses s. This allows us to define the symbolic transition relation $T(x,x')$ as in [9] but capturing the properties of parts and reactions, and to reason about dynamical properties of circuits using predicates such as $A[k]$, denoting that species A is available at step k. Arbitrary combinations of logical and dynamical constraints (including stability and oscillations as described in [8]) are defined using the operators $\{\vee, \wedge, \rightarrow, \leftrightarrow, \neg\}$ and can be generalized using additional SMT operators such as quantifiers.

3. CASE STUDIES

In Fig. 1, we illustrate the approach by generating circuit designs from a number of the constraints defined in Sec. 2. Using logical constraints, we identified circuits of different topologies that implement the same abstract interaction between components, *e.g.* self activation or repression, (Fig. 1-A). We then used dynamical constraints to encode an oscillation cycle and identify a design that resembles a Goodwin oscillator [5] (Fig. 1-B). Note that in this example the dynamical constraint from Fig. 1-B only requires that if species A is initially available, its state changes to not available and back in three successive steps, which is consistent with the Boolean dynamics of the identified circuit. More restrictive constraints such as sustained oscillations can be expressed as described in [8]. Detailed models of designs generated using this procedure can be studied further using the (stochastic) simulation functionality of GEC [4] (Fig. 1-B) but, currently, the formal guarantees do not extend directly to these representations. Finally, we illustrate the scalability of the approach on randomly generated databases with up to thousands of parts, where the number of promoters and pcr parts (m) is the same, as is the number of rbs and ter parts (n), each promoter is randomly assigned between 0 and 2 positive or negative regulators, the total number of parts is $2m + 2n$, and the database size is denoted by (m,n). Our method still leads to the identification of "toggle-switch" and "repressilator"-type circuits as shown in Fig. 1-C, where the time required for the identification of the first valid design (or showing that no such design exists) is reported.

4. CONCLUSION

In this abstract, we presented an SMT-based approach to the automated design of biological circuits, which allowed the specification of logical constraints defining abstract interaction topologies, as well as dynamical constraints of required circuit behavior, and could handle databases with thousands of parts. The approach is currently implemented within GEC but could also contribute to other biodesign automation tools. Here, we only considered reachability properties over Boolean circuit dynamics but additional SMT theories such as non-linear arithmetic and floating point numbers could allow dynamical constraints over more detailed models, for example by exploiting characterization data collected as in [7]. We only illustrated the SMT encoding of select properties, which can be extended to capture more detailed expression and regulation mechanisms. Finally, our current implementation only generates a single candidate design, which can be refined through additional specifications, but the approach is also capable of enumerating multiple designs.

5. REFERENCES

[1] L. de Moura and N. Bjørner. Z3: An Efficient SMT Solver. In *TACAS*, volume 4963 of *LNCS*. Springer, 2008.

[2] D. Densmore and S. Bhatia. Bio-design automation: software + biology + robots. *Trends in biotechnology*, 32(3), 2014.

[3] C. J. Myers. Platforms for Genetic Design Automation. In *Microbial Synthetic Biology*, volume 40 of *Methods in Microbiology*. Academic Press, 2013.

[4] M. Pedersen and A. Phillips. Towards programming languages for genetic engineering of living cells. *J R Soc Interface*, 6, 2009.

[5] O. Purcell, N. J. Savery, C. S. Grierson, and M. di Bernardo. A comparative analysis of synthetic genetic oscillators. *J R Soc Interface*, 7(52), 2010.

[6] F. Yaman, S. Bhatia, A. Adler, D. Densmore, and J. Beal. Automated Selection of Synthetic Biology Parts for Genetic Regulatory Networks. *ACS Synth Biol*, 1(8), 2012.

[7] B. Yordanov, N. Dalchau, P. K. Grant, M. Pedersen, S. Emmott, J. Haseloff, and A. Phillips. A Computational Method for Automated Characterization of Genetic Components. *ACS Synth Biol*, 2014.

[8] B. Yordanov, C. M. Wintersteiger, Y. Hamadi, and H. Kugler. Z34bio: An SMT-based framework for analyzing biological computation. *11th International Workshop on SMT*, 2013.

[9] B. Yordanov, Y. Wintersteiger, Christoph M., Hamadi, and H. Kugler. SMT-based Analysis of Biological Computation. In *NASA Formal Methods Symposium*, 2013.

Exploration of Design Principles and an End-to-End Workflow for Synthetic Biology using a Combinational Logic Circuit Library

Swati B. Carr
MCBB Program
36 Cummington St
Boston, MA 02215
+1 (857) 225-5110
swati610@bu.edu

Calin Belta
Dept of Mechanical Engineering
110 Cummington Mall
Boston, MA 02215
+1 (617) 353-9586
cbelta@bu.edu

Douglas M. Densmore
Electrical and Computer Engineering
8 St Mary St
Boston, MA 02215
+1 (617) 358-6238
dougd@bu.edu

ABSTRACT
In principle, the combined behavior of parts within a synthetic genetic circuit can be predicted from the independently characterized component parts. In practice, however, circuit function is not predictable; circuit assembly involves tedious trial and error and ad hoc tuning [6; 8; 10]. Synthetic biology is missing a set of design principles for its toolkit of genetic parts that will provide guidelines to get from parts to desired function faster. We are designing, building and testing a large set of NOT-gates (inverters) with two goals: (1) mining for patterns based on design and circuit expression, and (2) establishing a streamlined flow integrating automation and computer-aided tools wherever possible.

Keywords
Synthetic biology, logic gates, combinatorial logic.

1. INTRODUCTION
In principle, the combined behavior of genetic parts can be predicted from the independent characterization of each individual part or smaller units of genetic expression within a larger combinational genetic circuit [6; 8; 10]. However, in practice, we are currently unable to predict the functions of even the simplest combinational circuits. Synthetic circuits are either assembled ad-hoc and then tested and modified, or many iterations of the same circuit are built and testing in an effort to find the one that functions as expected [1; 7]. More importantly, since the behavior of a part changes when the context is changed, each already characterized part needs to be re-characterized in its final context [8]. The functions of genes are affected by their genetic, cellular and environmental contexts [2].

A possible approach to reducing the context-dependence is by using insulator elements that buffer synthetic circuits from genetic context [8]. While this method works very well in circuit designs in which genetic context varies in only one location, this approach may not be practical when synthesizing circuits that are large and have multiple points at which context-dependent variations may manifest. Another approach is to develop mathematical models based on experimental data from synthetic circuits that will allow us to accurately predict expression variations due to genetic context and guide us in choosing genetic parts accordingly.

2. CIRCUIT DESIGN
We have selected a four transcriptional unit (TU) design for our inverter circuits. The first transcriptional unit is a double inversion module driving the expression of the repressor inverting the signal. The final output is the RFP gene downstream of the repressor's cognate repressible promoter. GFP indirectly measures the level of repressor expressed by the circuit. We have removed all cis-regulation from our inverter designs.

With 4 TUs, 10 possible inducer-repressor combinations and 5 RBSes, the design space for our inverter library is very large. To avoid errors and omissions, and to make the task of designing and planning the assembly of our circuit library faster, we used the BioCAD tools **Eugene**, **Raven** (paper submitted) and **Pigeon**.

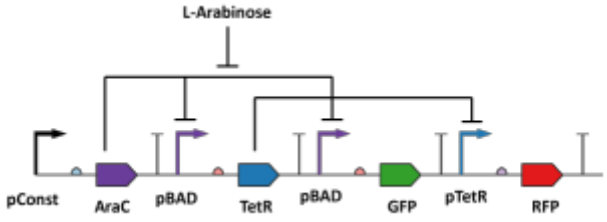


Figure 1: New Four Transcriptional unit (monocistronic) inverter. 5 RBSes allow for 5 expression level variants of each transcriptional unit, allowing for 625 variants of each inverter.

3. DEVELOPING A STREAMLINED SYNTHETIC BIOLOGY WORKFLOW
As the size and number of genetic circuits being built in tandem by synthetic biologists grows in number, it will become challenging to design and build them with intuition alone [9]. The end goal is to have an end-to-end workflow that is automated in all the key steps of **design**, **build**, and **test** in high-throughput volumes.

We provided Eugene the abstract circuit design and the list of parts. Eugene rules were used to create constraints to eliminate biologically nonsensical or non-orthogonal designs. Eugene designs were used in the assembly planner tool RavenCAD to generate an assembly plan including required oligos and effective use of intermediate parts already available. The Raven assembly

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

plan forms the template for Puppeteer, which generates the command for assembly of the circuits in a liquid handling robot.

A subset of the inverters were built in the lab and tested by flow cytometry in the lab. The results were analyzed using the tools developed by **BBN Technologies** [4] and data sheets will be generated using the data sheet generator, **Owl**. The expression data from the inverters will be used for pattern mining and the development of mathematical verification of designs [3; 5]. All parts, designs and intermediates will be stored on the CoSBi ICE Registry.

The goal is to have a process wherein a mathematical verification model is in place that can reject designs from a design space before assembly based on data models developed during our current work.

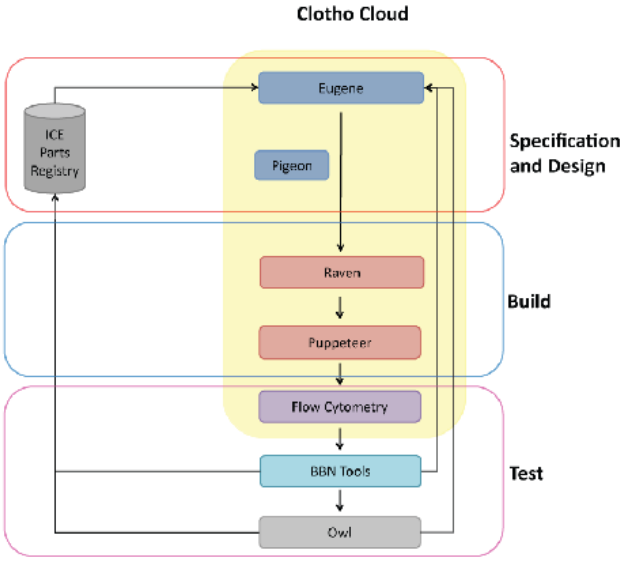


Figure 2: Streamlined workflow for synthetic biology. BioCAD tools and automation will be used wherever possible to create a seamless workflow

4. ACKNOWLEDGMENTS

I would like to thank Jake Beal for help with flow cytometry data analysis, Traci Haddock for advice, Ernst Oberortner for help with writing Eugene code, Swapnil Bhatia for his web tool PigeonCAD and Evan Appleton for his assembly tool RavenCAD.

5. REFERENCES

[1] ANDRIANANTOANDRO, E., BASU, S., KARIG, D.K., and WEISS, R., 2006. Synthetic biology: new engineering rules for an emerging discipline. *Mol Syst Biol 2*, 2006 0028. DOI= http://dx.doi.org/msb410007310.1038/msb4100073.

[2] ARKIN, A., 2008. Setting the standard in synthetic biology. *Nat Biotechnol 26*, 7 (Jul), 771-774. DOI= http://dx.doi.org/10.1038/nbt0708-771.

[3] AYDIN GOL, E., DENSMORE, D., and BELTA, C., 2013. Data-driven verification of synthetic gene networks. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, 4074-4079. DOI= http://dx.doi.org/10.1109/CDC.2013.6760513.

[4] BEAL, J., WEISS, R., DENSMORE, D., ADLER, A., APPLETON, E., BABB, J., BHATIA, S., DAVIDSOHN, N., HADDOCK, T., LOYALL, J., SCHANTZ, R., VASILEV, V., and YAMAN, F., 2012. An end-to-end workflow for engineering of biological networks from high-level specifications. *ACS Synth Biol 1*, 8 (Aug 17), 317-331. DOI= http://dx.doi.org/10.1021/sb300030d.

[5] BOYAM YORDANOV, E.A., RISHI GANGULY, EBRU AYDIN GOL, SWATI BANERJEE CARR, SWAPNIL BHATIA, TRACI HADDOCK, CALIN BELTA, DOUGLAS DENSMORE, 2012. Expreimentally Driven Verification of Synthetic Biological Circuits. In *Proceedings of the IEEE* (2012).

[6] ENDY, D., 2005. Foundations for engineering biology. *Nature 438*, 7067 (Nov 24), 449-453. DOI= http://dx.doi.org/nature0434210.1038/nature04342.

[7] KOBAYASHI, H., COLLINS, J.J., 2004. Programmable cells: Interfacing natural and engineered gene networks. *Proceedings of the National Academy of Sciences 101*, 22, 8414-8419. DOI= http://dx.doi.org/10.1073/pnas.0402940101.

[8] LOU, C., STANTON, B., CHEN, Y.J., MUNSKY, B., and VOIGT, C.A., 2012. Ribozyme-based insulator parts buffer synthetic circuits from genetic context. *Nat Biotechnol*(Oct 3). DOI= http://dx.doi.org/10.1038/nbt.2401nbt.2401.

[9] PURNICK, P.E.M. and WEISS, R., 2009. The second wave of synthetic biology: from modules to systems. *Nature Reviews Molecular Cell Biology 10*, 6, 410-422. DOI= http://dx.doi.org/10.1038/nrm2698.

