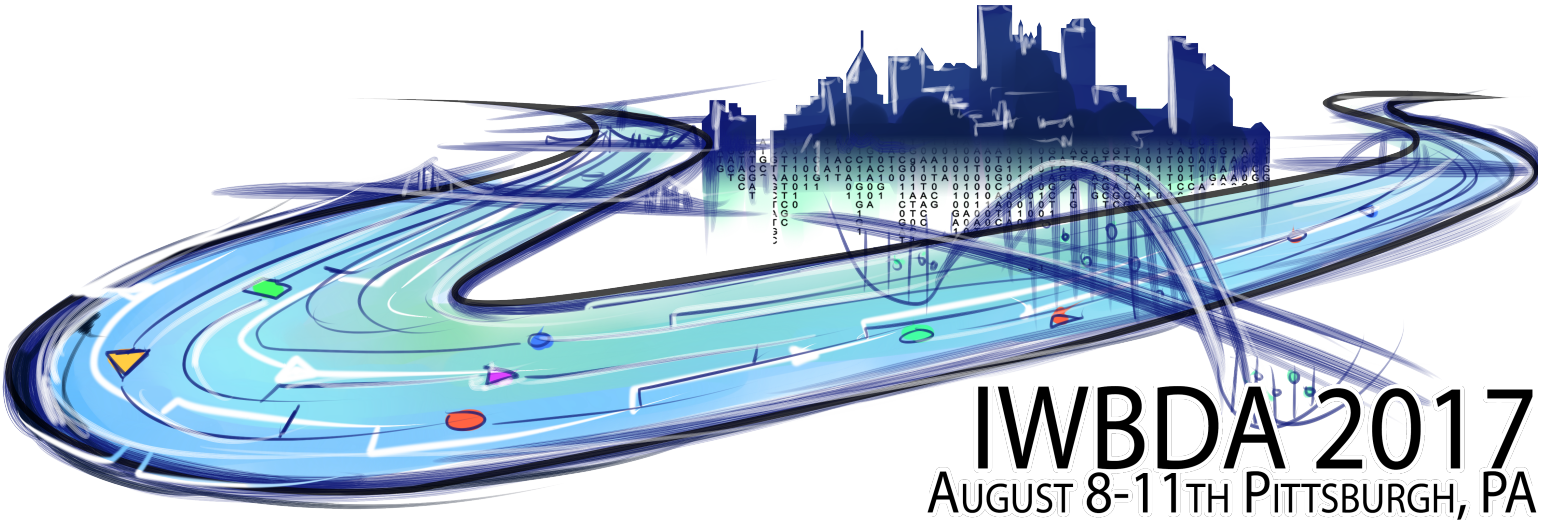


9th International Workshop on Bio-Design Automation
University of Pittsburgh
August 8-10th, 2017

Foreword

Welcome to IWBDA 2017!

The IWBDA 2017 Executive Committee welcomes you to Pittsburgh, PA, for the Ninth International Workshop on Bio-Design Automation (IWBDA). 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 14 poster presentations. Talks are organized into six sessions: Experiment-Machine Interfaces, Design Automation, Genetic Design Spaces, Modeling, Standards and Domain Specific Languages, and Machine Learning. In addition, we are very pleased to have three distinguished invited speakers: Dr. Caroline Ajo-Franklin from Lawrence Berkeley National Lab, Dr. Randy Rettberg from iGEM, and Prof. James Faeder from University of Pittsburgh.

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 SynbiCITE, Twist Bioscience, ACS Synthetic Biology, DSM, Raytheon BBN Technologies, Cytoscape, Amyris, Teselagen Biotechnology, Agilent Technologies and Minres Technologies for their support. We also thank The University of Pittsburgh for hosting and supporting IWBDA.

The following participants were provided financial support by our sponsors to attend IWBDA 2017

Evan	Appleton	Harvard Medical School
Hasan	Baig	Denmark Technical University (DTU)
Bryan	Bartley	University of Washington
Jacob	Becraft	Massachusetts Institute of Technology
Manuel	Giménez	Boston University
Yury	Ivanov	Boston University
Priya	Kapadia	Boston University
Ali	Lashkaripour	Boston University
Joshua	Lippai	Boston University
Curtis	Madsen	Boston University
Rizki	Mardian	Boston University
Shane	McCormack	Boston University
Tramy	Nguyen	University of Utah
Golestan Sally	Radwan	Royal Holloway University of London
Nicholas	Roehner	Boston University
Meher	Samineni	University of Utah
Radhakrishna	Sanka	Boston University
James	Scott-Brown	University of Oxford
Kestutis	Subacius	Boston University
Supreeta	Vijayakumar	Teesside University
Leandro	Watanabe	University of Utah

IWBDA 2017 Sponsors

Class + Table

Class

Algorithm

Host Institution

ACS Publications
Most Trusted. Most Cited. Most Read.

pubs.acs.org/synthbio

Submit Your Research to the IWBDA 2017 SPECIAL ISSUE

Submission Deadline:
SEPTEMBER 30, 2017

ACS Synthetic Biology, led by world-renowned scientists, is the cutting-edge forum where top scientists around the world publish their research in synthetic biology and systems bioscience. The journal is currently seeking submissions for the **2017 special issue** and encourages the use of SBOL visual for drawing genetic diagrams, and SBOL 2 for documenting the design of engineered genetic constructs.

EDITOR-IN-CHIEF

Christopher A. Voigt
Massachusetts Institute of Technology

AUTHOR BENEFITS

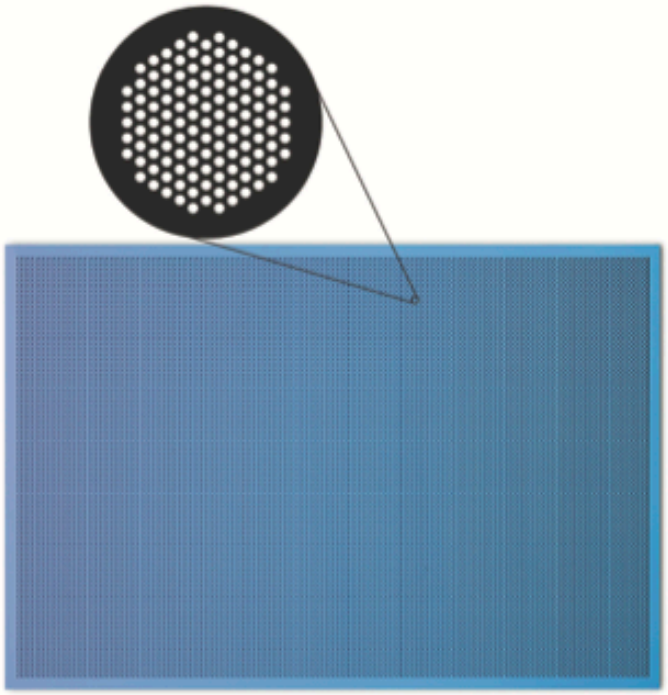
- RAPID PUBLICATION
- HIGHEST EDITORIAL STANDARDS
- NO PUBLICATION CHARGE
- HIGH IMPACT FACTOR
- GLOBAL EXPOSURE
- NEW OPEN ACCESS INITIATIVES

MINRES®
TECHNOLOGIES

Think Big, Screen Once

Accuracy and uniformity of oligo synthesis is critical when creating a CRISPR screening library. Twist Bioscience's technology enables massively parallel production of diverse high quality and accuracy oligo pools for generation of CRISPR gRNA libraries, allowing specific targeting and efficient screening.

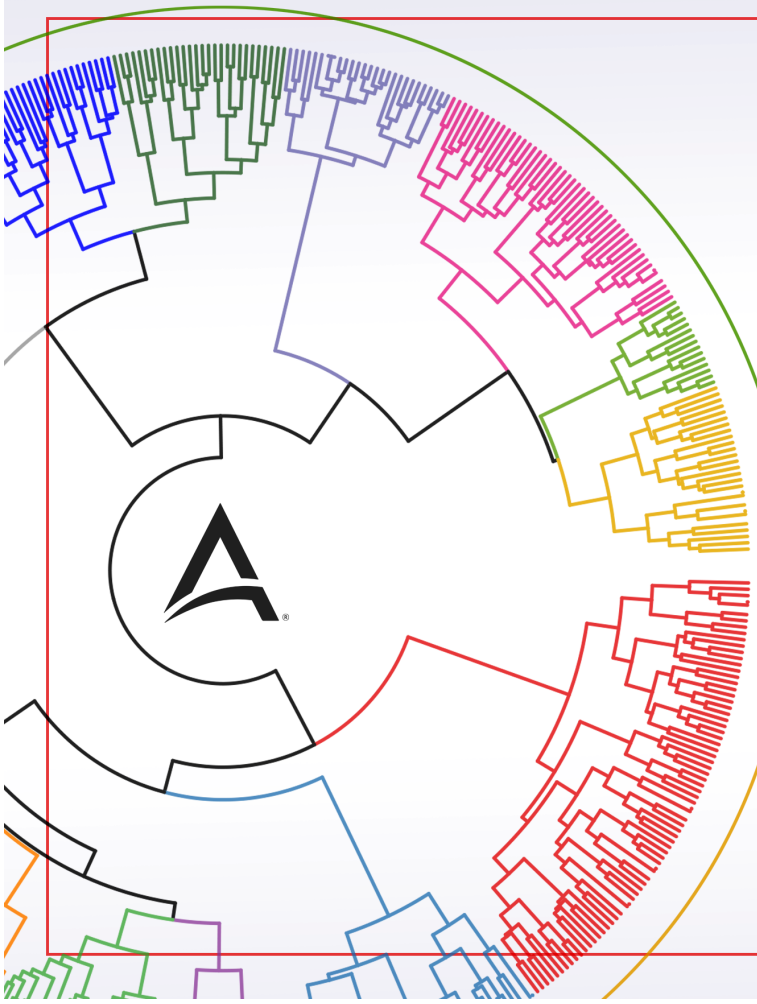
- Highly Accurate Synthesis For Specific Targeting
- Highly Uniform Synthesis Ensures Guide Representation
- Massively Parallel Guide Synthesis For Efficient Screening



TWIST'S SILICON-BASED PLATE WITH THOUSANDS OF CLUSTERS, EACH WITH 121 UNIQUE OLIGO SEQUENCES.

What can Twist do for you?
sales@twistbioscience.com

TWIST BIOSCIENCE



HIRING NOW

SCIENTIFIC COMPUTING
SOFTWARE ENGINEERING
& AUTOMATION

HERE AT AMYRIS
WE'RE BUILDING A COMPILER TOOLCHAIN
WHOSE TARGET ARCHITECTURE IS LIFE ITSELF

https://amyris.com/computing

Abstracts – Table of Contents

Oral Presentations

Poster Presentations

Organizing Committee

Executive Committee

General Chair
Natasa Miskov-Zivanov, University of Pittsburgh

Program Committee Chair
Nicholas Roehner, Boston University

Publication Chair
Evan Appleton, Harvard University

Web Chair
Aaron Adler, BBN Technologies

Finance Chair
Traci Haddock-Angelli, iGEM Foundation

Finance Co-Chair
Jacob Beal, Raytheon BBN Technologies

Local Chairs
Cheryl Telmer, Carnegie Mellon University; Khaled Sayed, University of Pittsburgh; Adam Butchy, University of Pittsburgh

Bio-Design Automation Consortium

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

Founders

Douglas Densmore, Boston University
Soha Hassoun, Tufts University
Marc Riedel, University of Minnesota
Ron Weiss, MIT

IWBDA 2017 Program

Tuesday, August 8th

08:00 - 07:30 (August 9th) — 2nd BDAthlon programming contest

Wednesday, August 9th

8:00 - 8:35 — Arrival, Breakfast, and Registration

8:35 - 8:50 — Opening Remarks
Natasa Miskov-Zivanov, General Chair

Talk Session I: Experiment-Machine Interfaces, Moderator: Evan Appleton

8:50 - 9:10 — Visual Expression of Experimental Protocols
James Scott-Brown and Antonis Papachristodoulou

9:10 - 9:30 — Automating Robots with Genetically Engineered Cells
Keith Heyde, John Lake and Warren Ruder

9:30 - 9:40 — Short Break

Keynote I

9:40 -10:40 — **Dr. Caroline Ajo-Franklin**
Engineering Microorganisms that Create and Communicate with Materials

Abstract: My research group uses biophysics and synthetic biology to engineer and explore the nanoscale interface between living microbes and inorganic materials. We are particularly interested in the basic mechanisms underlying charge transfer and assembly of materials at this living/non-living interface. In the first part of my talk, I will describe how we have engineered bi-directional electronic communication between living microbes and non-living systems using synthetic biology. By transplanting extracellular electron transfer pathways into the industrial organism Escherichia coli, we can confer upon these cells a molecularly-defined route to both accept and donate electrons to electrodes. Both current production and current consumption shift the metabolism of E. coli in well-defined ways, demonstrating that this electronic interface can control intracellular state. In the second part of my talk, I will describe how we have used surface-layer (S-layer) proteins as a programmable material. S-layer proteins form a highly ordered crystalline, yet porous, layer on the outermost cell surface of most species of bacteria and archaea. By engineering S-layers on the surface of different microorganisms, we can create arrays that modulate both the binding and mineralization of inorganic materials.

10:40 -11:10 — Coffee Break

Talk Session II: Design Automation, Moderator: TBD

11:10 - 11:30 — A Top-down Approach to Genetic Circuit Synthesis and Optimized Technology Mapping
Hasan Baig and Jan Madsen

11:30 - 11:50 — Heterologous Pathway Optimization using the Pathway Map Calculator
Sean Halper and Howard Salis

11:50 - 12:10 — Function-driven, Graphical Design Tool for Microfluidic Chips: 3DuF
Joshua Lippai, Radhakrishna Sanka, Ali Lashkaripour, and Douglas Densmore

12:10 - 12:20 — Short Break

Poster Pitches I

12:20 - 12:30 — Poster Pitches: 1 minute per poster

Lunch

12:30 - 12:35 — Announcements

12:35 - 13:30 — Lunch

Poster Session and Demos I

13:30 - 14:30 Poster Session

Keynote II

14:30 - 15:30 **Randy Rettberg** Title TBD

15:30 - 16:00 Coffee Break

Talk Session III: Genetic Design Spaces, Moderator: TBD

16:00 - 16:20 Mapping Genetic Design Space with Phylosemantics
Bryan A Bartley, Michal Galdzicki, Robert Sidney Cox, and Herbert M Sauro

16:20 - 16:40 Large-Scale Genetic Design: Moving From Designs to Design Spaces
Nicholas Roehner and Douglas Densmore

16:40 - 17:00 Modeling the Structural Determinants of mRNA Stability for Operon Design
Daniel Cetnar and Howard Salis

17:00 - 17:05 Announcements

Evening Activities

18:00 - 21:00 Conference Dinner at The Porch at Schenley Plaza.

Thursday, August 10th

8:00 - 8:30 Arrival, Breakfast, and Registration

Talk Session IV: Modeling, Moderator: Curtis Madsen

8:30 - 8:50 iBioSim 3: A Tool for Model-Based Genetic Circuit Design
Leandro Watanabe, Tramy Nguyen, Michael Zhang, Chris Myers, Curtis Madsen, and Nicholas Roehner

8:50 - 9:10 Standard Enabled Model Generator for Genetic Circuit Design
Göksel Misirli, Tramy Nguyen, James McLaughlin, Chris Myers, and Anil Wipat

9:10 - 9:30 Test-SFM: An automated model test system for systematic development of sequence-to-function models
Alexander Reis and Howard Salis

9:30 - 9:40 Short Break

Keynote III

9:40 - 10:40 **Prof. James Faeder** Rule-based modeling of cellular processes

Abstract: In this talk I will discuss challenges faced in developing detailed models of biochemical networks, that encompass large numbers of interacting components. Many of these same challenges arise iin synthetic biology. Although simpler coarse-grained models are often useful for gaining insight into biological mechanisms, detailed models are often necessary to understand how molecular components work in the network context and essential to developing the ability to manipulate such networks for practical benefits. The rule-based modeling (RBM) approach, in which biological molecules can be represented as structured objects whose interactions are governed by rules that describe their biochemical interactions, is the basis for addressing multiple scaling issues that arise in the development of large scale models. Currently available software tools for RBM, such as BioNetGen, Kappa, and Simmune, enable the specification and simulation of large scale models, and these tools are in widespread use by the modeling community. I will review some of the developments that gave rise to those capabilities and describe our current efforts broaden the appeal of these tools as well as to better enable collaborative development of models through re-use of existing models and improving visual representations of models. It is expected that a number of these advances will also be useful in synthetic biology.

10:40 - 11:10 Coffee Break

Talk Session V: Standards and Domain Specific Languages, Moderator: TBD

11:10 - 11:30 A Test Suite for Compliance Testing of Software Support for the Synthetic Biology Open Language
Meher Samineni and Chris Myers

11:30 - 11:50 Toward Quantitative Comparison of Fluorescent Protein Expression Levels via Fluorescent Beads
Jacob Beal, Nicholas Delateur, Brian Teague, Ron Weiss, John Sexton, Sebastian Castillo-Hair, and Jeffrey J. Tabor

11:50 - 12:10 mLSI Design with MINT
Radhakrishna Sanka, Joshua Lippai, and Douglas Densmore

Poster Pitches II

12:10 - 12:20 Poster Pitches: 1 minute per poster

Lunch

12:20 - 12:25 Announcements

12:25 - 13:30 Lunch

Poster Session and Demos II

13:30 - 14:30 Poster Session

Talk Session VI: Machine Learning, Moderator: TBD

14:30 - 14:50 From machine reading to discrete models of cellular signaling
Khaled Sayed, Cheryl Telmer, and Natasa Miskov-Zivanov

14:50 - 15:10 Garuda: A Synthetic Biology Platform that Learns from Data
Rizki Mardian, Manuel Gimenez, Marcia Sahaya Louis, Radhakrishna Sanka, and Douglas Densmore

15:10 - 15:30 Forward Engineering of Optimal CRISPR Guide RNA Sequences via Machine Learning
Riley Doyle, Mark Dunne, Neil Humphryes-Kirilov, and Leigh Brody

15:30 - 16:00 Coffee Break

Discussion Session, Leader: TBD

16:00 - 17:00 Discussion Topic: TBA

Closing Remarks and Awards

17:00 - 17:15 Closing remarks
Natasa Miskov-Zivanov, General Chair

Friday, August 11th

09:00 - 17:00 2nd Annual Introductory SBOL Workshop

Keynote Presentation

Caroline Ajo-Franklin

Engineering Microorganisms that Create and Communicate with Materials

Dr. Caroline Ajo-Franklin is a Staff Scientist at Lawrence Berkeley National Laboratory (LBNL). Her scientific training started in Chemistry; she earned a B.S. in Chemistry at Emory University in 1997 and received her Ph.D. in Chemistry from Stanford University in 2004. She then trained as postdoctoral fellow in Synthetic Biology with Pam Silver at Harvard Medical School. She started her independent research career in 2007 at Lawrence Berkeley National Lab and was promoted to Career Staff Scientist in 2014. Dr. Ajo-Franklin has built a strongly interdisciplinary research program focused on engineering and exploring the interface between living organisms and non-living materials. Reflecting this, she now holds appointments at the Molecular Foundry, the Molecular Biophysics and Integrated Bioimaging Division, and the Synthetic Biology Institute at LBNL.

Her research group uses biophysics and synthetic biology to engineer and explore the nanoscale interface between living microbes and inorganic materials. She is particularly interested in the basic mechanisms underlying charge transfer and assembly of materials at this living/non-living interface. In the first part of her talk, she will describe how they have engineered bi-directional electronic communication between living microbes and non-living systems using synthetic biology. By transplanting extracellular electron transfer pathways into the industrial organism Escherichia coli, they can confer upon these cells a molecularly-defined route to both accept and donate electrons to electrodes. Both current production and current consumption shift the metabolism of E. coli in well-defined ways, demonstrating that this electronic interface can control intracellular state. In the second part of her talk, she will describe how they have used surface-layer (S-layer) proteins as a programmable material. S-layer proteins form a highly ordered crystalline, yet porous, layer on the outermost cell surface of most species of bacteria and archaea. By engineering S-layers on the surface of different microorganisms, they can create arrays that modulate both the binding and mineralization of inorganic materials.

Keynote Presentation

Randy Rettberg

President and Founder of the iGEM Foundation

Randy Rettberg is the President and Founder of the iGEM Foundation. The foundation's primary programs are the International Genetically Engineered Machine competition (iGEM), which is the premier educational program in synthetic biology, and the iGEM Registry of Standard Biological Parts, the largest collection of standard biological parts for the synthetic biology community. iGEM began at MIT in 2003 with a January undergraduate design course. The first intercollegiate competition was in 2004. By 2015, iGEM has grown to over 280 teams in 37 countries, including 40 high school teams.

Prior to 2001, Randy was an executive in the computer industry working for Apple Computer and Sun Microsystems. Randy began his computer and networking career at Bolt Beranek and Newman as part of the ARPANET development team. At BBN, Randy worked on the early Internet and computer architecture. In 2001, after a 29-year tech career, Randy returned to MIT to work with Dr. Tom Knight to develop the field of synthetic biology.

Keynote Presentation

James Faeder

Rule-based modeling of cellular processes

Prof. James Faeder is Associate Professor of Computational and Systems Biology at the University of Pittsburgh School of Medicine. He is also Co-Director of the Joint Carnegie Mellon--University of Pittsburgh Ph.D. Program in Computational Biology and Department Vice Chair for Educational Programs. His research focuses on computational modeling of cell regulatory networks. His research combines development of novel methodologies with applications to specific systems of biological and biomedical relevance, including the immune system and cancer. He collaborates actively with experimental scientists both within the University of Pittsburgh as well as nationally and internationally. His work has been published in many high-profile journals including Science Signaling, Nature Methods, PLOS Computational Biology, and Bioinformatics. Dr. Faeder was a founding organizer of the well-known q-bio Conference on Cellular Information Processing, is a member of the Board of Reviewing Editors of Science Signaling, and has served as ad hoc member of numerous NIH study sections.

In this talk, he will discuss challenges faced in developing detailed models of biochemical networks, that encompass large numbers of interacting components. Many of these same challenges arise in synthetic biology. Although simpler coarse-grained models are often useful for gaining insight into biological mechanisms, detailed models are often necessary to understand how molecular components work in the network context and essential to developing the ability to manipulate such networks for practical benefits. The rule-based modeling (RBM) approach, in which biological molecules can be represented as structured objects whose interactions are governed by rules that describe their biochemical interactions, is the basis for addressing multiple scaling issues that arise in the development of large scale models. Currently available software tools for RBM, such as BioNetGen, Kappa, and Simmune, enable the specification and simulation of large scale models, and these tools are in widespread use by the modeling community. He will review some of the developments that gave rise to those capabilities and describe our current efforts broaden the appeal of these tools as well as to better enable collaborative development of models through re-use of existing models and improving visual representations of models. It is expected that a number of these advances will also be useful in synthetic biology.

Allan Kuchinsky Scholarship

Curtis Madsen

Dr. Curtis Madsen is a postdoctoral researcher in Prof. Doug Densmore's Cross-Disciplinary Integration of Design Automation (CIDAR) lab and Prof. Calin Belta's Hybrid and Networked Systems (HyNeSs) lab at Boston University. He received his Ph.D. in computer science from the University of Utah in 2013 working with Prof. Chris J. Myers on stochastic analysis of synthetic genetic circuits. Many of the contributions of his dissertation including the utilization of stochastic model checking and simulation to perform design space exploration of genetic circuits are implemented in the genetic design automation (GDA) tool, iBioSim. After completing his Ph.D., he worked as a research associate at Newcastle University in the Interdisciplinary Computing and Complex BioSystems (ICOS) research group under Prof. Anil Wipat and Dr. Paolo Zuliani. His two main focuses in ICOS were the development of a repository for the storing and publishing of Synthetic Biology Open Language (SBOL) data known as the SBOL Stack and the development of a tool that performs parameter set synthesis on biological models specified using the Systems Biology Markup Language (SBML) and corresponding time series data. Curtis currently works on the Bio Cyberphysical Systems (BioCPS) for Engineering Living Cells project. His research interests include the development of algorithms and techniques to improve the design and verification of biological systems, electronic circuits, and cyber-physical systems and the application of formal methods to problems from a variety of disciplines.

The third annual Allan Kuchinsky Scholarship to IWBDA is being generously sponsored by Agilent and Cytoscape.

Previous recipients
2016 Nicholas Roehner
2015 Swapnil Bhatia

Agilent Technologies

Cytoscape

Visual Expression of Experimental Protocols

James Scott-Brown
Department of Engineering Science,
University of Oxford
Parks Road
Oxford OX1 3PJ
james.scott-brown@eng.ox.ac.uk

Antonis Papachristodoulou
Department of Engineering Science,
University of Oxford
Parks Road
Oxford OX1 3PJ
antonis@eng.ox.ac.uk

ABSTRACT

Using robots to automate laboratory tasks could increase throughput and reproducibility, but requires experimental protocols to be specified in a computer-readable format. We present a new user interface (`'Lists of Liquids'`) for specifying experimental protocols by directly manipulating a task-graph: rather than having to specify individual liquid handling operations, the user can simply specify that particular lists of `Aliquots` should be combined as either a Cartesian product or convolution, and the system will plan a series of liquid handling steps to achieve this. This is intended to provide a higher-level interface that may make the creation of protocols faster and less error prone.

As an alternative to the graphical interface, the same abstractions are also provided through a python package and a textual domain-specific language.

KEYWORDS

high-level programming language; lab automation; rapid prototyping; reproducibility

ACM Reference format:
James Scott-Brown and Antonis Papachristodoulou. 2017. Visual Expression of Experimental Protocols. In *Proceedings of International Workshop on Bio-Design Automation, Pittsburgh, PA, US, August 2017 (IWBDA'17),* 2 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

There have been various attempts at constructing formal languages for expressing experimental protocols. These are intended to reduce ambiguity compared to natural language and enable either a robot to automatically execute a protocols, or a computer to guide a human operator through a protocol. These include both textual interfaces (e.g. BioCoder [1], PaR-PaR [6], Aquarium [4], Autoprotocol [8], Antha [7], and Symbolic Lab Language [5]) and graphical interfaces (e.g. BioBlocks [3], Wet Lab Accelerator [2], and proprietary tools from robotic systems manufacturers).

However, these interfaces place the primary focus on the specific liquid handling steps to be performed, rather than their intended results. We present here an interface that reverses this focus, allowing the user to specify operations at a higher level, and have the computer plan the lower-level details. In our system, the user expresses what they want to achieve at a high level, and the system determines what movement of liquids between physical locations are necessary to achieve this.

Regardless of which interface is used, it is possible to produce a range of outputs, including a diagram of the specified operations, a list of what each physical well will ultimately contain, and an executable protocol in Autoprotocol format.

IWBDA'17, August 2017, Pittsburgh, PA, US
2017. ACM ISBN . . . $15.00
https://doi.org/10.475/123_4

2 MODELLING CONCEPTS

We keep track of `Aliquot` objects, which contain a specified `Volume` of one or more liquids. Some of these are initially present, and can be regarded as inputs to the protocol (e.g. reagents or samples to be analysed). Others are defined as the result of mixing operations that are part of the protocol.

Two or more `Aliquots` can be combined to obtain a new `Aliquot`. Alternatively, an `Aliquot` can be combined with a `Volume`, producing a new `Aliquot` object with the same proportional composition but the specified volume.

As well as combining two single objects, it is possible to combine two lists of objects. This can be done in two ways: given two lists x and y, the Cartesian product/`cross` product independently combines every element of x with every element of y, whereas the convolution/`zip` operation combines each element of x with only the corresponding element of y.

3 GRAPHICAL INTERFACE

The graphical interface is implemented as a web-application. The back-end is written in Python using the Flask framework; the front-end is written in Javascript using the D3 library. Once a user has logged in they are able to create a new protocol, or edit a protocol that they have previously created.

Protocols can be represented as Directed Acyclic Graphs, and drawn as node-link diagrams in which the nodes are `Aliquot` objects (or lists of `Aliquot` objects), and the operations that combine these (e.g. `zip` and `cross`). Volumes are labeled on the corresponding arrows.

Right-clicking on the background displays a context menu that allows the user to add an initially present `Aliquot` object.

The user can click on a node and drag onto another to indicate that they should be combined; this creates nodes corresponding to the combination operation and the resulting

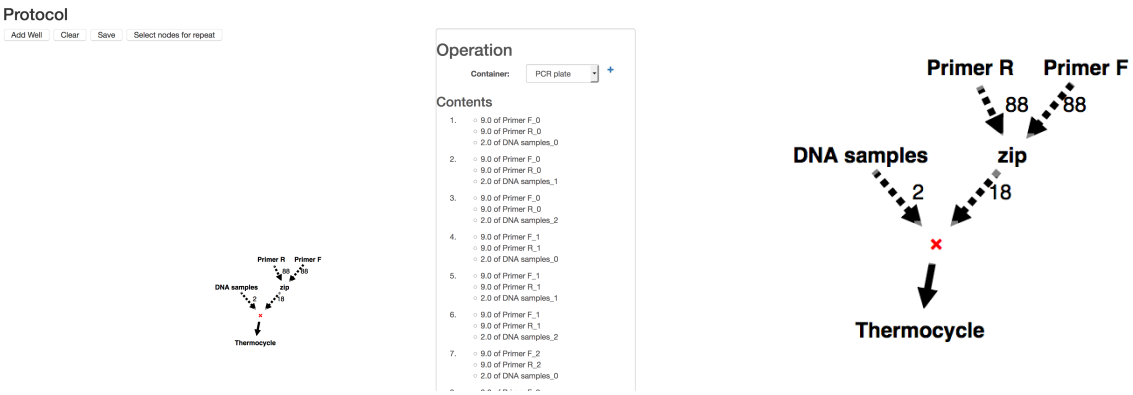


Figure 1: Screenshot of editor (left) and close up view (right). The protocol is for a PCR experiment, in which 2 µL of each DNA sample is separately mixed with 0.8 µL of each left primer (and 0.8 µL of the corresponding right primer), combined with 16.4 µL of PCR master mix, and thermocycled. In the corresponding domain-specific language, the liquid-handling part of this protocol could be specified as `16.4 microliter from master_mix + 2 microliter from sample_plate CROSS (0.8 microliter from left_primers ZIP 0.8 microliter from right_primers)`. In alternative systems, the user would instead specify individual liquid handling operations, such as which DNA samples should be transferred to which locations on a PCR plate.

object. A context menu allows the operation type to be changed.

The diagram visually distinguished different ways in which things can be combined: a solid arrow indicates that the corresponding `Aliquot`s remained in the same well as things were done to them. When two things are combined, the first could be added to the second (dashed arrow from first, solid arrow from second), the second could be added to the first (dashed arrow from second, solid arrow from first), or both could be added to a new, empty, container (dashed arrows from both). By default, it is assumed that the `Aliquot`s that were dragged are added to the `Aliquots` they were dragged onto, but this can be changed via a context-menu.

Other operations that perform some kind of processing (rather than simply combining liquids), such as thermocycling, can also be created by right-clicking on the background; dragging from an `Aliquot` to the newly-created node then indicates what the operation should be performed on.

Clicking on a node selects it, displaying its details in an information panel through which they can be edited. For `Aliquot` nodes, this displays the contents of each element of the list, which cannot be directly edited. For processing operations, this allows the relevant parameters to be edited (e.g. temperatures and durations for thermocycling).

4 FUTURE WORK

Future work will expand the range of output options, so that `Lists of Liquids` can be interfaced with more systems, and validate it by using it practically for wet lab experiments.

It is also intended to provide additional user-interface refinements, including the ability to select a subgraph of the complete task graph, and collapse this into a single block to reduce visual clutter, or indicate that it should be repeated.

A link to a demo of the tool, and its source code, will be available at http://sysos.eng.ox.ac.uk/tebio/protocols.

ACKNOWLEDGMENTS

J.S-B acknowledges funding through the EPSRC & BBSRC Centre for Doctoral Training in Synthetic Biology (EP/L016494/1), and from the Dstl. AP acknowledges funding from the EPSRC (EP/M002454/1).

REFERENCES

[1] Vaishnavi Ananthanarayanan and William Thies. 2010. BioCoder: A programming language for standardizing and automating biology protocols. *Journal of Biological Engineering* 4, 1 (2010), 13. https://doi.org/10.1186/1754-1611-4-13

[2] Maxwell Bates, Aaron J. Berliner, Joe Lachoff, Paul R. Jaschke, and Eli S. Groban. 2017. Wet Lab Accelerator: A Web-Based Application Democratizing Laboratory Automation for Synthetic Biology. *ACS Synthetic Biology* 6, 1 (jan 2017), 167–171. https://doi.org/10.1021/acssynbio.6b00108

[3] Vishal Gupta, Jesús Irimia, Iván Pau, and Alfonso Rodríguez-Patón. 2017. BioBlocks: Programming Protocols in Biology Made Easier. *ACS Synthetic Biology* (jan 2017). https://doi.org/10.1021/acssynbio.6b00304

[4] Eric Klavins. 2016. Aquarium. http://klavinslab.org/aquarium.html. (2016). [Online; accessed 25-April-2017].

[5] Emerald Cloud Lab. 2017. How It Works. http://emeraldcloudlab.com/how-it-works. (2017). [Online; accessed 25-April-2017].

[6] Gregory Linshiz, Nina Stawski, Garima Goyal, Changhao Bi, Sean Poust, Monica Sharma, Vivek Mutalik, Jay D. Keasling, and Nathan J. Hillson. 2014. PR-PR: Cross-Platform Laboratory Automation System. *ACS Synthetic Biology* 3, 8 (2014), 515–524. https://doi.org/10.1021/sb4001728

[7] Synthace. 2017. Antha. http://www.antha-lang.org. (2017). [Online; accessed 25-April-2017].

[8] Transcriptic. 2016. Autoprotocol. http://autoprotocol.org. (2016). [Online; accessed 25-April-2017].

Automating Robots with Genetically Engineered Cells

Extended Abstract[†]

Keith C. Heyde
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, USA
kheyde@andrew.cmu.edu

John R. Lake
University of Pittsburgh
300 Technology Drive
Pittsburgh, USA
jol85@pitt.edu

Warren C. Ruder
University of Pittsburgh
300 Technology Drive
Pittsburgh, USA
warrenr@pitt.edu

ABSTRACT

We have demonstrated a new method for controlling the behavior and motion of mechatronic systems using genetically engineered organisms. We developed a fully automated system that cultures synthetically engineered living cells within a microfluidic channel. This was accomplished using open-source syringe pumps designed and built in-house. We coupled this fully automated bioreactor with an epifluorescence microscope that imaged the entrapped cells once every ten minutes. Then, using custom-built software, the images were processed, allowing us to compute a single digital signal based on the relative abundance of red or green fluorescent proteins. This signal was then used to control the motion of a robot through a simulated environment. To close the information loop, these simulated robots were allowed to interact with different simulated carbon sources. The corresponding physical carbon source was then injected into the microfluidic channel. In this manner, engineered cells acted as both a sensing platform and as a processing unit for the simulated robot.

CCS CONCEPTS

• **Applied computing → Systems biology**

KEYWORDS

Synthetic Biology, Systems Biology, Robotics, Medical Robotics, Life Science Automation, Biomedical Engineering, Computational Biology, Instrumentation Design

ACM Reference format:

K.C. Heyde, J.R. Lake, and W.C. Ruder. 2017. IWBDA 2017 Conference Abstract.

1 INTRODUCTION

The natural world has served as inspiration for the design of automated robotic systems. By mimicking key physiological traits such as the surface properties of gecko feet[1], engineers have been able to build impressive artificial system that inform us about the natural world while optimizing performance. Similarly, the natural world has served as inspiration for control and processing components. Mutations, heredity, and fitness constraints that drive the evolution of living organisms served as inspiration for genetic algorithms, and the complex adaptive behavior of living brains led to the development of neural networks[2].

However, genetic processes and neural plasticity are not the only features that control behavior in the natural world. Increasingly, the microbiome, the collection of microorganisms living within a host organism, has been shown to affect the health and behavior of the host[3]. However, understanding the complexities of these interkingdom signaling networks is challenging due to the vast spatial and genetic heterogeneity of the microbiome[4].

Inspired by the strong connection between the microbiome and the host organism's behavior, we set out to develop a minimal, simplified system that allowed us to automate the behavior of a mechatronic "host" system using the state of a population of engineered living "microbiome" cells. We did this by building a fully automated bioreactor system that used mechatronic hardware and controls in conjunction with a microfluidic chip and an epifluorescence imaging system. Upon imaging the phenotypic state of our entrapped bacteria, we would process the image and use the resulting relative intensities of green or red fluorescent proteins to drive the motion of a robot.

2 EXPERIMENTAL DETAILS

2.1 Synthetically Engineered Cells

We cultured a population of genetically engineered *Escherichia coli* (*E. coli*) cells that contained a genetic toggle switch. This genetic feature is characterized by two bistable states of mutual repression. In practice, a well-engineered toggle switch behaves like a digital bit, with two mutually exclusive states. For our system, these two states were flipped by exposing cells to either arabinose or Isopropyl β-D-1-thiogalactopyranoside (IPTG). The two states corresponded to the production of a green fluorescent protein and a red fluorescent protein (mCherry) respectively. This feature created a sustained optical response from the engineered cells that corresponded to a transient chemical pulse.

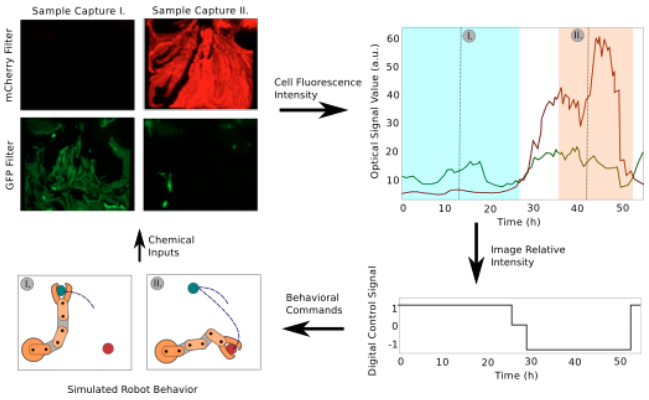


Figure 1: Using Engineered Living Cells to Automate the Behavior of a Simulated Robot

2.2 Microfluidic Bioreactor and Liquid Handling

We designed and fabricated a microfluidic bioreactor based off of previously published work. The bioreactor design was chosen to keep *E. coli* cells entrapped within an optical plane, while still maintaining the cells within an exponential phase of growth. We coupled this bioreactor with a set of three, custom-built syringe pumps controlled with a computer-linked Arduino microcontroller. Using a piezoelectric pressure transducer, the syringe pumps were able to control effluent fluid pressure. By filling the syringes with different concentrations of media, arabinose, and IPTG, the chemical composition of the fluid pumped into the microfluidic bioreactor could be well regulated.

2.3 Image Acquisition and Processing

The microfluidic bioreactor was placed on top of a Nikon Ti-E microscope and images were captured using a mounted Andor® Zyla Scientific CMOS camera. Once images were taken and background calibrated, using the Nikon Elements software package, a custom-built software package would compare the relative intensity of the green and red proteins. This relative intensity would then be sent to a finite state machine that governed a digital signal used to control robot behavior. All custom-built software was coded in Python 2.7.

2.4 Robot Simulation and System Feedback

We simulated a manipulator robot within a 2D virtual environment using a custom-built python script. The behavior of this robot was governed by the digital signal sent to it from the finite state machine described in section 2.3. Depending on the incoming signal, the simulated robot would actuate through a 2D arena containing targets corresponded to arabinose or IPTG. When the robot was in contact with one of the targets, then it would send a signal to the syringe pumps described in section 2.2, which would inject the corresponding chemical into the microfluidic channel.

2.5 System Integration and Automation

After building and testing each component of our system individually, we ran three trials of the integrated system. Real-time updates to the robot state and position were visible throughout the experiment, but not human intervention occurred during the experiment and results were only analyzed following the completion of the trials.

3 RESULTS AND DISCUSSION

We found that our integrated feedback system allowed genetically engineered cells to effectively control the motion of a robot in a predictable manner. As our cells were engineered to contain a genetic toggle switch, we expected that they would be able to automate the robot behavior to switch between two bistable states. We designed our experiment so that this bistability could be demonstrated. Our results indicated that this was true, with sustained cellular expression of mCherry or GFP (but not both) driving the robot to move in a predictable and consistent manner. We hope to expand this work by including more nuanced robot behaviors as well as more sophisticated regulatory gene networks to our cells.

4 CONCLUSIONS

We have demonstrated a novel system that allows synthetically engineered cells to control the motion of a simulated robot. Our system serves as a minimal “gut-brain axis” allowing a biomimetic “microbiome” to control the behavior of a “host” robot. We hope to use this system to explore how synthetically engineered cells may be used to automate mechatronic systems and to better understand the signaling dynamics between the microbiome and host organisms. We expect our findings will impact fields ranging from medicine to automation design for the life sciences.

ACKNOWLEDGMENTS

The authors gratefully acknowledge support from award N00014-17-12306from the Office of Naval Research of the USA and award FA9550-13-1-0108 from the Air Force Office of Scientific Research of the USA. Additionally, this material is based upon work supported by the National Science Foundation Graduate Research Fellowship.

REFERENCES

[1]Sitti, M., and Fearing, R. S. (2003) Synthetic gecko foot-hair micro/nano-structures for future wall-climbing robots. *IEEE International Conference on Robotics and Automation*

[2] Ayers, J. and Witting, J. (2007) Biomimetic approaches to the control of underwater walking machines. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*

[3] Montiel-Castro A. J., et al. (2013) The microbiota-gut-brain axis: neurobehavioral correlates, health, and society. *Frontiers in Integrative Neuroscience 7*

[4] Le Chatelier et al. (2013) Richness of human gut microbiome correlates with metabolic markers. *Nature*

A Top-down Approach to Genetic Circuit Synthesis and Optimized Technology Mapping

Hasan Baig and Jan Madsen
Department of Applied Mathematics and Computer Science
Technical University of Denmark
{haba, jama}@dtu.dk

1. INTRODUCTION

Genetic logic circuits are becoming popular as an emerging field of technology. They are composed of genetic parts of DNA and work inside a living cell to perform a dedicated boolean function triggered by the presence or absence of certain proteins or other species.

In this work, we introduce a top-down approach to synthesize genetic logic circuit. This approach is based on translating high-level description of genetic circuit (in the form of boolean function) to its low-level representation in the form of SBOL [1] notation. This approach is implemented in the *Gene*tic *Tech*nology mapping tool, *GeneTech*. It takes the Boolean expression of a genetic circuit as input, and then first optimize it. It then synthesizes the optimized Boolean expression into NOR-NOT form in order to construct the circuit using the real NOR/NOT gates available in the genetic gates library [2]. In the end, *GeneTech* performs technology mapping to generate all the feasible circuits, with different genetic gates, to achieve the desired logical behavior.

There are some existing tools which supports technology mapping of genetic circuits including Cello [2] and iBioSim [3]. *GeneTech* differs from these tools by generating all feasible genetic circuits from a Boolean expression. This work is originally inspired from the processes of optimization and technology mapping of electronic circuits in the *electronic design automation* (EDA) industry. In EDA, the combinatorial circuit optimization is always required to implement the circuit with the minimum number of logic gates [4]. This area-efficient implementation of digital circuits not only helps reducing the size of electronic devices but also avoid wasting power and redundant resources.

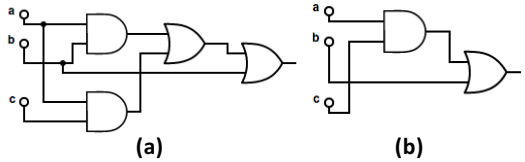


Figure 1. Digital circuit of the expression *ab+b+ac.*
(a) Original circuit. (b) Optimized circuit having two gates.

In order to get the insight of logic optimization, consider the digital circuit for the Boolean expression, $ab + b + ac$, shown in Figure 1(a). In this figure, the circuit consists of four logic gates. After running the optimization algorithm, the number of gates in the circuit reduces down to two while preserving the original functionality, as illustrated in Figure 1(b).

This optimization of digital electronic circuits seems simple and straight forward. However, the optimization and technology mapping of genetic circuits is not similar to electronic circuits. This is because the input and output quantities of electronic circuits are the same i.e. voltage, and therefore the electronic gates can easily be cascaded together. On the contrary, the input and output quantities of genetic gates are different, and therefore the signal matching has to be considered while mapping genetic gates on the circuit. This makes it very challenging to integrate genetic logic gates to construct complex genetic circuits. Similar to the above process of optimizing digital logic in electronic circuits, we want to avoid having redundant logic in genetic circuits as well.

2. METHODOLOGY

Two different ways to represent the same boolean logic or digital circuit are the *minterm* and *maxterm* canonical forms. Minterms are also called the *products* because the variables (or literals) in the Boolean expressions are represented as the *logical AND*. Maxterms are referred to as *sums* because the variables (or literals) are represented as the *logical OR*. Therefore, the same Boolean function can either be expressed as the *sum of products/minterms* (SOP) or the *product of sums/maxterms* (POS), as shown in equation (1). In this example, the left-hand side represents the SOP form and the right-hand side represents its equivalent POS form.

$$ab + b + ac = (a + b + c)(a + b + \bar{c})(\bar{a} + b + c) \quad (1)$$

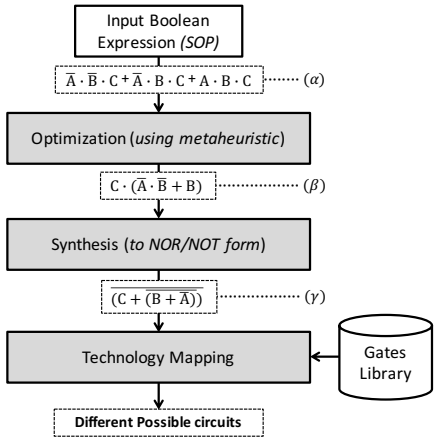


Figure 2. The technology mapping flow of *GeneTech*.

The flow of genetic technology mapping in *GeneTech* is shown in Figure 2. It takes the raw Boolean expression in the SOP form and then first optimize it using the *simulated annealing* (SA) [5] optimization algorithm. The goal of optimization at this step is to reduce the number of variables (or literals) in the expression while keeping the output logic the same. Reducing the number of literals in the Boolean expression results in the reduction of logic components required to obtain the desired logic.

To construct real genetic circuits, *GeneTech* uses the gates library from [2], which consists of genetic gates in the form of NOR and NOT functions. Therefore, to map the genetic gates on the Boolean expression, it is necessary to bring it into NOR/NOT form. Hence, when the Boolean expression is optimized, it then goes to a process of synthesis, as shown in Figure 2. Once the Boolean expression is available in NOR/NOT form, a mapping

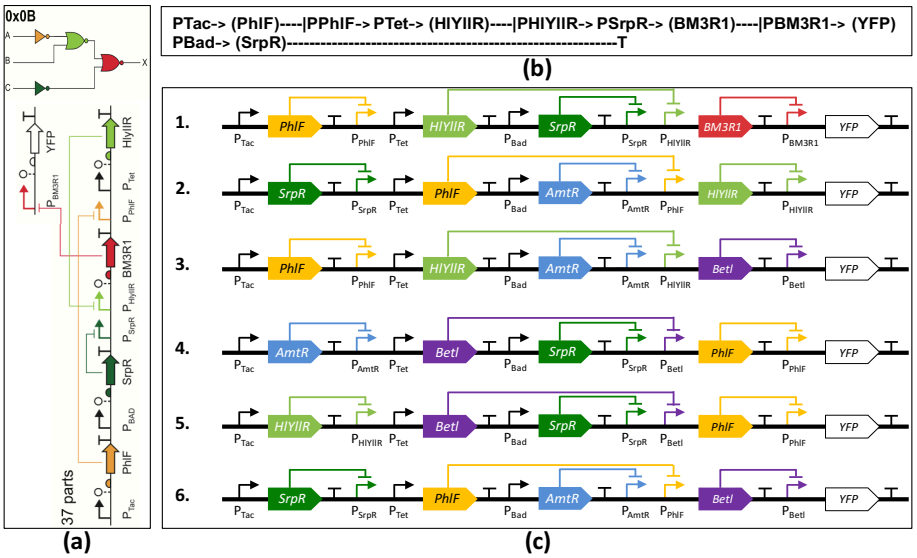


Figure 3. Experimental results of *GeneTech* for 0x0B [2]. (a) Circuit schematic and SBOL representation of 0x0B shown in [2]. (b) Circuit representation generated by *GeneTech*. (c) The SBOL notations of all possible circuits generated by *GeneTech* to achieve the same logic of the circuit 0x0B.

algorithm of *GeneTech* checks for the logic components in the gates library and find all feasible genetic circuits.

We have extracted the genetic gates by analyzing the SBOL notations of all the circuits shown in [2] and organized them in separate lists of genetic NOT and NOR gates. The algorithm is based on a deterministic depth-first search approach and maps the genetic gates on the deepest most elements in the expression. For example, in the expression (γ) shown in Figure 2, all NOT gates from the library which are compatible with $\overline{A}$ are selected first. Then the mapping algorithm checks for any available NOR gate with one of the input as B and the other input matching to the outputs of any of the NOT gates selected previously for $\overline{A}$. If any such NOR gate is found, the algorithm then search for another NOR gate with the inputs compatible to the output of first NOR gate and the output of any of the NOT gates available for $\overline{C}$. In this way, all the compatible components are used to achieve the same boolean functionality with different possible genetic circuits.

While constructing genetic circuits, *GeneTech*, avoid using those genetic gates which generate the same output protein. This is to make sure that the signals of the gates do not interfere with each other.

3. EXPERIMENTATION AND RESULTS

We performed experiments on the genetic circuits shown in [2]. Due to space limitation, the results of one circuit, 0x0B, are shown in Figure 3. Figure 3 (a) shows the schematic and SBOL representation of the circuit 0x0B obtained directly from [2].

The input expression of the circuit 0x0B is obtained from the truth table given in [2], which is shown as expression (α) in Figure 2. After optimization, it is reduced to the expression (β) shown in Figure 2. Afterwards the synthesis is performed to bring this expression into NOR/NOT form shown as (γ) in Figure 2. Figure 3(b) shows the multi-line text string format (similar to SBOL notation) which is used by *GeneTech* to represent the structure of a generated circuit. In Figure 3(b), promoters are shown with the symbol "->", the proteins are represented by round braces "( )", and the repression is indicated by the symbol "----|" or "T". Figure 3(b) indicates that the P_{Tac} promoter generates a protein PhlF which in turn represses the output promoter P_{PhlF}. The promoter P_{PhlF} together with the promoter P_{Tet} generate the protein HlYllR, which represses the output promoter P_{HlYllR}. In the second line, promoter P_{Bad} generates the protein SrpR which supresses its corresponding output promoter P_{SrpR}. This promoter P_{SrpR} together with the promoter P_{HlYllR} generate the protein BM3R1, which represses the activity of the output promoter P_{BM3R1}. The promoter P_{BM3R1} is used to produce the output indicator, the *yellow fluorescent protein* (YFP). The SBOL notation of Figure 3(b) is shown as notation 1 in Figure 3(c). Figure 3(c) shows the SBOL representations of the circuits generated by *GeneTech* tool. This figure shows that the *GeneTech* tool, beside suggesting the solution given in [2] (Figure 3(c)-1), it also finds all other possible circuits to achieve the same logic function using other genetic components available in the gates library [2].

4. SUMMARY

In Cello [2], circuits are constructed by selecting the appropriate genetic components, based on matching their threshold levels, through non-deterministic search using simulated annealing algorithm. Therefore, for every compilation of the same code in Cello, the generated circuit may contain the same or different genetic components. On the contrary, *GeneTech* gives the number of possible solutions to achieve the same logic. With the correct set of parameters, the threshold levels of these circuits can then be obtained using D-VASim [6] and then can be verified in the laboratory. More design constraints can be added in *GeneTech* to make sure that the circuits generated by this tool would work in the laboratory. Furthermore, *GeneTech*, at its current state, supports technology mapping of genetic gates based on repression. In future, it will be upgraded to support genetic gates based on other technologies.

5. REFERENCES

[1] B. Bartley, *et al*., "Synthetic Biology Open Language (SBOL) Version 2.0.0.," *J. Integr. Bioinform.*, 2015.

[2] A. A K. Nielsen, *et al*., "Genetic circuit design automation", *Science*, vol. 352, no. 6281, pp. 7341, 2016.

[3] N. Roehner and C. J. Myers, "Directed Acyclic Graph-Based Technology Mapping of Genetic Circuit Models," *ACS Synth. Biol*, 2014.

[4] Giovanni De Micheli, "Synthesis and Optimization of Digital Circuits," McGraw-Hill Series in Elec. and Comp. Engg., 1994.

[5] S. Kirkpatrick, *et al*., "Optimization by Simulated Annealing", *Science (80)*, vol. 220, no. 4598, pp. 671–680, 1983.

[6] H. Baig and J. Madsen, "Simulation Approach for Timing Analysis of Genetic Logic Circuits," *ACS Synth. Biol*., Feb. 2017.

Heterologous Pathway Optimization using the Pathway Map Calculator

Sean Halper
Pennsylvania State University
Department of Chemical Engineering
University Park, Pennsylvania 16802
sxh456@psu.edu

Howard Salis
Pennsylvania State University
Department of Chemical Engineering
Department of Biological Engineering
University Park, Pennsylvania 16802
salis@psu.edu

ABSTRACT

The optimization of heterologous metabolic pathways poses a combinatorial challenge as pathway size increases that hinders forward engineering efforts. To overcome these challenges, we developed an automated algorithm that uses a small number of characterized pathway variants to identify the expression-productivity relationship of a many-enzyme pathway, called a Pathway Map, by combining elementary mode analysis, kinetic metabolic modeling, model reduction, de-dimensionalization, and genetic algorithm optimization. We confirm the algorithm makes accurate prediction of optimal enzyme expression levels for pathways of varying size and complexity using in silico pathway examples, and have successfully optimized multiple heterologous pathways in E. coli with this method, the most recent being the NADPH regenerating Entner-Doudoroff pathway of *Z. mobilis*. The algorithm is agnostic to the genetic parts used to vary expression and the host organism of the pathway, broadening its pathway optimization applications.

CCS CONCEPTS

• **Applied computing** → **Biological networks**; *Metabolomics / metabonomics*; Systems biology;

KEYWORDS

Metabolic engineering, kinetic modeling, structured machine learning, design of experiments

ACM Reference format:
Sean Halper and Howard Salis. 2017. Heterologous Pathway Optimization using the Pathway Map Calculator. In *Proceedings of , Pittsburgh, PA USA, August 2017 (IWBDA'17),* 2 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Heterologous metabolic pathways enable organisms to manufacture a wide variety of chemical products, but to achieve economic viability in a scaled-up process, the pathway being heterologously expressed must be tuned to maximize productivity and titer. To this end, heterologous metabolic pathway optimization is currently carried out by constructing and characterizing many pathway variants, incorporating different regulatory genetic parts such as promoters [1], ribosome binding sites (RBS) [5] and origins of replication to vary the enzymes'expression levels [3].

IWBDA'17, August 2017, Pittsburgh, PA USA
2017. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

However, these design approaches present a difficult challenge, particularly when optimizing many-enzyme heterologous pathways, such as natural product pathways. Synergistic effects between enzymes makes the collective tuning of enzyme expression a requirement for an optimal pathway. Larger pathways pose a high-dimensional combinatorial problem that library-based approaches cannot adequately address, since the number of experimental measurements needed to inform pathway design decisions is too vast for the throughput of conventional characterization methods. Characterizing a tiny fraction of these pathways will yield a low chance of finding the pathway's optimal enzyme expression levels, unless a design approach that can use sparse, high throughput characterization data is used **(Figure 1A)**.

Here, we describe an algorithm, the Pathway Map Calculator, which learns the non-linear relationship between a pathway's enzyme expression levels and its end-product productivities. Using this relationship, called a Pathway Map, we predict the relative enzyme expression levels that maximize the pathway's end-product productivity. The Pathway Map enables efficient optimization of the pathway's enzyme expression levels, provides insights into the pathway's rate-limiting steps, and facilitates prioritization of protein engineering efforts to improve the slowest enzymes'kinetics **(Figure 1B)**.

2 THEORY AND CALCULATIONS

2.1 Required Inputs

The algorithm requires two types of inputs: first, an experimental data-set consisting of end-product measurements and relative enzyme expression levels for each characterized pathway variant; and second, a candidate network that lists the enzymes'reactions and the corresponding metabolite stoichiometries **(Figure 1B)**. Inputted end-product measurements may be assayed using any proportional measurement, such as LC-MS, GC-MS, enzyme-linked assays, or fluorescent biosensors. Inputted relative enzyme expression levels may be derived from either measurements or model predictions of the genetic parts'activities.

2.2 Model Generation and Parameterization

The algorithm automatically generates a system of de-dimensionalized, reduced differential equations that quantify the relationship between the metabolic network s enzyme expression levels and its metabolic fluxes. We use elementary mode analysis [8] to enumerate the candidate reaction network's elementary flux modes, and to convert all known metabolic fluxes into model constraints.

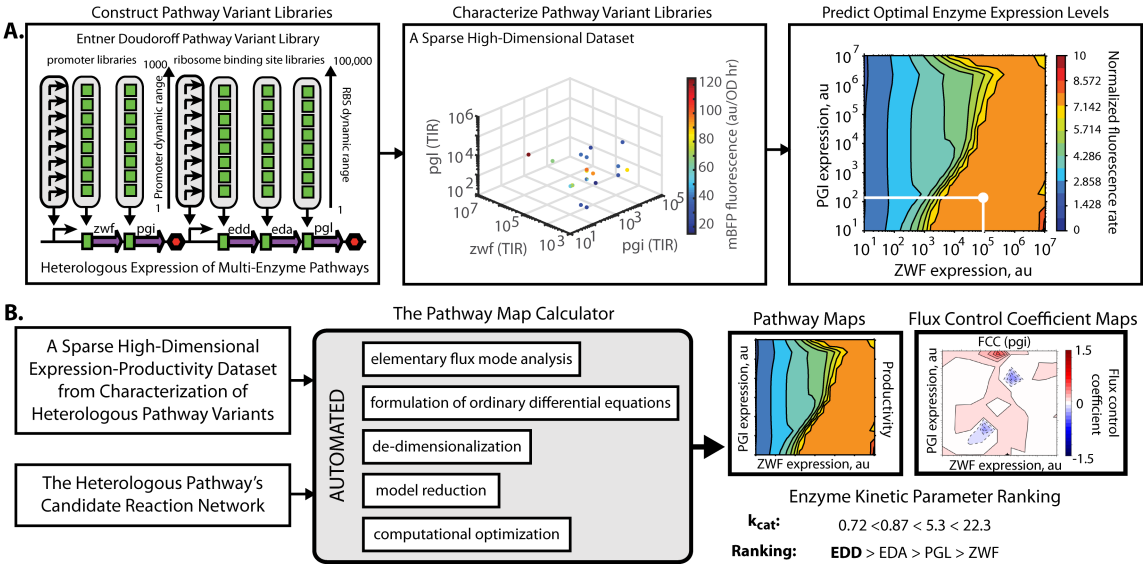


Figure 1: Overview of the Pathway Map Calculator. A) The design pipeline using the Pathway Map Calculator. After initially characterizing the enzyme-productivity relationship of a pathway with a small, coarse, non-optimal pathway variant library, the algorithm enables rapid prediction of that pathway's behavior with different enzyme expression levels, informing further design-build-test rounds B) Overview of the Pathway Map Calculator's operation. Accepting two inputs, the algorithm processes that data into 3 outputs: Pathway Maps, flux control coefficient maps, and rankings for protein engineering

We next generate the kinetic metabolic model, formulated as a system of ordinary differential equations, to quantify the rates of all reactions in terms of their metabolite and enzyme concentrations. Each enzyme-catalyzed reaction is broken down into a set of reversible first-order and second-order reactions according to the enzyme's reaction mechanism. We then apply dedimensionalization and model reduction, inspired by ensemble modeling [6, 7], to reduce the degrees of freedom required to parameterize the kinetic model and apply bounds to the model parameters.

We parameterize the kinetic model using a genetic algorithm to assign kinetic parameter values, and fit the resulting predictions to the provided pathway variant data, resulting in a predictive, mechanistic model of the pathway's behavior.

3 RESULTS

3.1 Outputs

The Pathway Map Calculator produces Pathway Maps, smooth, differentiable relationships linking the expression of a pathway's enzymes to its end-product productivities. All predictions are quantitative, experimentally actionable, and re-usable for a variety of applications. Additionally, the algorithm can provide mechanistic insights into a pathway's rate-limiting step via the calculation of flux control coefficients for each enzyme in the pathway [2], as well as prioritization of protein engineering efforts via estimations of each enzyme's Michaelis-Menten kinetics.

3.2 Applications

We have validated the algorithm with many in-silico examples, finding that the Pathway Map Calculator can predict optimal pathway variants for a large pathway using a small library of pathway variants. We have applied the algorithm to several pathways to optimize their productivities, including our most recent effort, the Entner Doudoroff pathway of *Zymomonas mobilis* [4] for regenerating NADPH.

The algorithm's web interface can be accessed online at: https://salislab.net/software/PathwayMapCalculator

REFERENCES

[1] H. Alper, C. Fischer, E. Nevoigt, and G. Stephanopoulos. 2005. Tuning genetic control through promoter engineering. *Proceedings of the National Academy of Sciences of the United States of America* 102 (2005), 12678–12683.
[2] D. A. Fell. 1998. Increasing the flux in metabolic pathways: A metabolic control analysis perspective. *Biotechnology and Bioengineering* 58 (1998), 121–124.
[3] V. K. Mutalik, J. C. Guimaraes, G. Cambray, C. Lam, M. J. Christoffersen, Q.-A. Mai, A. B. Tran, M. Paull, J. D. Keasling, A. P. Arkin, and D. Endy. 2013. Precise and reliable gene expression via standard transcription and translation initiation elements. *Nat. Meth.* 10 (2013), 354–360.
[4] C. Y. Nn, I. Farasat, C. Maranas, and H. M. Salis. 2015. Rational Design of a Synthetic Entner-Doudoroff Pathway for Improved and Controllable NADPH Regeneration. *Metabolic Engineering* 29 (2015), 86–96.
[5] H. M. Salis, E. A. Mirsky, and C. A. Voigt. 2009. Automated design of synthetic ribosome binding sites to control protein expression. *Nat. Biotech.* 27 (2009), 946–950.
[6] Y. Tan, J. G. L. Rivera, C. a. Contador, J. a. Asenjo, and J. C. Liao. 2011. Reducing the allowable kinetic space by constructing ensemble of dynamic models with the same steady-state flux. *Metabolic Engineering* 13 (2011), 60–75.
[7] L. M. Tran, M. L. Rizk, and J. C. Liao. 2008. Ensemble modeling of metabolic networks. *Biophysical Journal* 95 (2008), 5606–5617.
[8] C. Wagner. 2004. Nullspace Approach to Determine the Elementary Modes of Chemical Reaction Systems. *The Journal of Physical Chemistry B* 108 (2004), 2425–2431.

Function-driven, Graphical Design Tool for Microfluidic Chips: 3DuF

Joshua Lippai[1,2], Radhakrishna Sanka[1,2], Ali Lashkaripour[2,3] and Douglas Densmore[1,2]

[1]Department of Electrical & Computer Engineering, Boston University, Boston, MA
[2]Biological Design Center, Boston University, Boston, MA
[3]Department of Biomedical Engineering, Boston University, Boston, MA
{jlippai,sanka,lashkari,dougd}@bu.edu

1. INTRODUCTION

The use of microfluidic chips for applications in biology to reduce the cost, time, and difficulty of automating experiments, while promising, has proven to have barriers to entry. In particular, the cost of the equipment required for manufacturing techniques like soft lithography, the difficulty in designing functional microfluidic chips, and the time associated with manufacturing them have made rapid production for prototyping and iterative design difficult. Our lab's microfluidics design flow is capable of automating much of the design process of microfluidic chips using the paradigm of defining them as primitives placed on a layout grid and exporting standard formats for use in fabrication [1]. 3DuF, a design tool that allows the user to carry out the placement and connection of primitives through a browser-based GUI, simplifies the design process to specifying the primitives through parameters and using a pointer to connect them with channels. But this approach assumes that the designer knows exactly what physical dimensions the primitives need for the chip to perform adequately for experiments, which may not be the case if sufficient literature or a fluid dynamics expertise are not present. By communicating with DAFD, our lab's currently in-development database and model-fitting framework, 3DuF will be able to define microfluidic primitives for placement on chip layouts not only through physical dimensions, but also by specific performance metrics desired of the primitives' functions, which will result in automatically generated dimensions for those primitives. This will allow chip design through the simple paradigm of using a GUI to place primitives and connect them with channels, while also making a useful definition of those primitives for the designer's needs less reliant on their fluid dynamics expertise.

2. 3DUF - MODULAR DESIGN AND STANDARD I/O

3DuF allows design of microfluidic chip layers by selecting from a predefined list of microfluidic primitives, called MINT [3], specifying the physical dimensions that define the physical characteristics of those primitives, and placing them on a web page canvas where they can be connected with channels [2]. All of this is done with a point-and-click GUI, removing the need to draw a chip entirely by hand or learn a programming language to define the chip, publicly available online at http://www.3duf.org. Additionally, 3DuF's ability to read chip layouts from a JSON standard defined in our lab and render based on those, as well as to export chip designs in standard formats including SVG, make its designs compatible with a wide range of fabrication flows. In particular, in the context of our lab's other tools, 3DuF has been used as a rendering engine for chips designed by our automated place-and-route engine, Fluigi Core, as well as the visual component of our chip control page in the end-to-end GUI tool, Neptune. The SVG output from 3DuF has been used both in this context and with 3DuF as the standalone design tool to fabricate physical chips even with inexpensive methods like CNC milling in our fabrication flow, Makerfluidics [4]. The flow of these components working together, as well as the new component our parameter definition automation component, DAFD, would enable, is shown in Figure 1.

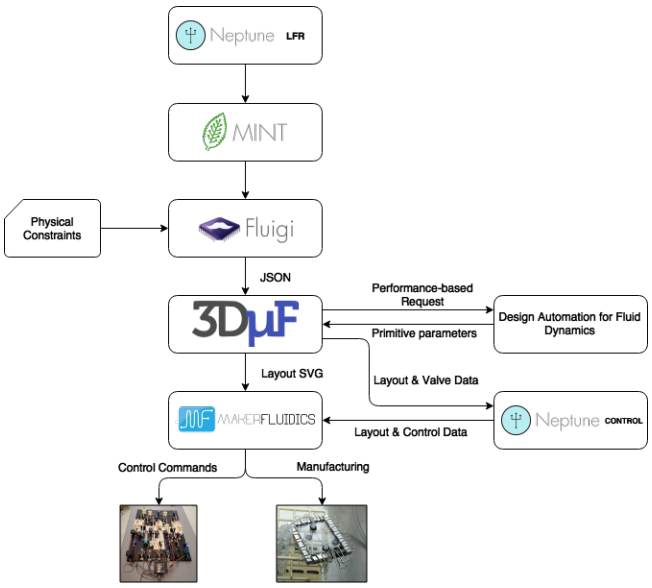


Figure 1: The connected flow of how all of our lab's microfluidic design, fabrication, and control components can work together

3. SIMPLIFYING DESIGN WITH DAFD

Our lab's developing tool for Design Automation for Fluid Dynamics, or DAFD, will further enhance 3DuF once it is

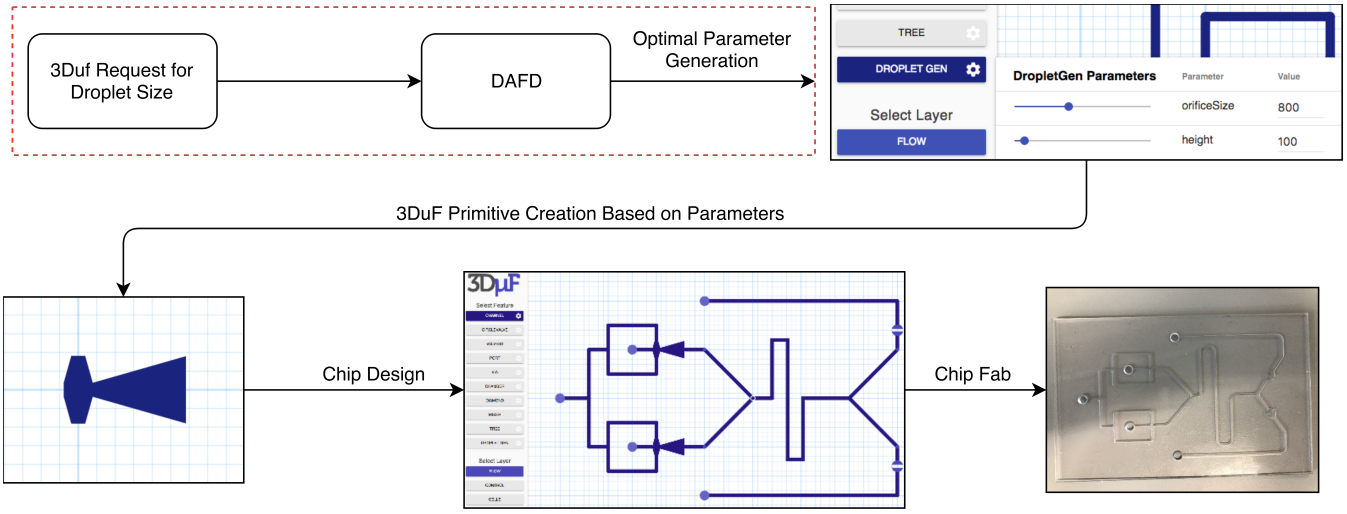


Figure 2: The process by which parameter definition for a droplet generator primitive and chip design take place. With DAFD added (dashed red box), manual specification of parameters will become unnecessary and the correct parameters based on a desired droplet size will automatically generated before the droplet generator primitive is used as part of a larger design.

incorporated by allowing definition of the parameters that define a primitive, as in the case of the droplet generator in Figure 2, using desired performance metrics rather than manual adjusting of the parameters. Figure 1 includes DAFD to show how it fits into the context of design using 3DuF: when creating a parameter, such as the droplet generator, 3DuF will query DAFD for parameters based on the specified metric. DAFD will, through either database lookup or, failing a stored result, model-based prediction, return parameter values for the primitive in question that should produce the desired physical behavior. 3DuF will then populate the parameter fields for the primitive in question with these values, and the designer is free to place it in the chip design as desired. As Figure 2 indicates, through 3DuF's design tools and output format, a chip can then be designed and sent to a fabrication workflow for physical manufacture.