[10] VOIGT, C.A., 2006. Genetic parts to program bacteria. *Current Opinion in Biotechnology 17*, 5, 548-557. DOI= http://dx.doi.org/10.1016/j.copbio.2006.09.001.

Nonlinear Biochemical Signal Processing via Noise Propagation

Kyung Hyuk Kim
Department of Bioengineering, University of Washington
Seattle, WA 98004
kkim@uw.edu

Hong Qian
Department of Applied Mathematics, University of Washington
Seattle, WA 98004
hqian@u.washington.edu

Herbert M. Sauro
Department of Bioengineering, University of Washington
Seattle, WA 98004
hsauro@u.washington.edu

ABSTRACT

We describe an intuitive, yet fully quantitative method for analyzing how biological noise affects cellular phenotypes based on identifying a system's nonlinearity and the signal propagation. We observe that noise can simultaneously enhance sensitivities in one behavioral region while reducing sensitivities in another. Using this effect we design and analyze various signal processing modules in gene regulatory networks and to verify the theory we construct synthetic genetic circuits in E. coli. The method developed here allows one to understand and engineer nonlinear biochemical signal processors based on noise-induced phenotypes.

Keywords

noise propagation, nonlinear signal processing, stochasticity, bistability, noise induced phenotype, linearizer

1. INTRODUCTION

Single-cell studies often show significant phenotypic variability due to the stochastic nature of intra-cellular biochemical reactions. When the numbers of molecules, e.g., transcription factors and regulatory enzymes, are in low abundance, fluctuations in biochemical activities become significant and such fluctuations âĂŞ noise âĂŞ can propagate through regulatory cascades, resulting in certain cellular phenotypes. Examples include cellular differentiation and lytic-lysogeny decision of virus infected cells. Noise-related phenotypes are often closely related to âĂŸnonlinearâĂŹ signal processing, but most studies on this subject have taken either computational or mathematical approaches that often lack systematic understanding of underlying mechanisms. Here, we provide a manageable and systematic approach in relation to noise-induced phenotypes.

2. RESULTS AND DISCUSSION

When a signal is processed nonlinearly, the signal shape can be deformed significantly; for example, an input signal that fluctuates in a sine wave can be transformed to a signal that fluctuates asymmetrically. This makes the mean level of the output different from the case without fluctuations, resulting in different input-output response curves at the average level for different strength of the signal fluctuations. When signals are noisy, a similar effect appears [1, 3]. We observe a typical pattern of the input-output response curve change: Noise can simultaneously enhance sensitivities in one behavioral region while reducing sensitivities in another. We name this phenomenon "stochastic focusing compensation" (SFC) [2]. SFC can be intuitively understood as a geometric effect of the reaction rate [2]. Mathematically, it is closely related to the Jensen's inequality.

We applied SFC to three signal processing modules. (1) We considered an incoherent feed-forward network that can function as a concentration detector. We enhanced the detection amplitude and sensitivity significantly by exploiting noise (Figure 1a, b). Here, the sensitivity refers to the relative change of steady-state response to parameter variation. (2) We investigated a mutual inhibition network that cannot show bistability without noise. SFC was used to induce bistability (Figure 1c, d). (3) We achieved linear amplification in simple cascade networks of various types of activation patterns without resort to any feedback (Figure 1e, f).

To verify the theories, synthetic genetic networks have been constructed in *E. coli* (MG1655Z1). Our preliminary experiments show that the strength of noise in gene expression levels can be controlled by using a library of ribosome binding sites, inducible promoters, and different copy-number plasmids (Figure 2). By applying this noise control mechanism, SFC is currently in the verification process.

3. CONCLUSIONS

Our quantitative approach for studying noise propagation showed that noise can increase or decrease the sensitivity (Hill coefficient) of transcription factor regulation. Our analysis method were applied to various gene regulatory networks to design noise-induced phenotypes. These examples illustrate how our sensitivity-based approach can be used to design, analyze, and optimize the functionalities of stochastic reaction networks by exploiting noise.

4. ACKNOWLEDGMENTS

The authors acknowledge the support from the National Science Foundation (NSF) (Molecular and Cellular Biosciences

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

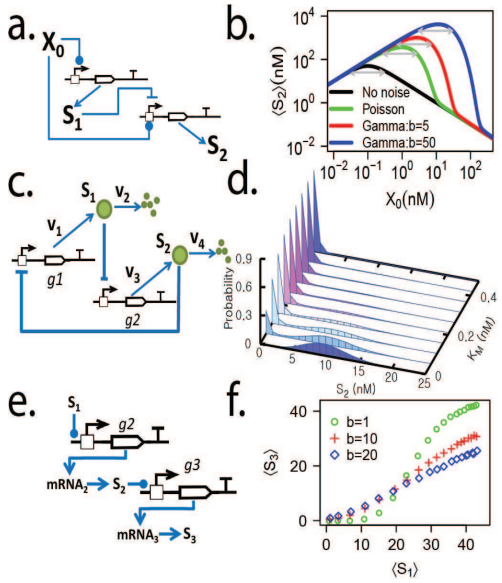


Figure 1: Stochastic focusing compensation: (a,b) noise-enhanced concentration detector, (c,d) noise-induced bimodality, and (e,f) noise-induced linearizer.

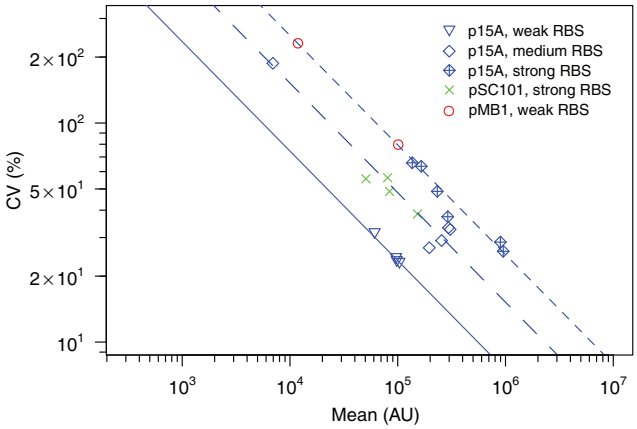


Figure 2: Mean vs. noise levels of GFP (BioBrick part, E0040)-expression under the *lac*-promoter (R0010) from plasmids of different copy-numbers in *E. coli* MG1655Z1 that constitutively expresses LacI. The noise and mean levels are inversely related (the lines of slope -0.5 were drawn for comparison). The noise level increases with the copy number of plasmids (weak RBS cases) and with transcription strength.

1158573 and Theoretical Biology 0827592).

5. REFERENCES

[1] S. P. Arold, E. Bartolák-Suki, and B. Suki. Variable stretch pattern enhances surfactant secretion in alveolar type ii cells in culture. *American Journal of Physiology-Lung Cellular and Molecular Physiology*, 296(4):L574, 2009.

[2] K. H. Kim, H. Qian, and H. M. Sauro. Nonlinear biochemical signal processing via noise propagation. *The Journal of chemical physics*, 139(14):144108, 2013.

[3] J. Paulsson, O. G. Berg, and M. Ehrenberg. Stochastic focusing: fluctuation-enhanced sensitivity of intracellular regulation. *Proceedings of the National Academy of Sciences*, 97(13):7148–7153, 2000.

An Aspect Oriented Design and Modelling Framework for Synthetic Biology

Extended Abstract

Philipp Boeing
Department of Computer Science
University College London

Darren Nesbeth
Department of Biochemical Engineering
University College London

Anthony Finkelstein
Department of Computer Science
University College London

Chris P Barnes
Department of Cell and Developmental Biology
University College London

ABSTRACT

Up until this point, work on synthetic biology has largely used a component-based metaphor for system construction. Here we present a design framework that steps away from this paradigm by utilising aspect oriented software engineering concepts. Though our approach does not rest upon this, we think that the notion of *concerns* is a powerful and biologically credible way of thinking about systems synthesis. By casting the design question in terms of concerns, our framework incorporates and builds upon many features of existing tools. This framework is written in Python which facilitates the adoption of standard software engineering tools and the interaction with existing biological modelling systems. The benefits of our approach are numerous and include improved conceptualisation of synthetic biological systems and a design workflow that iteratively adds complexity by adding more aspects to the design. We demonstrate the effectiveness of this framework by creating an oscillator system based around the *repressilator*.

1. INTRODUCTION

Synthetic biology aims to create a biological engineering field based on principles such as modularisation and abstraction. Until now, it has predominantly used a component-based metaphor inspired by electronic circuits: a genetic part encapsulates a specific sequence of DNA associated with a particular function, and parts are the basic building blocks of genetic circuits. As the ambition and scale of synthetic biology projects grows, better techniques for managing complexity are becoming increasingly necessary. Here we introduce a framework based on the principles of aspect oriented engineering which incorporates and builds upon many features of existing design tools. These include the constrained construction of biologically valid devices [4, 9], combinatorial design through abstract gene regulatory networks [3, 1] and allows for hierarchical, modular and interchangeable modelling of biological systems [8, 5, 7]. This framework is implemented in Python and supports the export of models in both SBOL and SBML.

2. ASPECT ORIENTED SOFTWARE ENGINEERING

AOSE relates to the modularization of concerns, defined as "specific requirements or considerations that must be addressed in order to satisfy an overall system goal." (AspectJ in Action). There are generally two categories of concerns: core concerns and cross-cutting concerns. Core concerns embody the main functionality of the system and in synthetic biology could refer to the gene network under construction. Cross-cutting concerns are often "system-level, peripheral requirements that cross multiple modules", (e.g. placing of terminators or interaction with a chassis system). Whereas OOP (Object Oriented Programming) is a suitable paradigm to modularize core concerns, it can not modularize cross-cutting concerns. Instead, code dealing with cross-cutting concerns gets tangled up in implementation of core concerns. AOP (Aspect Oriented Programming), the programming paradigm derived from AOSE, introduces new abstractions that achieve the modularization of all concerns, including cross-cutting concerns. AOP is generally implemented as an extension of OOP, although this is not strictly necessary.

3. THE PYTHON WEAVER FRAMEWORK

To achieve a modularization of core and cross-cutting concerns, we loosely follow the AOP model of AspectJ, an aspect oriented extension of the Java programming language, and introduce the following new constructs, on top of established concepts such as classes, interfaces and methods: Join Points and Point Cuts and Advice. A Join Point is an "identifiable point in the execution of a program". Traditionally, this refers to a method call, object instantiation, etc. In our synthetic biology specific framework, we consider a genetic circuit made of parts as a sequence of execution steps, with join points between them (Figure 1). A Point

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

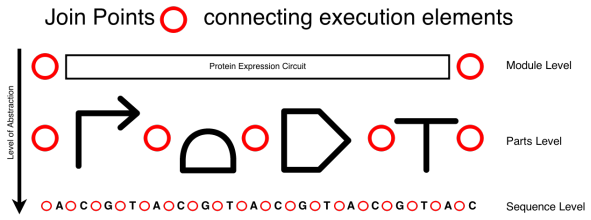


Figure 1: Possible join points in synthetic biology applications

Cut is a construct that selects particular Join Points. Then, an Advice can be used to define code that will be injected at particular Point Cuts. Point Cuts and Advice are bundled into class-like structures called Aspects. We have developed a "Weaver", implemented as a Python class, that weaves in the Advice code at the Join Points in the execution flow selected by the Point Cuts (Figure 2).

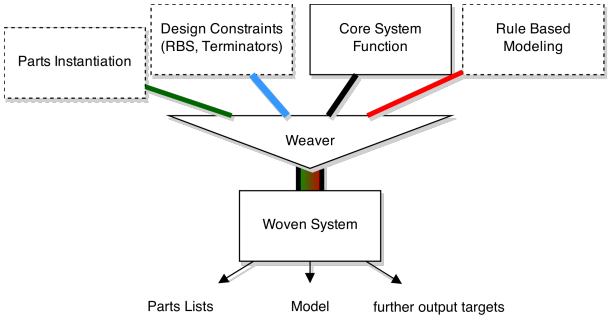


Figure 2: Conceptual model of the Python Weaver

4. REPRESSILATOR EXAMPLE

We demonstrate the utility of our novel framework on the design of a switchable oscillator system based around the *repressilator* [6]. We consider the following concerns:

- Oscillation abstract gene regulatory network (using abstract, non-determined transcription factors);
- Terminator and RBS design rules;
- Instantiation of abstract molecules;
- Two output versions: a standard GFP reporter, or alternatively, a GFP reporter dependent on an external inducer (that is, an additional on-switch).

The design starts with the two GFP coding concerns; one constitutively expressed GFP and another with a GFP downstream of an inducible promoter. An aspect is created that can weave the oscillation concern into either of the two output circuits. The aspect is able to change the promoter of the output circuit to make the output dependent on oscillation. If the promoter is unregulated, it must be replaced by one that is regulated by the oscillation output. However, if the promoter is already regulated, we need to insert an additional AND gate, based on the design by Wang *et. al.* [10], to preserve the original input. Lastly, we implement an aspect that can instantiate abstract molecules. Figure 3 shows a visualisation of weaving the aspects with the regulated GFP circuit. It was generated using an additional Weaver Output Advice, which generates a Pigeon visualisation specification for the design [2].