In practice, the interface with DAFD will be fully opt-in: upon selecting a primitive from the list of possible options in 3DuF's sidebar (buttons shown in Figure 2), the user will be able to choose definition by either standard manual adjustment of physical dimensions with sliders or a supported metric, such as droplet size for a droplet generator. In the latter case, the parameter values will be automatically populated as dictated by DAFD, resulting in the addition of the red dashed box in Figure 2 into 3DuF's operation.

4. CONCLUSION AND FUTURE WORK

While the specific metric of droplet size for a droplet generator is the only example showcased here, the principle of this design approach is applicable to performance metrics for a range of the primitives in our lab's MINT library, and 3DuF will be be able to apply this to others, such as the serpentine micro-mixer and droplet merging structures. By hooking into the database and model-based prediction tools of DAFD, this approach will allow 3DuF to enable design of microfluidic chips with relative ease for researchers.

5. REFERENCES

[1] H. Huang. *Fluigi: An end-to-end Software Workflow for Microfluidic Design*. PhD thesis, Boston University, 2015.

[2] J. Lippai, R. Sanka, A. Lashkaripour, R. Silva, and D. Densmore. 3duf - modular design for microfluidic chips. poster presented at EBRC Retreat 2017, Mar. 2017.

[3] R. Sanka, H. Huang, R. Silva, and D. Densmore. Mint - microfluidic netlist. poster presented at IWBDA 2016, Aug. 2016.

[4] R. Silva, A. Heuckroth, C. Huang, A. Rolfe, and D. Densmore. Makerfluidics: Microfluidics for all. poster presented at Synberc: Fall 2015, September 2015.

Mapping Genetic Design Space with Phylosemantics

Bryan A. Bartley[1], Michal Galdzicki[2], Robert Sidney Cox III[3], Herbert M. Sauro[1]
[1]University of Washington, Department of Bioengineering, Seattle, WA 98195
[2]Arzeda Corp, 2715 W Fort St, Seattle, WA 98199
[3]Prospect Bio, 2627 Hanover St, Palo Alto CA 94304

ABSTRACT

Synthetic biology often employs evolutionary and combinatorial approaches to generate large libraries of genetic variants. In a basic example of a variant library, the biological function of a single genetic component is varied by introducing point mutations. Ultimately, however, synthetic biologists are more concerned with generating combinatorial variants at the system level[1]. Such libraries may include different genetic components arranged in permutative configurations[2], especially order and orientation permutations. In more advanced cases, variants may even encode semantically different networks[3], such as different logic gates.

Here we demonstrate use of a phylosemantic tree to systematically map and explore genetic design space. The tree classifies combinatorial promoters into families which exhibit similar patterns of gene expression, revealing design patterns. Phylosemantics is a combination of phylogenetics and semantic alignments. Semantic alignments are not a new idea[4], [5], but to our knowledge, this is the first application of semantic alignments to engineered genetic systems, or, more specifically, *genetic architectures* in the context of synthetic biology. Applications may include rationally guided DNA assembly or comparative analysis of genetic architectures. We believe phylosemantics could be a useful abstraction technique for the biodesign community and might help synthetic biologists understand how a variety of related designs are related to each other. We are seeking industry or academic collaborators who are interested in applying phylosemantic approaches to a case study and who might be willing to share annotated sequence data.

1. RESULTS & DISCUSSION

The Cox combinatorial promoter library[2] is based on an abstract design composed of three operators arranged sequentially in distal, medial, and proximal positions (Fig. 1). The boundary between positions are defined by the -35 and -10 sigma70 RNA polymerase binding sites. Promoter variants were derived by varying operator types at each position (repressor, neutral, or activator) while also testing different sequence variants for a given type. For example repressor operators include variants of LacI or TetR operators, while activator variants include LuxR or AraC.

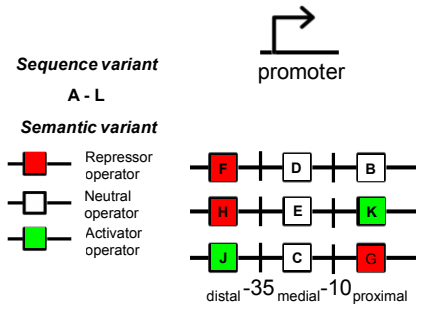


Fig 1. The Cox combinatorial promoter library

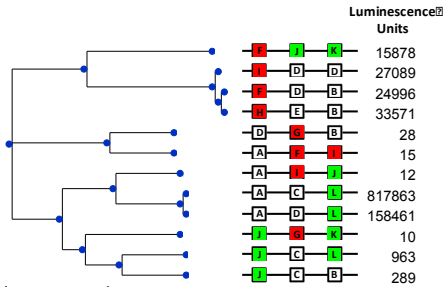


Fig 2. Phylosemantic tree of combinatorial promoters

Of 288 promoters, we selected 12 and mapped them with a phylosemantic tree (Fig 2). The length of branches of the tree correspond to semantic distance between variant designs. Tabulated next to each variant design are the basal levels of gene expression measured for each variant promoter. The advantage of graphing these data with a phylosemantic tree is that some design patterns become more apparent.

The first four variants are all related because they each have a repressor operator distally. Despite the presence of a repressor operator, these promoters exhibit high expression anyway. Repression in this family of promoters appears to fail. Thus, a design rule which may be inferred is that *repressor operators in distal position are ineffective.*

The middle cluster contains similar promoters with a medial repressor operator. Promoters with a medial repressor operator exhibit very low gene expression consistent with repression. This makes sense from a biophysical perspective—a repressor bound in medial position will sterically hinder RNA polymerase binding. A design rule may thus be stated that *repressor operators in medial position exhibit a pronounced "off" effect.*

The lower cluster is largely characterized by activators in the proximal position. This cluster is interesting because it suggests that proximal activator operators exhibit high basal expression when uninduced. The high expression levels coming from

proximal L activation operators suggest leaky expression. Consequently, it is likely that this activator does not have much dynamic range. In order to assess the full range of activation, we would need to graph induced conditions versus the uninduced conditions currently shown. *Activation operators in proximal position may have a poor range of response and be energetically costly due to leaky expression.*

These design rules are not new observations, and were originally described by Cox et al. Thus the design rules we inferred from the phylosemantic tree are aligned with the results of their previous study. The purpose of this exercise was to cross-validate the phylosemantic approach on the Cox library because the general design rules are already known. Some of these design rules can clearly be seen when presented in the context of the phylosemantic tree in Fig. 2. For brevity and clarity, our phylosemantic analysis only includes 12 of 288 promoters. Analysis of the full promoter library might reveal other interesting patterns.

2. METHODS

Phylosemantics, as we define it here, extends conventional phylogenetic approaches with the use of abstract genetic encodings and semantic weights. Phylogenetics is a well-established method for classifying gene variants into evolutionary families based on similarity in primary sequence structure. The typical workflow begins with multiple sequence alignment (MSA) of variant sequences. From the MSA, a pairwise distance matrix is obtained. In phylosemantics, this distance matrix is augmented with semantic weights which may be derived from an ontology, such as the Sequence Ontology[6], or a simple, custom ontology defined by the user. A phylosemantic tree is derived from the semantically-weighted distance matrix.

Phylosemantics requires genetic data that is annotated with terms from which a semantic score may be calculated. For this reason, standardized annotations based on ontologies are especially useful. Annotations may be computationally predicted, or created by biologists using annotation tools like sequence editors. The utility of phylosemantics will always be tied to the richness of annotated sequence data which is available. For this reason, we have leveraged the data standard called Synthetic Biology Open Language[7], [8] (SBOL) which supports rich annotations and easy exchange of genetic data.

For this study we used several open-source software libraries. For semantic representation of combinatorial promoters we used the C++ library LibSBOL 2.1.1[9]. For sequence alignment of abstract sequence encodings we used the C++ library Seqan 2.2.0[10]. For visualization of trees we used the Python phylogenomics module ETE Toolkit[11].

3. FUTURE WORK

So far we have only applied this approach to genetic designs consisting of biological parts arranged in primary sequence. In the future, more sophisticated semantic alignments could be performed, for example on hierarchical genetic structures or the networks they encode. These advanced approaches might become valuable as biological designs become ever more complex and the availability of standardized, well-annotated genetic data improves.

Phylosemantics encompasses a number of related approaches that might apply in different scenarios. For example, different formulae for calculating semantic distance can produce trees that are more useful for one type of analysis versus another. Another choice with interesting implications is whether to construct a rooted versus unrooted tree. Other scenarios in which phylosemantics might be useful include:

- Phylosemantic classification might be used to classify permutations of genes in different orientations.
- Phylosemantics could enable biodesign automation efforts by helping synthetic biologists plan rational assembly strategies given a collection of DNA templates.
- Phylosemantic classification might also be useful for biologists more interested in studying natural systems than building synthetic systems. For example, a comparative study of the architecture of variant proteasome systems might be helped by a more systematic approach[12].

We are seeking industry or academic collaborators who are interested in applying phylosemantic approaches to a case study and who might be willing to share annotated sequence data.

4. ACKNOWLEDGMENTS

This work was supported by NSF#1355909 Synthetic Biology Open Language Resource

5. REFERENCES

[1] C. C. Guet et al., “Combinatorial synthesis of genetic networks,” *Science (80-.).*, vol. 296, no. 5572, pp. 1466–1470, 2002.

[2] R. S. Cox et al., “Programming gene expression with combinatorial promoters,” *Mol. Syst. Biol.*, vol. 3, no. 1, p. 145, 2007.

[3] A. A. K. Nielsen et al., “Genetic circuit design automation,” *Science (80-.).*, vol. 352, no. 6281, p. aac7341, 2016.

[4] M. Schulz et al., “Retrieval, alignment, and clustering of computational models based on semantic annotations,” *Mol. Syst. Biol.*, vol. 7, no. 1, p. 512, 2011.

[5] J. Collado-Vides, “Towards a unified grammatical model of σ70 and σ54 bacterial promoters,” *Biochimie*, vol. 78, no. 5, pp. 351–363, 1996.

[6] K. Eilbeck et al., “The Sequence Ontology: a tool for the unification of genome annotations,” *Genome Biol.*, vol. 6, no. 5, p. R44, 2005.

[7] M. Galdzicki et al., “The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology.,” *Nat. Biotechnol.*, vol. 32, no. 6, pp. 545–50, 2014.

[8] N. Roehner et al., “Sharing structure and function in biological design with SBOL 2.0,” *ACS Synth. Biol.*, vol. 5, no. 6, pp. 498–506, 2016.

[9] “LibSBOL.” [Online]. Available: http://sbolstandard.org/software/libraries/.

[10] A. Döring et al., “SeqAn an efficient, generic C++ library for sequence analysis,” *BMC Bioinformatics*, vol. 9, no. 1, p. 11, 2008.

[11] J. Huerta-Cepas et al., “ETE 3: Reconstruction, analysis, and visualization of phylogenomic data,” *Mol. Biol. Evol.*, vol. 33, no. 6, pp. 1635–1638, 2016.

[12] R. De Mot, “Actinomycete-like proteasomes in a Gram-negative bacterium,” *Trends Microbiol.*, vol. 15, no. 8, pp. 335–338, 2007.

Large-Scale Genetic Design: Moving From Designs to Design Spaces

Nicholas Roehner
Boston University, US
nicholasroehner@gmail.com

Douglas Densmore
Boston University, US
dougd@bu.edu

1. INTRODUCTION

As synthetic biology increases in scope from genetic parts and devices to genomes and libraries [5], it becomes challenging to infer or specify common design patterns using descriptions of individual design variants. Given a collection of GenBank files or a spreadsheet specifying the design of thousands of constructs, it is a non-trivial task to analyze these constructs and determine the rules that govern their shared design. Even when informal verbal or written descriptions of these rules are provided, important details or edge cases can easily be omitted. This issue is compounded by the fact that formal mechanisms for tracking changes to design rules are often not in place, leading to information loss over the course of a project.

We have sought to address these issues by developing a platform (Knox) for manipulating rule-based genetic design spaces. Unlike previous approaches to rule-based genetic design [2, 3], Knox enables rapid sharing and discovery of design rules by providing operators for merging and intersecting design spaces. These operations are possible because Knox encodes design spaces as labeled directed graphs. We briefly describe this existing graph-based formalism for combinatorial genetic design [1] and discuss its application via Knox to biosynthetic gene clusters designed by our collaborators. These gene clusters can access many different small molecule products based on their inclusion/exclusion of enzymes belonging to different enzymatic classes.

2. DESIGN SPACE GRAPHS

Labeled graphs can be used to abstract entire design spaces for comparison and rule sharing. In Knox, each design space graph consists of a set of nodes and a set of directed edges between these nodes. The edges are labeled with sets of one or more DNA parts, which are depicted in Figure 2 and Figure 3 using symbols taken from the SBOL Visual standard [4]. Each path from a start node (empty green circle) to a stop node (empty red circle) then represents a linear concatenation of one DNA part from each edge into a genetic design. Each of these designs is "correct by construction," which means that they adhere to the rules encoded by the structure of the graph.

In this way, design space graphs can compactly store large numbers of genetic design variants. In addition, the design rules encoded by the structure of these graphs can be manipulated via straightforward graph operations. Knox provides a variety of operators for connecting and combining graphs, including those shown in Figure 1 and discussed in the case study of Section 3.

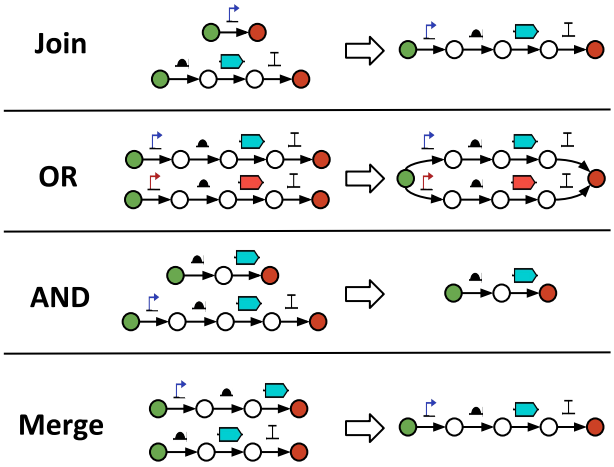


Figure 1: Subset of available operators for connecting and combining design space graphs in Knox.

3. CASE STUDY

We have applied Knox to track and manipulate genetic design spaces for gene clusters designed by our collaborators for combinatorial biosynthesis. This case study focuses on the application of Knox to a design space (Design Space A) that mixes-and-matches 16 enzymes in 6 different enzymatic classes (Figure 2), and another design space (Design Space B) that mixes-and-matches 48 enzymes in 11 different enzymatic classes (Figure 3). When visually appraising the design space graphs shown in Figure 2 and Figure 3, it is immediately apparent that only Design Space B features polycistronic genes, and that there is more specific pairing of ribosome binding sites (RBSs) with coding sequences (CDSs) in this design space. Less apparent is the fact that these design spaces share only 8 parts between them, but this can quickly be determined by applying the AND operator, which intersects the design spaces and produces a significantly smaller space, as reported in Table 1. Further analysis of the intersection space reveals that most of the shared parts are ribozymes, and only three shared parts form a longer motif: a ribozyme followed by an RBS and a CDS.

Table 1 also reports the results of applying the Join, OR, and Merge operators to Design Space A (40 nodes and 49 edges) and Design Space B (95 nodes and 139 edges). Unlike

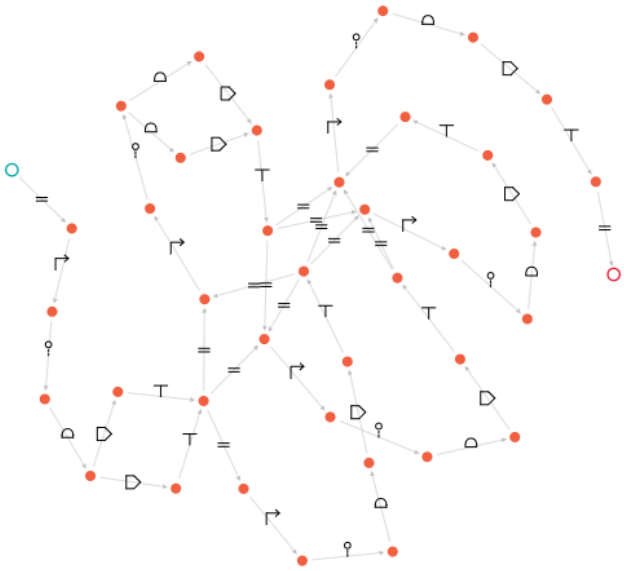


Figure 2: Graph for Design Space A encoding $1,512$ gene cluster designs capable of combinatorial biosynthesis. Each design adheres to rules implicitly encoded by the structure of the graph. These rules dictate part ordering and the inclusion/exclusion of different enzymatic classes, leading to the biosynthesis of different small molecule products.

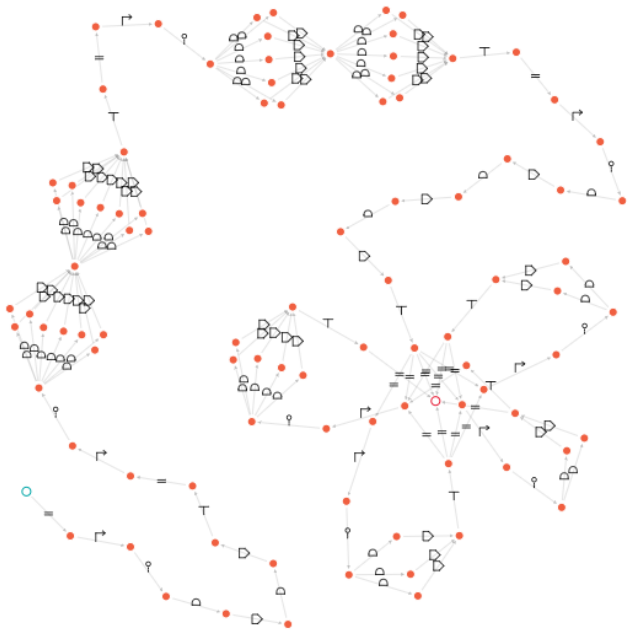


Figure 3: Graph for Design Space B encoding $2.3 * 10^{10}$ gene cluster designs capable of combinatorial biosynthesis. The first half of the graph is mostly linear and specifies the ordering of mandatory enzymatic classes. The second half dictates the inclusion/exclusion of optional enzymatic classes.

Table 1: Applying Operators to Spaces A & B

Operator	Time (s)	Nodes	Edges	Designs
Join	1.50	134	188	$3.5 * 10^{13}$
OR	1.71	133	188	$2.3 * 10^{10}$
AND	1.33	9	8	6
Merge	3.63	963	1361	$> 3.5 * 10^{13}$

the AND operator, which can be used to compare and prune design spaces, these operators are primarily used to expand design spaces, either connecting them in series (Join), connecting them in parallel (OR), or combining them based on their similarity (Merge). While the Merge operator can produce large design spaces that may contain malformed transcriptional units, these nonsense designs can be eliminated through subsequent applications of the AND operator. Importantly, neither of these operations would be possible to perform at scale without the aid of a machine.

4. CONCLUSION

Based on our initial testing of Knox with real-world test cases, we have demonstrated that automating the process of sharing design knowledge between projects in a rule-based manner can greatly improve quality control for large-scale genetic design. Not only does automation reduce the risk of human error when applying design rules, it can also reveal unexpected portions of a design space that might otherwise be overlooked due to human bias. Knox has the capability to be a platform for large-scale, rule-based genetic design.

5. ACKNOWLEDGMENTS

This work is supported by the DARPA Living Foundries award HR0011-15-C-0084. We thank Swapnil Bhatia (Boston University) for his development of the design space graph formalism leveraged by Knox. We also thank Rob Warden-Rothman and Raissa Eluere for sharing data on their biosynthetic gene cluster libraries. Finally, we thank Christopher Voigt (MIT) and D. Benjamin Gordon (MIT-Broad Foundry) for helpful discussions related to the presentation and application of Knox.

6. REFERENCES

[1] S. Bhatia, M. Smanski, D. Densmore, and C. Voigt. Genetic design via combinatorial constraint specification. Technical Report 2017-1, Boston University, May 2017.

[2] L. Bilitchenko et al. Eugene – a domain specific language for specifying and constraining synthetic biological parts, devices, and systems. *PLoS ONE*, 6(4):e18882, 2011.

[3] 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):2760–67, 2007.

[4] J. Y. Quinn et al. SBOL Visual: a graphical language for genetic designs. *PLoS Biol*, 13(12):e1002310, 2015.

[5] M. J. Smanski et al. Functional optimization of gene clusters by combinatorial design and assembly. *Nat. Biotechnol.*, 32:1241–1249, 2014.

Modeling the Structural Determinants of mRNA Stability for Operon Design

Daniel Cetnar
Chemical Engineering
Pennsylvania State University
University Park, PA
dpc5107@psu.edu

Dr. Howard Salis
Chemical and Biological Engineering
Pennsylvania State University
University Park, PA
salis@psu.edu

ABSTRACT

Single and multi-cistronic operons are the central architectural unit of all natural and engineered genetic systems in bacteria, including multi-regulator genetic circuits and multi-enzyme pathways. An operon's sequence ultimately determines the expression levels of its RNAs and proteins, though the sequence-to-function rules that dictate optimal operon design remain to be elucidated or fully tested. Currently, operons are designed by appending several pre-existing promoters, ribosome binding sites, coding sequences, and terminators. As a consequence, many engineered operons are full of overlapping, context-dependent, and undesired genetic elements that confound rational design and will inevitably break the operon's function. Building upon the Operon Calculator's existing fifteen design rules, we present quantitative sequence-function relationships controlling mRNA lifetimes in E. coli. Furthermore, we will show how these systematic experiments yield design rules that enable the Operon Calculator to achieve maximum tunable control over operon design.

CCS CONCEPTS

• **Applied computing → Life and medical sciences**; **Computational biology**;

KEYWORDS

Operon Design, RNAse E, RNAse III, mRNA Stability, Biophysical Modeling

ACM Reference format:
Daniel Cetnar and Dr. Howard Salis. 2017. Modeling the Structural Determinants of mRNA Stability for Operon Design. In *Proceedings of ACM Conference, Pittsburgh, PA, USA, August 2017 (IWBDA),* 2 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

Rapid and cost effective operon construction relies on the ability to avoid multiple design-build-test cycles by accurately modeling the expression and behavior of each gene in the operon. We developed the Operon Calculator as a multi-objective optimization tool to automate implementation of multiple sequence design constraints.

IWBDA, August 2017, Pittsburgh, PA, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.475/123_4

In particular, we have developed a sequence-to-function model of mRNA turnover to predict mRNA half-life in E. coli. Substantial previous work has identified the primary enzymes responsible for mRNA turnover, which are RNA endonucleases E and III. This work has identified RNAse E's primary role in the degradosome machinery and its preference for unstructured RNA. Likewise, RNAse III targets long paired RNA structures [2, 3, 7]. RNA-seq using the entire E. coli genome has opened new avenues to determine the particular sites targeted by the RNAses[1, 5]. Our approach builds on this body of information by using rational design to test particular mRNA structural motifs to quantitatively model RNA half-life. Identifying and avoiding particular structural motifs that cause degradation allows for more precise and reliable operon design.

2 EXPERIMENTAL RESULTS

Our experimental investigation into mRNA half-lives presents the key finding that a strong positive relationship between mRNA stability and translation rate exists. Using the RBS Library Calculator, we engineered ribosome binding sites to created operons that use the same promoter, fluorescent reporter, and terminator, but span three orders of magnitude with regard to translation rate [4].

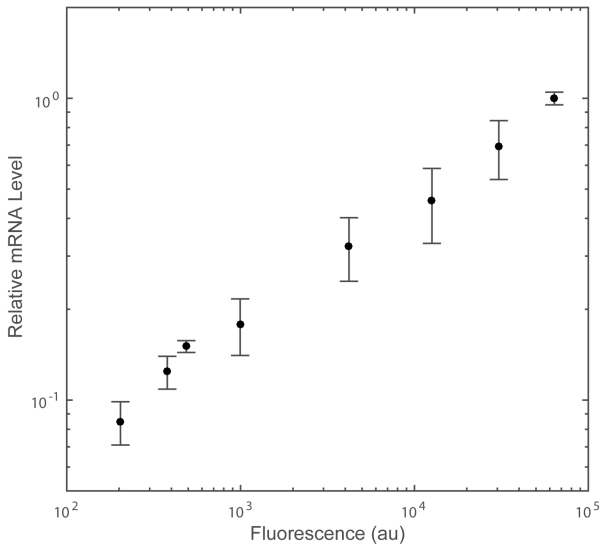


Figure 1: mRNA Stability as a Function of Translation Rate

Using real-time qPCR we found that mRNA levels drop an order of magnitude over this translation space. This result suggests that higher ribosome density protects mRNA transcripts from attack by RNAses as depicted in Figure 2. Bulky 70s ribosomes physically occlude RNAse binding sites, preventing RNAses from binding and cutting the mRNA transcript. This hypothesis will also be tested by investigating variably structured protein coding regions. Additional parameters such as the effect of the location of RNAse sites in the operon, such as the 5' UTR, protein coding region, intergenic region, and 3' UTR are all investigated. Furthermore, the required footprint size of the RNAses is also quantified.

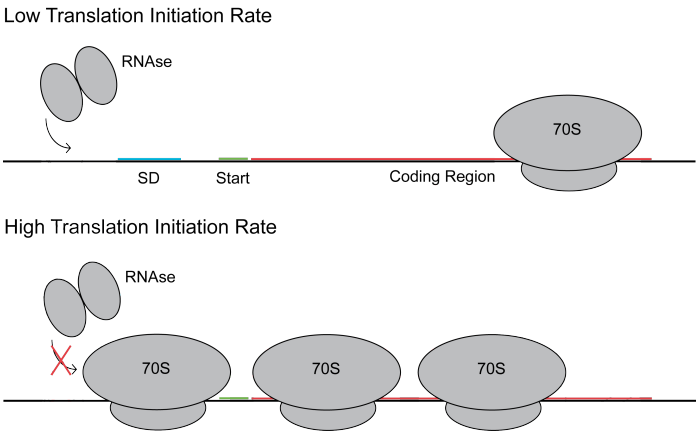


Figure 2: Ribosome Protection of mRNA Transcripts

3 MODELING APPROACH

Using the experimental mRNA lifetime results, a Markov model of mRNA transcript lifetime can be developed. Each state of the system is defined as a particular segment of the mRNA transcript, its binding status to RNAses or ribosomes, and whether or not it is cut. mRNA structures present on each particular segment of the mRNA are calculated using Vienna RNAfold [6]. Based on the significant effect of translation on mRNA accessibility to degradation, existing translational modeling is used to model the advancing ribosomes from one state to the next. [8]. Using the experimental results of the size of RNAse E and RNAse III sites, the probability of attack by RNAse III and E can be calculated and used to determine the overall lifetime of the mRNA transcript.

4 AVAILABILITY

The Operon Calculator is available on the Salis Lab website for both complete operon design and for the prediction of previously designed operons. Figure 3 shows the interface for the Operon Calculator. The top check boxes control the different design criteria for the entire operon. The model of mRNA stability will be incorporated into the 'Improve mRNA Stability' design rule function shown. In the next field, the user can either input a promoter or select a promoter from our database. In the following fields, the protein coding sequences can be entered along with the desired target translation rate. Additionally, the user can either input a terminator or select one from our database. Lastly, the user selects the host organism for the operon. The Operon Calculator is freely available to academic users. The Operon Calculator design mode is available at https://salislab.net/software/OperonCalculator_ForwardDesign and the evaluate mode is available at https://salislab.net/software/OperonCalculator_EvaluateAnnotatedOperon.

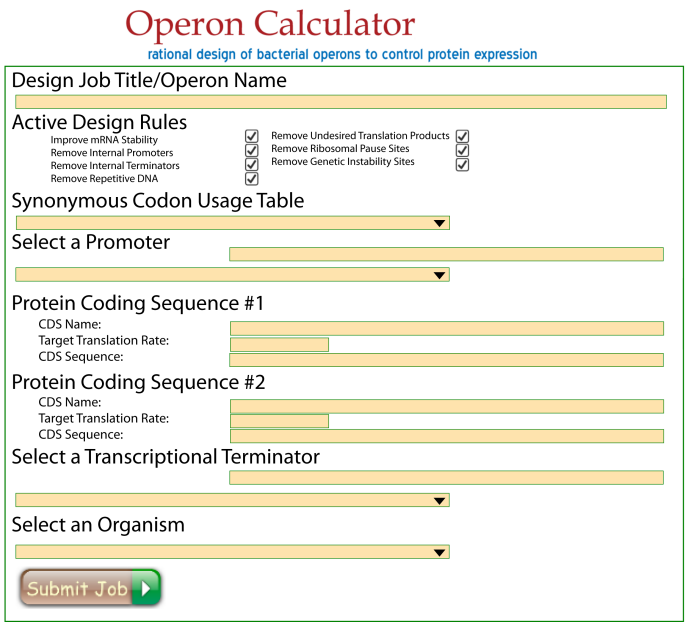


Figure 3: Operon Calculator Interface

5 CONCLUSION

Overall, the Operon Calculator provides a complete tool to design robust operons for predictable gene expression. We have demonstrated the importance of including quantitative mRNA degradation data when modeling the expression of an operon.

REFERENCES

[1] Huiyi Chen, Katsuyuki Shiroguchi, Hao Ge, and Xiaoliang Sunney Xie. 2015. Genome-wide study of mRNA degradation and transcript elongation in Escherichia coli. *Molecular systems biology* 11, 1 (2015), 781.

[2] Justin E Clarke, Louise Kime, David Romero, and Kenneth J McDowall. 2014. Direct entry by RNase E is a major pathway for the degradation and processing of RNA in Escherichia coli. *Nucleic acids research* 42, 18 (2014), 11733–11751.

[3] Donald L Court, Jianhua Gan, Yu-He Liang, Gary X Shaw, Joseph E Tropea, Nina Costantino, David S Waugh, and Xinhua Ji. 2013. RNase III: genetics and function; structure and mechanism. *Annual review of genetics* 47 (2013), 405–431.

[4] Iman Farasat, Manish Kushwaha, Jason Collens, Michael Easterbrook, Matthew Guido, and Howard M Salis. 2014. Efficient search, mapping, and optimization of multi-protein genetic systems in diverse bacteria. *Molecular systems biology* 10, 6 (2014), 731.

[5] Gina C Gordon, Jeffrey C Cameron, and Brian F Pfleger. 2017. RNA Sequencing Identifies New RNase III Cleavage Sites in Escherichia coli and Reveals Increased Regulation of mRNA. *mBio* 8, 2 (2017), e00128–17.

[6] Ronny Lorenz, Stephan H Bernhart, Christian Hoener Zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. 2011. ViennaRNA Package 2.0. *Algorithms for Molecular Biology* 6, 1 (2011), 26.

[7] George A Mackie. 2013. RNase E: at the interface of bacterial RNA processing and decay. *Nature Reviews Microbiology* 11, 1 (2013), 45–57.

[8] Yun-Bo Zhao and J Krishnan. 2014. mRNA translation and protein synthesis: an analysis of different modelling methodologies and a new PBN based approach. *BMC systems biology* 8, 1 (2014), 25.

iBioSim 3: A Tool for Model-Based Genetic Circuit Design

Leandro Watanabe
Dept. of Elect. and Comp. Engineering
University of Utah
l.watanabe@utah.edu

Tramy Nguyen
Dept. of Elect. and Comp. Engineering
University of Utah
tramy.nguyen@utah.edu

Michael Zhang
Dept. of Elect. and Comp. Engineering
University of Utah
michael.zhang@utah.edu

Chris Myers
Dept. of Elect. and Comp. Engineering
University of Utah
myers@ece.utah.edu

Curtis Madsen
Dept. of Elect. and Comp. Engineering
Boston University
ckmadsen@bu.edu

Nicholas Roehner
Dept. of Elect. and Comp. Engineering
Boston University
nroehner@bu.edu

1 INTRODUCTION

iBioSim [5] is a standard-enabled *genetic design automation* (GDA) tool that promotes the model-based design of genetic circuits and the sharing of these designs via community developed standards and data repositories. A high-level illustration of the key features of iBioSim is shown in Figure 1. The iBioSim design workflow leverages models and their analysis throughout to guide the design choices made when constructing a genetic circuits. The remainder of this abstract describes this workflow in further detail.

2 DNA CIRCUIT DESIGN

A genetic circuit design in iBioSim begins by using the SBOLDesigner [21] tool to select genetic parts from the SynBioHub part repository (formally known as the SBOL Stack [7]). This DNA-level design is expressed using version 2 of the *Synthetic Biology Open Language* (SBOL) [2, 15]. SBOLDesigner [21] is an intuitive sequence editor tool that is incorporated into iBioSim as a plugin. The structural layer of genetic designs can be viewed and created hierarchically in SBOLDesigner's canvas. SynBioHub is a repository for synthetic biology designs that allows storing and sharing genetic designs represented in SBOL. This feature facilitates model-based design of genetic circuits by providing the means to construct new designs from existing well-defined parts.

Next, the Virtual Parts model generator obtains *interaction* data, as described in [8, 9], from SynBioHub to add functional information to the SBOL description. For example, it adds the proteins that act as *transcription factors* for the *promoters*, as well as their *coding sequences* in the DNA-level design. These *protein components* are coupled with the *DNA components* constructed by SBOLDesigner along with their interactions into functional *module definitions*. Next, an SBOL to SBML converter [14] can be applied to translate this SBOL into a quantitative model expressed in the *Systems Biology Markup Language* (SBML) [3]. Since SBOL is used to represent qualitative models, the quantitative information required by SBML is inferred [14]. However, this SBML model can then be further refined and model parameters added using iBioSim's model editor. Any changes made can be mapped back to SBOL using the SBML to SBOL converter [10]. In this converter, many SBML elements are lost (e.g. events, rules, function definitions, etc.) because they do not have any purpose in SBOL. Note that these conversions are lossy since SBOL and SBML are used for different purposes. The resulting complete genetic circuit can be uploaded to SynBioHub to share and document the design.

3 ANALYSIS

iBioSim supports simulation of SBML models using a variety of different simulation methods, such as *ordinary differential equations* (ODEs) and *stochastic simulation*. In addition, *Markovian analysis*, such as *stochastic model checking* [6], can be used to reason about the robustness of the design. Lastly, the iBioSim tool allows the visualization of grid-based models of cellular populations [17].

iBioSim is the first software tool that is capable of simulating SBML models that utilize the *hierarchical model composition* (*comp*) [16] and *arrays* packages without flattening out these structures [19, 20]. Another feature of iBioSim's simulation capabilities is the ability to perform *flux balance analysis* (FBA) on SBML models encoded using the *flux balance constraints* (*fbc*) package [11]. FBA is quite useful when kinetic information about the model is unknown. In this case, reaction kinetics are replaced by flux bounds. The goal of FBA is to optimize an objective function (usually cell growth) according to a set of reaction flux constraints. One of the limitations of FBA is that it cannot express kinetic dynamics, since the species concentrations are not updated once the reaction fluxes are obtained. In order to overcome this limitation, *dynamic FBA* (DFBA) can be used. In DFBA, an FBA model is coupled to its counterpart represented with chemical kinetics, where kinetic parameters are obtained as a function of the fluxes. This allows the species in the FBA model to be updated dynamically. DFBA is not inherently supported in SBML, though using the *comp* package along with the *fbc* package addresses this limitation. A consensus has been reached, and a new scheme using these packages has been proposed to encode such models. iBioSim includes a new simulation method that is able to simulate such models.

Since one of the goals of iBioSim is to use standards for the interoperability between tools, the *Simulation Experiment Description Markup Language* (SED-ML) [18] is integrated into iBioSim to describe how a model should be analyzed and how the results are presented to the user. The SED-ML file along with results of the simulation can also be attached to designs stored in SynBioHub.

4 SYNTHESIS

While the workflow described in Figure 1 requires manual selections of parts for a genetic design, iBioSim also supports automated methods for part selection leveraging a process called technology mapping [13]. Rather than deriving a model from a circuit, this process begins by expressing the model in SBML, and a circuit is constructed by selecting parts that implement the desired function.

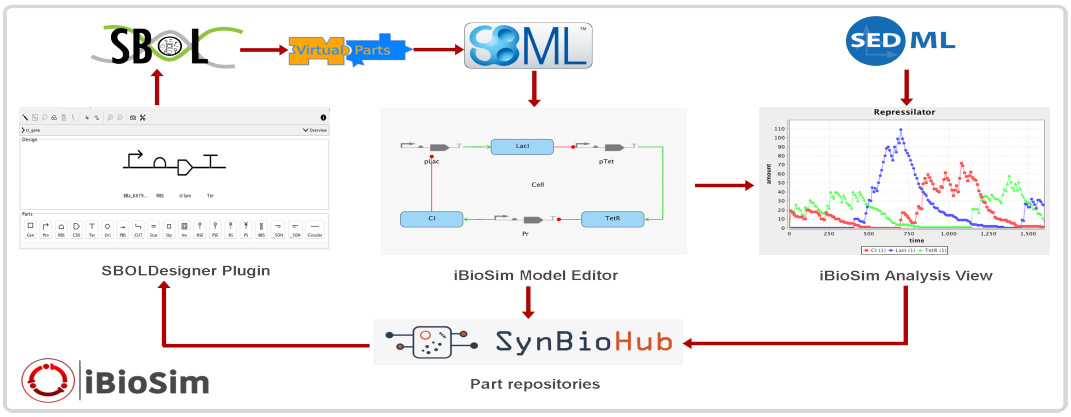


Figure 1: This is a high-level diagram of how the genetic circuit design workflow is supported by iBioSim. First, genetic parts encoded using SBOL are fetched from SynBioHub using the SBOLDesigner plugin to construct the DNA-level design encoding using SBOL. Next, the DNA design is augmented with interaction data using the Virtual Parts model generator, and the functional SBOL is converted into an SBML model. The resulting mathematical model can then be refined and parameters configured using iBioSim's model editor. The SBML model can be analyzed in iBioSim as described by an associated SED-ML document. The data created for SBOL parts, SBML model, and analysis can be shared and documented by uploading these artifacts to SynBioHub.

The key challenge that has to be addressed is that the parts selected must not interfere with each other. Namely, there should be no unintended interactions between the proteins produced by each of the parts of the design.

5 CONCLUSION

iBioSim is a GDA tool for the modeling, analysis, and design of genetic circuits that is being actively developed at the University of Utah. While some older analysis methods [4] and model generation methods [1] are written in C/C++, the majority of iBioSim is written in Java leveraging pure-Java libraries such as JSBML [12] and libSBOLj [22]. iBioSim is an open-source project available publicly at: https://github.com/MyersResearchGroup/iBioSim.

ACKNOWLEDGMENTS

The authors would like to thank Nathan Barker, Scott Glass, Kevin Jones, Hiroyuki Kuwahara, Nam Nguyen, Tyler Patterson, Jason Stevens, and Zach Zundel for their contribution to the development of earlier versions of iBioSim.

The authors of this work are supported by the National Science Foundation under Grant No. CCF-1218095 and DBI-1356041. 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.

REFERENCES

[1] N. A. Barker et al. Learning genetic regulatory network connectivity from time series data. *IEEE/ACM transactions on computational biology and bioinformatics*, 8(1):152–165, 2011.

[2] M. Galdzicki et al. The synthetic biology open language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nature biotechnology*, 32(6):545–550, 2014.

[3] M. Hucka et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, Mar. 2003.

[4] H. Kuwahara et al. Automated abstraction methodology for genetic regulatory networks. In *Transactions on computational systems biology VI*, pages 150–175. Springer, 2006.

[5] C. Madsen et al. Design and test of genetic circuits using iBiosim. *IEEE Design Test of Computers*, 29(3):32–39, June 2012.

[6] C. Madsen et al. Stochastic model checking of genetic circuits. *J. Emerg. Technol. Comput. Syst.*, 11(3):23:1–23:21, Dec. 2014.

[7] C. Madsen et al. The SBOL Stack: A platform for storing, publishing, and sharing synthetic biology designs. *ACS Synthetic Biology*, 5(6):487–497, 2016.

[8] G. Mısırlı et al. Bacillondex: An integrated data resource for systems and synthetic biology. *Journal of Integrative Bioinformatics (JIB)*, 10(2):103–116, 2013.

[9] G. Mısırlı et al. Data integration and mining for synthetic biology design. *ACS synthetic biology*, 5(10):1086–1097, 2016.

[10] T. Nguyen et al. A converter from the Systems Biology Markup Language to the Synthetic Biology Open Language. *ACS Synthetic Biology*, 5(6):479–486, 2016.

[11] B. G. Olivier and F. T. Bergmann. The systems biology markup language (sbml) level 3 package: Flux balance constraints. *Journal of Integrative Bioinformatics (JIB)*, 12(2):660–690, 2015.

[12] N. Rodriguez et al. JSBML 1.0: providing a smorgasbord of options to encode systems biology models. *Bioinformatics*, 31(20):3383, 2015.

[13] N. Roehner et al. Directed acyclic graph-based technology mapping of genetic circuit models. *ACS synthetic biology*, 3(8):543–555, 2014.

[14] N. Roehner et al. Generating Systems Biology Markup Language Models from the Synthetic Biology Open Language. *ACS Synthetic Biology*, 4(8):873–879, 2015.

[15] N. Roehner et al. Sharing structure and function in biological design with SBOL 2.0. *ACS Synthetic Biology*, 5(6):498–506, 2016.

[16] L. P. Smith et al. SBML level 3 package: Hierarchical model composition, version 1 release 3. *Journal of Integrative Bioinformatics (JIB)*, 12(2):603–659, 2015.

[17] J. T. Stevens et al. Dynamic modeling of cellular populations within iBiosim. *ACS Synthetic Biology*, 2(5):223–229, 2013.

[18] D. Waltemath et al. Reproducible computational biology experiments with SED-ML-the Simulation Experiment Description Markup Language. *BMC systems biology*, 5(1):198, 2011.

[19] L. Watanabe et al. Efficient analysis of Systems Biology Markup Language Models of cellular populations using arrays. *ACS synthetic biology*, 5(8):835–841, 2016.

[20] L. H. Watanabe et al. Hierarchical stochastic simulation algorithm for SBML models of genetic circuits. *Frontiers in bioengineering and biotechnology*, 2, 2014.

[21] M. Zhang et al. SBOLDesigner 2: An intuitive tool for structural genetic design. *ACS Synthetic Biology*, 2017.

[22] Z. Zhang et al. libSBOLj 2.0: A Java library to support SBOL 2.0. *IEEE Life Sciences Letters*, 1(4):34–37, Dec 2015.

Standard Enabled Model Generator for Genetic Circuit Design

Göksel Mısırlı
ICOS, School of Comp. Science
Newcastle University
gokselmisirli@gmail.com

Tramy Nguyen
Dept. of Elect. and Comp. Eng.
University of Utah
tramy.nguyen@utah.edu

James McLaughlin
ICOS, School of Comp. Science
Newcastle University
j.a.mclaughlin@newcastle.ac.uk

Chris Myers
Dept. of Elect. and Comp. Eng.
University of Utah
myers@ece.utah.edu

Anil Wipat
ICOS, School of Comp. Science
Newcastle University
anil.wipat@ncl.ac.uk

1 INTRODUCTION

A substantial amount of information is being produced about biological parts that can be used to implement complex designs. However, this information is usually available for human interpretation and often at the DNA sequence level. Computational modeling *in silico* is often exercised manually in order to predict the behavior of designs that can be implemented *in vivo* or *in vitro*. In these models, functional relationships and design constraints between parts in a design are captured in a formal modeling language for simulations. Although this approach may be sufficient for small designs and part libraries, automation of the model generation process is necessary to evaluate larger combinatorial design spaces.

Data standards are particularly important to facilitate design automation, and to pass information between different computational tools when implementing complex workflows. The *Synthetic Biology Open Language* (SBOL) [1, 3, 11] has emerged as an international standard to exchange genetic circuit designs. This standard is useful to specify designs in terms of constituent components. The order and sequences of these components in a design can be captured, and these designs can then be hierarchically used in more complex designs. Critically, SBOL supports capturing molecular interactions between these components. This information is invaluable when creating computational models. Deriving simulatable models, in the form of the *Systems Biology Markup Language* (SBML) [4] documents, from such SBOL designs has already been demonstrated [10]. In this earlier approach, interactions are manually provided.

Building upon these promising efforts, this paper presents a data integration based approach, enabled by data standards, to facilitate the automated creation of computational models from simple definitions of genetic circuits. These definitions may include minimum information that is necessary for DNA synthesis. Computational models are then constructed by extracting knowledge about these DNA components and other interacting entities such as proteins, small molecules, complexes, etc.

2 MODEL GENERATION WORKFLOW

Our model generation workflow is depicted in Figure 1. This workflow consists of three components. SynBioHub (which now incorporates the SBOL Stack [6]), a data repository that stores information about genetic parts and their interactions, iBioSim [5], a software for constructing and modeling genetic circuit designs,

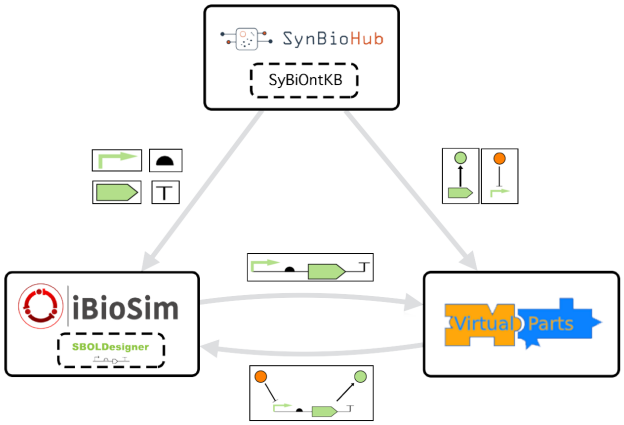


Figure 1: The workflow described in this abstract is summarized in this diagram. The process begins by designing a simple genetic design in iBioSim using the SBOLDesigner plugin. The parts for this design are retrieved from SynBioHub. The generated design is submitted to the VPR, which adds the interactions between the components to the original design. The resulting functional design retrieved in SBOL is translated into an SBML computational model, which can now be simulated using iBioSim.

and the Virtual Parts Repository (VPR) [9], a methodology for storing modular, composable models of parts, together with an interface that allows their composition into larger models. Each of these components is described in more detail in the following sections.

2.1 Data Integration

While a substantial amount of biological information has been produced, this data is often available in different formats and the meaning of data varies between different databases. To make the most of this data in synthetic biology, it is important that these heterogeneous datasets are integrated so that they can be used easily both by humans and software tools. SyBiOntKB [7] is an integrated dataset for synthetic biology applications that has initially been populated with an integrated *Bacillus subtilis* dataset [8]. SyBiOntKB is represented in RDF and the semantics of entities are defined

using the SyBiOnt ontology. SyBiOntKB is now hosted in a publicly shared SynBioHub instance (https://synbiohub.org). SyBiOntKB is then mined for genetic parts which are recorded back in SynBioHub in the form of SBOL objects. The SynBioHub database is backed-by an RDF triplestore and allows uploading, downloading SBOL documents and querying the underlying data using SPARQL.

2.2 Genetic Circuit Construction

In order to construct the DNA-level genetic circuit, our workflow uses the SBOLDesigner [12] plug-in within iBioSim[5]. SBOLDesigner can connect to SynBioHub in order to query for parts by their role, such as promoter, coding sequence, etc., and the user can then select a desired part, and download its sequence and other meta-data. This process can be repeated until a complete, DNA structural-level genetic design is produced.

2.3 Enriching SBOL Designs

The next step of the workflow uses the VPR to examine the DNA-level information provided by SBOLDesigner and enrich it with further details using data found in the SynBioHub repository. This work extends the VPR [9] API in providing functionality which enriches SBOL objects with information about interactions of DNA components with other biological molecules. The resulting enriched SBOL designs are returned from the VPR API now including additional design components for interacting proteins, small molecules, and protein complexes. Interactions such as the translation of proteins, the activation and inhibition of promoters, and complex formation are also incorporated. The following rules are applied for this process:

- A single SBOL `ModuleDefinition` entity is created for a given transcriptional unit design, which may be formed of promoters, ribosome binding sites, coding sequences and terminators. This entity is used to encapsulate molecular interactions between biological components.
- Biological molecules interacting with any of the DNA-based components are added to the `ModuleDefinition`.
- First-level interactions of proteins produced by the transcriptional unit are also included in the `ModuleDefinition`.
- If a biological molecule is not produced using all the entities in the enriched SBOL design, then the corresponding SBOL `FunctionalComponent` is marked as an input. Otherwise, the corresponding `FunctionalComponent` is marked as both input and output. These inputs and outputs are further used when creating hierarchical designs.

In addition to the syntax provided by the SBOL standard, defining the semantics has been an important aspect of the workflow presented here. For example, the VPR uses the *Systems Biology Ontology* (SBO) [2] terms when providing types of interactions, and roles of participants in each interaction, as described here [1]. These terms make the resulting SBOL documents further machine tractable and facilitate deriving models.

2.4 Deriving Dynamic Models

Expressing a model in SBML is necessary to verify the behavior of a design since SBOL, by design, does not include all of the information needed for dynamic simulation. To derive SBML from SBOL, a conversion tool is applied [10]. This tool uses ontology terms to help translate components from one standard to another. SBOL `ComponentDefinitions` are mapped to SBML `Species` and SBOL `Interactions` are mapped to SBML `Reactions`. The `ModuleDefinition` that the `Interaction`(s) are nested in are mapped to SBML `ModelDefinitions`. The mapped reactions use default kinetic laws and reaction rates, specified in [10]. Once the SBOL data has been converted to SBML, the default kinetic laws and reaction rates can be altered through the model editor provided in iBioSim. In the future, these rates parameters may be stored as annotations within the SBOL returned from the VPR. The SBML model constructed in this way can be simulated using a variety of methods, including ordinary differential equations (ODEs) or stochastic simulation methods.

3 CONCLUSIONS

This standard enabled design workflow is important to abstract the details of complexity when dealing with computational models. As demonstrated, simplifying the design process using a tool such as SBOLDesigner has significant benefits. Designs can relatively easily be created by users who can design circuits using DNA parts and still benefit from computational simulations. Moreover, this approach facilitates automation and exploring large designs spaces of biological systems. The standards SBOL and SBML are critical since they serve as domain specific languages that seamlessly connect all the tools within this workflow.

4 ACKNOWLEDGMENTS

The authors of this work are supported by The Engineering and Physical Sciences Research Council grant EP/J02175X/1 (G.M., J.M., A.W.) and the National Science Foundation under Grant No. CCF-1218095 and DBI-1356041 (T.N. and C.M.). 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 funding agencies.

REFERENCES

[1] J. Beal et al. Synthetic biology open language (SBOL) version 2.1.0. *Journal of Integrative Bioinformatics*, 13(3):30–132, 2016.
[2] M. Courtot et al. Controlled vocabularies and semantics in systems biology. *Mol. Syst. Biol.*, 7:543, 2011.
[3] M. Galdzicki et al. The synthetic biology open language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nature biotechnology*, 32(6):545–550, 2014.
[4] M. Hucka et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, Mar. 2003.
[5] C. Madsen et al. Design and test of genetic circuits using iBioSim. *IEEE Design and Test*, pages 32–39, 2012.
[6] C. Madsen et al. The SBOL Stack: A platform for storing, publishing, and sharing synthetic biology designs. *ACS Synthetic Biology*, 5(6):487–497, 2016.
[7] G. Misirli et al. Data integration and mining for synthetic biology design. *ACS Synthetic Biolology*, 5(10):1086–1097.
[8] G. Misirli et al. Bacillondex: An integrated data resource for systems and synthetic biology. *Journal of Integrative Bioinformatics*, 10(2):224, 2013.
[9] G. Misirli et al. Composable modular models for synthetic biology. *ACM Journal on Emerging Technologies in Computing Systems*, 11(3):22:1–22:19, Dec. 2014.
[10] N. Roehner et al. Generating systems biology markup language models from the synthetic biology open language. *ACS Synthetic Biology*, 4(8):873–879, 2015.
[11] N. Roehner et al. Sharing structure and function in biological design with SBOL 2.0. *ACS Synthetic Biology*, 5(6):498–506, 2016.
[12] M. Zhang, J. A. McLaughlin, A. Wipat, and C. J. Myers. SBOLDesigner 2: An intuitive tool for structural genetic design. *ACS Synthetic Biology*, 2017.

Test-SFM: An automated model test system for systematic development of sequence-to-function models

[Extended Abstract]

Alexander C. Reis
Pennsylvania State University
Department of Chemical Engineering
alex.reis@psu.edu

Howard M. Salis*
Pennsylvania State University
Departments of Chemical Engineering and Biological Engineering
salis@psu.edu

ABSTRACT

Gene expression models greatly accelerate the engineering of genetic systems by predicting sequence-function relationships and reducing trial-and-error experimentation. While improved models are needed to engineer more complex systems, it has been a challenge to identify and characterize new mechanisms controlling expression, motivating the development of methods to systematically test model accuracies and identify sources of error. To address this challenge, we developed an automated model test system that uses a growing database of 16789 characterized genetic systems to measure model accuracies, accept or reject mechanistic hypotheses, and identify areas for model improvement. Using the model test system, we compared the accuracies of six different models predicting bacterial translation initiation rates, identified new sequence determinants controlling gene expression, and rejected several hypothetical, long-suggested mechanisms controlling translation initiation. Automated model test systems will dramatically accelerate the development of improved gene expression models, and thereby transition synthetic biology into a mature engineering discipline.

Categories and Subject Descriptors

I.2.2 [**Computing Methodologies**]: Automatic Programming—*Automatic analysis of algorithms*; H1.1 [**Information Systems**]: Systems and Information Theory—*Information theory*; G.3 [**Mathematics of Computing**]: Discrete Mathematics—*Markov processes, Probabilistic algorithms (including Monte Carlo)*; J.3 [**Computer Applications**]: Life and Medical Sciences—*Biology and genetics*

Keywords

Automated model testing, sequence-expression database, biophysical models, information theory, hypothesis testing

*To whom correspondance should be addressed.

IWBDA '17 August 8-11 2017, Pittsburg, Pennsylvania

1. INTRODUCTION

It has been a grand challenge to transition synthetic biology into a mature engineering discipline, where genetic systems are reliably designed, built, and tested to reprogram cellular behavior with desired outcomes. Quantitative models play a central role in synthetic biology's design-build-test cycle by predicting the function of a candidate genetic system, before it's constructed, and therefore reduce trial-and-error experimentation. For example, gene expression models can be combined with system-level models of genetic circuits and metabolic pathways to predict how changes in system architecture, component expression levels, and host genome control organismal decision-making and the biosynthesis of desired chemicals. Overall, more accurate sequence-function models are becoming essential to correctly build genetic systems with many parts.

In mature engineering disciplines, test systems are routinely used to verify that models and software systems generate predictions and outcomes within specified performance requirements [1]. Model test systems are run whenever an existing model is modified or when new models are proposed to ensure consistent improvements in accuracy across the widest possible range of inputs. Model test systems also accelerate the discovery of new interactions by identifying the factors that contribute to model error. Within the life sciences, the CASP [4], DREAM [5], and IMPROVER [3] competitions have served a somewhat analogous purpose, where researchers are challenged to apply computational modeling to solve complex problems, for example, predicting protein structure from sequence, identifying disease genetic traits, and reverse-engineering gene regulatory networks. A key theme of these test systems is that truly novel mechanisms are far more discoverable once state-of-the-art models are challenged to predict the outcome of a large and diverse experimental data-set.

2. WORKFLOW

Here, we present `test-sfm`, the first model test system for gene expression models, capable of evaluating quantitative model predictions on a compiled database of 16789 characterized genetic systems with highly diverse DNA sequences and measured functions (Figure 1). The model test system facilitates uploading and managing experimental data using `pandas` dataframes. It includes a smart model-wrapping interface that pulls required values to pass as arguments to

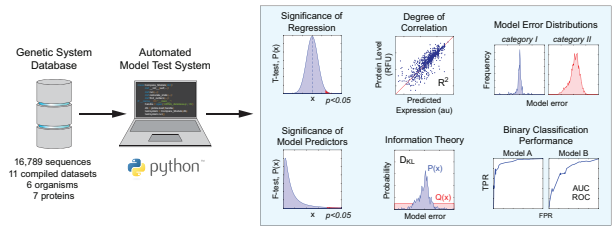


Figure 1: The inputs and outputs of the model test system. The model test system systematically evaluates the predictions of proposed models to accept or reject specific hypotheses and identify sources of model error.

the `Python` wrapped models. Output provided by `test-sfm` includes correlation statistics, information theory metrics, model error statistics, and ROC-curve analysis in the case that models are binary classifiers.

2.1 Automated Model Testing

Using the automated model test system, we evaluated the accuracies and sources of error for six different models of bacterial translation initiation rate (RBS Calculator v1.0 [6], v1.1, v2.0, UTR Designer, RBS Designer, EMOPEC) on the genetic system database. 856 genetic systems were characterized individually, using flow cytometry or spectrophotometry to measure reporter expression levels (856IC), and the remaining 15933 sequences were characterized using a new technique, called Flow-seq, that uses flow-assisted cell sorting (FACS), bar-coding, and next-generation sequencing (RNA-seq/DNA-seq) to characterize transcription and translation rates (15933FS)[2]. The model test system revealed clear differences in model accuracies, and provides a definitive comparison of these models these unified datasets (Table 1).

Table 1: Model Comparisons on the 856IC subset

Model	R^2	D_{KL}	MC (bits)	AUROC
RBS Calculator v1	0.51	0.22	1961.5	0.76
RBS Calculator v2	0.73	0.33	2744.2	0.94
UTR Designer	0.47	0.23	2035.0	0.75
RBS Designer	0.28	0.14	1368.0	0.59
EMOPEC	0.29	0.19	1315.3	0.76

2.2 Model Capacity (Information Theory)

We found several examples where specific models evaluated on one sub-group of data could have substantially different apparent accuracies than when evaluated on the entire data-set. By treating the model test system as a communication process, governed by the principles of information theory, we derived a new accuracy metric that quantifies the amount of information encoded within a model, the model capacity (MC), considering the diversity of sequences (the Shanon sequence entropy, H_{seq}), the number of distinct experimentally-observable outcomes (N_{bins}, and the model's error distribution (H_{model})(Equation 1). This newly proposed MC metric provides an objective way to assess both the model and experimental data-set together, facilitating correct cross-dataset comparisons.

$$MC = H_{seq}(N_{bins} - 1)(1 - \frac{H_{model}}{H_{random}}) \quad (1)$$

2.3 Hypothesis Testing

We adapted the model test system to test existing mechanistic hypotheses, and to search for predicted physical properties of a genetic system (e.g. RNA structure, sequence motifs, folding dynamics) associated with increased model error. Hypotheses can be implemented as extensions to existing models and can be rapidly prototyped using `test-sfm`. Types of hypothesis that can be explored include model-implementation hypotheses, species-specific hypotheses, mechanistic hypotheses, and more.

3. DISCUSSION

The genetic systems in the database consist of non-regulated genes, which can be used to test other sequence-to-function models of gene expression, such as models of translation elongation, translational coupling, and mRNA degradation. However, as the database grows and the complexity of the genetic systems increases, the model test system can be used to test other relevant models that predict the function of other sequence-encoded parts or systems. For example, models that predict the function of individual parts, such as the strengths of terminators, switching of riboswitches, or cleavage rates of ribozymes could be tested on corresponding data. System level models, such as those that predict genetic circuit function, could be tested with the same approach on databases of characterized circuits. As these sequence-to-function models continue to improve, so does our capacity to build functional and robust genetic systems.

The model test system is implemented in Python and is available at https://github.com/reisalex/test-sfm.

4. REFERENCES

[1] I. Jacobson, G. Booch, and J. Rumbaugh. *The unified software development process*. Addison-wesley Reading, 1999.

[2] S. Kosuri, D. B. Goodman, G. Cambray, V. K. Mutalik, Y. Gao, A. P. Arkin, D. Endy, and G. M. Church. Composability of regulatory sequences controlling transcription and translation in escherichia coli. *PNAS*, 2013.

[3] P. Meyer, J. Hoeng, J. J. Rice, R. Norel, J. Sprengel, K. Stolle, T. Bonk, S. Corthesy, A. Royyuru, M. C. Peitsch, and G. Stolovitzky. Industrial methodology for process verification in research (improver): toward systems biology verification. *Bioinformatics28*, 2012.

[4] J. Moult. A decade of casp: progress, bottlenecks and prognosis in protein structure prediction. *Current Opinion in Structural Biology*, 2005.

[5] R. J. Prill, D. Marbach, J. Saez-Rodriguez, P. K. Sorger, L. G. Alexopoulos, X. Xue, N. D. Clarke, G. Altan-Bonnet, and G. Stolovitzky. Towards a rigorous assessment of systems biology models: The dream3 challenges. *PLoS ONE*, 2010.

[6] H. M. Salis, E. A. Mirsky, and C. A. Voigt. Automated design of synthetic ribosome binding sites to control protein expression. *Nature Biotechnology*, 2009.

A Test Suite for Compliance Testing of Software Support for the Synthetic Biology Open Language

Meher Samineni
School of Computing
University of Utah
mehersam251@gmail.com

Chris Myers
Dept. of Elect. and Comp. Engineering
University of Utah
myers@ece.utah.edu

KEYWORDS

SBOL, data standards, validation, compliance testing

1 INTRODUCTION

Data standards are integral for interoperability between software applications, since they provide guidelines for how data can be meaningfully exchanged and in a uniform manner. Standards provide a bridge for applications to share and translate data; however, they do not guarantee that applications are compatible to perform a data exchange or that any translated data is legal and valid. This paper proposes a methodology that validates applications and their compliance to the data standard. Although this paper describes the application of this methodology to a particular standard, the *Synthetic Biology Open Language* (SBOL) [2, 4, 5, 8], it should be applicable compliance testing for other data standards as well.

SBOL is an emerging standard that provides a format for describing the structural and functional information about a genetic design. The structural description includes the chemical makeup, i.e. sequence data, whereas the functional expresses the behavior and interactions of a design. The SBOL data standard is paired with the SBOL Visual standard [4] which provides a set of symbols that are used to visually describe genetic designs.

Recently, we conducted a survey of synthetic biology software developers to determine the level to which their applications support SBOL. A list of software that support SBOL is shown in Table 1 (more details in [6]). As the information from the survey is self-reported by the application's developer, a methodology is required to actually test software compliance. There are three different elements for testing: support of SBOL Visual, an application's ability to import SBOL data, and an applications ability to export SBOL data. SBOL Visual must be manually inspected to ensure the correct usage of symbols. Additionally, to verify an application can export SBOL data, the *SBOL Validator* tool can check the validity of the SBOL files produced [8]. Verifying that SBOL data is correctly imported requires a round-trip test. A round-trip test consists of importing SBOL data into an application and then exporting the imported data. A comparison is performed of the imported and exported data to ensure that no semantically important data has been transformed or lost. If the comparison produces no semantic differences, then the application can correctly import SBOL data. While this test strategy could be excruciatingly tedious if the comparison is performed manually, the ideal scenario is to instrument the software through an interface that enables the tool to import and export SBOL data programmatically, them perform the comparison. The remainder of this paper describes an SBOL test suite, and a methodology for compliance testing using this test suite.

Table 1: A partial list of software supporting SBOL. An up-to-date list is maintained on `http://sbolstandard.org`. The function column indicates if the tool is a (R)epository, (S)equence design tool, (G)enetic circuit design tool, (M)odeling and simulation tool, or a (V)isualization tool. The SBOL column indicates if it supports SBOL (V), as well as import and/or export of SBOL (1) or (2).

	Function					SBOL				
							Import		Export	
Name	R	S	V	G	M	V	1	2	1	2
BOOST		•					•	•	•	•
Cello				•	•	•				•
DeviceEditor		•	•	•	•	•		•		
DNAPlotLib			•			•	•	•	•	•
Eugene		•		•		•	•		•	
Finch	•	•	•	•		•				•
GenoCAD	•	•		•		•		•		•
GeneGenie		•							•	
Graphviz			•			•				
ICE	•	•	•			•		•		•
iBioSim			•	•	•	•	•	•	•	•
j5		•			•	•	•		•	
MoSeC		•		•					•	
Pigeon			•			•				
Pinecone	•	•				•		•		•
Pool Designer		•					•	•		
Proto BioCompiler			•	•	•	•			•	
SBOLDesigner	•	•	•			•	•	•	•	•
SBOLme				•						•
ShortBol		•		•						•
SynBioHub	•		•			•	•	•		•
Tellurium		•		•	•			•	•	
TeselaGen		•	•			•		•		
TinkerCell			•	•	•	•	•		•	
VisBOL			•			•		•		
VirtualParts	•			•	•	•	•		•	•

2 SBOL COMPLIANCE TEST SUITE

To test a software tool's compliance to the SBOL data standard, a series of existing SBOL files are utilized including published SBOL 1 files that have been converted to SBOL 2 [1, 2], simple examples from the SBOL specification [3], and other miscellaneous files created to test the SBOL Java library [7]. This section briefly describes an analysis of this test suite (more details in [6]). In particular, it organizes this test suite to determine its coverage of the SBOL data model. The algorithm for analyzing the test suite begins by reading in each example file into an **SBOLDocument** `libSBOLj`. The data types within each example are identified and a count is associated for each data type. The purpose of this is to understand the extent of the SBOL data model utilized within each example. The algorithm then organizes the examples within groups of clusters based on the same type of data contained. The clusters contain a set of SBOL data types common to a set of examples. Relationships are then

created between the clusters by choosing a cluster and marking it as a parent. Then, this cluster is compared to the remaining clusters and a direct relation exists if the parent cluster contains a superset of the data types contained by the compared cluster, and no other cluster exists that also has a superset of the data types and a subset of those in the parent cluster. The resulting relationship forms a partially order set (POSET) representation the test suite clusters.