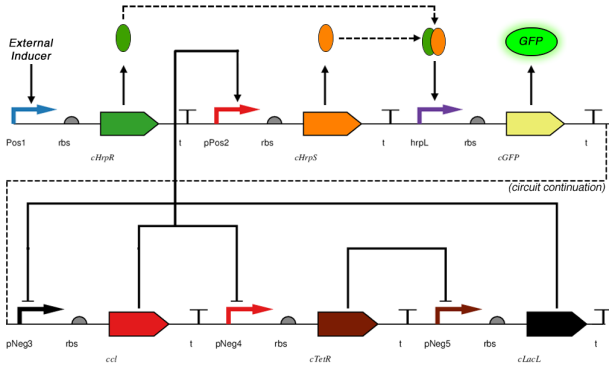


Figure 3: Visualisation of the repressilator example.

5. REFERENCES

[1] J. Beal, R. Weiss, D. Densmore, A. Adler, E. Appleton, J. Babb, S. Bhatia, N. Davidsohn, T. Haddock, J. Loyall, R. Schantz, V. Vasilev, and F. Yaman. An End-to-End Workflow for Engineering of Biological Networks from High-Level Specifications. *ACS synthetic biology*, 1(8):317–331, Aug. 2012.

[2] S. Bhatia and D. Densmore. Pigeon: A Design Visualizer for Synthetic Biology. *ACS synthetic biology*, 2(6):348–350, June 2013.

[3] L. Bilitchenko, A. Liu, S. Cheung, E. Weeding, B. Xia, M. Leguia, J. C. Anderson, and D. Densmore. Eugene–a domain specific language for specifying and constraining synthetic biological parts, devices, and systems. *PloS one*, 6(4):e18882, 2011.

[4] Y. Cai, B. Hartnett, C. Gustafsson, and J. Peccoud. A syntactic model to design and verify synthetic genetic constructs derived from standard biological parts. *Bioinformatics*, 23(20):2760–2767, Oct. 2007.

[5] D. Chandran and H. M. Sauro. Hierarchical Modeling for Synthetic Biology. *ACS synthetic biology*, 1(8):353–364, Aug. 2012.

[6] M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, Jan. 2000.

[7] C. F. Lopez, J. L. Muhlich, J. A. Bachman, and P. K. Sorger. Programming biological models in Python using PySB. *Molecular Systems Biology*, 9:646, 2013.

[8] 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.

[9] M. Pedersen and A. Phillips. Towards programming languages for genetic engineering of living cells. *Journal of the Royal Society, Interface / the Royal Society*, 6 Suppl 4:S437–50, Aug. 2009.

[10] B. Wang, R. I. Kitney, N. Joly, and M. Buck. Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology. *Nature communications*, 2:508, 2011.

Towards Rule-based Knowledge-Based Systems for Synthetic Biology

Ernst Oberortner, Audrey Lewis, Douglas Densmore
Department of Electrical and Computer Engineering
Boston University
{ernstl,alewis13,dougd}@bu.edu

ABSTRACT

The design of novel synthetic biological systems involves various design decisions. Biologists mine knowledge usually from the literature and utilize the acquired knowledge in the decision making process, making it hard to reenact the design rationale of functioning biological systems. Moreover, expert knowledge gets lost. We introduce the applicability of Knowledge-Based Systems (KBS), enabling to electronically share, exchange, and communicate design rationale formulated as rules. The ultimate goal is to computationally support biologists in the decision making process and to understand and learn how and why biological systems function or fail to function.

1. MOTIVATION

The design paradigm of synthetic biological systems proposes to recursively decompose a system under design into encapsulated sub-components that either perform basic biological functions ("parts") or human-engineered functions ("devices") [5]. To responsibly engineer robust, reliable, and secure biological systems, parts and devices must then be composed according to biological "rules". Rules specify conditions, characteristics, and interactions of the parts' and devices' function in isolation and when composed together. We infer that design-specific rules encapsulate knowledge on the functioning of synthetic biological systems.

When iteratively composing parts and devices into complex systems, biologists face various design decisions. Specifically:

- The **Selection** design decision focuses on selecting appropriate parts, devices, or sub-systems based on desired characteristics. To provide some examples: What type must the parts have? Should only "strong" promoters be selected? How do the selected promoters influence the expression rate of the selected coding sequences?
- The **Arrangement** design decision deals with an appropriate ordering of a system's parts, devices, and sub-systems. To provide some examples: Does every gene require an upstream promoter? What orientation must the components have? When and where should desired regulatory interactions occur?

For each design decision it is hard to choose among appropriate alternatives with no or little knowledge on the biological function. Currently, biologists mine the literature in the decision making process or follow trial-and-error and combinatorial design approaches. Design specifications of biological systems and their building-blocks must be enriched with rules, enabling to infer and reason biological knowledge based on existing knowledge. Such approach will computationally support synthetic biologists in the decision making processes of forward engineering novel systems. Also, an automated and standardized knowledge transfer will facilitate learning and understanding of the functioning and language of biology.

2. BACKGROUND ON KBS

Knowledge-Based Systems (KBS) are clearly separated into a Knowledge-Base (KB) and an Inference Engine (IE). Also, a KBS usually provides interfaces for humans and machines, allowing them to interact with the KBS [2].

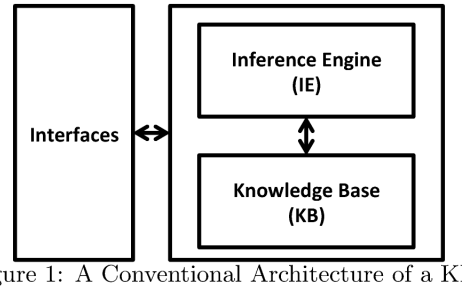


Figure 1: A Conventional Architecture of a KBS

The KB of a rule-based KBS contains various types of knowledge. *Facts* are problem-specific and immutable pieces of information and originate, for example, from experiments, observations, or novel discoveries. Depending on the offered interfaces, facts can either be inserted programmatically or manually, as well as by the IE. *Rules* can encapsulate general knowledge as well as specify constraints about valid relations among the facts. Rules are usually specified in the following form:

IF <CONDITION> THEN <CONCLUSION>

Conclusions are implications of conditions and can utilized to nest rules and to perform appropriate actions if the conditions are satisfied.

Depending on the application of the KBS, the IE performs various tasks. For example, an IE asserts the conditions of the rules on the facts in the KB and performs the rules' conclusions or actions. By specifying appropriate conditions, conclusions, and actions, an IE can infer novel facts or rules automatically, for example, by machine learning algorithms.

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

A prominent subset KBS are so-called *Expert Systems*, in which the knowledge — either facts or rules — stems from domain experts, such as synthetic biologists. Usually expert systems also provide additional interfaces for the domain experts.

Depending on the requirements and application, a KBS consists of additional components. For example, an "Explanation" component can describe how certain conclusions were deduced or why certain actions were performed.

3. CASE STUDY

Now, we demonstrate the applicability of a KBS on a case study designing a genetic AND gate [1]. Currently, we develop *Sparrow*, a rule-based KBS for synthetic biology offering, at the time of this writing, a primitive set of interfaces that can be accessed programmatically. *Sparrow* is build upon the JBoss Drools rule engine[1], enabling rapid prototyping of synthetic biology designs and their associated rules.

First, we manually specify a library of 13 parts in Eugene [3] and insert them as *facts* into the Sparrow KB. We also specify relationships among four orthogonal repressor-promoter pairs as presented in [7]. For example, the $AmeR$ repressor gene is orthogonal to the $pAmeR$ promoter.

Next, we select appropriate parts from the library. For example, the Sparrow query TYPE=="PROMOTER" returns all promoter parts. According to [7], an AND gate consists of 17 parts arranged in a specific order. We specify appropriate positioning constraints in Eugene. Based on the number of parts in the library, Eugene returns $12,288$ AND gates, which we store as *facts* into the Sparrow KB.

However, most of the $12,288$ AND gates contain common failure modes of genetic circuits [4]. Hence, we engineered knowledge from [4], formulated two common failure modes as rules, and stored them into the Sparrow KB. We have chosen the "Orthogonality" and "Recombination" failure modes since both encapsulate knowledge to support the Selection and Arrangement decision making in designing genetic circuits. We formulate the rules as follows:

$$\begin{gathered}\text{IF } R1 \text{ is not orthogonal to } pR1 \wedge \\ R2 \text{ is not orthogonal to } pR2 \wedge \\ R3 \text{ is not orthogonal to } pR4 \wedge \\ R1 \text{ is not equal to } R2 \wedge \\ R2 \text{ is not equal to } R3 \wedge \\ R3 \text{ is not equal to } R1 \\ \text{THEN The design is valid}\end{gathered}$$

The first three conditions formulate the "Orthogonality" failure mode and the last three conditions specify the "Recombination" failure mode. All conditions are composed using the logical AND ($\wedge$) operator. In this case study, we conclude that an AND gate design is valid if it complies with those five specified conditions. Based on the specified rule, the IE of the Sparrow KBS finds 144 rule-compliant AND gate designs.

As demonstrated, six conditions — engineered from the literature [4] — prune the $12,288$ AND gates to 144 AND gates, also enabling to computationally exchange, communicate, and share the formulated rule.

[1] `https://www.jboss.org/drools/`

4. CONCLUSION

Galdzicki et al. [6] utilize semantic web technologies to develop a KB of standard biological parts, enabling efficient querying of biological parts based on characteristics, such as type, DNA sequence, or orientation.

One drawback of a KBS for synthetic biology lies in the development and maintenance costs, such as the initial insertion of biological knowledge. Furthermore, the probability that a KBS will provide wrong answers because of faulty inferences and reasons will never disappear. A KBS is most powerful when its KB, IE, and interfaces are tailored for a very specific application domain or biological organisms in synthetic biology. Here, we demonstrated how to engineer rules from the literature. However, knowledge must be engineered, represented, and specified in close collaboration with domain experts.

Benefits of a synthetic biology KBS lie in a standardized, automated knowledge transfer of how and why synthetic biological systems work or fail under a specific set of conditions. A KBS can then be further utilized to *explain* how and when synthetic biological systems function and behave as programmed. Explanations can then be further utilized to *learn* common design principles to design and build robust, reliable, and secure synthetic biological systems. Learned design principles can then then be further utilized for *education*. We believe that Knowledge-based Systems are crucial in engineering functioning, reliable, and secure synthetic biological systems, will help to save time and money, and will enable the automate and standardize synthetic biology engineering processes.

Acknowledgements

The work was funded under NSF grant 1147158 and by the Agilent Technologies' Applications and Core Technology University Research (ACT-UR) program.

5. REFERENCES

[1] Anderson, J. C., Voigt, C., and Arkin, A. P. Environmental signal integration by a modular AND gate. *Molecular systems biology 3*, 133 (Jan. 2007), 133.

[2] Beierle, C., and Isberner, G. K. *Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen*. Vieweg, 2006.

[3] Bilitchenko, L., Liu, A., Cheung, S., Weeding, E., Xia, B., Leguia, M., Anderson, J. C., and Densmore, D. Eugene: A domain specific language for specifying and constraining synthetic biological parts, devices, and systems. *PLoS ONE 6*, 4 (04 2011), e18882.

[4] Brophy, J. A. N., and Voigt, C. A. Principles of genetic circuit design. *Nature Methods 11*, 5 (Apr. 2014), 508–520.

[5] Endy, D. Foundations for engineering biology. *Nature 438*, 7067 (2005), 449–453.

[6] Galdzicki, M., Rodriguez, C., Chandran, D., Sauro, H. M., and Gennari, J. H. Standard biological parts knowledgebase. *PLoS ONE 6*, 2 (02 2011).

[7] Stanton, B. C., Nielsen, A. A. K., Tamsir, A., Clancy, K., Peterson, T., and Voigt, C. A. Genomic mining of prokaryotic repressors for orthogonal logic gates. *Nat Chem Biol 10*, 2 (Feb. 2014), 99–105.

Boston University Center of Synthetic Biology (CoSBi) ICE Repository

Stephanie Paige, Sonya Iverson, Aaron Heuckroth, Swati Carr, Traci Haddock, Douglas Densmore
Center of Synthetic Biology
Boston University
Boston, MA 02215 USA
spaige, siverson, awh, swati610, thaddock, dougd@bu.edu

INTRODUCTION

Synthetic Biology as a discipline is witnessing the emergence of numerous repositories of biological "Parts" [1]. Organizations hosting such services include the International Genetically Engineered Machine Competition (iGEM) (*partsregistry.org*), the Joint BioEnergy Institute (JBEI) (*public-registry.jbei.org*), the Joint Genome Institute (JGI) (*jgi.doe.gov*), the Synthetic Biology Engineering Research Center (SynBERC) (*registry.synberc.org*), and Cambridge University (*plantfab.org*). As a result, a "web of registries" concept has emerged which would allow locally curated registries to be "linked" so that information exchange is facilitated more easily and standards are established. Moreover this would send a strong message to the community that interacting registries based around similar technologies represent not only a means to quickly deploying registries but more importantly a step in the unification of the space.