A graphical representation of the resulting POSET for the example test suite is shown in Figure 1. The diamond source nodes within the graph represent a superset of the data types common to a group of examples. Each source node denotes a set of examples with unique superset of the data types. In other words, no other node exists that includes the same data types and more. The remaining portion of the graph consists of paths made up of nodes that follow parent-child relationships. Pathways exist as a child node stems from one or more parent nodes and each child node consists of examples that contain a subset of the SBOL data types contained by its immediate parent. These pathways are significant because they provide a test strategy to narrow down which data types are being correctly supported. The yellow colored nodes represent clusters that include only structural data types (**Sequences**, **ComponentDefinitions**, etc.) while the green colored nodes represent clusters that also include functional data types (**Models**, **ModuleDefinitions**, etc.). For example, source Node *6* contains one example with fifteen data types represented in the example and there exists no example representing these data types and more. Node *11* stems from *6* with three examples representing nine types and are also a subset of the types found in the previous node. Node *25* which contains one example with two data types that are also within node *11* and so on.

In the SBOL data model, there are nineteen classes excluding abstract classes. Given this, the maximum number of data types that exists in a set of examples is fifteen types as represented by Node *6*. Furthermore, there does not exist any example with every data type represented. While there does not exist an all-inclusive example, every single data type is at least represented once within an example. One last key insight is the imbalance in the types of data existing within the examples. Sixty-nine percent of the examples represent only structural data classes, while only thirty-one percent of the examples include functional data type.

3 DISCUSSION

In creating a methodology for analyzing SBOL software applications and their support of the SBOL Standard, an algorithm is created to analyze an SBOL test suite. The inspection of the examples results in a graph that can be used to test SBOL applications and verify the self-reported data from the survey. In particular, one test case can be selected from each source node and used to test the application, if an example fails, then one test case would be drawn from all child clusters. This process would repeat until the sink node is reached or no further failures are discovered. By analyzing the point at which examples succeed, we can accurately determine the data classes that the software supports. In the future, we plan to expand the example test suite to fill some gaps in the testing landscape, and apply these examples to test SBOL software compliance. In extension, we plan to develop an SBOL test-runner that automates testing of import and export of SBOL data through applications using the testing strategies previously discussed.

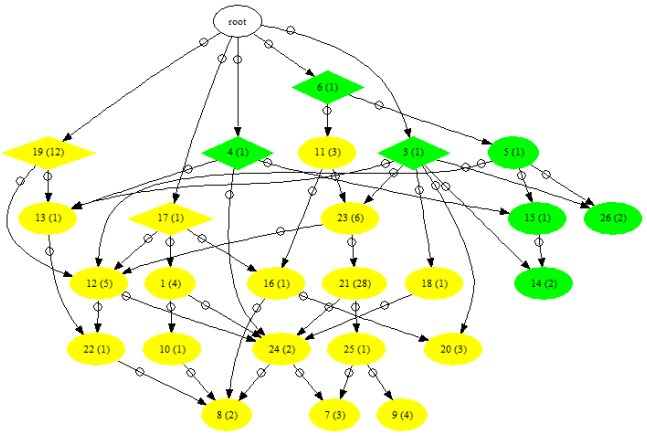


Figure 1: A graphical representation of SBOL test suite and their relations based on the data types supported. The nodes are clusters of examples with the same SBOL data types that are numbered arbitrarily for easy referencing. The numbers within the parentheses are the number of examples in the cluster. Diamond nodes are source nodes with unique supersets of data types. Nodes colored yellow indicate clusters that only include structural SBOL data types. Green nodes represent clusters that also include functional SBOL data types. All of the examples analyzed are available at `https://github.com/SynBioDex/SBOLTestSuite/tree/master/valid/SBOL2.0`

ACKNOWLEDGMENTS

The authors would like to thank Zhen Zhang, Tramy Nguyen, and Zach Zundel for their help in developing the examples used in the test suite. The authors of this work are supported by the National Science Foundation under Grant No. CCF-1218095 and DBI-1356041. 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.

REFERENCES

[1] N. Adames et al. Genolib: a database of biological parts derived from a library. *IEEE Life Sciences Letters*, 1(4):34–37, 2016.

[2] B. Bartley et al. The synthetic biology open language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nature Biotechnoloy*, (32):545–550, 2014.

[3] J. Beal et al. Synthetic biology open language (SBOL) version 2.1.0. *J. Integrative Bioinformatics*, 13(3), 2016.

[4] J. Quinn et al. SBOL visual: A graphical language for genetic designs. *PLoS Biol*, 13(12):e1002310, 12 2015.

[5] N. Roehner et al. Sharing structure and function in biological design with SBOL 2.0. *ACS Synthetic Biology*, 5(6):498–506, 2016.

[6] M. Samineni et al. Software compliance testing for the synthetic biology open language, 2017. Bachelor's Thesis.

[7] Z. Zhang et al. libSBOLj 2.0: a java library to support SBOL 2.0. *IEEE Life Sciences Letters*, 1(4):34–37, 2016.

[8] Z. Zundel et al. A validator and converter for the synthetic biology open language. *ACS Synthetic Biology*, 2016.

Toward Quantitative Comparison of Fluorescent Protein Expression Levels via Fluorescent Beads

Jacob Beal
Raytheon BBN Technologies
Cambridge, MA, USA
jakebeal@ieee.org

Nicholas DeLateur, Brian Teague, Ron Weiss
MIT, Cambridge, MA, USA
{delateur, teague, rweiss}@mit.edu

John Sexton, Sebastián Castillo-Hair, Jeffrey J. Tabor
Rice Univ., Houston, TX, USA
{john.t.sexton,smc9, jeff.tabor}@rice.edu

1. MOTIVATION

Establishing an effective engineering discipline always requires standardized and comparable units of measurement. Such measurements serve as a means of communication between the people and machines interacting with a project, ensure compatibility between components, and allow prediction of the results of design decisions. Regulating gene expression is foundational for organism engineering, and flow cytometry is an excellent means of quantifying large numbers of single cell gene expression measurements. At present, however, flow cytometry data is still often acquired in arbitrary or relative units, without standardizing the measurement by comparison to an independent reference material (i.e., one enabling precise calibration of measurements). Some have proposed standardizing to a biological cultured reference material (e.g., [3]), but fluorescence from such materials varies strongly, unpredictably, and often not proportional to the samples it is intended to be a reference for, thus resulting in a large degree of uncertainty in measurement.

In contrast, stable reference materials have been developed, in the form of beads with a defined fluorescence quantified in terms of molecules of equivalent reference fluorophores (ERF; alternately MEF or ME[fluorophore]) [5]. These reference materials have been primarily employed in medical applications of flow cytometry, which typically use a small number of standard dyes rather than a wide range of fluorescent protein variants, and where the goals of measurement are typically focused on the "digital" goal of classifying cells into distinct populations, rather than the more "analog" goal of precisely quantifying levels of gene expression.

Fluorescent beads have already been used as a reference material for engineering gene expression in a number of studies, including making high-precision circuit predictions (e.g., [2]), engineering novel biological sensors (e.g., [4]), and debugging circuit design problems (e.g., [1]). We now aim to validate these methods through interlaboratory studies and to develop supporting methods and recommended practices that will simplify widespread adoption of well-defined units in flow cytometry, thus accelerating scientific development and simplifying the engineering of biological organisms.

2. USAGE SCENARIOS

We have identified four key usage scenarios for bead-based comparison of fluorescence measurements, along with key target applications motivating their development. In these scenarios, an "ERF-quantified laser/filter combination" means that a set of beads has been assigned ERF values for their intensity when excited with a particular laser frequency and observed through a particular optical filter (e.g., via the process in [5]). If a significantly different laser or filter are used, then the quantification does not apply. The four scenarios (also illustrated in Figure 1) are, in increasing levels of complexity:

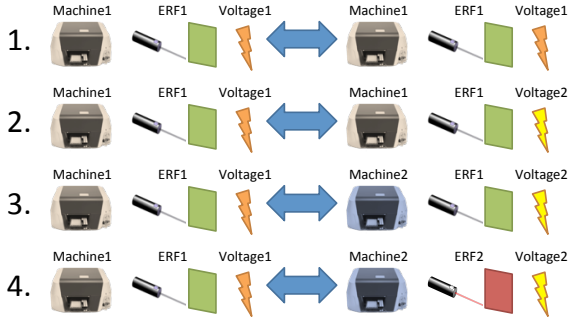


Figure 1: Scenarios for use of ERF-calibrated beads in comparing flow cytometry data.

1. Comparison of samples with the same ERF-quantified laser/filter combination, on the same machine, with the same voltage settings. *Target application: fusion of data sets.*
2. Comparison of samples with the same ERF-quantified laser/filter combination, on the same machine, but different voltage settings. *Target applications: fusion of data sets, extension of data range.*
3. Comparison of samples with the same ERF-quantified laser/filter combination, but different machines (making voltage comparison moot). *Target applications: fusion of data sets, validation of material or method transfer.*
4. Comparison of samples with different ERF-quantified laser/filter combinations (making machine and voltage comparison moot). *Target applications: fusion of data sets, validation of material or method transfer, comparison of multiple signals.*

Note that in no case are the target applications focused on comparison of a cell sample to ERF-quantified beads, per se. Rather, the goals are focused on comparison of cell samples, as enabled by comparing each sample to a set of beads.

IWBDA 2017 Pittsburgh, USA
Funding for this work comes from the NSF Expeditions in Computing Program Award #1522074 as part of the Living Computing Project. We also gratefully acknowledge support from David Ross and Ariel Hecht of NIST in the form of data and discussions.

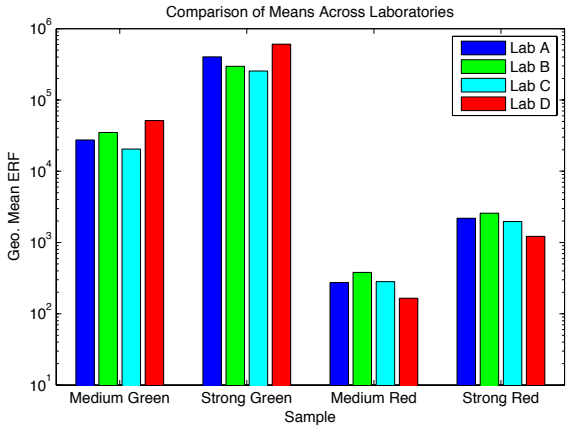


Figure 2: Pilot study measurements show a 1.43-fold geometric standard deviation of mean ERF measurements across laboratories (laboratories indicated by color; red is the laboratory with unmatched channels).

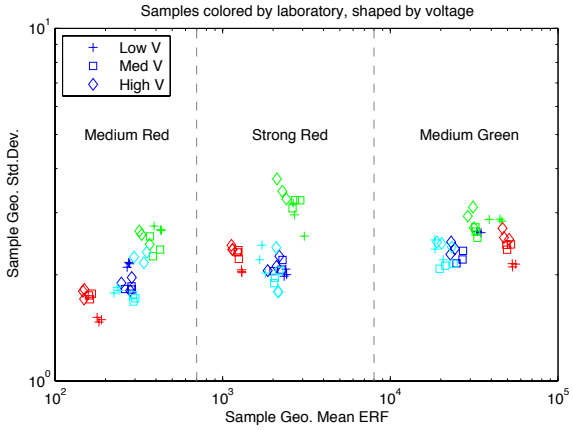


Figure 3: Geometric mean and standard deviation for individual samples are clustered tightly for each laboratory and for each voltage within a laboratory.

Thus, validity of measurements depends primarily on the relationship between cell samples and beads remaining stable across space and time, rather than the actual relationship between cellular fluorescent protein and ERF, which is much more difficult to assure.

3. PILOT STUDY RESULTS

As an initial test of using ERF-quantified beads to establish common units for Scenarios 1-3, we conducted a pilot interlaboratory study of bead-calibrated measurements of *E. coli* expressing GFP and mCherry across four flow cytometers, each in a different laboratory. Three of the flow cytometers had closely matching filters for GFP and mCherry, while the fourth had significantly different filters for both. Five sets of samples of *E. coli* were prepared at one of the four laboratories: empty vector, medium GFP expression, strong GFP expression, medium mCherry expression, and strong mCherry expression. Each sample was then split into aliquots and shipped frozen, to be measured at each laboratory in three independently prepared replicates of each sample at three channel voltages, chosen to spread measurements across the range of each instrument.

Figure 2 and Figure 3 present statistical results from these samples computed by gating for cell events using a gate based on the empty vector sample, calibrating to ERF units using the provided ERF values for SpheroTech RCP-30-5A beads of the appropriate batch (MEFL for green, MEPTR for red), then computing geometric mean and standard deviation (geometric statistics are used throughout due to the fact that the distribution of each sample is roughly log-normal, as is often observed with flow cytometry).[1] Figure 2 compares the geometric mean of ERF values for each laboratory. Note that the geometric means are all fairly close to one another: their standard deviation is 1.43-fold across all laboratories and 1.23-fold for the three laboratories with matched channels. Examining the geometric mean and standard deviation of the events in each individual sample (Figure 3), we find that laboratory-to-laboratory variation amongst laboratories with matched channels is on the same order as replicate-to-replicate variation (standard deviation 1.05-fold) and voltage-to-voltage variation (standard deviation 1.11-fold).

[1]The strong GFP samples, however, had a very low density of events, and so for those samples we report the mode of the upper distribution component instead.

4. CONTRIBUTIONS AND FUTURE WORK

As expected, these preliminary results indicate that fluorescent beads can be used for precise quantitative comparison of fluorescent protein expression, even despite some degree of variation in instrument configuration. We are now in the process of scaling up to a larger and more comprehensive interlaboratory study, with which we hope to definitively establish the efficacy of commercially available ERF-quantified calibration beads for quantification of fluorescent protein expression. Further goals include validating methods for comparison of the expression levels of different fluorescent proteins, improving analytical software to make these methods readily accessible, and quantifying the sources of variation.

5. REFERENCES

[1] S. B. Carr, J. Beal, and D. M. Densmore. Reducing dna context dependence in bacterial promoters. *PLOS ONE*, 12(4):e0176013, 2017.

[2] N. Davidsohn et al. Accurate predictions of genetic circuit behavior from part characterization and modular composition. *ACS Syn. Bio.*, 4(6):673–681, 2014.

[3] J. R. Kelly et al. Measuring the activity of biobrick promoters using an in vivo reference standard. *Journal of Biological Engineering*, 3(4), 2009.

[4] P. Ramakrishnan and J. J. Tabor. Repurposing Synechocystis PCC6803 UirS–UirR as a UV-violet/green photoreversible transcriptional regulatory tool in e. coli. *ACS Syn. Bio.*, 5(7):733–740, 2016.

[5] L. Wang, P. DeRose, and A. K. Gaigalas. Assignment of the number of equivalent reference fluorophores to dyed microspheres. *Journal of Research of the National Institute of Standards and Technology*, 121:264–281, June 2016.

mLSI Design with MINT

Radhakrishna Sanka[1], Joshua Lippai[1] and Douglas Densmore[1]

[1]Department of Electrical & Computer Engineering, Boston University, Boston, MA

{sanka,jlippai,dougd}@bu.edu

1. INTRODUCTION

Despite these many applications for microfluidics in augmenting the synthetic biology workflow, adoption of microfluidics as a potential experimental and test platform has been slow in the synthetic biology community at large. The majority of synthetic biologists lack both the expertise knowledge in fluid dynamics and microfluidic design and the expensive capital equipment for microfluidic fabrication. Layout of microfluidic designs by hand is both time consuming and error prone. In addition, experiments involving mLSI also require a complex control platform including both software and hardware for valve control and fluid manipulation.

In [3] the authors undertook an herculean effort to design and fabricate the MLSI device with only the aid of tools such as AutoCAD and Solidworks most researchers who use mircofluidics today spend a large portion of their time in manually drawing out every physical feature that constitutes the physical device. We found that in the previous versions of MINT [5, 2] generated layouts that could degrade the performance of the architecture that was specified. In this work we leverage MINT [5] and its ability to specify physical design constraints to easily define large designs and Fluigi to automatically generate the physical layout of these designs and report the updates made to the MINT specification standard and the architectural updates made to Fluigi to process these constraints.

2. MINT

Previous versions of MINT [5], required the user to specify each of the components and lay them out the connections between each of the component. At that version, generating grids of reaction chambers required the was a tedious task which required the user to write scripts that could generate MINT that would be processed for generating a physical design. This process of defining individual components and connections was arduous and hindered the user's ability to specify geometric constraints on the generated layout.

Various geometric constraints such as "GRID", "BANK" and "TREE" would require the relative component positions and orientations to be fixed. The implication of statements such as "GRID" are that it also requires the place and route tool to process and satisfy the geometric constraints given by the designer. In order to satisfy these constraints, an additional stage in the process that would generate macro cells and algorithms that will optimize the placement within these macro cells will also be implemented.

In addition to being able to add geometric constraints. The updates to the Fluigi and MINT architecture now abstract the layers for manufacturing and the component connectivity, hence the layers can be defined as either "FLOW", "CONTROL", "INTEGRATION" without any worry about how many feature depths are present in the device. The components in the "FLOW" layer include everything that make the device functional. The components in the "CONTROL" layer are typically control the components in the "FLOW" layer. Finally the components in the "INTEGRATION" layer will include all the components that will require external integrations and imply hard constraints onto placement algorithms to ensure that external integrations are not blocked by other features. This redesigned architecture can accommodate multiple fabrication protocols for the devices.

3. FLUIGI

Fluigi is the Place and Route tool that is used for automatically generating the physical layout of the microfluidic chip. Figure 1 show a high level description of the various stages where a MINT description of the device is converted into a physical device. The gray box consists of a parser that was generated using ANTLR [4], the purple box consists of the bulk of algorithms that will generate the physical design, the red box consists of a routines that check if the generated design is valid or not and finally the orange box consist of plug-ins that will generate the design outputs.

The new extensions to MINT will require changes to how the device is modeled in Fluigi. The updated process is described in Algorithm 2. In order to accommodate the new constraints that will be introduced in this work, the Microfluidic Device model as described in [2] will be completely restructured.

4. CONCLUSION

Physical design automation remains to be an actively researched problem [1] in the microfluidics space but the control of these MLSI devices remains yet to be integrated with the physical design automation flow. Using the current virtual microfluidic device model Fluigi[2] currently generates control sequences for a single architecture of microfluidic devices. The ability to group components and the facility to export the device to to a standard format will further allow the device descriptions to be imported into future tools that generate control sequences for arbitrary biology protocols in addition to designing large scale microfluidic designs.

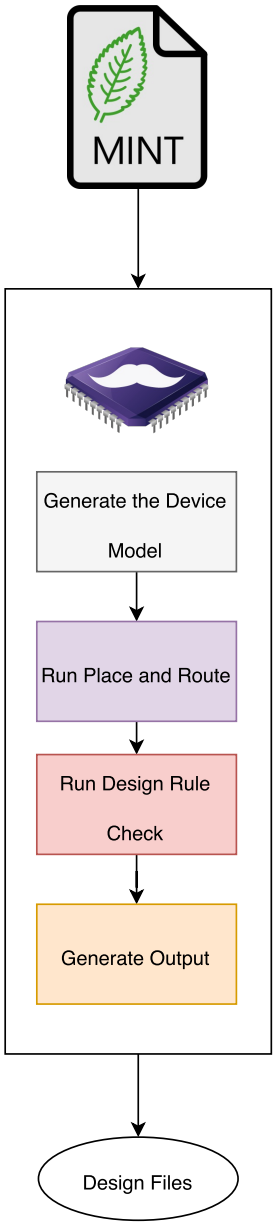


Figure 1: Fluigi Flow - This diagram describes the various stages of the Fluigi Place and Route tool that generates the design files that will be sent for manufacturing.

Require: $N :=$ MINT Description
Ensure: N is a valid description
$D := generate\_device(N)$
Require:
$LB := (\{FLOW, CONTROL, INTEGRATIOM\}\ in\ D$
for all LB **do**
  $G := extract\_groups(FLOW)$
  $C := generate\_placement\_cells(D, G)$
  $FP := generate\_flow\_placements(D, C)$
  for $Pin\ FP$ **do**
    $place\_flow(P)$
    $CP := place\_control(CONTROL, P)$

    $IP := place\_integration(INTEGRATION, CP, FP)$
  end for
  $route(P, CP, FP)$
  $D := import\_place(D, P, CP, IP)$
  $RESULT = design\_rule\_check(D)$
  if $RESULT$ **then**
    $generate\_output()$
  else
    $redo()$
  end if
end for

Figure 2: Place and Route: The above algorithm describes the place and route process and highlights the dependencies between the layers within each individual LAYER BLOCK that consists of a FLOW, CONTROL and INTEGRATION. These dependencies determine the order in which they are placed and routed. The bulk of the placement is determined during the $place\_flow()$.

5. REFERENCES

[1] I. E. Araci and P. Brisk. Recent developments in microfluidic large scale integration. *Current opinion in biotechnology*, 25:60–68, 2014.

[2] H. Huang. *Fluigi: An end-to-end Software Workflow for Microfluidic Design*. PhD thesis, Boston University, 2015.

[3] J. Melin and S. R. Quake. Microfluidic Large-Scale Integration: The Evolution of Design Rules for Biological Automation. *Annual Review of Biophysics and Biomolecular Structure*, 36(1):213–231, 2007.

[4] T. Parr and K. Fisher. LL(*): The Foundation of the ANTLR Parser Generator. In *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '11, pages 425–436, New York, NY, USA, 2011. ACM.

[5] R. Sanka, H. Huang, R. Silva, and D. Densmore. Mint - microfluidic netlist. poster presented at IWBDA 2016, Aug. 2016.

From Machine Reading to Discrete Models of Cellular Signaling

Khaled Sayed
Department of Electrical and Computer Engineering,
University of Pittsburgh, PA, 15213
USA
k.sayed@pitt.edu

Cheryl A. Telmer
Department of Biological Sciences
Carnegie Mellon University,
Pittsburgh, PA, 15213
USA
ctelmer@cmu.edu

Natasa Miskov-Zivanov
Department of Electrical and Computer Engineering, Bioengineering,
Computational and Systems Biology
University of Pittsburgh, PA, 15213
USA
nmzivanov@pitt.edu

ABSTRACT

We describe here our approach to translating machine reading outputs, obtained by reading biological signaling literature, to discrete models of cellular networks. Our approach starts with creating a representation format that allows for receiving the information from literature and converting it into mechanistic discrete models. The representation format can also be considered as a scaffold for the retrieved information and as a guide for the reading engines. Here, we use outputs from three different reading engines, and describe our approach to translating their different features.

KEYWORDS

Discrete Logical Modeling, Cell Signaling Networks, Automated Literature Reading.

1 INTRODUCTION

Biological knowledge is voluminous; it is nearly impossible to read all scientific papers on a single topic. When building a model of a particular biological system, researchers usually start by searching for existing relevant models and by looking for information about system components and their interactions in published literature.

Although there have been attempts to automate the process of model building [1], most often modelers conduct these steps manually, with multiple iterations between (i) information extraction, (ii) model assembly, (iii) model analysis, and (iv) model validation through comparison with most recently published results. To allow for rapidly modeling the complexity of diseases like cancer, and for efficiently using ever-increasing amount of information in published work, we need representation standards and interfaces such that these tasks can be automated. This, in turn, will allow researchers to ask informed, interesting questions that can improve our understanding of health and disease.

To this end, the contributions of the work presented in this abstract include: a ***novel standardized representation*** of the information used to create executable models of cellular signaling and an ***approach to effectively translate*** machine reading output to discrete models of cellular signaling.

2 FRAMEWORK OVERVIEW

To automatically incorporate new reading outputs into models, we have developed a reading-modeling-explanation framework, called DySE (Dynamic System Explanation), outlined in Figure 1. This framework allows for (i) expansion of existing models or assembly of new models using machine reading, (ii) analysis and explanation of models and (iii) generation of machine-readable feedback to reading engines. In this paper, we describe *the front end of the framework, the translation from reading outputs to the list of elements, and their influence sets, including any available context information.*

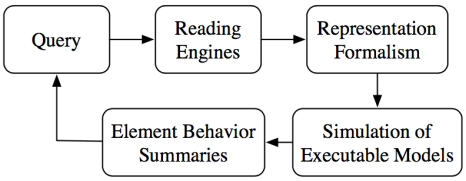


Figure 1: DySE framework.

3 MODEL REPRESENTATION FORMAT

To enable fast and accurate translation from a reading engine output to our modeling format, our models are first created in the form of a spreadsheet. It is important to note here that the spreadsheet representation does not include final update rules, that is, the spreadsheet version of the model is further translated into an *executable model* as an input to a simulator. Each row in the spreadsheet corresponds to one specific model element (i.e., modeled system component), and the columns are organized in several groups: 1) information about the modeled system component, 2) information about the component's regulators, and 3) information about knowledge sources. This format enables straightforward model extension to represent both additional system components as new rows in the spreadsheet, and additional component-related features by including new columns in the spreadsheet. The addition of new columns occurs as various parts of the framework become more sophisticated.

The first group of fields in our representation format includes **system component-related** information. This information is either used by the executable model, or kept as background information to provide specific details about the system component when creating a hypothesis or explaining outcomes of wet lab experiments.

A. `Name` – full name of element, e.g., "Epidermal growth factor receptor".
B. `Nomenclature ID` – name commonly used in the field for cellular components, e.g., "EGFR" is used for "Epidermal growth factor receptor".
C. `Type` – these are types of entities used by reading engines such as protein, chemical, RNA, gene, biological processes.

D. `Unique ID` – we use identifiers corresponding to elements that are listed in databases such as UniProt, PubChem, HGNC, MeSH [2].
E. `Location` - we include subcellular locations such as plasma membrane, cytoplasm, nucleus or the extracellular space.
F. `Location identifier` – we use location identifiers from Gene Ontology (GO) database.
G. `Cell line` – obtained from reading output.
H. `Cell type` – obtained from reading outputs.
I. `Tissue type` – obtained from reading output.
J. `Organism` – obtained from reading output.
K. `Executable model variable` – variable names currently include above described fields B, C, E, and H.

The second group of fields in our representation includes **component regulators-related** information that is mainly used by executable models, with a few fields used for bookkeeping, similar to the first group of fields.

L. `Positive regulator nomenclature IDs` – list of positive regulators of the element.
M. `Negative regulator nomenclature IDs` – list of negative regulators of the element.
N. `Interaction type` – for each listed regulator, in case it is known whether interaction is direct or indirect.
O. `Interaction mechanism` – for each known direct interaction, if the mechanism of interaction is known. Examples of such mechanisms are Activation, Inhibition, Binding, or Phosphorylation.
P. `Interaction score` – for each interaction, a confidence score obtained from reading.

The third group of fields in our representation includes **interaction-related provenance** information

Q. `Reference paper IDs` – for each interaction, we list IDs of published papers that mention the interaction. This information is obtained directly from reading output.
R. `Sentences` – for each interaction, we list sentences describing the interaction. This information is obtained directly from reading output

4 FROM READING TO MODEL

We obtain outputs from three reading engines, namely REACH [3], RUBICON [4], and Leidos table reading (LTR)[5]. The REACH engine can extract simple direct and indirect interactions such as activation and inhibition as well as extracting nested interactions that have an element "C" which can positively or negatively regulate the activation or inhibition of element "B" by element "A". The RUBICON engine provides two reading outputs, one for *direct* and one for *indirect interactions*. For the indirect interactions, it creates a chain of elements that starts with the regulator and ends with the regulated element and includes the intermediate elements forming a path from the regulator to the regulated elements. The Direct output file contains additional information such as Confidence and Tags. The Confidence indicates how *confident* the reading engine is about the extracted interaction, and it can be LOW, MODERATE, and HIGH. The Tags includes *epistemic tags* such as 'implication', 'method', 'hypothesis', 'result', 'goal', or 'fact'. Additionally, LTR engine performs table reading and can extract more detailed biological information since tables tend to describe a highly specific experiment about interacting components.

The three reading engines provide output files with similar but not exactly the same format as the format described in section 3. For example, RUBICON output files have three additional columns, namely Confidence, Tags, and the intermediate elements of a regulation path. REACH output includes information about the formation and dissociation of protein complexes (two elements) in one row, which needs to be translated into the format in section 3 where each row represents only one element. Figure 2 shows an example of the regulation of protein complexes where the formation of protein complex AB can be activated or inhibited by element C. Also, Table 1 shows how we translate REACH output with interaction that includes a complex into our format.

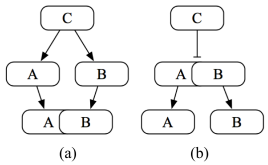


Figure 2: Regulation of Complexes. (a) complex formation, (b) complex dissociation.

Table 1: Translating REACH output for complexes into DySE format

		Fig. 2(a)		Fig. 2(b)	
		Element Name	PosReg	Element Name	NegReg
REACH		{A,B}	C	{A,B}	C
DySE	Row1	A	B and C	A	B and C
	Row2	B	A and C	B	A and C

5 CONCLUSION

This paper describes the design of a standardized format for the processing of machine reading outputs from multiple readers. Our automated framework allows for executable models to be assembled and tested at a completely new scale and to incorporate information at a level not previously possible. We hope that this formalized representation of research findings for the purpose of creating dynamic models will improve knowledge and increase our understanding of biological systems to guide future studies.

REFERENCES

[1] N. Miskov-Zivanov, "Automation of Biological Model Learning, Design and Analysis," in *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, 2015, pp. 327-329.
[2] Bioresources. *Biological entities IDs*. Available: https://github.com/clulab/bioresources/wiki/KBs
[3] M. A. Valenzuela-Escárcega, G. Hahn-Powell, T. Hicks, and M. Surdeanu, "A domain-independent rule-based framework for event extraction," *ACL-IJCNLP 2015,* vol. 127, 2015.
[4] G. A. Burns, P. Dasigi, A. de Waard, and E. H. Hovy, "Automated detection of discourse segment and experimental types from the text of cancer pathway results sections," *Database,* vol. 2016, p. baw122, 2016.
[5] S. Sloate, V. Hsiao, N. Charness, E. Lowman, C. Maxey, G. Ren*, et al.*, "Extracting Protein-Reaction Information from Tables of Unpredictable Format and Content in the Molecular Biology Literature," presented at the Bioinformatics and Artificial Intelligence (BAI), New York, USA, 2016.

Garuda: A Synthetic Biology Platform that Learns from Data

Rizki Mardian[1], Manuel Gimenez[2], Marcia Sahaya Louis[1], Radhakrishna Sanka[1], and Douglas Densmore[1, 2]

[1]Department of Electrical and Computer Engineering, Boston University, Boston, MA
[2]Department of Biomedical Engineering, Boston University, Boston, MA

1. INTRODUCTION

Recent advances in genetics and genomics technologies have made it possible for scientists to produce large-scale functional data to characterize biological process in a relatively a short time. Although that it should be obvious that many studies may benefit from these massively available datasets, surprisingly only few efforts have attempted to interface synthetic biology with computational sciences that deal with data.

In this work, we introduced Garuda (short for: Genetic Automation: Recommendation Unit and Data Analyzer), a programming-biology platform that consists of collections of data mining, pattern analysis, and machine-learning algorithms tailored for synthetic biology purpose. As a computational application that learns from data, Garuda is attached to a centralized data management tool, named Clotho [1]. Garuda runs pattern analysis and data mining techniques to come up with certain insights from hidden information (i.e. information that was not stored by users explicitly), and meaningful recommendations may guide users for designing or even debugging their experiments. As the data management part of Garuda allows cross-experimental data-aggregation between multiple users, this tool can also be employed as a knowledge-sharing mechanism in a project involving a large number of researchers.

As proof-of-concepts, we have utilized Garuda in two different study cases: 1) a task of finding the subtle sources of toxicity, and 2) a task of characterizing genetics circuits.

2. OVERVIEW

Garuda is a computational platform with a novel capability to extract hidden patterns across large datasets and produce meaningful insights from this data. General overview of Garuda's architecture is shown in Figure 1.a below.

In a high level, Garuda system can be divided into two different sub-systems, the database and the machine learning layer. The first-mentioned deals with data persistence and data management. Garuda is designed as a modular client-side application that can be coupled with any database as a back-end for data storage. Currently, Garuda supports Clotho (previously developed at the CIDAR lab) and Synbiohub (previously known as SBOL) which has been utilized broadly in synthetic biology communities. The second-mentioned consists of collections of machine learning algorithms that can be employed to solve various problems as defined by users. As these algorithms reside within Garuda as independent modules (color-coded in Figure 1.a: red for unsupervised learning, blue for supervised learning, and green for regression model), users can easily add or modify new scripts to support different models or functionalities. Presently, Garuda supports Java and Python, and provides API for communication with other applications through RESTful connection. Garuda also provides a nice web-based user interface through which users can feed the machine learning engine with data and questions.

3. RESULTS

3.1 Sources of Toxicity

Toxicity is one of the most challenging problems in molecular and synthetic biology to date. Certain combination of genetic parts may not operate as expected when they are put together, even when there is no direct harmful interaction between individual components. Tracing all combinations by hand or ad-hoc debugging process to find the toxicity sources can be very tedious.

In contrast, Garuda can tackle this problem in an automated fashion by performing machine learning-based evaluation over large genetic design datasets. Using multiple regression, Garuda identifies possible toxic candidates by filtering the most significant parameters that deviate the normal distribution of a pool of healthy genetic parts.

Figure 1.b shows the performance test of the regression model over different parameters. For these purpose, we have generated artificial data consisting n-number of genetic designs ($n = 100 - 1000$), with the following assumption. Each design is constructed from m-number of parts ($m = 2 - 10$), and each part is either randomly assigned as "toxic" or "healthy". Therefore, we have a list of toxic parts and a list of healthy parts, and each list is associated with one normally distributed population with mean=μ (μ_t, μ_h) and standard deviation=σ (σ_t, σ_h), representing the toxic and healthy distribution. When the design consists of any part from the toxic lists it is then assigned with a growth-rate drawn from the toxic distribution (otherwise, from the healthy distribution).

Multiple regression model is then employed to predict the candidates for the toxic parts and plotted in two different modes: predictive (red-lines), and accurate (blue-lines). From our observations, the model worked with 90% of accuracy if Garuda was provided with 500 designs, less than 25% of toxic parts, and the performance was better when the toxicity-healthiness are derived from two well-spread toxic and healthy distributions.

3.2 Genetic Circuits Characterization

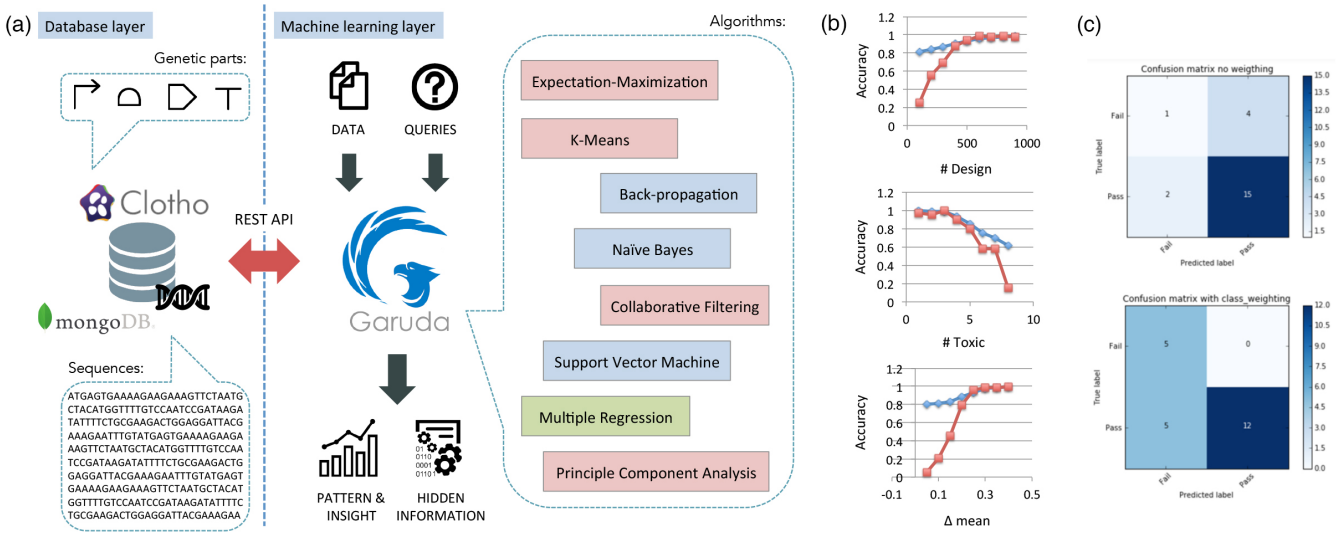


Figure 1: (a) General architecture of Garuda, (b) Simulation results of the sources of toxicity problem, (c) Simulation results of the genetic circuits characterization problem

In-silico Bio-Design Automation (BDA) tools, such as Cello [2], provide abstraction and accelerate the design and implementation of genetic circuits. But these circuits are sometimes do not behave as expected. This is due to the inherent high complexities of living organisms and also limited by the scope of understanding of hidden biological phenomena. Iterative design-build-test-learn cycles to improve the design reliability and achieve a higher rate of the correctly working circuits can be extravagant in terms of time and resources.

In this work, we have used Garuda to generate models to characterize experimental data from Cello. Garuda used a black-box filtering approach to classify the in-silico designs, into two classes: "defective" and "functional" (in supervised learning, this module is termed data classifier). In particular, Linear Support Vector Machine (SVM) algorithm [3] was selected as the problem solving approach.

Figure 1.c shows the simulation results of two different models of Linear SVM (with no weighting, and with class weighting). The dataset contains 55 different circuits, which is divided into training and testing data, with the ratio 3:2 (this ratio is arbitrarily chosen based on typical performance of linear SVM). We have created a mxn matrix for the occurence of parts within each circuit (with m =number of circuit, and n =number of unique parts), and utilized this as the feature vectors for the model. For example, when a circuit consists of one particular part, then a value of '1' is assigned to its correspondent column in the matrix (or a value of '0' if otherwise). We then trained the model. The no-weighting model all data points have equal weight, while the class-weighting model assigned different weights to parts according to how often they are occured in the whole circuits. From our observation, we found that SVM with class-weighting model has better performance.

4. DISCUSSION

To date, many Bio-Design Automation tools have been introduced and they have shaped innumerable important breakthrough in synthetic biology. However, most of these tools (if not all) still lack in the following features: 1) task-specific, 2) requires extensive libraries, 3) reusability issues when working with similar problems but different libraries, and vice-versa, and 4) does not benefit from multiple users knowledge.

On the other hand, large amount of data are produced daily from many different labs thanks to the advance in genetics and genomics technology nowadays. We have developed Garuda, a programming-biology platform that learns from data. As the field of synthetic biology is continously growing, and more data are being produced everyday, Garuda has a big potential to guide and assist experimentalists, with its novel capability to mine hidden information for large datasets.

While Garuda is intended to be an application-agnostic and data-hungry tool (as long as it is provided with data, custom applications can be re-written easily), we have provided two different study cases as the proof of concept. Future directions of this work include experimental validation and cross-validating the predictions to improve the accuracy of the tool through reinforcement learning mechanisms. Garuda will also be publicly available online (for non-commercial purpose) via Web-UI and RESTful API.

5. ACKNOWLEDGEMENTS

The authors thank Dr. Benjamin D. Gordon for the discussions. This work has been funded by the US Defense Advanced Research Projects Agency (DARPA) Living Foundries award HR0011-15-C-0084.

6. REFERENCES

[1] Densmore, D, et. al. *A platform-based design environment for synthetic biological systems* The Fifth Richard Tapia Celebration of Diversity in Computing Conference: Intellect, Initiatives, Insight, and Innovations, New York, NY, USA, pp. 24-29, 2009

[2] Nielsen, A. A. K., et. al. *Genetic circuit design automation.* Science vol. 352 (6281), 2016

[3] Noble, W. S *What is a support vector machine?.* Nature Biotechnology 24, pp. 1565-1567, 2006

Forward Engineering of Optimal CRISPR Guide RNA Sequences via Machine Learning

Riley Doyle*
Desktop Genetics Ltd
28 Hanbury Street
London, UK E1 6QR
rileyd@desktopgenetics.com

Mark Dunne
Desktop Genetics Ltd
28 Hanbury Street
London, UK E1 6QR
markd@desktopgenetics.com

Neil Humphryes-Kirilov
Desktop Genetics Ltd
28 Hanbury Street
London, UK E1 6QR
neilh@desktopgenetics.com

Leigh Brody
Desktop Genetics Ltd
28 Hanbury Street
London, UK E1 6QR
leighb@desktopgenetics.com

1 MOTIVATION

The popularity of CRISPR RNA-guided genome editing makes it an ideal test case developing automated nucleic acid design systems. First generation CRISPR genome editing has yielded a number of decision rules that are used by bench scientists to select one or more guide RNA sequences for a particular application [2]. These decision rules use sequence-level and contextual features of the predicted RNA-guided endonuclease (RGEN) cut sites in the target genome to predict the activity, specificity, and post-repair outcome of the RGEN in real world experiments.

2 MODEL DEVELOPMENT

We assembled a data warehouse of 640,455 of experimentally tested guide RNAs from public sources [4][5], collaborators, and our own projects to systematically test 4,906 predictors of guide RNA performance (model features) across multiple species, cell lines, and protocol variations. We tested whether various on-target and off-target predictors [2][6][1][3] generalize across species and cell lines. We also assessed various measures of guide performance, devised appropriate statistics for normalizing guide performance data across experiments, and investigated how prediction accuracy changes with incremental increases in the amount of prior data. We applied these results to derive improved predictors and measures of guide RNA activity and specificity.

3 MODEL EVALUATION

For sequence-based predictors, the derived weights for the 1-mer (single base) features were similar to the result shown previously by Doench and Xu. Both the identity and position of a base have an effect on guide performance and the magnitude of influence increases with proximity to the guide's PAM site. Our human and mouse models assigned the largest weights to the 3-mer nucleotide triplets, including the known stop codon TAA, and GC content at the expense of biochemical features like transcript representation. The distribution of predictor magnitudes followed a reasonably smooth power law distribution from between 0.005 and 0 with roughly the top half of magnitudes between 0.005 and 0.001.. There were no "break away" predictors that dominated the model. The widely used predictor of percent peptide, the distance of the cut site from the start codon into the coding DNA sequence, received a low weight of 3488 of 4906 overall predictors, consistent with our prior, unpublished work.

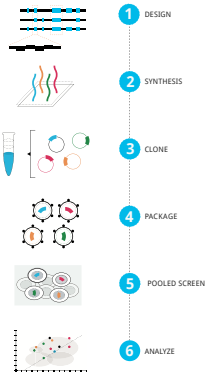


Figure 1: Overview of major steps in the protocol, adapted from [5] and [4]. The representation of this plasmid pool was measured by NGS and transfected into a packaging cell line. Virons where harvested, assayed for titer, and used to transduce target cell lines. Cell cultures were sampled at several time points following transduction. DNA was extracted and space sequence counts were determined via deep sequencing using primers specific to a constant portion of the viral progenome.

4 DISCUSSION

Our investigation yielded several results that contradicted our hypotheses and suggest the need for context-aware algorithms. First, models of guide RNA performance did not generalize effectively between species, yet did generalize across cell lines with some loss in

*Presenting Author

IWBDA 2017, August 2017, Pittsburg, PA USA

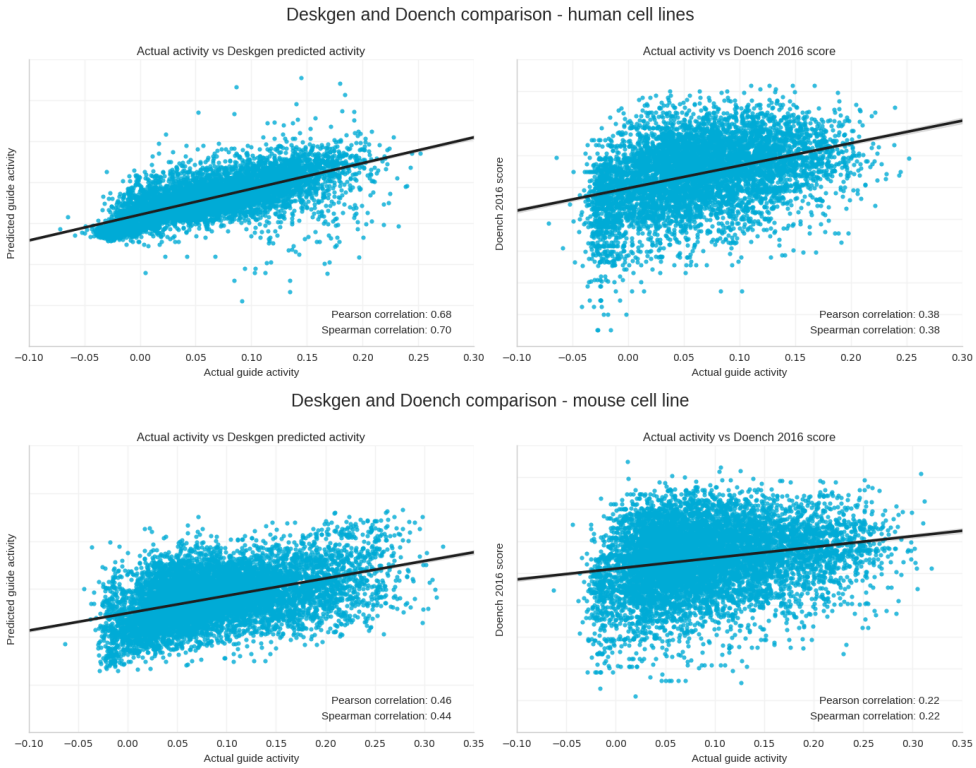


Figure 2: Comparison of predicted guide RNA performance with actual guide RNA performance. On the left is the model generated as part of this work for guide RNAs targeting the human (upper) and mouse (lower) genomes. On the right are the predictions the model developed in [1] for these same guide RNAs. Models were evaluated by comparing both Spearman and Pearson correlation coefficients across an unseen data set for guide RNAs targeting the human and mouse genomes. For comparison, repeat measurements of these same guide RNAs have an inter-replicate Spearman correlation of 0.78. Similar relative differences where obtained for the mouse guide RNAs, but with lower Spearman correlations of 0.44 and 0.22 respectively.

accuracy. Second, the diversity of the prior data set and the processing of experimental measurements had larger effects on predictor performance than model complexity or volume of prior data. Third, effective predictors of guide RNA performance at the population level may be overridden by the functional context of each target sequence. Finally, the quality and breadth of this prior knowledge about the target genome (i.e. annotation tracks) had a large impact on prediction accuracy. These findings suggest there is a significant unmet need for better data resources for non-human genomes and for best practices for characterizing and comparing RNA-guided endonuclease activity and specificity between experiments.

REFERENCES

[1] John G Doench, Nicolo Fusi, Meagan Sullender, Mudra Hegde, Emma W Vaimberg, Katherine F Donovan, Ian Smith, Zuzana Tothova, Craig Wilen, Robert Orchard, Herbert W Virgin, Jennifer Listgarten, and David E Root. 2016. Optimized sgRNA design to maximize activity and minimize off-target effects of CRISPR-Cas9. *Nature biotechnology* 34, 2 (feb 2016), 184–91. https://doi.org/10.1038/nbt.3437

[2] John G Doench, Ella Hartenian, Daniel B Graham, Zuzana Tothova, Mudra Hegde, Ian Smith, Meagan Sullender, Benjamin L Ebert, Ramnik J Xavier, and David E Root. 2014. Rational design of highly active sgRNAs for CRISPR-Cas9-mediated gene inactivation. *Nature biotechnology* 32, 12 (dec 2014), 1262–7. https://doi.org/10.1038/nbt.3026

[3] Fuguo Jiang, David W. Taylor, Janice S. Chen, Jack E. Kornfeld, Kaihong Zhou, Aubri J. Thompson, Eva Nogales, and Jennifer A. Doudna. 2016. Structures of a CRISPR-Cas9 R-loop complex primed for DNA cleavage. *Science* 351, 6275 (2016), 867–871. https://doi.org/10.1126/science.aad8282

[4] Ophir Shalem, Neville E. Sanjana, Ella Hartenian, Xi Shi, David A. Scott, Tarjei S. Mikkelsen, Dirk Heckl, Benjamin L. Ebert, David E. Root, John G. Doench, and Feng Zhang. 2014. Genome-Scale CRISPR-Cas9 Knockout Screening in Human Cells. *Science* 343, 6166 (2014), 84–87. https://doi.org/10.1126/science.1247005 arXiv:NIHMS150003

[5] Tim Wang, Jenny J. Wei, David M. Sabatini, and Eric S. Lander. 2014. Genetic Screens in Human Cells Using the CRISPR-Cas9 System. *Science* 343, 6166 (2014), 80–84. https://doi.org/10.1126/science.1246981 arXiv:NIHMS150003

[6] Han Xu, Tengfei Xiao, Chen-Hao Chen, Wei Li, Clifford A Meyer, Qiu Wu, Di Wu, Le Cong, Feng Zhang, Jun S Liu, Myles Brown, and X Shirley Liu. 2015. Sequence determinants of improved CRISPR sgRNA design. *Genome research* 25, 8 (aug 2015), 1147–57. https://doi.org/10.1101/gr.191452.115

Algorithms for selecting fluorescent reporters

Evan Appleton[1,2,*], Prashant Vaidyanathan[3,4,*], David Tran[4], Alex Vahid[4], George Church[1,2] and Douglas Densmore[3,4]

[1]Department of Genetics, Harvard Medical School, Boston, MA
[2]Wyss Institute for Biologically Inspired Design, Boston, MA
[3]Biological Design Center, Boston University, Boston, MA
[4]Department of Electrical & Computer Engineering, Boston University, Boston, MA
[*]These authors contributed equally to this work
{prash,avahid,djtran,dougd}@bu.edu,{evan_appleton}@hms.harvard.edu,
{gchurch}@genetic.med.harvard.edu

1. INTRODUCTION

Molecular biologists and biochemists rely on the use of fluorescent probes (fluorophores) to take measurements of their model systems. These fluorophores fall into various classes (i.e. fluorescent dyes, fluorescent proteins, etc.), but they all share some general properties and required equipment for data acquisition. Specifically, all fluorophores have an excitation and emission spectra and require some light source (typically a laser) to excite them and light detectors with filters to capture emitted light within a range of wavelength. While they all share some common properties, selecting an ideal fluorphore or set of fluorophores for a particular measurement technology or designing the measurement technology parameters around a decided set of fluorophores is a multidimensional problem that is difficult to solve solve with *ad hoc* methods.

Historically, decisions on fluorescent reporter selection and fluorescence detection machine specifications were done by studying individual fluorophores and referencing curated guides[3] by experts in the field. More recently, some software tools have been developed and put online to help biologists make selections of fluorophores by plotting data for multiple fluorophres on individual lasers and filters to aid in this optimization[1, 2]. While these resources contain many fluorophore spectra, no existing tool will plot excitation and emission spectra for fluorophores on multiple lasers simultaneously and correct them for brightness and other important physical properties. Moreover, there are no known tools that house algorithms for selecting an optimized or optimal set of fluorophores based on maximizing signal-to-noise ratio for each signal or optimizations for other physical properties like oligomerization.

Here we detail the computational complexity of solving these problems and describe a set of algorithms for solving these problems efficiently to arrive at solutions optimized for a maximum signal-to-noise ratio, material cost, and other additional parameters. We have housed these algorithms in a web-based software tool that contains curated data sets for large sets of available fluorophores and measurement machine components.

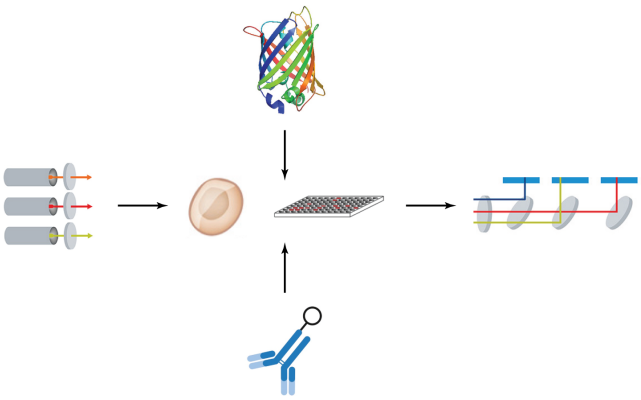


Figure 1: Fluorescent reporters are either engineered to be produced in the cell as fluorescent proteins or applied in vitro after fixation with conjugated antibodies or other techniques. The cells are then fed into a measurement machine where lasers excite the reporters and their emitted light is captured by filtered light detectors.

2. COMPUTATIONAL COMPLEXITY

The task of exhaustively searching and choosing the right set of fluorescent reporters for a given cytometer, is a problem with factorial complexity. The problem statement can be formulated as:

PROBLEM 1. *For a set of Fluorophores X, and a cytometer C, which has Y lasers, where each laser $y_i \in Y$ has a set of Detectors Z_i, find a set $X_n \in X$ (where $\mid X_n \mid = n$) such that the average signal-to-noise (SNR) is the highest for any X_n.*

SNR for a specific fluorophore $x \in X_n$ for a detector z is defined as:

$$\frac{\text{emission spectra of x within z}}{\text{sum of all emission spectra of } x_i \in \{X_n - \{x\}\} \text{ in z}} \quad (1)$$

To exhaustively search for X_n, every permutation of fluorophores has to be checked against every combination of all available detectors (Z is the union of all detectors of every

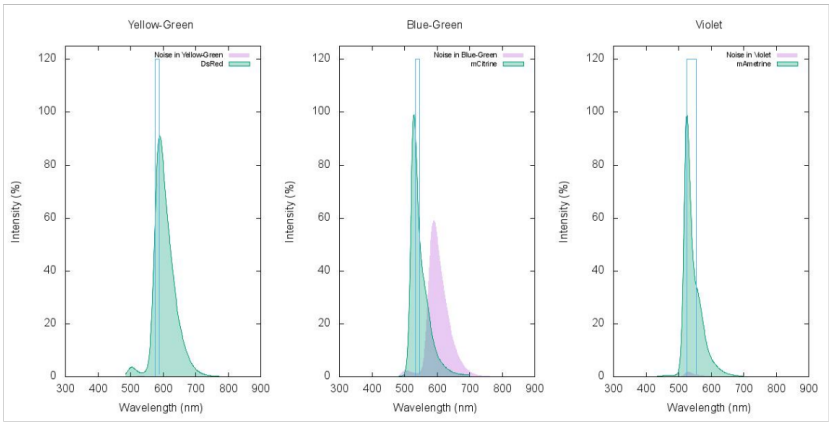


Figure 2: An example of a graph produced by the web tool, showing the signal and noise in a specific detector. In this case, the Signal in any detector is shown in green while noise is shown in purple. In this run, 3 fluorophores were chosen out of 14 possible options for the cytometer in Boston University: DsRed in detector E of the Yellow-Green Laser with an SNR value of 75.506, mCitrine in detector A of the Blue-Green Laser with SNR 30.271, and mAmetrine in detector B of the Violet laser with SNR 27.044.

laser in Y). The overall complexity can be generalized as: .

$$^{x}P_{n}.^{z}C_{n}$$

3. WEBTOOL

The algorithms are connected to a front-end user interface in which users can download large sets of example data or upload their own data to evaluate both their current choices of fluorophore usage and machine configurations and compare them to optimized solutions. Users can enter the cytometer settings by uploading a standard cytometer configuration file (which contains the laser and detector metadata) as well as a list of all available fluorophores along with relevant metadata for each fluorophore (such as normalized emission and excitation spectrum). Users can then specify the number of fluorophores, n, they wish to use in their circuits concurrently and can choose one of the following algorithms to get the best n fluorophores that have the highest average SNR ratio over all fluorophore-detector combinations:

- Exhaustive search: In this approach, every permutation of n fluorophores are compared against every combination of n detectors in the cytometer. The permutation with the least overall SNR value is chosen.
- Beam search: In this approach, only a subset of combinations of detectors are considered based on the Riemann Sum of the excitation spectrum of a fluorophore compared to the width of the detector. The detector combinations are preprocessed and the top p% permutations are chosen where x is specified by the user.
- Simulated annealing: In this approach, a random permutation of fluorophores and combination of detectors are chosen as the starting point of the algorithm. The algorithm runs for a fixed number of iterations where at each iteration, a random fluorophore or detector is replaced if the new SNR is greater than the current SNR or if the annealing schedule allows a lower SNR score.

The result of each run is specified as a set of fluorophores along with the corresponding laser and detector, and the SNR value for that specific detector. The web-tool also displays a graph showing the signal and noise for each detector. Fig 2 is an example of one such graph.

The web-interface will be able to increase the quality of the signal-to-noise in the measurements that users must take and in the cases of fluorescent dyes, minimize their cost of material expense.

4. EXPERIMENTAL VALIDATION

Since fluorescent proteins are often not expressed in equal amounts, dyes can bind with different affinity, and lasers and detectors can be toggled for sensitivity, we needed to validate that our predicted signal-to-noise ratios matched observed signal-to-noise ratio on a measurement machine. We ran an experiment in which we expressed five different fluorescent proteins in individual *E. coli* strains and three fluorescent dyes conjugated to non-fluorescent beads provided by the dye manufacturer.

To test how signal-to-noise was impacted by adjustments in laser power and voltage, we tested each strain and dyed beads at different voltages on each of five lasers and their respective optimal detector voltages. After correcting data for brightness and excitation and emission percentages according to machine settings, normalizing to absolute units with rainbow calibration beads, and subtracting autofluorescence, we found that our data was consistent with prior findings that increasing laser power and detector voltage scaled linearly with the produced signal but generally decreased signal-to-noise ratio. To confirm that our predictions with these corrections were consistent, we tested the same set of strains on a different measurement machine and produced the predicted results.

5. REFERENCES

[1] Fluorophores.org. http://www.fluorophores.tugraz.at/, 2011.

[2] Spectra database hosted at the university of arizona. http://www.spectra.arizona.edu/, 2011.

[3] N. C. Shaner, P. A. Steinbach, and R. Y. Tsien. A guide to choosing fluorescent proteins. *Nature methods*, 2(12):905–909, 2005.

An automated cell-free method for the measurement of mammalian gene-expression

Margarita B. Kopniczky
Imperial College London
m.kopniczky@imperial.ac.uk

Kirsten Jensen
Imperial College London
k.jensen@imperial.ac.uk

David W. McClymont
Imperial College London
d.mcclymont@imperial.ac.uk

Paul S. Freemont
Imperial College London
p.freemont@imperial.ac.uk

ABSTRACT

A cell-free Transcription-Translation (TX-TL) method[1] is proposed for the measurement of mammalian gene-expression where acoustic liquid handling is used to assemble reactions, thus generating data in high-throughput. Several technical challenges were solved in the process of integrating wetware-hardware-software to get a reliable workflow. The key features of the basic TX-TL reaction are presented and the experimental design for library screening or the characterization of regulatory elements is outlined.

1 INTRODUCTION

The tediousness and costliness of tissue-culture methods is one of the factors that limit the growth of the mammalian synthetic biology field. In contrast, cell-free TX-TL reactions can be assembled fast and take only 5 hours to run, use little reagents and can be automated (Figure 1).

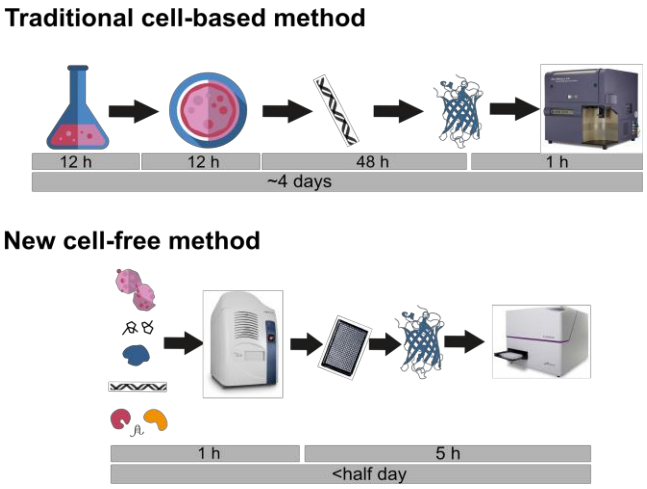


Figure 1: A transfection experiment involves manual steps, takes days and results are obtained by flow cytometry or microscopy. In contrast, TX-TL experiments can be automated using liquid handling technology and the output is measured by a fluorescent plate-reader.

Tip-based liquid handling is a popular choice in lab automation, but operates in the µL range and is slow, two factors which are major disadvantages for the TX-TL method. Microfluidics can operate with droplets at low volumes but is often limited in throughput and the flexibility of experimental design. Acoustic liquid handling technology has the advantage of working with micro-plates ideal for high-throughput, and is fast (750,000 liquid transfers per day) with nL accuracy. Furthermore, Labcyte's Echo machines were also an ideal solution because their software takes instructions that are easy to generate programmatically.

2 METHODS

2.1 Acoustic liquid handling automation

The CherryPick software from Labcyte takes a CSV file with a list of liquid transfer steps to carry out. This file is generated by method-specific code written especially for TX-TL experiments that takes high-level parameters such as the number of replicates, DNA constructs etc. (Figure 2). The experiments are designed so that each given DNA construct is tested in combination with all the regulators, at all the specified concentrations. Blank reactions with no DNA and Positive controls with a specified DNA concentration are also added to the list of reaction types. A key feature is that each TX-TL reaction is randomly assigned to a well on a 384 well-plate in order to distribute error evenly across the plate and counter any potential position associated effects. The last row of the plate is reserved for FITC standards, which are optionally added for MEFL (Molecules of Equivalent Fluorescein) unit conversion. Finally, the amounts of the reagents needed for a given experiment are also calculated by the algorithm.

The output includes three files. One contains instructions for the biologist about how to prepare the source plate containing the reagents. The second contains machine instructions for the liquid transfer steps and the third stores all the information about the given experiment for downstream data analysis.

2.2 Molecular Biology

Plasmids were purified (QIAGEN Plasmid Plus Kits) and their concentration was quantified using 20nl PicoGreen reagent in a

total reaction volume of 20ul in 384 well plates (Thermo Fisher P11496). The TX-TL reactions were assembled from the 1-Step Human Coupled IVT Kit (Thermo Fisher 88881) in a total reaction volume of 2uL in clear bottom small-volume 384 plates (Greiner 788 096), sealed, and measured with bottom-optic setting of a Biotek SynergyMx plate reader.

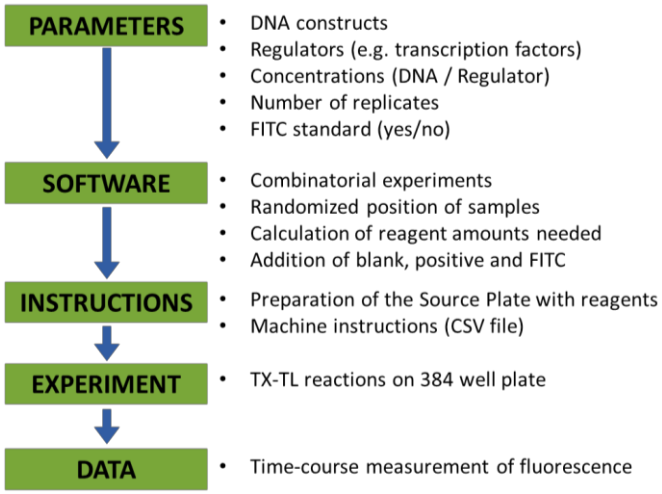


Figure 2: The automated TX-TL workflow relies on custom software to generate both the human readable and the machine instructions for carrying out an experiment.

3 RESULTS

3.1 Method optimization

Several technical problems had to be tackled when taking the lab-bench method to an automated workflow. First, the total reaction volume and the expensive TX-TL component were minimized to a 2µL total volume and 60% TX-TL component. Second, DNA purification and quantification methods were optimized. In order to neutralize any effects resulting from the plate layout or the order that samples are added to the plate, a randomization step in the assignment of the location of each sample to a well was included in the code. However, this first increased the error to an unacceptable extent and it was found that the homogeneity of the TX-TL mix was compromised at a centrifugation step, which was then eliminated to solve the problem. Lastly, the MEFL conversion of fluorescence and a constant positive control constitutive GFP construct provides the check on the quality of the TX-TL activity.

3.2 Characterization of the TX-TL reaction

Time course measurements show saturation of the reaction at 5h, which was chosen as the time point for data analysis on Figure 3B. The saturated GFP levels reflect the production rates in the linear phase of expression. With respect to DNA concentration, the reaction saturates at about 4nM for the positive control DNA (pCFE-GFP).

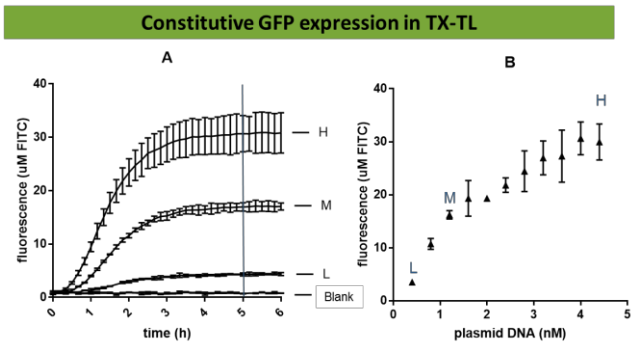


Figure 3: A: Time course measurement of fluorescence in TX-TL reactions. Blank reaction without DNA, low (L), medium (M) and high (H) template DNA concentrations are shown, corresponding to figure B. Error bars represent standard deviation from 3 replicates. B: Saturation curve of pCFE-GFP plasmid constitutively expressing GFP at 5h time point. Blank corrected fluorescence is shown, error bars represent standard deviation from 3 replicates.

4 DISCUSSION

The potential range of applications is wide and the new method may be applied for screening and characterizing new regulatory elements for mammalian synthetic biology and sample a genetic and experimental design space otherwise unreachable by traditional tissue-culture methods. Furthermore, the approach of assembling TX-TL reactions with the presented method can be applied to other organisms. In-depth data was acquired to feed models of gene-expression regulation in *B. megaterium* cell-free this way, for example[2].

The software tools are in the process of being made available online in the form of a Web Application for those with little coding experience. Interfacing with the liquid handling platform through the downloadable machine instruction files will lower the barrier for a wider user base. Not only the testing, but also the assembly of genetic constructs can be automated and this approach, where web application software is used to generate instructions for automated platforms is used extensively at the London DNA Foundry. Building up the software tools and protocol databases for automation by acoustic liquid handling methods could greatly benefit the synthetic biology community.

ACKNOWLEDGMENTS
This work was supported by EPSRC.

REFERENCES

[1] Hodgman, C.E. and Jewett, M.C., 2012. *Cell-free synthetic biology: thinking outside the cell*. *Metabolic engineering*, 14(3), pp.261-269.

[2] Moore, S.J., MacDonald, J.T., Weinecke, S., Kylilis, N., Polizzi, K.M., Biedendieck, R. and Freemont, P.S., 2016. *Prototyping of Bacillus megaterium genetic elements through automated cell-free characterization and Bayesian modelling*. *bioRxiv*, p.071100.

[3] Storch, M., Casini, A., Mackrow, B., Fleming, T., Trewhitt, H., Ellis, T. and Baldwin, G.S., 2015. *BASIC: a new biopart assembly standard for idempotent cloning provides accurate, single-tier DNA assembly for synthetic biology*. *ACS synthetic biology*, 4(7), pp.781-787.

Automated creation of temporal properties from protein interaction data

Emilee Holtzapple
University of Pittsburgh
3700 O'Hara St
Pittsburgh, PA 15213
erh87@pitt.edu

Cheryl Telmer
Carnegie Mellon University
4400 Fifth Ave.
Pittsburgh, PA 15213
ctelmer@cmu.edu

Natasa Miskov-Zivanov
University of Pittsburgh
3700 O'Hara St
Pittsburgh PA 15213
nmzivanov@pitt.edu

ABSTRACT

Models of biological systems are usually validated manually, through very labor-intensive comparisons with published experimental observations making the process slow and imprecise. We propose a tool to automate the creation of temporal logical properties from data. These properties can represent steady states or transient changes of individual model elements, or relative values of different model elements. Automation of property creation from data can significantly speed up and improve both model testing and extension to create a better understanding and explanations of biological behaviors.

CCS Concepts

• **Applied computing → Life and medical sciences → Computational biology → Biological networks**

Keywords

System properties; signaling cascades; protein interactions; model validation.

1. INTRODUCTION

The creation of *in silico* models of signaling cascades can significantly aid in biomedical research. Not only can these models reduce the expense and time of using *in vivo* disease models, but they can also utilize the vast amount of publicly available biological data. However, testing computational models can also be time-consuming. We propose an automated pipeline that can assist in validating models, and testing additions to the model. (Figure 1)

2. RECRUITMENT OF HIGH-CONFIDENCE INTERACTION DATA

To check established model output and recruit new model elements, high-confidence data must be gathered for all interactions. The Search Tool for the Retrieval of Interacting Genes/Proteins (STRING)[1] stores several different types of data on protein-protein or protein-gene interactions, including co-expression, homology, and experimental data. The STRING API can be utilized to quickly mine interaction data. A Python script was written to handle API requests and return the output (Figure 2).