We present our deployment of a repository based on the Inventory of Composable Elements (ICE) [2] architecture here at the Boston University Center of Synthetic Biology (CoSBi ICE). CoSBi ICE will serve as the outward facing parts repository for the core faculty of CoSBi (Jim Collins, Douglas Densmore, Ahmad Khalil, and Wilson Wong) as well as many of the numerous affiliated faculty members in bioinformatics, physics, and molecular cell biology and biochemistry. It will be used by over 50 undergraduate, graduate, and postdoctoral researchers and in projects funded by the National Science Foundation (NSF), Office of Naval Research (ONR), and the Defense Advanced Research Projects Agency (DARPA).

Specifically CoSBi ICE will have a number of attributes that make it a unique entry amongst ICE based registries:

- CIDAR MoClo library - this is a collection of over 154 MoClo [3] Parts publically available electronically through the registry and physically via Addgene plasmid numbers directly integrated into the repository.
- Integration with Eugene [4] web services to create Part and Device permutations based on constraints which are electronically stored. This provides a built in design space exploration mechanism.
- Integration path with Benchling (*benchling.com*) to serve as a workflow to connect a working commercial design suite to an open source parts repository. This will build upon community data exchange standards such as SBOL (*sbolstandard.org*) [5].
- Integration with Clotho 3.0 to exchange data with numerous Clotho apps including Raven (*ravencad.org*) [6] and Puppeteer via a lightweight Javascript API.

CoSBi ICE enhancements will be provided directly to the larger ICE community so that future instances of ICE will also be able to use these features as future official ICE releases.

REFERENCES

1. Peccoud J, Blauvelt MF, Cai Y, Cooper KL, Crasta O, et al. (2008) Targeted development of registries of biological parts. PLoS One 3: e2671.
2. Ham TS, Dmytriv Z, Plahar H, Chen J, Hillson NJ, et al. Design, implementation and practice of JBEI-ICE: an open source biological part registry platform and tools. Nucleic Acids Res 40: e141.
3. Weber E, Engler C, Gruetzner R, Werner S, Marillonnet S A modular cloning system for standardized assembly of multigene constructs. PLoS One 6: e16765.
4. Bilitchenko L, Liu A, Cheung S, Weeding E, Xia B, et al. Eugene--a domain specific language for specifying and constraining synthetic biological parts, devices, and systems. PLoS One 6: e18882.
5. Quinn, J., et al., *Synthetic Biology Open Language Visual (SBOL Visual)*. BioBricks RFC 93, 2013. Version 1.0.0.
6. Appleton, E., et al., *Interactive Assembly Algorithms for Molecular Cloning*. Nature Methods, 2014.

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

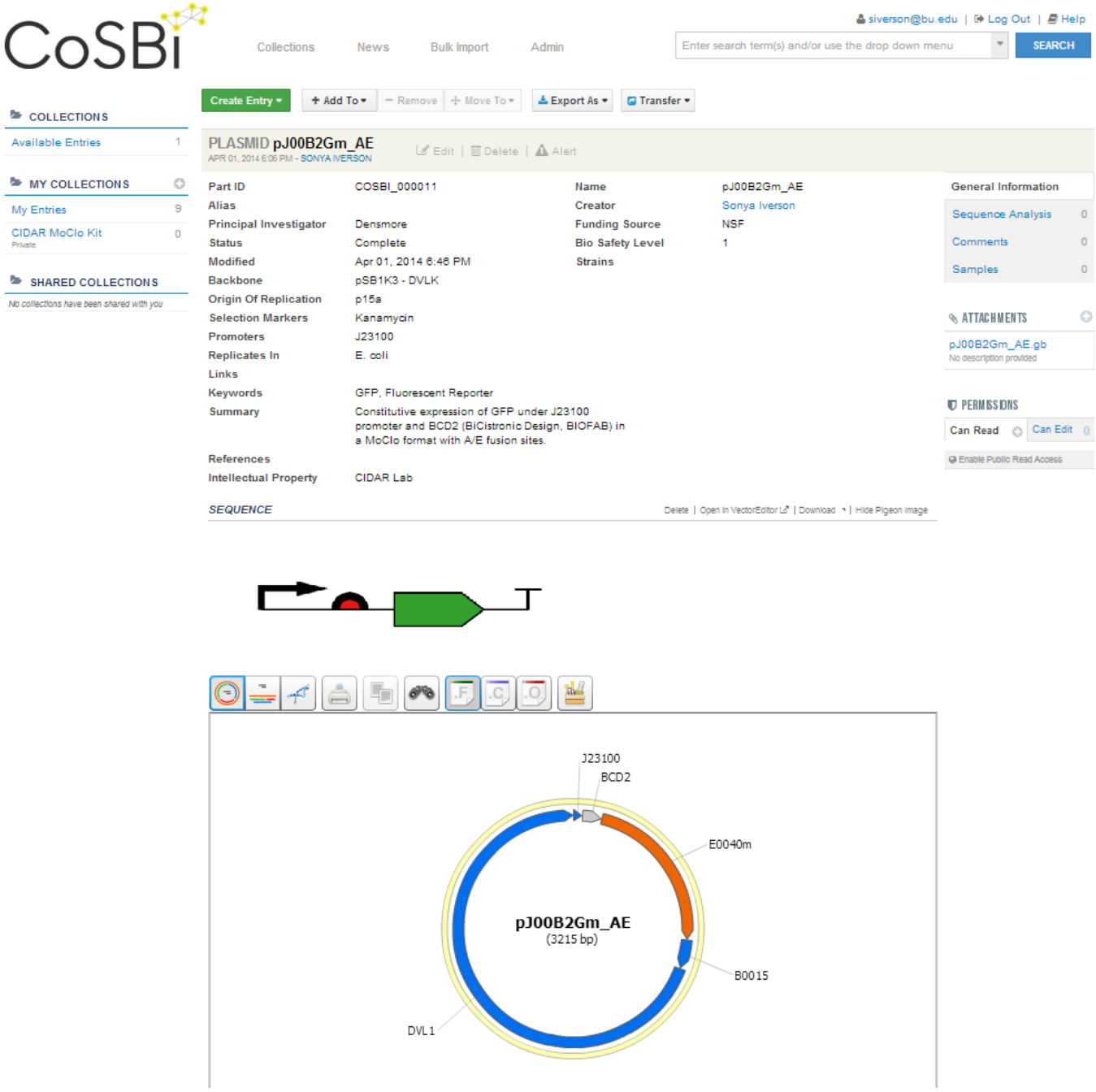


Figure 1: Screenshot of the CoSBi ICE initial deployment. It uses the basic ICE architecture while also including backend extensions to make it work with a number of bio-design automation initiatives at Boston University. The registry is located at https://cidar.bu.edu/cosbi-registry/

A Distributed and Interconnected Biological Part Registry

Hector A. Plahar
Joint BioEnergy Institute
Emeryville, CA
haplahar@lbl.gov

Joanna Chen
Joint BioEnergy Institute
Emeryville, CA
joannachen@lbl.gov

Timothy S. Ham
Joint BioEnergy Institute
Emeryville, CA

Zinovii Dmytriv
Joint BioEnergy Institute
Emeryville, CA

Jay D. Keasling
Joint BioEnergy Institute
Emeryville, CA
jdkeasling@lbl.gov

Nathan J. Hillson
Joint BioEnergy Institute
Emeryville, CA
njhillson@lbl.gov

ABSTRACT
In this article, we describe the Inventory of Composable Elements (ICE)[1], a biological part registry that features a web based infrastructure that enables interaction with other registries.

Keywords
Synthetic Biology, DNA, Part Registry, Distributed Services

1. INTRODUCTION
Advances in synthetic biology research and genetic engineering, along with the availability of rapid and reliable DNA sequencing has resulted in a steady increase in the number of complex engineered biological devices. Maintaining information about the constituent parts of these devices, along with the ability to share them with other members in the synthetic biology community as well characterized components with the goal of reusability and interoperability, is a challenge that is not fully addressed by existing registry software platforms.

2. ICE REGISTRY PLATFORM
The Inventory of Composable Elements (ICE) is an open source registry platform for storing and managing information about biological parts. It provides a feature rich and extensible web based application platform with support for storing generic biological parts in addition to plasmids, microbial strains and *Arabidopsis* seeds.

A major aim of ICE is to provide the Synthetic Biology community with a registry platform that provides robust data storage for DNA components, integrated tools for part characterizing and verification, as well as simple mechanisms for secure access and information sharing with other users and software tools. The source code for the platform is made available on Github at https://github.com/JBEI/ice. It is released under the Berkeley Software Distribution (BSD) license, which is a permissive free software license that imposes minimal restrictions on how the software can be used or redistributed. A public instance of ICE is also made available at https://public-registry.jbei.org to the Synthetic Biology community for easy evaluation of the software and also to enable biological parts to be uploaded and made publicly available.

2.1 Features
The ICE platform incorporates an increasingly growing list of features that aid in managing, organizing and sharing biological part information. It provides support for uploading sequence information using GenBank, FASTA or the Synthetic Biology Open Language (SBOL) [2] format, with seamless inter-conversion between formats.

A collections management feature allows users to organize parts into ad-hoc or logical collections for ease of access. There are also options available for adding, removing and moving parts between collections. Access permission can be provided to individual parts or collections on a per user or group basis to provide detailed sharing capabilities. This feature ensures that the user has fine-grained control over who is allowed to see or edit their data.

Advanced full text and BLAST+ search capabilities enable ease of parts discovery.

ICE also includes a bulk import feature that enables users to upload information about several parts at once, using either the available interface (see Figure 1) or uploading a file in the comma separated value (CSV) format.

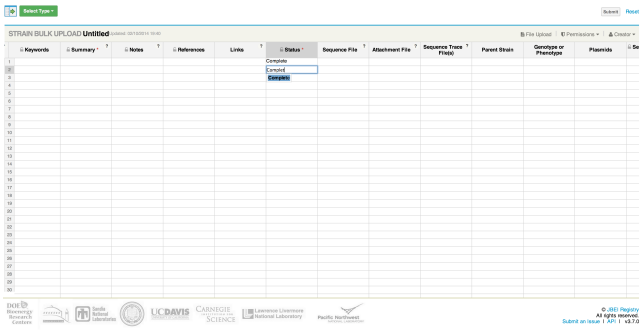


Figure 1: ICE Registry bulk import interface.

Another important feature that is incorporated in ICE is the sample tracking system. It allows the physical location of the biological parts to be associated with the records in the registry.

Users are able to request additional features on the project's Github page.

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

2.2 Integrated Tools

Graphical applications and other tools are integrated into the ICE platform to aid in sequence design, annotation and verification.

2.2.1 VectorEditor

VectorEditor (see Figure 2) is the largest and most complex integrated tool. It is capable of real time DNA editing with live vector map display, restriction site visualization, sophisticated feature annotation and annealing temperature calculation. Vector editor can import and export GenBank and SBOL files for data exchange with other sequence manipulation programs. Like ICE, VectorEditor is also open source software that is made freely available to the community at https://github.com/JBEI/vectoreditor.

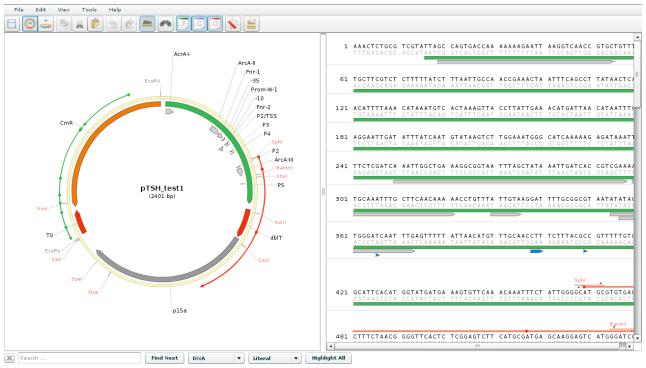


Figure 2: VectorEditor

2.2.2 SequenceChecker

Sequence Checker is also another integrated tool, which enables sequencing trace files in the form of .ab1, or FASTA files to be aligned onto a target sequence for DNA construct verification (see Figure 3). It overlays multiple trace files onto a plasmid vector map and visually highlights any mismatches.

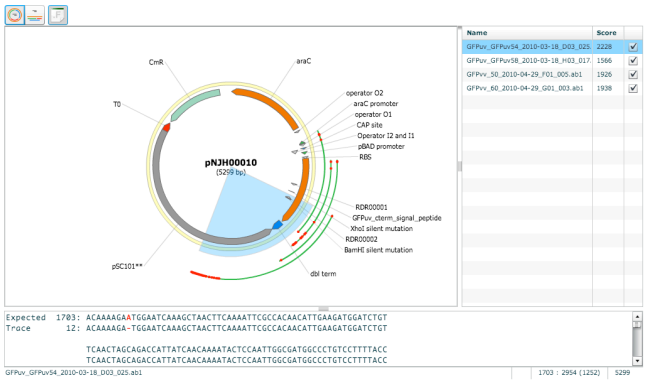


Figure 3: SequenceChecker

2.2.3 Pigeon

ICE also incorporates Pigeon [3], which is a web based tool developed at Boston University to programmatically generate synthetic biology design visualizations. Design visualizations are useful in encouraging sequence characterization and pointing out when the wrong feature types are used.

2.3 Web of Registries

One distinguishing feature of ICE is the inclusion of a distributed software platform to enable efficient sharing of biological elements across labs (see Figure 4) in the synthetic biology research community, thus expanding the search space of biological constructs. It offers community collaboration capabilities, referred to as "Web of Registries" which enables scientist to publish and share their datasets across multiple ICE installations in potentially difference geographic locations.

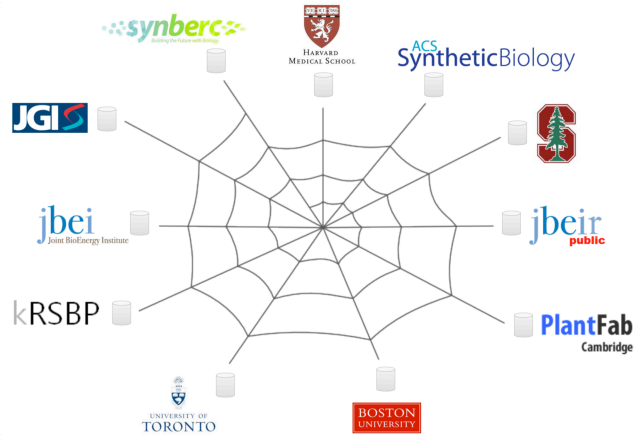


Figure 4: Current and anticipated research labs and institutions participating in web of registries

3. ACKNOWLEDGMENTS

This work was conducted by the Joint BioEnergy Institute was supported by the Office of Science, Office of Biological and Environmental Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

4. REFERENCES

[1] Ham TS, Dmytriv Z, Plahar H, Chen J, Hillson NJ, Keasling JD. (2012) *Design, implementation and practice of JBEI-ICE: an open source biological part registry platform and tools*. Nucleic Acids Res 40(18):e.141

[2] M. Galdzicki et al. *Synthetic Biology Open Language (SBOL) version 1.1.0* BBF RFC #87

[3] S. Bhatia and D. Densmore. *Pigeon: a design visualizer for synthetic biology*. ACS Syn. Bio., April 2013

Generating Systems Biology Markup Language Models from the Synthetic Biology Open Language

Nicholas Roehner
Dept. of Bioengineering
University of Utah
Salt Lake City, Utah 84112
n.roehner@utah.edu

Chris J. Myers
Dept. of Electrical and Computer Eng.
University of Utah
Salt Lake City, Utah 84112
myers@ece.utah.edu

1. INTRODUCTION

As the *Synthetic Biology Open Language* (SBOL) [1] is extended to exchange data on the intended function as well as structure of genetic designs, care must be taken to avoid duplicating existing standards that may be leveraged to represent genetic function, such as the *Systems Biology Markup Language* (SBML) [3]. Recently, a new data model for the next version of SBOL has been proposed. Instead of directly representing the quantitative function of a genetic design, this data model refers to mathematical models written in other standards and merely represents the qualitative function of the design in order to inform the creation of these models. In this way, the proposed SBOL data model can serve as a means of loosely coupling qualitative and quantitative data on the function of a genetic design, thereby enabling synthetic biologists with different areas of expertise to work on different functional aspects of the same exchangeable design.

In order to facilitate interdisciplinary collaboration and tighter connections between qualitative and quantitative data on genetic function, *genetic design automation* (GDA) tools are needed to help automate the process of creating quantitative models based on qualitative models. Accordingly, this abstract presents a methodology for generating SBML models from the proposed SBOL data model and uses the SBML and SBOL for a LacI inverter (one half of the genetic toggle switch [2]) as an example.

2. SBOL FOR LACI INVERTER

Under the proposed SBOL data model, the qualitative function of the LacI inverter can be described using a module that (1) instantiates the DNA, protein, and small molecule components of the LacI inverter as signals and (2) asserts the regulatory and gene expression interactions occurring between these signals. Fig. 1 is a *Unified Modeling Language* (UML) diagram that depicts a SBOL module containing three interactions and six signals. Each interaction has a type derived from the *Systems Biology Ontology* (SBO) [5], a controlled vocabulary for systems biology terms, and refers to its participating signals indirectly via participations. Each participation has a role that is also derived from a SBO term and indicates what a signal does or has done to it in an interaction.

3. SBML FOR LACI INVERTER

Under the data model for Level 3 Version 1 of SBML, the quantitative function of the LacI inverter can be described using a model that (1) captures the chemical species of the LacI inverter and the reactions that produce/consume them and (2) supplies the rate laws for these reactions to facilitate simulation and mathematical analysis. Fig. 2 is a UML diagram that shows a SBML model containing a list of two reactions that together reference a total of five species. Each reaction has one or more lists of species references that identify which species are its reactants, products, and modifiers. Furthermore, each reaction and species reference can include a SBO term to more concretely specify its type or role in a biochemical or genetic context.

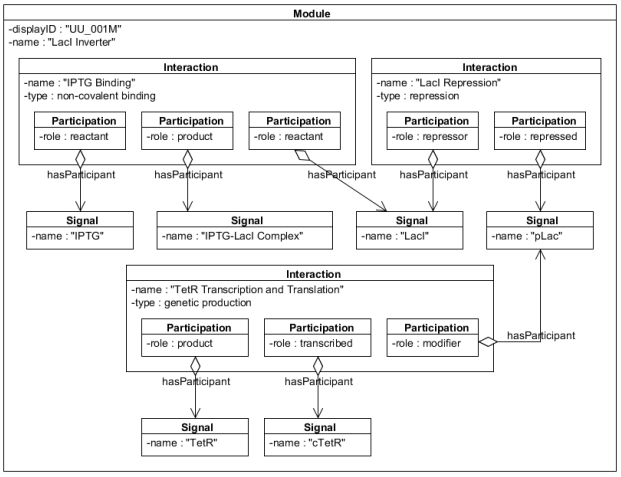


Figure 1: A UML diagram of the LacI inverter under the proposed data model for the next version of SBOL.

4. MODEL GENERATION

During model generation, a SBOL module describing a genetic circuit is translated to a SBML model using a particular mapping function. The rules for applying this mapping function are as follows:

1. For each protein signal, small molecule signal, and complex signal in a SBOL module, add a species and a degradation reaction with a mass-action rate law to a SBML model.
2. For each promoter signal in the SBOL module, add a promoter species to the SBML model.

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

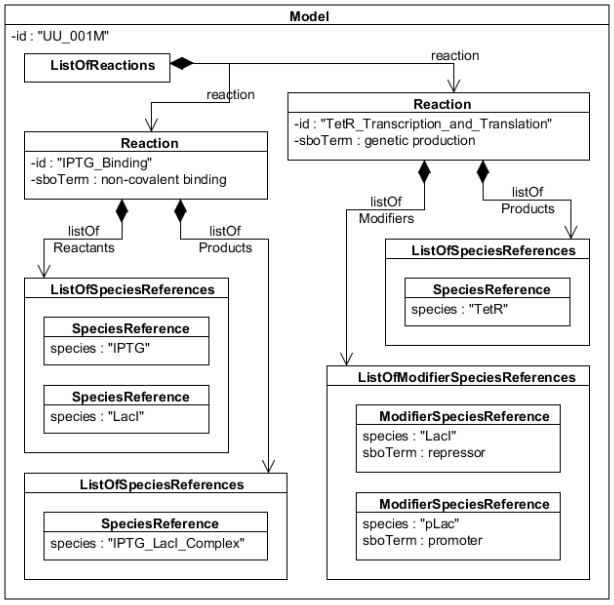


Figure 2: A UML diagram of the LacI inverter under Level 3 Version 1 of the SBML data model.

3. For each genetic production interaction in the SBOL module, add a genetic production reaction to the SBML model with a Hill function rate law.

4. For each promoter signal in the SBOL module that participates as a modifier in a genetic production interaction, add a promoter species reference to the list of modifiers for the corresponding genetic production reaction in the SBML model.

5. For each promoter signal in the SBOL module that is activated/repressed in an activation/repression interaction and participates as a promoter in a genetic production interaction, add N activator/repressor species references to the list of modifiers for the corresponding genetic production reaction in the SBML model, where N is the number of transcription factor signals in the SBOL module that participate as activators/repressors in the activation/repression interaction.

6. For each protein signal in the SBOL module that participates as a product in a genetic production interaction, add a species reference to the list of products for the corresponding genetic production reaction in the SBML model.

7. For each non-covalent binding interaction in the SBOL module, add a non-covalent binding reaction to the SBML model with a mass-action rate law.

8. For each protein signal, small molecule signal, and complex signal in the SBOL module that participates as a reactant or product in a non-covalent binding interaction, add a species reference to the list of reactants or products for the corresponding non-covalent binding reaction.

5. CONCLUSIONS

As implemented in our GDA software `iBioSim` [4], this methodology enables users to generate a quantitative SBML model from a qualitative SBOL module that documents the molecular interactions of a genetic circuit. The SBML model can then be annotated with its corresponding SBOL using a previously developed annotation methodology [7] to tightly couple these quantitative and qualitative descriptions of genetic function. Note that the mapping function used in this approach is only one of many possible mappings from SBOL to SBML, or from SBOL to another modeling standard. In the future, other mappings will be developed for different modeling tasks, much in the same way that different fonts exist for the same set of alphanumeric characters.

Because the proposed SBOL data model is not capable of encoding quantitative parameters, the SBML rate laws generated by this methodology are populated with default parameters that must be customized by the user. A more detailed description of these rate laws and their parameters can be found in [6]. In the near future, the SBOL data model might be extended with the capacity to store data on quantitative parameters and measurements, thereby providing a firmer foundation for GDA tools to generate different mathematical models for different design tasks that nevertheless conform to the same basic data set.

6. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. CCF-1218095. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

7. REFERENCES

[1] M. Galdzicki et al. SBOL: A community standard for communicating designs in synthetic biology. *Nat. Biotechnol.*, 32(6), 2013.

[2] T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in Escherichia coli. *Nature*, 403:339–342, 2000.

[3] M. Hucka et al. The Systems Biology Markup Language (SBML): a medium for representation and exchange of biochemical network models. *Bioinform.*, 19(4):524–531, 2003.

[4] C. Madsen, C. Myers, T. Patterson, N. Roehner, J. Stevens, and C. Winstead. Design and test of genetic circuits using iBioSim. *IEEE Design and Test*, 29(3):32–39, 2012.

[5] N. Juty and N. Novere. Systems Biology Ontology. In *Encyclopedia of Systems Biology*, pages 2063–2063. Springer New York, 2013.

[6] N. Nguyen, C. Myers, H. Kuwahara, C. Winstead, and J. Keener. Design and analysis of a robust genetic Muller C-element. *J. Theor. Biol.*, 264(2):174–187, 2010.

[7] N. Roehner and C. J. Myers. A methodology to annotate Systems Biology Markup Language models with the Synthetic Biology Open Language. *ACS Synth. Biol.*, 3(2):57–66, 2014.

Incorporating loading effects into the automated design of complex biocircuits

[Extended Abstract]

Andras Gyorgy
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139
gyorgy@mit.edu

Domitilla Del Vecchio
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139
ddv@mit.edu

ABSTRACT

The ability to accurately predict the behavior of a complex system from that of the composing modules has been instrumental to the development of engineering systems. Here, we present a framework accounting for loading effects among biological components which carries substantial conceptual analogy with the theory of electrical systems. Within this framework, the description of each component includes quantities similar to impedance. Accounting for these impedance-like quantities facilitates the reliable and modular design of large-scale biocircuits as part of an automated workflow.

Keywords

gene networks, impedance, modularity, retroactivity

1. INTRODUCTION

It was proposed that biology can be understood, just like engineering, in a modular fashion [7]. If biological parts and modules behaved the same way in isolation as when part of a larger network, the design of complex biocircuits would be simpler. Unfortunately, a module's behavior is often affected by its context [1]. One source of context-dependence is retroactivity [2], a phenomenon in which a downstream component changes the behavior of an upstream component as a result of sequestering chemical species. For instance, the frequency and amplitude of a clock's oscillations can be largely affected by a load [3].

In a bottom-up approach to engineer biological systems, simple modules are combined to create larger ones [9]. Among other factors [1], retroactivity makes it necessary to re-design modules through a lengthy and *ad hoc* process every time they are inserted into a different context [10]. Therefore, to design biocircuits in an automated and modular fashion, we must account for retroactivity during the design phase to ensure the reliable functioning of the implemented systems.

This paper addresses this issue by incorporating retroactivity into the description of promoters, and by further combining these retroactivities to predict how the behavior of modules change upon interconnection in the context of biodesign automation, e.g., in the framework presented in [6].