Existing Model Elements	New Elements
JUN	MAPK9
ATF3	SMAD3
MAPK8	

ATF3	SMAD3	score:0.971|escore:0.36|dscore:0.9|tscore:0.589
JUN	SMAD3	score:0.989|escore:0.396|dscore:0.9|tscore:0.837
MAPK8	SMAD3	score:0.867|ascore:0.07|escore:0.387|tscore:0.78
MAPK9	JUN	score:0.998|escore:0.662|dscore:0.9|tscore:0.96
MAPK9	ATF3	score:0.456|escore:0.178|tscore:0.366
MAPK9	MAPK8	score:0.986|pscore:0.008432|escore:0.864|dscore:0.9|tscore:0.015328

Figure 1: Example network (top) with sample STRING output (bottom). The overall score is composed of several different subscores - experimental (escore), textming (tscore), phyletic (pscore), database (dscore), and coexpression (tscore).

2.1 Existing models

To use the Python script to verify a model, all existing model elements are compared for interactions through STRING. The output, a list of interactions and the associated STRING confidence score, will then be used to create properties for model validation. The output is a list of interactions with a total score above a specified threshold. The total scores range from 0 (least confident) to 1 (most confident). The total score is composed of seven different subscores. However, the relevant subscore is the experimental score (escore), which estimates the confidence of a direct interaction between two objects by methods such as a Western blot. By only considering the escore, all indirect interactions (such as coexpression) can be discarded.

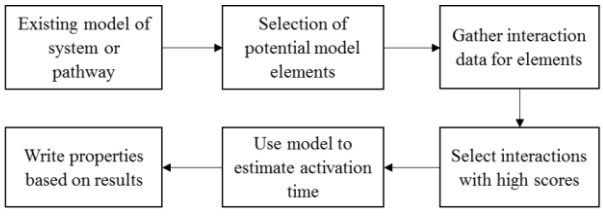


Figure 2: The automated pipeline for the creation of system properties to extend an existing model.

2.2 New models

To add new elements to an existing model, experimental data on any possible interactions is needed. The potential model elements can come from a variety of sources: differentially expressed genes (DEGs), biomedical literature, or other experimentally-based methods. These potential model elements are then cross-referenced with every existing model element in STRING. Based on the e-score, new model elements can be selected. This step eliminates the need for extensive knowledge of the biological signaling pathways to identify new elements.

3. PROPERTY CREATION

After selection of interactions with high e-scores, properties can be created that describe the behavior of the element within the model. For example, a property indicating that protein Jun reaches high levels early after stimulation start and remains high for long period of time can be written as follows:

F[100]G[900](Jun=HIGH).

where F[100] indicates that Jun reaches HIGH level by step 100 during simulation, and G[900] indicates that Jun remains at high level for 900 steps.[2]

For model extension, the new elements can be added, using the time and expression data to accurately describe its behavior. For model verification, the final state of each element should be compared to the existing model. For either method, the final output is a list of properties that describes the model. By using biological data to automate model validation, the speed and accuracy of this process can be improved.

4. MODEL VALIDATION

Creation of system properties for model validation should be done independently of the existing model framework. To do so, all data used for property creation should come from experimental data. When possible, time-course data should be used to inform property creation. However, the emphasis should be on the final state of the object.

5. MODEL EXTENSION

To estimate the timing of model element activation or deactivation, the existing model framework can be used. By identifying which element is downstream, the activation time of the upstream regulator can be used. When the upstream regulator has reached a steady state in the model, that time point can be marked as the point of activation for the downstream element. STRING can also be used to determine the direction of action between the two objects, and to provide experimental data that may include time-courses

6. CONCLUSION

Using a system of temporal logical properties to represent cellular signaling networks can be utilized for model extension or validation. Furthermore, automating the creation of these properties allows for fast testing of built models, and can significantly speed up the process of identifying models that best represent real systems.

Our next steps will include additional methods to increase accuracy and speed of property creation. These approaches will also incorporate a function to estimate direction of action between two objects based on their biological functions. For example, the direction of action between a phosphorylated protein and a phosphatase can be reasonably estimated.

7. ACKNOWLEDGMENTS

Thanks to Adam Butchy and Khaled Sayed for helpful discussions.

8. REFERENCES

[1] Szklarczyk, D., et al. (2015). "STRING v10: protein-protein interaction networks, integrated over the tree of life." *Nucleic Acids Res 43*(Database issue): D447-452.

[2] Miskov-Zivanov, N., et al. (2016). High-level modeling and verification of cellular signaling. 2016 IEEE International High Level Design Validation and Test Workshop (HLDVT).

Automated Portable Microfluidic Bioreactor for Synthetic Biology

Extended Abstract†

John R. Lake
Carnegie Mellon University
300 Technology Drive
Pittsburgh, USA
jol85@pitt.edu

Keith C. Heyde
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, USA
kheyde@andrew.cmu.edu

Warren C. Ruder
University of Pittsburgh
300 Technology Drive
Pittsburgh, USA
warrenr@pitt.edu

ABSTRACT
Due to its ease-of-use, microfluidics has become a ubiquitous tool for researchers within the field of synthetic biology. With the ability to optically image cells reliably, microfluidics has allowed researchers to take cellular-level measurements at a low cost. However, despite the low cost for building microfluidic devices, the peripheral equipment required for fluidic control are expensive and difficult to customize. The result is that although microfluidic tools are empowering for scientific advancement, controlling experiments accurately and with a variety of inputs and conditions remains challenging. Here, we present a set of low-cost, open-source tools that enable researchers to design well-controlled, customized microfluidic experiments at a low cost. We do this by leveraging additive manufacturing and commonly available 3D printing technologies. We have built a suite of microfluidic peripheral systems that include a fluid handling system, a temperature controller, and an optical monitoring device. These systems form a low-cost, feedback-controlled, automated microfluidic bioreactor in which living cells may be cultured and analyzed in real-time. It is our objective to use these open-source components to automate the process of cell culture, phenotypic selection, and epigenetic actuation. We expect our system to affect the fields of systems and synthetic biology.

CCS CONCEPTS
• **Applied computing → Systems biology**

KEYWORDS
Automated Bioreactor, Microfluidics, Synthetic Biology, Additive Manufacturing

ACM Reference format:
K.C. Heyde, J.R. Lake, and W.C. Ruder. 2017. IWBDA 2017 Conference Abstract.

1. INTRODUCTION
The facile approach for using well-established photolithography techniques[1] to design and fabricate microfluidic devices has enabled research across many scientific disciplines, particularly within bioengineering and biomedical research[1-5]. Organ-on-a-chip systems promise to revolutionize the investigation of underlying biological mechanisms and allow for simplified testing of the efficacy of various treatments of disease[6, 7]. Microfluidic systems have also been employed to enable automated synthetic biology assembly techniques to occur within a small package, like Golden Gate and Gibson assembly techniques[8]. Additionally, microfluidics has been used as a means for culturing bacteria and other living cells for live-cell imaging to observe the resulting behavior caused by synthetic gene circuits[9]. However, peripheral systems required for liquid handling, temperature regulation, and imaging are large and expensive, limiting the potential impact of microfluidic technology. Here we detail a set of tools that may be used to lower the overall cost of establishing a microfluidic experimental system. We expect these tools will be empowering for researchers looking to establish fast, customizable workspaces and for researchers looking to run low-cost microfluidic studies in parallel with their existing experimental infrastructure.

2. EXPERIMENTAL METHODS
2.1 Additive Manufacturing
As 3D printing technologies have become more widely adopted, 3D printing systems have become less expensive and easier to use. For the design of the majority of the physical components for the mechanical pumps and miniaturized microscope developed, we have employed 3D printing wherever possible. This approach both allows for rapid prototyping of our designs and enables these design to be freely shared with other researchers for their own use.

2.2 Live-cell Fluorescence Microscopy
Living cells are loaded and sustained by pumping culture media through the microfluidic channels of our designed device (Figure 1). In order to automate this process, we developed a custom fluidic pump and pressure sensor system that enable cell culturing. Our automated system provides well-regulated flow ensuring a constant stream of nutrients is delivered to the entrapped cell culture.

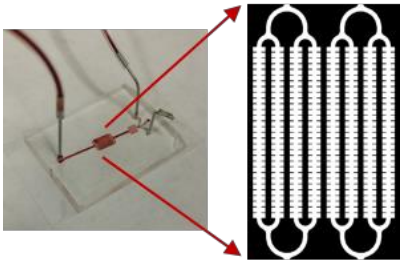


Figure 1: The microfluidic device used for testing. *E. coli* bacteria are confined in traps adjacent to main flow channels.

We are able to evaluate the efficacy of our flow-control system by entrapping synthetically engineered *E. coli* bacteria within our microfluidic channel. These cells were engineered to contain a genetic toggle switch[10], as well as other simple synthetically engineered gene regulatory networks. By examining fluorescent proteins under an epifluourescence microscope, we were able to regularly measure cellular response to influence media, while evaluating the pressure control profile of our liquid handling device. Crucially, by coupling multiple syringe pump actuators to a single microfluidic device, we were able to well-regulate the influent chemical compositions. This is important as it allowed us

to selectively induce populations of entrapped bacteria, modifying their epigenetic states using our fluidic handling system.

Our system can sense and control the pressure of incoming fluid, the system temperature, and the optical states of cells within our bioreactor. These processes can be handled by software, allowing automated culturing and experimentation.

3. RESULTS AND DISCUSSION

3.1 Automating Synthetic Biology Testing

The automated bioreactor system presented here offers a novel approach for experimentation with living cells that have been synthetically engineered. The separate subsystems developed here can also be more widely used outside of synthetic biology experimentation, to include many microfluidics research projects. For instance, the syringe pressure pumps developed here (Figure 2) allow for a low-cost and stable means of providing pressure driven flow for any microfluidic device. These pumps provide two modes of operation and can provide relatively high performance microfluidic flow control for a range of applications for a cost of approximately $110[11].

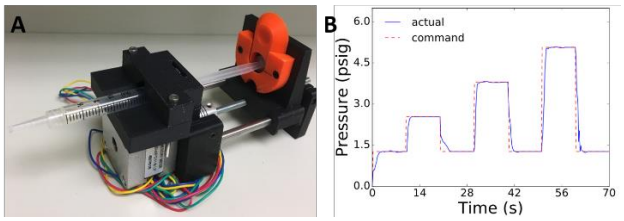


Figure 2: A) Syringe pressure pump designed for driving flow in microfluidic devices. B) Data collected shows our pressure pumps can accurately control pressure within a microfluidic device relative to a command over time.

Additionally, the miniaturized epifluorescent microscope being developed (Figure 3) provides a means of measuring the fluorescent reporter response from synthetically engineered cells confined in a microfluidic chip. By integrating off-the-shelf optical components with a 3D-printed housing, we have been able to design and construct a low-cost epifluorescent microscope, capable of imaging the fluorescent response of the synthetically engineered *E. coli* bacteria we used for testing.

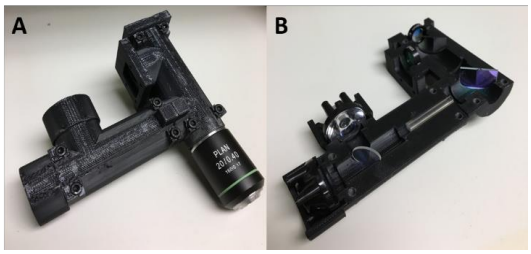


Figure 3: A) Miniaturized microscope body and objective lens for epifluorescent microscopy. B) Interior of the body houses the optical components.

3.2 Portable Microfluidic Bioreactor

Experimentation was focused on extended duration tests to ensure cells can be sustained for prolonged periods of many days at a time. As the onboard cells here are genetically modified organisms (GMOs), the closed system also isolates these GMOs from the surrounding environment. This can allow for the use of synthetically engineered living systems outside of the laboratory, for example, as a field-deployed biosensor or for metabolic optimization. Additionally, the portability of this system could be used for educational purposes, allowing for demonstrations of synthetic circuits in real-time for students in classrooms without access to lab spaces.

4. CONCLUSION

Our low-cost, open-source microfluidic bioreactor allows scientists and engineers to culture and examine synthetically engineered living organisms at a fraction of the cost of the typical lab set-up. We expect our system to enable both systems and synthetic biologists by driving down the cost of experimentation while enabling customization and portability. We look forward to improving the automation and resiliency of our system and deploying it for more comprehensive experiments.

5. ACKNOWLEDGMENTS

The authors acknowledge funding from Office of Naval Research award N00014-17-12306 and from the U.S. National Science Foundation Graduate Research Fellowship Program.

6. REFERENCES

1. Sia, S.K. and G.M. Whitesides, *Microfluidic devices* fabricated *in poly (dimethylsiloxane) for biological studies.* Electrophoresis, 2003. **24**(21): p. 3563-3576.
2. Mondragón-Palomino, O., et al., *Entrainment of a* population *of synthetic genetic oscillators.* Science, 2011. **333**(6047): p. 1315-1319.
3. Clausell-Tormos, J., et al., *Droplet-based microfluidic platforms for the encapsulation and screening of mammalian cells and multicellular organisms.* Chemistry & biology, 2008. **15**(5): p. 427-437.
4. Ashraf, M.W., S. Tayyaba, and N. Afzulpurkar, *Micro electromechanical systems (MEMS) based microfluidic devices for biomedical applications.* International journal of molecular sciences, 2011. **12**(6): p. 3648-3704.
5. Kim, Y. and R. Langer, *Microfluidics in Nanomedicine.* Reviews in Cell Biology and Molecular Medicine, 2015.
6. Bhatia, S.N. and D.E. Ingber, *Microfluidic* organs-*on-chips.* Nat Biotech, 2014. **32**(8): p. 760-772.
7. Samiei, E., M. Tabrizian, and M. Hoorfar, *A review of digital microfluidics as portable platforms for lab-on a-chip applications.* Lab on a Chip, 2016.
8. Shih, S.C., et al., *A versatile microfluidic device for automating synthetic biology.* ACS synthetic biology, 2015. **4**(10): p. 1151-1164.
9. Prindle, A., et al., *A sensing array of radically coupled genetic /*`biopixels/'. Nature, 2012. **481**(7379): p. 39-44.
10. Gardner, T.S., C.R. Cantor, and J.J. Collins, *Construction of a genetic toggle switch in Escherichia coli.* Nature, 2000. **403**(6767): p. 339-342.
11. Lake, J.R., K.C. Heyde, and W.C. Ruder, *Low-Cost Feedback-Controlled Syringe Pressure Pumps for* Microfluidics *Applications.* PLoS ONE, In Press.

Design Automation based on Fluid Dynamics

Ali Lashkaripour[1,2], Radhakrishna Sanka[2,3], Joshua Lippai[2,3] and Douglas Densmore[2,3]

[1]Department of Biomedical Engineering, Boston University, Boston, MA
[2]Biological Design Center, Boston University, Boston, MA
[3]Department of Electrical & Computer Engineering, Boston University, Boston, MA
{lashkari,sanka,jlippai,dougd}@bu.edu

1. INTRODUCTION

Microfluidic devices provide researchers with numerous advantages such as high throughput, increased sensitivity and accuracy, lower cost, and reduced reaction time [13, 11, 6]. However, design, fabrication, and running a microfluidic device are still heavily reliant on expertise [2, 1, 8]. Recent studies suggest micro-milling can be a semi-automatic, inexpensive, and simple alternative to common fabrication methods [3]. Micro-milling does not require a clean-room, mask aligner, spin-coater, and Plasma bonder, thus cutting down the cost and time of fabrication significantly. Moreover, through this protocol researchers can easily fabricate microfluidic devices in an automated fashion eschewing levels of expertise required for typical fabrication methods, such as photolithography, soft-lithography, and etching [14]. However, designing a microfluidic chip that meets a certain set of requirements is still heavily dependent on a microfluidic expert, several days of simulation, and numerous experiments to reach the required performance. To address this, studies have reported random automated design of microfluidic devices based on numerical simulations for micro-mixing [12]. However, random design generation is heavily reliant on time-consuming simulations carried out beforehand, and is prone to error due to the accuracy limitations of the numerical method. On the other hand, by using micro-milling for ultra-fast and inexpensive fabrication of microfluidic devices and Taguchi design of experiments for state-space exploration of all of the geometric parameters, we are able to generate a database of geometries, flow rates, and flow properties required for a single primitive to carry out a specified microfluidic task.

In this work, we report a modular automated design tool for microfluidic devices. DAFD (Design Automation based on Fluid Dynamics) enables researchers to design a microfluidic chip component-by-component. DAFD will output geometric parameters (alongside with fluid and flow properties) that can be incorporated into a standardized design description of these microfluidic components [10], hence allowing the design layouts to be automatically generated for the whole chip using Fluigi [4] .

2. DAFD - DROPLET GENERATION

Droplet generation was chosen as the first candidate of study due to its superior control over sample volume and concentration (ideal for biological reactions), and the complex interdependence between geometry and fluid dynamics. The first step of characterizing a geometry for a given microfluidic primitive (droplet generator, mixer. etc.) starts with Taguchi design of experiments to minimize the number of required geometrical variations to explore the state-space [9]. Then each geometry will be tested for different fluid and flow properties. Once the data is gathered, it will be fed to an ANFIS (Adaptive Network-based Fuzzy Inference System) model to train a predictive model. Once trained, ANFIS is able to create predictive models based on experimental data through artificial intelligence and statistical analysis [5]. Finally, its predictive accuracy will be verified experimentally and fed to DAFD back-end as shown in Figure 2.

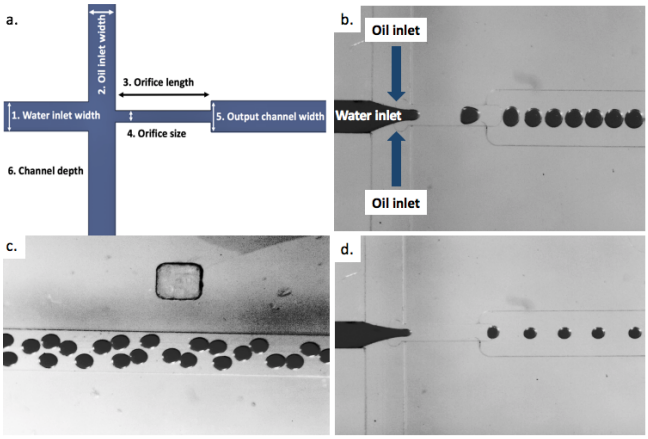


Figure 1: a. Geometric parameters of microfluidic cross-junction droplet generation, including water inlet width, oil inlet width, orifice length, orifice size, output channel width, and channel depth. b. Cross-junction microfluidic droplet generation. c. Droplets are counted and measured using image processing. d. Changes in geometry and flow rates results in altered droplet radius and generation rate.

The cross-junction flow-focusing droplet generator was used for the study due to its superior control over droplet size and its geometry that can be fully defined with six parameters as shown in Figure 1 [7]. For each parameter five different levels were considered; having six parameters in total , without Taguchi design of experiments, 15625 different geometries would have to be tested. However, through Taguchi design of experiments, this number will be reduced to 25. For each geometry, several water and oil flow rates will be examined to study their effect on droplet size and generation rate, and the data will be fed to DAFD database.

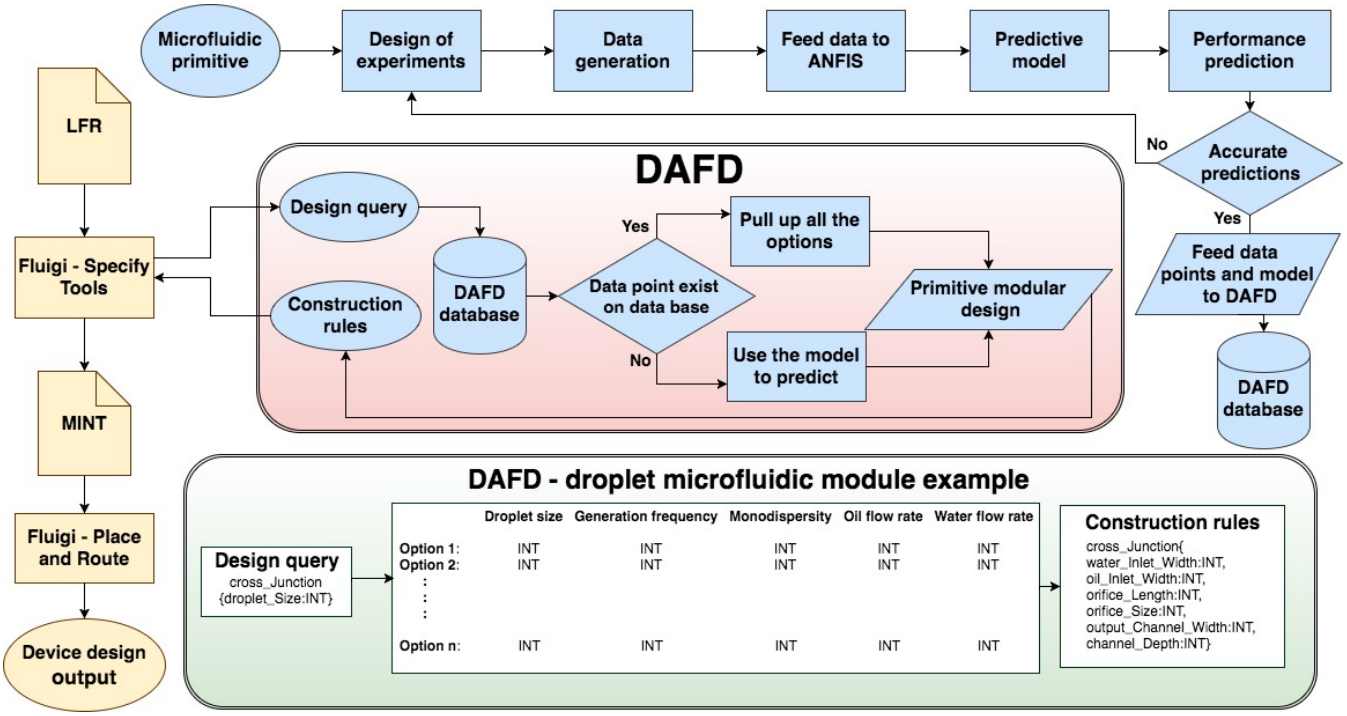


Figure 2: Yellow: Fluigi workflow for automated design of microfluidic devices, interface with DAFD. Blue: DAFD algorithmic approach for characterizing and modeling a microfluidic primitive into a database. Pink: Each design query will go through DAFD database to find an exact geometry or approximation based on predictive model. Green: An example of design query and construction rules for microfluidic cross-junction droplet generation. (*INT: Integer)

3. INTEGRATION WITH FLUIGI

DAFD lowers the barrier of entry into microfluidics by reducing the amount of prior knowledge necessary to design microfluidic devices. On the other hand, Fluigi as a new paradigm for physical design of microfluidic devices, is an automated CAD tool to generate the layout of the microfluidic device based on high-level description. Through the Fluigi workflow, the tool will send design queries to DAFD using the abstract Liquid Flow Relations (LFR) described by the user to design a microfluidic device. DAFD, in turn, will query its database for an exact geometry for the corresponding primitive; if one does not exist, it will use its predictive model to approximate the geometry for the described task as shown in Figure 2.

4. CONCLUSION AND FUTURE WORK

By considering the basic microfluidic primitives required to carry out common biological protocols in a microfluidic chip, such as droplet generators, micro-mixers, cell-traps, and gradient generators, and fully characterizing them to build a large enough database and an accurate predictive model, DAFD will enable researchers to design a functional microfluidic chip based on their needs without any prior microfluidic knowledge. We are also aiming to model the other mentioned microfluidic primitives in the future.

5. REFERENCES

[1] I. E. Araci and P. Brisk. Recent developments in microfluidic large scale integration. *Current opinion in biotechnology*, 25:60–68, 2014.

[2] D. T. Chiu, D. Di Carlo, P. S. Doyle, C. Hansen, R. M. Maceiczyk, R. C. Wootton, et al. Small but perfectly formed? successes, challenges, and opportunities for microfluidics in the chemical and biological sciences. *Chem*, 2(2):201–223, 2017.

[3] D. J. Guckenberger, T. E. de Groot, A. M. Wan, D. J. Beebe, and E. W. Young. Micromilling: a method for ultra-rapid prototyping of plastic microfluidic devices. *Lab on a Chip*, 15(11):2364–2378, 2015.

[4] H. Huang. *Fluigi: An end-to-end Software Workflow for Microfluidic Design*. PhD thesis, Boston University, 2015.

[5] J.-S. Jang. Anfis: adaptive-network-based fuzzy inference system. *IEEE transactions on systems, man, and cybernetics*, 23(3):665–685, 1993.

[6] A. Lashkaripour, A. Abouei Mehrizi, M. Rasouli, and M. Goharimanesh. Numerical study of droplet generation process in a microfluidic flow focusing. *Journal of Computational Applied Mechanics*, 46(2):167–175, 2015.

[7] W. Lee, L. M. Walker, and S. L. Anna. Role of geometry and fluid properties in droplet and thread formation processes in planar flow focusing. *Physics of Fluids*, 21(3):032103, 2009.

[8] M. Rasouli, A. Abouei Mehrizi, and A. Lashkaripour. Numerical study on low reynolds mixing oft-shaped micro-mixers with obstacles. *Transport Phenomena in Nano and Micro Scales*, 3(2):68–76, 2015.

[9] R. K. Roy. *A primer on the Taguchi method*. Society of Manufacturing Engineers, 2010.

[10] R. Sanka, H. Huang, R. Silva, and D. Densmore. Mint - microfluidic netlist. poster presented at IWBDA 2016, Aug. 2016.

[11] T. M. Squires and S. R. Quake. Microfluidics: Fluid physics at the nanoliter scale. *Reviews of modern physics*, 77(3):977, 2005.

[12] J. Wang, P. Brisk, and W. H. Grover. Random design of microfluidics. *Lab on a Chip*, 16(21):4212–4219, 2016.

[13] G. M. Whitesides. The origins and the future of microfluidics. *Nature*, 442(7101):368–373, 2006.

[14] D. P. Yen, Y. Ando, and K. Shen. A cost-effective micromilling platform for rapid prototyping of microdevices. *Technology*, pages 1–6, 2016.

Discrete Modeling of the JAK2/STAT5 Signaling pathway to Determine Important Negative Feedback Mechanisms

Adam A Butchy[1,*], Alexandre Terrier[2,*], Cheryl Telmer[3], James Faeder[4], and Natasa Miskov-Zivanov[2,4]

[1]Department of Bioengineering, University of Pittsburgh, PA
[2]Department of Electrical and Computer Engineering, University of Pittsburgh, PA
[3]Biological Sciences, Carnegie Mellon, PA
[4]Computational and Systems Biology, University of Pittsburgh, PA
*These authors contributed equally to this work
Contact: nmzivanov@pitt.edu

1. Introduction

The JAK (Janus Kinase) and STAT (Signal Transducers and Activators of Transcription) families of proteins are a large group of signaling molecules that are involved in many important signaling pathways in cells. There are at least 3 main types of JAK proteins and 6 different STATs and together, they are responsible for activating some of the most important cellular functions such as cell proliferation, differentiation, migration and apoptosis [1,2,3].

The JAK2/STAT5 pathway (simplified and shown in Figure 1) is incredibly important and known to play a vital role in macrophage activation during the immune response [4], and cell differentiation [5]. However, despite its importance, it is not well understood.

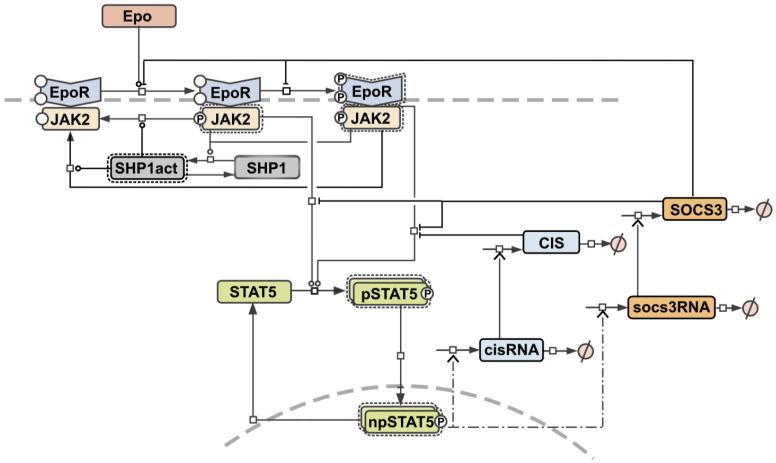


Figure 1: A simplified version of the JAK2/STAT5 signaling pathway. This contains all the key elements necessary for STAT5 phosphorylation and transcription of the downstream proteins SOCS3 and CIS. Specifically, how CIS and SOCS3 work in the negative feedback loop is not well understood. This figure is from [6].

We focus on elucidating the JAK2/STAT5 signaling pathway through comparing model simulations against experimental results. ***The theory behind this is that a model that is able to most accurately match the experimental results must be the model that most closely resembles the system in cells.*** For this, we are expanding on the work of J. Bachmann et al. and their mathematical model of this pathway [6]. The model developed by J. Bachmann et al. was able to fit the experimental data they collected but they were unable to determine if each regulation they modeled was present in the true biological system. Indeed, they hypothesized that some of their negative feedback regulations were not actually present in the native reaction pathway and that their model could be further simplified. ***In this same vein, we hypothesized that we could simplify their mathematical model using Boolean Modeling and reduce the number of negative feedback regulations while accurately matching their experimental results.***

2. Discrete Modeling

In discrete modeling, elements are only allowed to have discrete values. Interactions between elements in the model are described using a simple notation which is translated into a series of logical rules. Once the initial values have been set for each element in our model, our simulator is able to simulate the model by computing the next value of an element by only knowing it's previous value and the previous values of the elements that interact with it. This is done until the model reaches steady state (i.e. no element's value changes, a cycle is reached, or a specified number of time steps have finished).

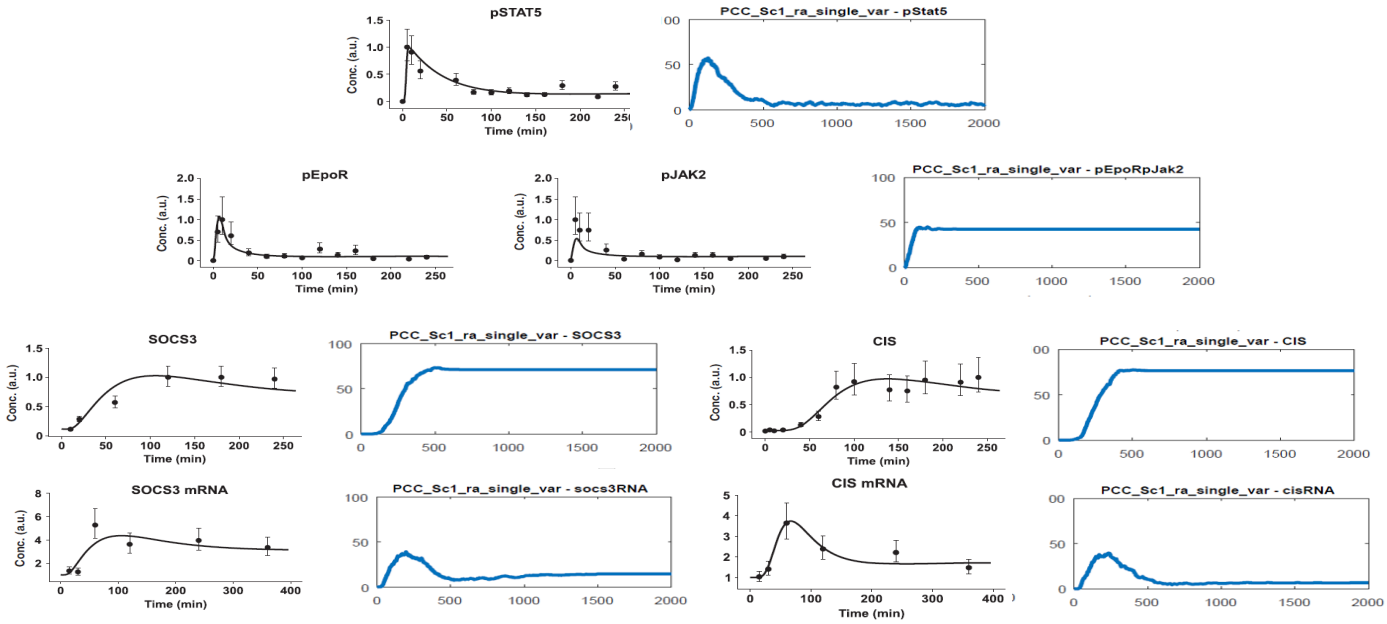


Figure 2: Comparison between experimental results and our simulation's results. In black on the left are the results for pStat5, pEpoR, pJak2, SOCS3, CIS, SOCS3mRNA, CISmRNA. On the right, in blue are the corresponding results produced by our model.

In order to observe dynamic time-course data from these experiments, an asynchronous model must be simulated numerous times and the values of each element to be saved over the course of each run. In this way the elements' levels over the number of trials can be summed and a dynamic graph can show the maximum expression of all trials normalized to 100% activity.

3. Simulation

We were able to create a baseline Boolean model based on the Jak2/Stat5 mathematical model. Once we have established a baseline Boolean model that matches the behavior of the experiment and mathematical model, our goal was to test which inhibitions are vital to the system, and which ones are redundant. We began systematically removing negative feedback regulation of SOCS3 and CIS on their upstream regulators and comparing the new model against the baseline. When the removal of a regulation compromised the accuracy of the model, it was ruled invalid and the next model was created

4. Conclusion

We set out to determine if the feedback regulation present in the JAK2/STAT5 signaling pathway could be simplified from a preliminary mathematical model [6]. We were able to identify 3 negative feedback regulations present in the mathematical model that were not necessary for the accuracy of the discrete model. These regulations were redundant in nature for two different reasons: either they regulated the receptor that began the pathway (a feedback mechanism that is exceedingly rare and therefore not likely to be present in this system) or because there were two negative feedback regulations that were essentially identical (due to the fact that the negative regulators were sharing the same behavior). We can conclude that our Boolean model is a simpler and more elegant model that accurately represents the behavior of the JAK2/STAT5 signaling pathway.