2. MAIN RESULT

Take the reaction $\emptyset \underset{\delta}{\overset{\xi(t)}{\rightleftharpoons}}$ x from [2] describing the production and degradation of transcription factor (TF) x, yielding

$$\dot{x} = \xi(t) - \delta x. \tag{1}$$

Next, consider the addition of DNA load to x (e.g., a logic gate using x as input), modeled by the reversible binding reaction $\mathrm{x} + \mathrm{p} \underset{k_-}{\overset{k_+}{\rightleftharpoons}} \mathrm{c}$, where p and c are the empty and x-bound promoters by having x bound to the promoter p. As a result, the dynamics of x change from (1) to

$$\dot{x} = \frac{1}{1+R(x)}\left[\xi(t) - \delta x\right], \tag{2}$$

where $R(x) = \frac{\eta/k_d}{(1+x/k_d)^2}$ is the retroactivity of the promoter with $k_d = k_-/k_+$ being the dissociation constant of x to the promoter, and $\eta = p + c$ is the total concentration of the promoter [2]. The higher the DNA copy number η and the lower the dissociation constant k_d, the greater the retroactivity $R(x)$. Comparing (1) with (2), greater $R(x)$ causes more substantial changes in the dynamics of x, that is, the effect of the downstream component on the behavior of the upstream component increases with $R(x)$ [2].

For the promoter of gene i, the *retroactivity* R_i introduced in [5] captures the retroactive effects arising from the binding of the regulators of gene i changing the dynamics of the regulators, as shown in (2) for the regulator x. In [5] the formula for computing R_i is given as a function of measurable biochemical parameters: the DNA copy number of the promoter and the dissociation constants of the regulators are required. Combinatorial regulation (when multiple transcription factors regulate the expression of a gene) and multimerization (when multiple molecules of the same transcription factor first need to form a complex which then can regulate the expression of a gene) are both considered in [5].

Modules are (small) networks of interconnected genes with specific functions, such as toggle switches and oscillators.

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

The loading effects inside a module are described by the *internal retroactivity* R of a module [5], which can be calculated considering the retroactivity R_i of each promoter and the topology of the module (i.e., which gene regulates which). Neglecting R can lead, for instance, to the design of non-functional clocks [5]. This could happen, for example, in the framework presented in [6] for the automated design of biocircuits. However, this limitation can be easily overcome by simply incorporating the retroactivity terms when evaluating circuit performance, which is straightforward once retroactivity is part of the description of a promoter, as proposed here.

The behavior of modules is affected by retroactivity arising due to intermodular connections upon interconnection. For instance, in the case of the toggle switch [4], adding loads to either repressor sequesters molecules from the repression of the other repressor in the toggle switch, yielding slower switching characteristics compared to the isolated case [5]. These effects can be described by the *scaling retroactivity* S and *mixing retroactivity* M of a module [5], which can be computed similarly to the internal retroactivity R, by considering the retroactivity R_i of each promoter and the network topology (for details, see [5]). In particular, let the dynamics of module A and B given by $\dot{x}^A = f^A(x^A, x^B, \dot{x}^B)$ and $\dot{x}^B = f^B(x^B, x^A, \dot{x}^A)$, respectively, where x^A and x^B are the concentration vectors of TFs in module A and B [5], so that when the modules are not interconnected, we have

$$\begin{pmatrix} \dot{x}^A \\ \dot{x}^B \end{pmatrix} = \begin{pmatrix} f^A(x^A, x^B, \dot{x}^B) \\ f^B(x^B, x^A, \dot{x}^A) \end{pmatrix}. \quad (3)$$

Upon interconnection, the dynamics of the modules become coupled as the matrix

$$\begin{bmatrix} I + (I + R^A)^{-1} S^B & (I + R^A)^{-1} M^B \\ (I + R^B)^{-1} M^A & I + (I + R^B)^{-1} S^A \end{bmatrix}^{-1} \quad (4)$$

pre-multiplies the right-hand side of (3), see [5] for details (R^X, S^X and M^X are the internal, scaling and mixing retroactivity of module X, respectively, for $X \in \{A, B\}$).

3. CONCLUSIONS

The above presented results can be integrated into an automated design workflow, such as [6], as follows (Figure 1). First, when selecting promoters, their description should include their retroactivity R_i, similarly to the impedance of an electrical component. Then, when combining parts to form functional modules, the specification of a module should incorporate its internal, scaling and mixing retroactivity (R, S and M, respectively). These quantities are similar to the input impedance of electrical modules. Once these retroactivities are incorporated into the description of the modules, the parameters of the modules can be optimized so that the input-output behaviors of modules do not change appreciably upon interconnection ("retroactivity-matching", see [5]). Incorporating retroctivity into the description of promoters and modules as detailed in [5] thus takes us one step closer to the modular and predictable design of complex biocircuits in an automated fashion.

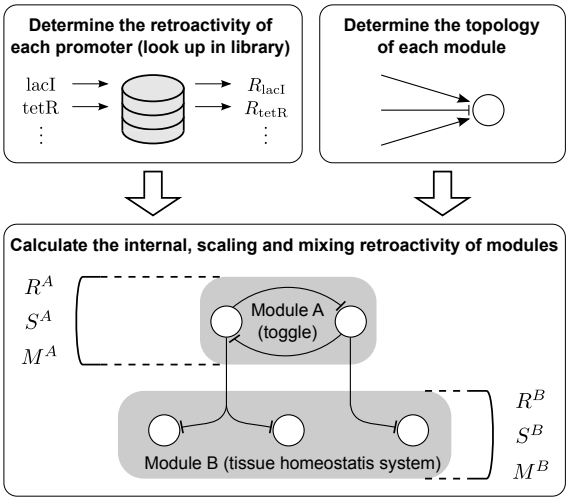


Figure 1: Illustration of how to incorporate retroactivity into to description of biocircuits. For instance, in the architecture proposed in [8], the toggle switch [4] (upstream module) is regulating an artificial tissue homeostasis system (downstream module). For details, see [5].

4. REFERENCES

[1] S. Cardinale and A. P. Arkin. Contextualizing context for synthetic biology – identifying causes of failure of synthetic biological systems. *Biotechnology Journal*, 7(7):856–866, 2012.

[2] D. Del Vecchio, A. J. Ninfa, and E. D. Sontag. Modular cell biology: retroactivity and insulation. *Nature/EMBO Molecular Systems Biology*, 4(161), 2008.

[3] E. Franco, E. Friedrichs, J. Kim, R. Jungmann, R. Murray, E. Winfree, and F. C. Simmel. Timing molecular motion and production with a synthetic transcriptional clock. *PNAS*, 108(40):E787, 2011.

[4] T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in *Escherichia coli*. *Nature*, 403:339–342, 2000.

[5] A. Gyorgy and D. D. Vecchio. Modular composition of gene transcription networks. *PLoS Computational Biology*, 10(3):e1003486, 2014.

[6] L. Huynh, A. Tsoukalas, M. Koppe, and I. Tagkopoulos. SBROME: A scalable optimization and module matching framework for automated biosystems design. *ACS Synthetic Biology*, 2:263–273, 2013.

[7] D. A. Lauffenburger. Cell signaling pathways as control modules: complexity for simplicity? *PNAS*, 97(10):5031–5033, 2000.

[8] M. Miller, M. Hafner, E. Sontag, N. Davidsohn, S. Subramanian, P. Purnick, D. Lauffenburger, and R. Weiss. Modular design of artificial tissue homeostasis: robust control through synthetic cellular heterogeneity. *PLoS Computational Biology*, 8(7):e1002579, 2012.

[9] P. E. M. Purnick and R. Weiss. The second wave of synthetic biology: from modules to systems. *Nature Reviews Molecular Cell Biology*, 10(6):410–422, 2009.

[10] A. L. Slusarczyk, A. Lin, and R. Weiss. Foundations for the design and implementation of synthetic genetic circuits. *Nature Reviews Genetics*, 13(6):406–20, 2012.

Optimizing Product Yield Through Identifying Gene Expression Fold Changes

Sara Amr Amin
Department of Computer Science
Tufts University
Sara.Amin@tufts.edu

Mona Yousofshahi
Department of Computer Science
Tufts University
Mona.Yousofshahi@tufts.edu

Soha Hassoun
Department of Computer Science
Tufts University
soha@cs.tufts.edu

ABSTRACT
We describe in this paper using Simulated Annealing the identification of gene expression fold changes to enhance yield. We evaluated this approach to identify interventions needed to maximize antibody production in the Chinese Hamster Ovary (CHO) cell.

1. INTRODUCTION
Engineering biological systems promise to enhance the production of beneficial products such as therapeutics, drugs and biofuels. One challenge in engineering biological systems is identifying cellular interventions that maximize the flux (rate) of producing desired products. Cellular optimization problems are formulated in terms of two kinds of variables: flux variables and control (decision) variables that correspond to the presence or absence of up- or down-regulation for each possible reaction. The overall objective of the optimization procedure is to tune these variables optimally to maximize the production of a target metabolite. Mathematically, the solution must satisfy several constraints including: steady state constraints on the metabolic network, a minimum biomass production above a given threshold, and uptake values for some select fluxes. One example optimization tool is OptKnock, which uses a bilevel optimization framework to identify gene knockouts (deletions) while optimizing the coupled objectives of metabolite overproduction and biomass formation[1]. OptReg utilizes a similar mathematical formulation to identify sets of genes that should be jointly up- or down-regulated[2]. CCOpt utilizes chance-constrained programming, where constraints are met with some probabilistic guarantees, to identify genes that must be modified[3]. While previous approaches focused on identifying genes that must be modified, there are experimental benefits in additionally identifying minimal enzyme fold changes for the selected gene set.

2. METHODS
We apply in this paper Simulated Annealing (SA) to find the gene set that must be up-regulated and the relevant minimal fold changes needed to maximize the desired flux. SA is a probabilistic metaheuristic used to find a good approximate solution of a global optimum for a problem with a large feasible solution space. Each solution specifies a gene set that must be modified and the relevant fold changes. Each fold change can be one of three values, 2x, 5x, or 10x, compared to a 1x non-regulated fold change. SA begins by selecting a solution at random. The goodness of a solution is evaluated by the SA fitness function, which computes the maximum yield given steady-state, thermodynamic and uptake constraints. The fold changes modify the constraints by imposing new (increased) upper bounds on fluxes. The fitness calculation considers the total fold changes by favoring a solution with smaller fold changes in enzyme activities over another when the two solutions have the same production rate.

A new solution is generated by permuting an existing one. The extent of permutations is correlated with the cooling schedule. At the beginning of the simulation and at high temperatures, both the gene set and the fold changes are permuted. As the temperature cools, the permutation is more likely to involve only one fold change. We utilized a geometric method for the cooling schedule. To speed SA and for practical considerations, the number of possible modifications was restricted.

3. RESULTS
We evaluated our computational framework by identifying interventions needed to maximize antibody production in the Chinese Hamster Ovary (CHO) cell [4]. The CHO cell model consists of 24 metabolites and 34 reactions. Due to the small model size, we restricted the number of interventions to 4. SA identifies four up-regulation targets, with three fold change values of 10 and one fold change of 2. The resulting antibody yield was 3861 nmol/10^6cells/day compared to the original antibody production of 525 nmol/10^6cells/day. SA identified two other inferior solutions, with 5 and 10 fold changes instead of 2, with the same yield.

4. REFERENCES
[1] Burgard, Anthony P., Priti Pharkya, and Costas D. Maranas. "OptKnock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization." *Biotechnology and bioengineering* 84.6 (2003): 647-657.
[2] Pharkya, Priti, and Costas D. Maranas. "An optimization framework for identifying reaction activation/inhibition or

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

elimination candidates for overproduction in microbial systems." *Metabolic engineering* 8.1 (2006): 1-13.
[3] Yousofshahi, Mona, et al. "Probabilistic strain optimization under constraint uncertainty." BMC systems biology 7.1 (2013): 29.
[4] Nolan, Ryan P., and Kyongbum Lee. "Dynamic model of CHO cell metabolism."Metabolic engineering 13.1 (2011): 108-124.

Owl: Electronic Datasheet Generator for Synthetic Biology

Evan Appleton, Jenhan Tao, Traci Haddock, and Douglas Densmore
Center of Synthetic Biology, Boston University, Boston, MA, USA
{eapple, thaddock, dougd}@bu.edu, jenhantao@gmail.com

1. MOTIVATION

As the field of synthetic biology grows, the number of synthetic genetic components used by the community expands. Each genetic part is studied by individual groups, but is often poorly documented and communicated to the rest of the community. Furthermore, when this information is conveyed, it is often in variable formats and can be time consuming to communicate if many parts are under study.

Given this already large and rapidly growing body of information, a number of 'genetic parts registries' have emerged. Registries of synthetic genetic parts such as BioBricks, JBEI, JGI, SynBERC and BIOFAB, have approached the question of how to best share and store data in recent years. It has become apparent that the synthetic biology community should form a concerted effort to share data on genetic parts and other biological components in a standardized way[1]. To address this need, we have created an online tool for datasheet generation, called Owl (owlcad.org), to help users generate datasheets with a common formats and iconography[2].

2. WHAT IS OWL?

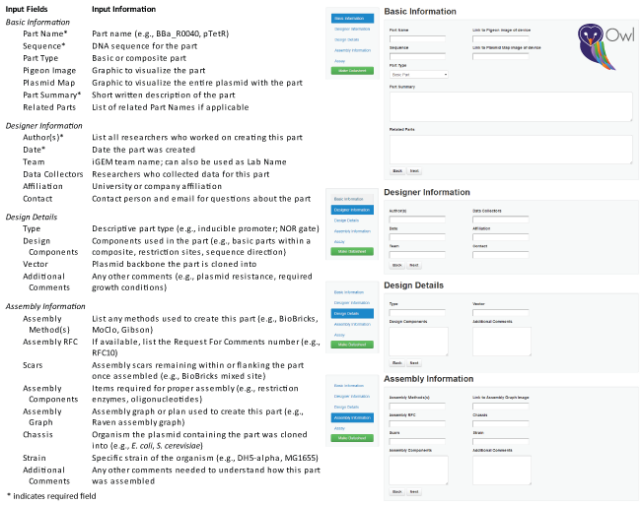
Owl is a bio-design automation tool that generates electronic datasheets for synthetic biological parts. Owl datasheets describe parts using a common format for ease of use for interacting with other tools and for sharing with other researchers. Data can be retrieved automatically from existing repositories and be changed and added to in the Owl UI. Owl uses the data to generate an HTML page with a standard layout that can be saved locally. Here we present the Owl software tool, its current UI, a description of current input data for generating a datasheet, and an example datasheet.

3. REQUIRED AND OPTIONAL FIELDS

Owl datasheets have required fields for each datasheet (**Fig. 1**). This information is meant to represent the minimum information required to define a part with which data can be experimentally associated. This would change based upon the input data model. Here we require that a part physically exists to have a datasheet. All other fields are optional.

4. AUTOMATED DATA POPULATION

When generating a new datasheet from an existing Registry page, Owl parses and extracts information for a specific part to populate fields on the datasheet, namely: Part Name (ex: BBa_B0034), Part Description, Part Type (ex: RBS), Date entered, Part Author, and Sequence. The user can then go through each section manually and add more information to the datasheet such as experiments they have run with the part of interest.



Input Fields	Input Information
Basic Information	
Part Name*	Part name (e.g., BBa_R0040, pTetR)
Sequence*	DNA sequence for the part
Part Type	Basic or composite part
Pigeon Image	Graphic to visualize the part
Plasmid Map	Graphic to visualize the entire plasmid with the part
Part Summary*	Short written description of the part
Related Parts	List of related Part Names if applicable
Designer Information	
Author(s)*	List all researchers who worked on creating this part
Date*	Date the part was created
Team	iGEM team name; can also be used as Lab Name
Data Collectors	Researchers who collected data for this part
Affiliation	University or company affiliation
Contact	Contact person and email for questions about the part
Design Details	
Type	Descriptive part type (e.g., inducible promoter; NOR gate)
Design Components	Components used in the part (e.g., basic parts within a composite, restriction sites, sequence direction)
Vector	Plasmid backbone the part is cloned into
Additional Comments	Any other comments (e.g., plasmid resistance, required growth conditions)
Assembly Information	
Assembly Method(s)	List any methods used to create this part (e.g., BioBricks, MoClo, Gibson)
Assembly RFC	If available, list the Request For Comments number (e.g., RFC10)
Scars	Assembly scars remaining within or flanking the part once assembled (e.g., BioBricks mixed site)
Assembly Components	Items required for proper assembly (e.g., restriction enzymes, oligonucleotides)
Assembly Graph	Assembly graph or plan used to create this part (e.g., Raven assembly graph)
Chassis	Organism the plasmid containing the part was cloned into (e.g., *E. coli*, *S. cerevisiae*)
Strain	Specific strain of the organism (e.g., DH5-alpha, MG1655)
Additional Comments	Any other comments needed to understand how this part was assembled

* indicates required field

Figure 1: An example Owl datasheet

5. FUTURE WORK

While the Owl tool only currently parses the BioBricks Registry, it will be expanded to parse additional registries. In anticipation of additional data models, we will allow users to upload a configuration file to govern what fields and types of inputs should be included in the datasheet. To accommodate for different desired formats, future versions will allow users to upload a file to specify a layout configuration file.

6. ACKNOWLEDGEMENTS

The authors acknowledge the 2012 & 2013 BU iGEM Team, S. Bhatia, E. Oberortner, and S. Iverson for useful discussions and their feedback on the fields included in the datasheet.

7. REFERENCES

[1] Arkin, A. Setting the standard in synthetic biology. *Nature Biotechnology 26* (2008), 771–774.

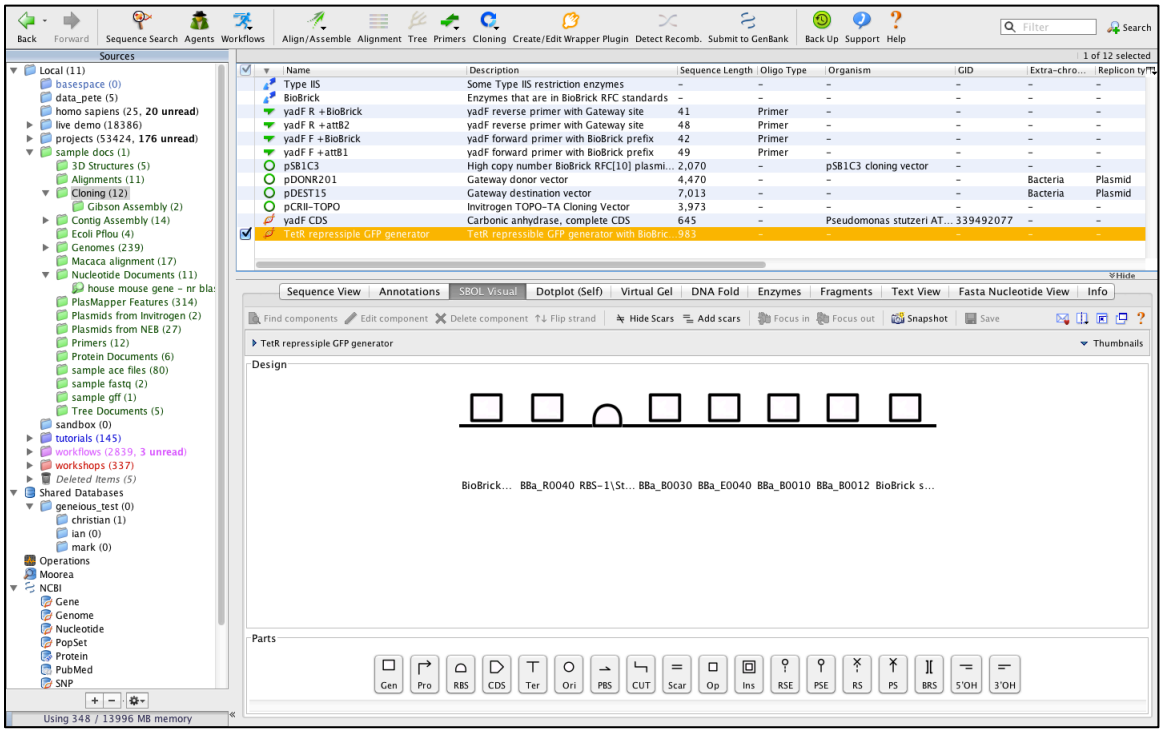
[2] Bhatia, S., and Densmore, D. Pigeon: a design visualizer for synthetic biology. *ACS Synthetic Biology* (2013).

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

Synthetic Biology Open Language Designer

Christian Olsen*[1], Kashef Qaadri[1], Helen Shearman[2], Hilary Miller[2]
[1] Biomatters, Inc. 185 Clara St, Suite 101A San Francisco, CA 94107
[2] Biomatters, Ltd. L2, 76 Anzac Ave Auckland, 1010 New Zealand

The Synthetic Biology Open Language (SBOL) Designer is a synthetic biology design tool for visualizing and creating designs expressed using SBOL. SBOL is an open-source data exchange standard for descriptions of genetic parts, devices, modules, and systems. SBOL defines both a serialization format based on RDF/XML and a set of visual icons to graphically depict functional information encoded by nucleic acid sequences. The SBOL Designer has been integrated with the Biomatters' Geneious R7 sequence analysis/alignment/assembly platform. The SBOL Designer can be used to visualize the sequences created in Geneious using SBOL visual icons, edit the design using the SBOL view, and see the results immediately inside Geneious R7. The SBOL Designer allows users to create designs using SBOL visual icons, edit SBOL designs in Geneious R7, save designs in an SBOL RDF/XML file, and import DNA components from an SBOL parts registry. The SBOL designer uses Standard Biological Parts Knowledgebase (SBPkb) to import DNA components from the Registry of Standard Biological Parts at MIT. Other SPARQL Protocol and RDF Query Language (SPARQL) endpoints can be defined in the tool to use components from a different source. This is useful for accessing parts databases and provides tools for collaboration by allowing synthetic biologists and genetic engineers to electronically exchange designs across multiple sites. This poster will cover the features and benefits of the SBOL designer and will provide insight on the latest Synthetic Biology tools.



IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

1) Michal Galdzicki, Kevin P. Clancy, Ernst Oberortner, Matthew Pocock, Jacqueline Quinn, Cesar A. Rodriguez, Nicholas Roehner, Mandy L. Wilson, Laura Adam, J. Christopher Anderson, Bryan A. Bartley, Jacob Beal, Deepak Chandran, Joanna Chen, Douglas Densmore, Drew Endy, Raik Grünberg, Jennifer Hallinan, Nathan J. Hillson, Jeffrey D. Johnson, Allan Kuchinsky, Matthew Lux, Goksel Misirli, Jean Peccoud, Hector A. Plahar, Evren Sirin, Guy-Bart Stan, Alan Villalobos, Anil Wipat, John H. Gennari, Chris J. Myers, and Herbert M. Sauro. SBOL: A community standard for communicating designs in synthetic biology. Figshare, 2013; doi:10.6084/m9.figshare.762451

2) Michal Galdzicki, Cesar Rodriguez, Deepak Chandran, Herbert M. Sauro, and John H. Gennari (2011) Standard Biological Parts Knowledgebase. PLoS ONE. 6(2): e17005.

3) Jim Rapoza (2 May 2006). "SPARQL Will Make the Web Shine". eWeek. Retrieved 4/4/14.

4) Toby Segaran, Colin Evans, Jamie Taylor (2009).Programming the Semantic Web. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. p. 84. ISBN 978-0-596-15381-6.

A top-down approach to genetic circuit synthesis

Sune Mølgaard Laursen, Jakob Jakobsen Boysen, Jan Madsen
Department of Applied Mathematics and Computer Science
Technical University of Denmark
jama@dtu.dk

1 Introduction

One of the ultimate goals of synthetic biology is the engineering of dedicated cell behavior, by introducing completely new sequences of DNA (called the genetic circuit) into the DNA of the cell. A genetic circuit represents a gene regulator network which is triggered by a combination of external signals, such as chemicals, proteins, light or temperature, to emit signals to control gene expression or metabolic pathways accordingly.

Today, new genetic circuits are to a large extent formed in an experimental bottom-up approach, where the genetic circuit is assembled from a standard library of well-defined genetic circuits or parts of such (e.g., the BioBricks library), to match an intended behavior. This approach resembles the early days of microelectronics circuit design. However, when the complexity of the genetic circuits grows, there will be a strong need for a top-down approach, where the dedicated cell behavior is expressed in a high-level language and then synthesized into the most efficient genetic regulator network.

We present a top-down genetic circuit synthesis framework, which synthesizes a new efficient genetic circuit from a high-level logical description. Our approach is based on 3 phases; 1) genetic logic optimization, 2) genetic circuit mapping, and 3) quantitative genetic circuit analysis.

2 Genetic circuit synthesis

The genetic circuit synthesis process (illustrated in Fig. 1) is highly inspired by the techniques developed for the synthesis of electronic circuits. However, logic high and low are defined by the concentration of molecules, e.g., proteins, and there are some very important differences to electronic synthesis, which challenges the synthesis of genetic circuits,

Compatibility: In electronic circuits the output from an arbitrary circuit can always be used as input to another circuit. In genetic circuit synthesis the ”wiring” is established through diffusion of particular proteins, hence, compatibility between input- and output-proteins of the library parts must be ensured.

Orthogonality: Cross-talk can have devastating effects in genetic circuits, so circuits or parts with the same intermediate proteins are not allowed in the genetic circuit. This further implies that library parts cannot be reused, except for amplification of protein concentrations.

Size: The synthesized genetic circuits should fit into a single cell. Currently only 20 orthogonal promoters have been identified, [2], which effectively defines an upper limit of the complexity of a genetic circuit. As our knowledge and models get better, this limit will most likely increase, and if translational regulation using sRNA is considered, the upper limit may be extended already today, as promoters are not influenced by sRNA.

Stochastic nature: The stochastic nature of gene expression, makes analytical assessment of models of genetic circuits very difficult, if not impossible, and simulations have to be carried out to give good assessments of the quality.

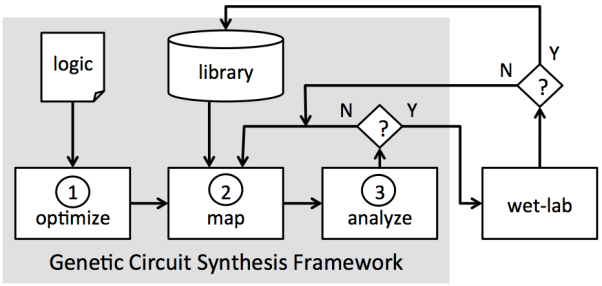


Figure 1: Genetic circuit synthesis framework.

In the following, we present the 3 phases of our genetic circuit synthesis framework in more details.

2.1 Genetic logic optimization

The first phase converts the logical input description of the desired cell behaviour into a logic expression. To ensure low circuit complexity of the synthesized circuit, the logic expression is minimized to a minimal Boolean expression using e.g. the *Quine-McClusky algorithm*. In the following, we will use the minimized expression

```
CI = (GFP') + (IPTG lacI)
```

to illustrate the phases of the genetic circuit synthesis.

2.2 Genetic circuit mapping

The minimized expression is given to a *genetic circuit mapper* that finds different coverings of the logic expression by library circuits that makes up possible candidate solutions. The mapping algorithm uses an *And-Inverter Graph* (AIG)[1], and performs an explorative top-down pattern matching using all the library circuits. If the entire graph can be fully covered while ensuring the aforementioned constraints, a design candidate has been found. In Fig. 2b and c, two possible solutions are shown.

[1]A Directed Acyclic Graph using only ”and” and ”not” operators.

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

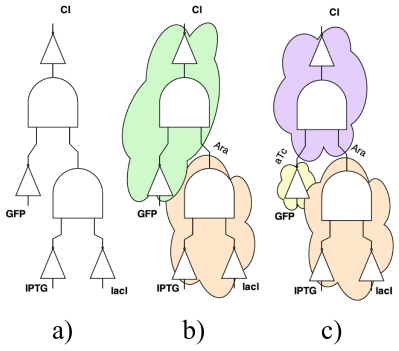


Figure 2: Possible coverings of a simple logic expression. *a*) The AIG representation of the expression. *b*) Design candidate 1. *c*) Design candidate 2.

Fig. 2a shows the AIG representation of the minimized expression of the motivating example. Assuming the simple library of the following 4 genetic circuits `CI = (GFP') + (Ara)` (green), `Ara = (IPTG lacI)` (red), `CI = (aTc) + (Ara)` (purple) and `aTc = GFP'` (yellow), the genetic circuit mapper finds two solutions, b) and c) in Fig. 2.

The genetic circuit mapper makes a first ranking of the solutions based on an objective function, adapted from [1], which defines as the cost C:

$$C = 2^{N_R} + 2^{N_A} + N_{IMP}$$

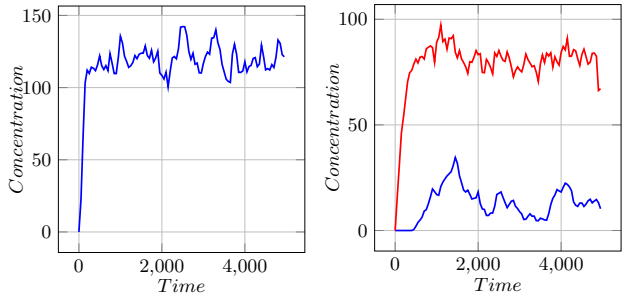
Here N_R, N_A and N_{IMP} are the total number of repressor promoters, activator promoters and intermediate proteins respectively. The cost C represents the complexity or the realisability of the circuit, not the quality, thus a low cost means a lower realisation complexity.

With better understanding of the complexity of realising the genetic circuits in the laboratory (indicated as wet-lab in Fig. 1), the objective function can be updated accordingly. The cost is dependant on the number of intermediate proteins thus design candidates composed of few library circuits or parts are automatically favored, as there will be fewer internal protein interactions, thus less internal complexity.

2.3 Quantitative genetic circuit analysis

Due to the highly stochastic effects as well as the coarse assumption of considering genetic behaviour as just logical highs and lows, the solutions cannot be guaranteed to perform as the logical expression dictates, hence validation through simulation is necessary. If the simulations show acceptable behavior, a wet-lab experiment should be carried out and only if this results in an acceptable behavior, the new genetic circuit is accepted and saved as a part in the library for later use. Our framework does not yet include the wet-lab validation, but genetic circuit models are specified in *SBML* , transformed to *stochastic petri nets* internally and are stochastic simulated using *Gillespie's direct method* . As estimating parameters of models (such as rate parameters) is still a huge problem, we assume the modelling technique and simulation work correctly, if the correct parameters are applied, hence, we are currently providing parameters to the models that make the models behave as intended.

Based on the ranking of the solutions proposed by the genetic circuit mapper, each solution is validated through stochastic simulation of all combinations of input protein concentration levels, i.e. high and low. The evaluation starts with the lowest cost C solution. As an example, Fig. 3 shows the concentration level of CI over time for the two candidate solutions, in the case where all inputs are kept low. On the basis of this simulation, we reject design candidate 2 as it does not perform as predicted, i.e. it does not reach a steady state. A closer examination, which can be performed in the framework, reveals that the output concentration of the intermediate `aTc` protein from *yellow* is not high enough to fully activate the input of *purple*. An automatic evaluation step could possibly detect this issue and duplicate *yellow* until satisfactory levels are reached, this automatic adjustment is not part of the current framework, but is part of our future work.



(a) Design candidate 1. **(b) Design candidate 2.**

Figure 3: Simulations of design candidates 1 and 2 from Fig. 2, blue is `CI`, red is `aTc`. Here simulated for absence of all input proteins.

Further simulations of design candidate 1 with all possible input combinations reveal that the solution will work (given the library and parameters). It further identifies concentration levels that can be considered high and low respectively for the new genetic circuit.

3 Discussion and summary

The details of the simulation of design candidate 2 should be used by the genetic circuit mapper to modify the genetic circuit in order to possibly obtaining a circuit that should not be rejected by simulation. Thus assessments of solutions can be used to decide if the logical behaviour is satisfied and to adjust design candidates if not.

The assessment is a time-consuming step of the tool flow, which is why automising this step and incorporating results into the genetic circuit mapper is very important.

We have briefly presented our top-down genetic circuit synthesis framework that covers the initial steps: logical behaviour can be specified as a logic expression, which can be minimised. The minimised expression is then input to a genetic circuit mapper using AIG as internal representation of logic behaviour. The output of the genetic circuit mapper is one or more design candidates specified as models represented as SPNs. These models can be stochastic simulated and the logical behaviour obtained from these simulations can be (currently manually) verified against the initial input behavior.

4 References

[1] M. A. Marchisio and J. Stelling. Automatic design of digital synthetic gene circuits. *PLoS Computational Biology*, 7(2), 2011.

[2] V. A. Rhodius, T. H. Segall-Shapiro, B. D. Sharon, A. Ghodasara, E. Orlova, H. Tabakh, D. H. Burkhardt, K. Clancy, T. C. Peterson, C. A. Gross, and C. A. Voigt. Design of orthogonal genetic switches based on a crosstalk map of sigmas, anti-sigmas, and promoters. *Molecular Systems Biology*, 9(1), Oct. 2013.

Using the Method of Types to Improve Adjacency Testing for Elementary Flux Mode Computation

Extended Abstract

Ehsan Ullah
Department of Computer Science
Tufts University
ehsan.ullah@tufts.edu

Shuchin Aeron
Department of Electrical and Computer Engineering
Tufts University
shuchin@ece.tufts.edu

Soha Hassoun
Department of Computer Science
Tufts University
soha@cs.tufts.edu

ABSTRACT

Pathway analysis has played an important role in systems biology to understand, modify and optimize cellular behavior. Elementary flux mode (EFM) analysis is one of the widely used pathway analysis approaches to engineer microorganisms. EFM analysis identifies all independent, thermodynamically feasible pathways in a metabolic network operating at steady-state. EFMs analysis is a computationally challenging task and improvements in existing algorithms is required to reduce runtime. We propose *Method of Types* to reduce runtime of EFM analysis. The results of this new method on different test cases have shown that the number of comparisons can be reduced upto 78% compared to existing approaches.

1. INTRODUCTION

EFM analysis is equivalent to identification of extreme rays of a convex polyhedral. Mathematically, the feasible steady-state flux space of a network with m internal metabolites and n reactions can be represented as a pointed convex polyhedral if all the reactions are irreversible:

$$\mathbf{P} = \{\mathbf{v} \in \mathbb{R}^n : \mathbf{S} \cdot \mathbf{v} = \mathbf{0} \text{ and } v \geq 0\} \tag{1}$$

where $\mathbf{S}$ is an $m \times n$ stoichiometric matrix of the network, $\mathbf{v}$ represents a steady-state flux and $\mathbf{v}$ represents fluxes in $\mathbf{v}$. The double description method [2] creates an alternative representation of the flux space by transforming the representative matrix of the network $\mathbf{A}$ to an equivalent representative matrix $\mathbf{R}$. Given $\mathbf{A}$, the steady-state flux space can be described as:

$$\mathbf{P} = \{\mathbf{v} = \mathbf{R} \cdot \boldsymbol{\lambda} \; \forall \; \lambda \geq 0\} \tag{2}$$

where $\mathbf{R}$ represents the set of EFMs.

Any steady-state flux distribution can be represented as a non-negative linear combination of EFMs. For the double-description method, each row of matrix $\mathbf{A}$ represents a constraint. Each column in $\mathbf{R}$ represents a ray of the convex cone. The constraints are iteratively satisfied to generate extreme rays (EFMs), which are new columns in $\mathbf{R}$. In each iteration, rays in $\mathbf{R}$ are divided into the following three groups based on the current constraint: positive rays J^+, negative rays J^- and zero rays J^0. The constraint is satisfied by combining rays from J^+ and J^-. Extreme rays (independent vectors) are generated by performing an adjacency test on the rays to be combined. If the rays are adjacent, a new extreme ray is generated using Gaussian Combination by satisfying the current constraint. After processing the current constraint, matrix $\mathbf{R}$ is updated.

2. METHODS

In the double-description method, adjacency testing is the most computationally challenging step. To check the adjacency of two rays, a newly generated ray is compared with existing rays. A ray r can be represented as a set of reactions P participating in the pathway represented by the ray. Consider a new ray r_n generated by combining rays r_1 and r_2. The set of reactions P_n corresponding to r_n can be represented as:

$$P_n = P_1 \cup P_2 \tag{3}$$

Rays r_1 and r_2 are non-adjacent if for a ray r_i in $\mathbf{R}$ the following holds:

$$P_i \nsubseteq P_n \; \forall \; r_i \neq r_1, r_2 \tag{4}$$

The above condition is tested for all the possible combinations of rays from J^+ and J^- against the existing rays in J^+, J^- and J^0. Terzer and Stelling proposed bitpattern trees to reduce the number of comparisons [4]. Bitpattern trees are considered to be state of the art data structure for efficient adjacency testing. We propose *Method of Types* [1] to further reduce the number of comparisons. Let $|P_n|$ represents the cardinality of the set P_n. Equation 4 holds iff the following is true:

$$|P_i| \leq |P_n| \tag{5}$$

Based on equation 5, any newly generated ray r_n should only be compared with existing rays having cardinality less than that of the newly generated ray. We define cardinality to be a *type* of rays and group of existing rays based on their *types*. A group of *type* n will contain rays with cardinality less than or equal to n. In our approach, a newly generated ray with cardinality n will only be compared to the rays with

IWBDA 2014, June 11–12, 2014, Boston, Massachusetts, USA.

type n rays. Bitpattern trees can be used to implement each ray *type*.

3. RESULTS

We have applied this new method on test cases of various sizes including Chinese Hamster Ovarian cell [3], *Escherichia coli* [6], and *Helicobacter pylori* [5]. We have implemented the *Method of Types* on an existing tool for EFM computation gEFM [7]. The results of this new method on the test cases have shown that the number of comparisons can be reduced upto 78% compared to that of gEFM.

4. CONCLUSION

The *Method of Types* provides an improvement over the existing approaches by reducing the number of comparisons performed during adjacency testing. The reduction in number of comparisons is dependent on many factors such as network topology and constraint ordering. In worst case, the number of comparisons performed in *Method of Types* would be equal to the existing approaches.

5. ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant No. 0829899.

6. REFERENCES

[1] I. Csiszar. The method of types [information theory]. *Information Theory, IEEE Transactions on*, 44(6):2505–2523, Oct 1998.

[2] T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall. The double description method. *Annals of Mathematics Studies*, 2:51–73, 1953.

[3] L. Quek, S. Dietmair, J. KrÃűmer, and L. Nielsen. Metabolic flux analysis in mammalian cell culture. *Metab Eng*, 12(2):161–71, 2010.

[4] M. Terzer and J. Stelling. Accelerating the computation of elementary modes using pattern trees. *Algorithms in Bioinformatics*, 4175:333–343, 2006.

[5] I. Thiele, T. D. Vo, N. D. Price, and B. Palsson. Expanded metabolic reconstruction of *Helicobacter pylori* (iIT341 GSM/GPR): an in silico genome-scale characterization of single- and double-deletion mutants. *Journal of Bacteriology*, 187(16):5818–5830, 2005.

[6] C. T. Trinh, P. Unrean, and F. Srienc. Minimal *Escherichia coli* cell for the most efficient production of ethanol from hexoses and pentoses. *Appl Environ Microbiol*, 74(12):3634–43, 2008.

[7] E. Ullah, C. Hopkins, S. Aeron, and S. Hassoun. Decomposing biochemical networks into elementary flux modes using graph traversal. In *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics*,