5. References

[1] J. S. Rawlings, K. M. Rosler, and D. A. Harrison, "The JAK/STAT signaling pathway," *J. Cell Sci.*, vol. 117, no. 8, 2004.
[2] D. S. Aaronson and C. M. Horvath, "A Road Map for Those Who Don't Know JAK-STAT," *Science (80-.).*, vol. 296, no. 5573, 2002.
[3] J. J. O'Shea, M. Gadina, and R. D. Schreiber, "Cytokine Signaling in 2002," *Cell*, vol. 109, no. 2, pp. S121–S131, Apr. 2002.
[4] A. L. Mui, H. Wakao, A. M. O'Farrell, N. Harada, and A. Miyajima, "Interleukin-3, granulocyte-macrophage colony stimulating factor and interleukin-5 transduce signals through two STAT5 homologs.," EMBO J., vol. 14, no. 6, pp. 1166–75, Mar. 1995.
[5] M. Azam, C. Lee, I. Strehlow, and C. Schindler, "Functionally Distinct Isoforms of STAT5 Are Generated by Protein Processing," Immunity, vol. 6, no. 6, pp. 691–701, 1997.
[6] J. Bachmann *et al.*, "Division of labor by dual feedback regulators controls JAK2/STAT5 signaling over broad ligand range," *Mol. Syst. Biol.*, vol. 7, no. 1, pp. 516–516, Apr. 2014

Examining the thermodynamic basis of differential gene expression *in vitro*

Extended Abstract

Grace Vezeau*
The Pennsylvania State University
gev107@psu.edu

Howard M. Salis†
The Pennsylvania State University
salis@psu.edu

ABSTRACT

Cell-free expression systems have been used, since their inception, as test platforms to investigate biological phenomena. Today, synthetic biologists use coupled transcription-translation (TX-TL) cell-free systems as a quick, simple, and high-throughput method to prototype biological parts and pathways. Here, we quantify the differences in the biophysics of gene expression between the *in vitro* and *in vivo* environments. Increases in the concentration of crowding agents in vitro, and the concentration of salts, increase overall gene expression *in vitro*, while decreasing the proportional range of expression between low- and high-translation rate constructs. We relate this difference to the apparent temperature, β, and the effect of crowding agents and salts on this thermodynamic parameter. The difference in β between the *in vivo* and *in vitro* environments presents a challenge for translating the performance of biological parts and systems prototyped *in vitro* to *in vivo* applications.

KEYWORDS

Biophysical modeling, cell-free expression, macromolecular crowding

ACM Reference format:
Grace Vezeau and Howard M. Salis. 1997. Examining the thermodynamic basis of differential gene expression *in vitro*. In *Proceedings of the International Workshop on Bio-Design Automation, Pittsburgh, PA USA, August 2017 (IWBDA)*, 2 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Cell-free expression systems (CFSs) are a powerful tool to characterize biological phenomena. By purifying the cellular gene expression machinery, and combining amino acids, NTPs, and small-molecule cofactors, the intracellular process of gene expression can be recapitulated. CFSs allow researchers the freedom of working with biological parts and machinery in an open system, making perturbations to that system far simpler than if they were working with a living cell.

As the field of synthetic biology has emerged, it has embraced the use of TX-TL CFSs as a flexible platform which can be used to build circuits, produce bioproducts, and sense pathogens [7, 10, 13]. By bringing the time-to-experiment for testing new genetic parts down to several days from as long as several weeks, TX-TL-based assays can accelerate the design-build-test cycle, while greatly remediating any issues of toxicity related to overexpression [6].

In particular, synthetic biologists have recently begun using TX-TL as a molecular biology "breadboard" to prototype genetic parts, circuits, and even entire pathways. Regulatory sequences, such as promoters and ribosome binding sites (RBSs), have been prototyped *in vitro* in both both *E. coli* and *Bacillus subtilis*-derived cell-free systems [2, 8]. A variety of circuits, ranging from repressilators to phage integrase circuits, have been tested in TX-TL systems as well [1, 9]. Even pathways, such as those synthesizing valinomycin and butanol, have been successfully prototyped in TX-TL [7, 14].

However, *in vitro* results may not directly correspond to *in vivo* behavior. Here, we use a biophysical model of translation initiation to demonstrate that the relative differences in gene expression in different contexts can be related to differences in the thermodynamic β of the system they are tested in, and that tuning the concentration of macromolecular crowding agents leads to changes in β [11].

2 β *IN VIVO* AND *IN VITRO*

The rate of protein production of a construct is proportional to the Gibbs free energy of translation initiation of that construct, according to the following:

$$r = Ke^{-\beta \Delta G_{total}} \quad (1)$$

Where K is a constant representing the effect of mRNA level, reporter quantum yield, total 30S ribosomal subunit concentration, and initiation at alternative ribosome binding sites. ΔG_{total} is calculated using a biophysical model of translation initiation [4]. β inside the bacterial cell has previously been measured to be approximately 0.45, or 1120 °C [4, 5, 11].

Figure 1 shows the fit of β to a dataset of 169 individually characterized sequences [4]. We chose seven constructs, previously observed to be well-predicted by RBS Calculator 2.0, to determine the effects of macromolecular crowding on the rate of translation initiation *in vitro* [4].

*Department of Agricultural and Biological Engineering
†Department of Agricultural and Biological Engineering, Department of Chemical Engineering

IWBDA, August 2017, Pittsburgh, PA USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

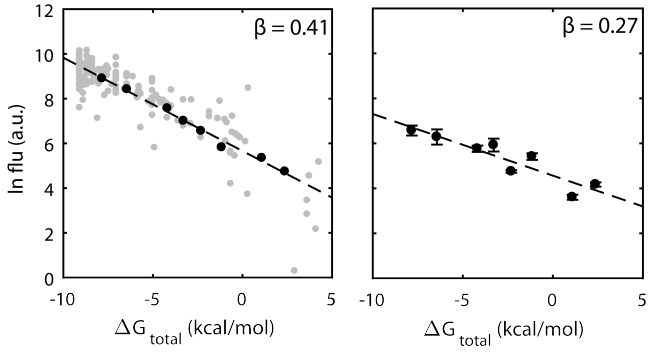


Figure 1: β *in vivo* and *in vitro*. Fluorescence of 169 sequences characterized *in vivo* vs. ΔG_{total} are shown on the left, in gray. Sequences in black were chosen for further characterization *in vitro*. Fluorescence vs. ΔG_{total} in TX-TL is shown on the right.

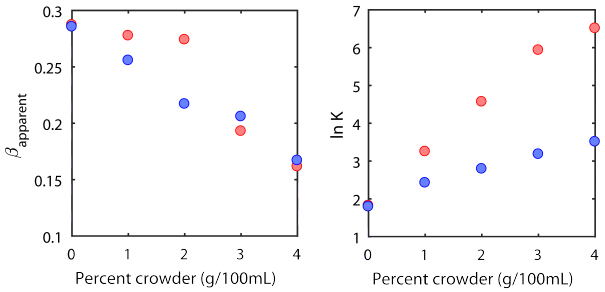


Figure 2: Effect of crowders on *ln* K and β. Apparent β of constructs tested in varying % PEG-8000 (red) and Ficoll400 (blue) are shown on the left. *ln* K of constructs tested in varying % PEG-8000 (red) and Ficoll400 (blue) are shown on the right.

A cell-free expression system was prepared as described by [12]. 5 uL reactions, with 2 nM plasmid DNA template, were incubated at 29°C. Constructs expressing mRFP1 from the J23100 promoter, with designed RBSs, were screened [4]. The steady-state mRFP1 fluorescence level for each treatment, a measure of the total fluorescent protein production, was used to quantify translation initiation rate.

Initial screens of these same constructs in TX-TL showed that the value of β *in vitro* differs significantly from the value obtained *in vivo*. Despite the fact that the *in vitro* environment is 30-fold more dilute than the cellular environment, the apparent β is lower.

3 EFFECT OF CROWDERS ON PARAMETERS OF GENE EXPRESSION

Macromolecular crowding is known to affect the equilibria of biochemical reactions and interactions due to excluded volume effects [3]. To examine the role that macromolecular crowding has on the translation rate and β of our selected constructs, we added differing amounts of two neutral crowding agents, PEG-8000 and Ficoll 400, to TX-TL. Examining K and β separately allows the effect of crowding on transcription and translation, respectively, to be determined

Adding either crowder resulted in increases in gene expression for all constructs tested. As shown in Figure 2, β consistently decreased with increased % (g/100mL) of crowder added, while K increased. The effects of both crowders on β, and therefore translation initiation, were similar. PEG-8000, however, increased transcription to a much greater degree than did Ficoll400.

4 CONCLUSIONS

Although the behavior of genetic parts *in vitro* is qualitatively similar to their behavior *in vivo*, the thermodynamic β varies between the two systems. Changing the degree of crowding in TX-TL further decreases beta. Although TX-TL is a useful platform to prototype genetic parts and systems intended to be used *in vivo*, further studies are needed to determine how *in vitro* performance translates *in vivo*.

REFERENCES

[1] Georgios Artavanis, Victoria Hsiao, Clarmyra A Hayes, and Richard M Murray. 2016. The role of single occupancy effects on integrase dynamics in a cell-free system. *bioRxiv* (2016), 059675.

[2] James Chappell, Kirsten Jensen, and Paul S. Freemont. 2013. Validation of an entirely in vitro approach for rapid prototyping of DNA regulatory elements for synthetic biology. *Nucleic Acids Research* 41, 5 (2013), 3471–3481. https://doi.org/10.1093/nar/gkt052

[3] R John Ellis. 2001. Macromolecular crowding: obvious but underappreciated. *Trends in biochemical sciences* 26, 10 (2001), 597–604.

[4] Amin Espah Borujeni, Anirudh S. Channarasappa, and Howard M. Salis. 2014. Translation rate is controlled by coupled trade-offs between site accessibility, selective RNA unfolding and sliding at upstream standby sites. *Nucleic Acids Research* 42, 4 (2014), 2646–2659. https://doi.org/10.1093/nar/gkt1139

[5] Iman Farasat, Manish Kushwaha, Jason Collens, Michael Easterbrook, Matthew Guido, and Howard M Salis. 2014. Efficient search, mapping, and optimization of multi-protein genetic systems in diverse bacteria. *Molecular systems biology* 10 (2014), 731. https://doi.org/10.15252/msb.20134955

[6] Jonathan Garamella, Ryan Marshall, Mark Rustad, and Vincent Noireaux. 2016. The All E. coli TX-TL Toolbox 2.0: A Platform for Cell-Free Synthetic Biology. *ACS Synthetic Biology* 5, 4 (2016), 344–355. https://doi.org/10.1021/acssynbio.5b00296

[7] Ashty S. Karim and Michael C. Jewett. 2016. A cell-free framework for rapid biosynthetic pathway prototyping and enzyme discovery. *Metabolic Engineering* 36 (2016), 116–126. https://doi.org/10.1016/j.ymben.2016.03.002

[8] Richard Kelwick, Alexander J. Webb, James T. MacDonald, and Paul S. Freemont. 2016. Development of a Bacillus subtilis cell-free transcription-translation system for prototyping regulatory elements. *Metabolic Engineering* 38 (2016), 370–381. https://doi.org/10.1016/j.ymben.2016.09.008

[9] Henrike Niederholtmeyer, Zachary Z. Sun, Yutaka Hori, Enoch Yeung, Amanda Verpoorte, Richard M. Murray, and Sebastian J. Maerkl. 2015. Rapid cell-free forward engineering of novel genetic ring oscillators. *eLife* 4, OCTOBER2015 (2015). https://doi.org/10.7554/eLife.09771

[10] Keith Pardee, Alexander A. Green, Tom Ferrante, D. Ewen Cameron, Ajay Daleykeyser, Peng Yin, and James J. Collins. 2014. Paper-based synthetic gene networks. *Cell* 159, 4 (2014), 940–954. https://doi.org/10.1016/j.cell.2014.10.004

[11] Howard M Salis, Ethan A Mirsky, and Christopher A Voigt. 2009. Automated design of synthetic ribosome binding sites to control protein expression. *Nature biotechnology* 27, 10 (2009), 946–50. https://doi.org/10.1038/nbt.1568

[12] Zachary Z Sun, Clarmyra A Hayes, Jonghyeon Shin, Filippo Caschera, Richard M Murray, and Vincent Noireaux. 2013. Protocols for implementing an Escherichia coli based TX-TL cell-free expression system for synthetic biology. *Journal of visualized experiments : JoVE* (2013), e50762. https://doi.org/10.3791/50762

[13] Melissa K. Takahashi, Clarmyra a. Hayes, James Chappell, Zachary Z. Sun, Richard M. Murray, Vincent Noireaux, and Julius B. Lucks. 2015. Characterizing and prototyping genetic networks with cell-free transcriptionâĂŞtranslation reactions. *Methods* (2015). https://doi.org/10.1016/j.ymeth.2015.05.020

[14] Tiffany Zhou. 2016. Prototyping a valinomycin biosynthesis pathway within a cell-free transcription-translation (TX-TL) system. *bioRxiv* (2016), 091520.

Fluigi Cloud

A Cloud CAD Platform for Microfluidics

Kestutis Subacius[1], Priya Kapadia[1], Shane McCormack[1], Anish Asthana[1], Radhakrishna Sanka[1], and Douglas Densmore[1]

[1]Department of Electrical and Computer Engineering, Boston University, Boston, MA

{kestas,priyak,aonanam,asthana,sanka,dougd}@bu.edu

1. INTRODUCTION

With microfluidic large scale integration [Thorsen et al. 2002] and the emergence of many new synthetic biology technologies, there is an ever increasing benefit in using computer automated design (CAD) tools for scaling designs to larger and more complex applications. In 2015 Xin Han et al. demonstrated the effective delivery of CRISPR-Cas9 [Cong et al. 2013] to cells, which are normally difficult to transfect, using a microfluidic membrane device [Han et al. 2015]. To help researchers and engineers realize microfluidics for new synthetic biology applications, it is pertinent that they have access to CAD tools to facilitate the design process. Fluigi Cloud is an online platform designed with this goal in mind. It provides a suite of software tools for microfluidic CAD. This work describes some applications of Fluigi Cloud and the role it plays in the greater ecosystem of microfluidic design and synthetic biology.

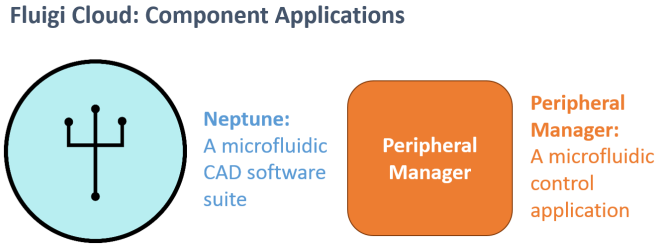


Figure 1: Components in the first release of Fluigi Cloud

As shown in Figure 1, the first release of Fluigi Cloud is made up of two components: Neptune and the Peripheral Manager. The application Neptune provides an environment for researchers to develop designs within the Fluigi Software Workflow [Huang 2015]. The Peripheral Manager provides an interface to control microfluidic hardware and experiments. This framework was designed with the goal of scalability in mind, with the software architecture built to incorporate future microfluidic CAD tools.

2. ARCHITECTURE

Fluigi Cloud is built to support three main functions critical to a successful CAD workflow:

1. A database and file system for microfluidics to be be designed on the Cloud.
2. A robust system to execute CAD jobs on the Cloud.
3. An interface that allows the microfluidic experiments to be controlled from the Cloud.

This architecture is outlined in Figure 2. Fluigi Cloud was developed within a NodeJS [Nod 2017] Express framework [Exp 2017]. The application is hosted on an AWS EC2 instance, and interfaces with the AWS S3 static file system [AWS 2017] for object storage. A MongoLab hosted database is used for data model storage in MongoDB [Mon 2017]. CAD jobs are executed on the EC2 instance, with the system built to support a modular and expandable number of new software tools. Currently, Fluigi Cloud supports the translation of LFR files to MINT files, and the compilation of MINT files into microfluidic schematics. Fluigi Cloud interfaces with a local application to control microfluidics that were created in the Cloud environment.

The rest of the abstract will discuss the two applications supported under Fluigi Cloud's first release: Neptune and the Peripheral Manager.

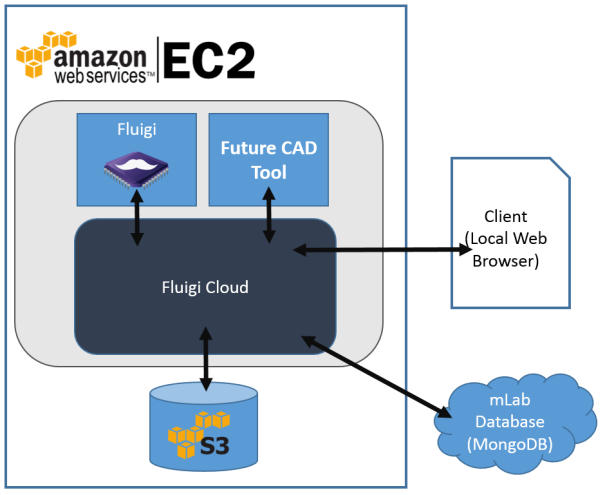


Figure 2: Architecture of Fluigi Cloud

3. NEPTUNE

Neptune is an application designed to incorporate the end-to-end microfluidic design procedure under the Fluigi Workflow. With Neptune, users can describe their microfluidic design idea in a high level specification called LFR (liquid

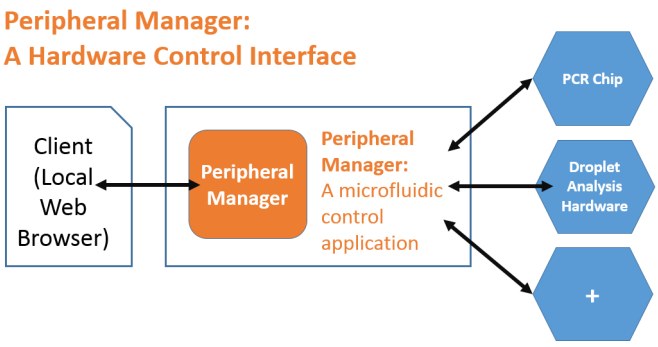


Figure 3: Peripheral Manager

flow relations), or in a more descriptive MINT (microfluidic netlist), specification [Sanka et al. 2016]. Neptune provides an environment where these descriptions can be written and then compiled into realizable design schematics with Fluigi. Hence, Neptune is the main CAD tool that generates microfluidic designs from high level descriptions. These design schematics can then be fabricated with photolithography, or with the MakerFluidics protocol [Silva et al. 2015].

4. PERIPHERAL MANAGER

The Peripheral Manager is the application that provides a link between Fluigi Cloud and the physical microfluidic device. This application is run as a server from the local client, and it was built to support connection from external application, (such as Fluigi Cloud) and to support connection to external devices (such as microcontrollers used to control microfluidic experiments). This application is developed in a NodeJS framework, and can be used both independently of and in conjunction with Fluigi Cloud. The function of the Peripheral Manager is illustrated in Figure 3.

5. FLUIGI CLOUD IN THE GREATER CONTEXT OF SYNTHETIC BIOLOGY

The greater vision of Fluigi Cloud is to provide a coherent and seamless suite of software tools to make microfluidic design and research easier. As Figure 4 illustrates, the goal of Fluigi Cloud is to incorporate all aspects of microfluidic design. This includes computer automated design tools to push the boundaries of design complexity, enable access to automated and low-cost fabrication protocols that can be adopted by many labs, and finally to provide control software to allow experiments to be run directly from the Cloud. This all places Fluigi Cloud as an ideal ecosystem for researchers and engineers to not only design novel devices, but also to collaborate and share their designs in a unified space. Future releases of Fluigi Cloud will aim to create this space, always with the goal of enabling synthetic biologists to push the frontier of what is possible in Microfluidics.

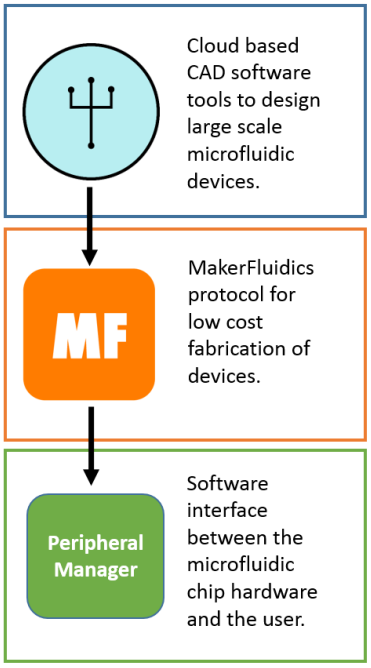


Figure 4: End-to-end microfluidic workflow

6. REFERENCES

[AWS 2017] 2017. Amazon Web Services, Simple Storage Service. (2017). https://aws.amazon.com/s3/

[Exp 2017] 2017. Express. (2017). https://expressjs.com/

[Mon 2017] 2017. MongoDB. (2017). https://www.mongodb.com/

[Nod 2017] 2017. Node JS. (2017). https://nodejs.org/

[Cong et al. 2013] Le Cong, F. Ann Ran, David Cox, Shuailiang Lin, Robert Barretto, Naomi Habib, Patrick D. Hsu, Xuebing Wu, Wenyan Jiang, Luciano A. Marraffini, and Feng Zhang. 2013. Multiplex Genome Engineering Using CRISPR/Cas Systems. *Science* 339, 6121 (2013), 819–823. DOI: http://dx.doi.org/10.1126/science.1231143

[Han et al. 2015] Xin Han, Zongbin Liu, Myeong chan Jo, Kai Zhang, Ying Li, Zihua Zeng, Nan Li, Youli Zu, and Lidong Qin. 2015. CRISPR-Cas9 delivery to hard-to-transfect cells via membrane deformation. *Science Advances* 1, 7 (2015). DOI: http://dx.doi.org/10.1126/sciadv.1500454

[Huang 2015] Haiyao Huang. 2015. *Fluigi: An end-to-end Software Workflow for Microfluidic Design.* Ph.D. Dissertation. Boston University. http://cidarlab.org/wp-content/uploads/2016/02/ch_2015_thesis.pdf

[Sanka et al. 2016] Radhakrishna Sanka, Haiyao Huang, Ryan Silva, and Douglas Densmore. 2016. MINT - Microfluidic Netlist. poster presented at IWBDA 2016. (Aug. 2016). http://cidarlab.org/wp-content/uploads/2016/09/MINT-IWBDA-2016-Poster-Template-copy.pdf

[Silva et al. 2015] Ryan Silva, Aaron Heuckroth, Cassie Huang, Aparna Rolfe, and Douglas Densmore. 2015. MakerFluidics: Microfluidics for all. Poster presented at Synberc: Fall 2015. (September 2015). http://cidarlab.org/wp-content/uploads/2016/02/Synberc_MakerFluidics_Poster_Silva.pdf

[Thorsen et al. 2002] Todd Thorsen, Sebastian J. Maerkl, and Stephen R. Quake. 2002. Microfluidic Large-Scale Integration. *Science* 298, 5593 (2002), 580–584. DOI: http://dx.doi.org/10.1126/science.1076996

A Novel Flexible Library for Representation of the Biological Systems in Computer Algorithms

Mehrshad Khosraviani

Dept. of Computer Engineering & IT, Amirkabir University of Technology, 424, Hafez Ave.
Tehran, Iran

mkhosraviani@aut.ac.ir

Sepehr Najjarpour

Fanap Co., No. 123, 12th Noavari St.,
Pardis Technology Park, 20th km of Damavand Road
Tehran, Iran

s.najjarpour@fanap.ir

ABSTRACT

Given recent developments in the graph-based approaches, used in synthetic and systems biology, and the variety types of the graphs employed by those methods, it is becoming more obvious that the requirements for developing a comprehensive graph library are being met. This paper focuses on providing a new graph library using a novel flexible datamodel for full representation of the biological systems with the support of their functional and topological properties.

CCS CONCEPTS

• **Software and its engineering → Software libraries and repositories;** *Software design engineering;* Software architectures**.** • **Applied computing → Biological networks;** *Bioinformatics*; Systems biology.

KEYWORDS

Design automation; biological systems; graph library; three-tier architecture, flexible datamodel.

1. INTRODUCTION

Beyond the production of a huge amount of biological data, there is a real challenge to represent and use the data, as they are given, in computer algorithms and tools developed for analysis and design-automation of the biological systems. Moreover, due to the complexity of relationships and the peculiarities of the data, the interpretation of most of biological systems is often difficult. However, graphs are the well-known mathematical abstractions which can be used to get insights into studying the topological properties of the biological systems and the functioning of them.

Given both pure and hybrid graph-based approaches currently, present fascinating challenges with respect to the functional and topological properties of the biological systems, we decided to provide and develop a new well-structured and procedurally-rich graph library.

Although some of the graph libraries, *e.g*., QuickGraph [1] and boost [2] provide generic graph data structures and algorithms for the different programming languages, it cannot be used for the efficient and flexible representation of the non-typical graphs, including bipartite graphs (Petri net graphs) [3], signed graphs [4], and hypergraphs (AND/OR graphs) [5][6]. Indeed, the computer algorithms suffer from the lack of a single comprehensive library to consider (non-)typical types of graphs.

Furthermore, in the NoSQL databases such as Neo4j, users face a fixed graph type, *i.e*., a normal OR graph, which must be changed by them before exploiting. By contrast, the relational database gives the program developers access to the flexible types of the graphs. In other words, using the graphs of any different types, not using an inflexible type, helps the users to efficiently solve a range of problems, especially in the field of synthetic and systems biology. For example, in Neo4j, due to the lack of some specific graph types like the AND/OR graphs, the users will be pushed for superficial implementation of these types of the graphs. Therefore, because of these shortcomings, we proposed a novel flexible datamodel through which access to the best graph-based solution of a variety of the problems in different biological systems will be possible.

These are precisely the motivations behind this work in which we give our solution (based on a novel flexible datamodel) to the aforementioned shortage of the conventional graph libraries, and indeed, it does, as we shall see in the following.

2. MATERIAL AND METHOD

2.1 Architecture

This section deals exclusively with architecture principles of the proposed datamodel of our graph library.

Due to the several reasons such as central security, easy access to the data, scalability and flexibility, our solution offered for the problem of designing a single comprehensive graph library is based on the three-tier architecture (Figure 1). This architecture is a data-intensive client-server by which an application is broken down into three logical functional-separate tiers (layers). Each of these has well-defined interfaces. However, to design and develop an efficient library, we focused on the only data and business tiers for the implementation and the future developments.

2.1.1 Data Tier

Data-tier of the architecture (of the proposed library) is one of the complete layers designed to store a variety of graph types. The design, performed with the optimal methodologies, has been resulted in meaningfulness during the data storage. Accordingly, no extra software is needed to resolve types of the recording data.

As a simple example, assume that $Rx: Ci + Cj \longrightarrow Ct$ and $Ry: Cj \longrightarrow Ct$ are two bioreactions of a given biochemical network. Figure 2A shows the hypergraph corresponding to these two bioreactions, and Figure 2B shows how the corresponding data are stored and managed in our proposed datamodel.

Figure 1. Flow diagram of the 3-Tiered client-server architecture

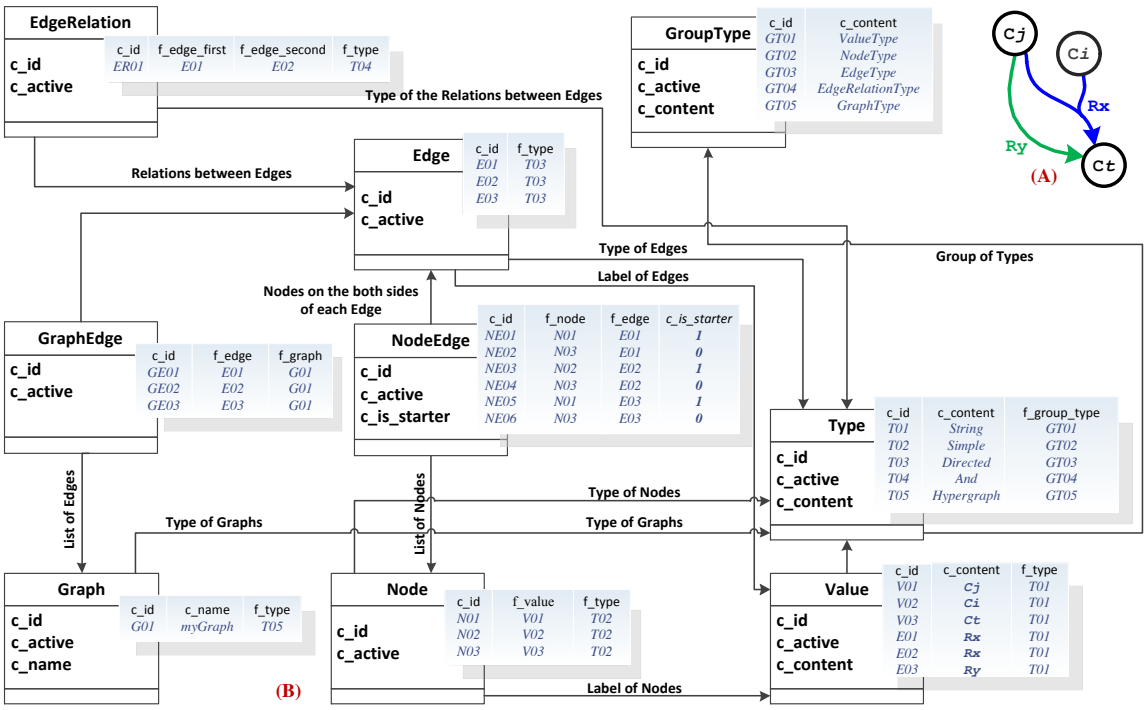


Figure 2. Overview of the datamodel of our proposed library: **(A)** An exemplary hypergraph, and **(B)** The original format of datamodel, plus the completed tables (entities) by considering the properties of the hypergraph.

Using the proposed datamodel, each type of the graphs can be saved in computer programs for the next utilizations. The steps include the following:

- Name and type of a given graph G must be specified in the "Graph" table.
  - Note that storing type of a graph G is done through the "Type" table.
- Vertices v_i and edges e_i of G will be detailed in the "Node" and "Edge" tables, respectively.
  - The "Type" table is also used to put the type of v_i and e_i in storage space. For example, using this table and assignment of two (or more) types of v_i, it will be able to represent bipartite (multipartite) graphs in the computer. The key attributes of edges e_i such as (un)directed, (un)weighted and (un)signed must be also kept into this table.
  - If v_i and e_i of the graph were to be labeled, the "Value" table is used to keep up labels. Since the labels can be of various data types including integer, string and so forth, this table helps us to consider them without any concern. Additionally, if either v_i or e_i (or both) has no label, this table occupy less (no) memory space.
  - The initial/terminal vertices from/to which an edge start/point, specifies in the "NodeEdge" table.
- Data of the "GraphEdge" table includes the information combined from the graphs and their corresponding edges. Actually, this table denotes the edges e_i which are in possession of a graph G.
- If there are any special types of the relations, such as the "AND" relations (AND-edges), between the edges of graph G, they must be stated in the "EdgeRelation" table.
- The contents stored in the "Type" table will be categorized and demonstrated in the "GroupType" table.

2.1.2 Business-Tier

The business-tier guarantees accuracy and correctness of the graph properties inserted, displayed and updated in datamodel. Additionally, whenever a new graph type is to be needed, it must be added through this tier.

3. CONCLUSION

With abundant information on biological systems such as gene regulatory networks, metabolic pathways, and signal transduction networks, there is pressing need to create of a comprehensive library to describe biological systems and processes. Therefore, we proposed a flexible datamodel (of a graph library) to feasibly represent the non-typical graphs as the typical ones.

4. REFERENCES

[1] De Halleux, J. 2012. "QuickGraph, Graph Data Structures and Algorithms for .NET".

[2] Siek, J., Lee, L., and Lumsdaine, A. 2000. "The boost graph library (BGL)."

[3] Wingender, Edgar. 2011. *Biological Petri Nets*. IOS Press.

[4] McDonald, D., Waterbury, L., Knight, R., & Betterton, M. D. 2008. Activating and inhibiting connections in biological network dynamics. *Biology Direct*. 3, 1, 49.

[5] Yeung, M., Thiele, I., and Palsson, B. Ø. 2007. Estimation of the number of extreme pathways for metabolic networks. *BMC bioinformatics*, 8, 1, 363.

[6] Khosraviani, M., Saheb Zamani, M., and Bidkhori, G. 2016. FogLight: an efficient matrix-based approach to construct metabolic pathways by search space reduction. *Bioinformatics*. 32, 3, 398-408.

Phoenix: A Systems Engineering Approach to Genetic Circuit Synthesis

Curtis Madsen[1], Evan Appleton[2], Prashant Vaidyanathan[1], Rizki Mardian[1], Cristian-Ioan Vasile[3], Katherine Elkind[1], Alexander Bennett[1], Fan Cao[1], Masakazu Nagata[3], Calin Belta[1], and Douglas Densmore[1]

[1]Boston University, Boston, MA
[2]Harvard Medical School, Boston, MA
[3]Massachusetts Institute of Technology, Cambridge, MA

1. INTRODUCTION

Synthetic biologists typically create new genetic circuits in informal, iterative specify-design-build-test cycles. Currently, the field relies on ad hoc methods, sometimes aided by simulations and mathematical modeling, though recent advances have led to the development of software tools capable of computationally designing functional genetic circuits using *Boolean logic* [5]. Biologists are, however, often concerned with how their systems behave spatially and temporally, while most existing tools are primarily focused on the steady-state function of the circuit and fail to capture these types of performance metrics. The ability to formally describe genetic circuit behavior over time with performance parameters represents a powerful move towards building more complex and robust genetic circuits.

In addition to providing performance specifications for desired genetic circuits, synthetic biologists can utilize ideas from systems engineering, an interdisciplinary field that focuses on building and maintaining complex engineering systems over their entire life cycles, to produce more robust genetic circuits. Applying standards and core concepts of related sub-fields such as performance engineering and reliability engineering is a promising way to advance synthetic biology's potential applications in research areas such as stem cell research, cellular medicine, and cellular environmental monitoring and sensing.

To address the need for more expressive specifications and engineering methods in synthetic biology, we have developed Phoenix, a Genetic Systems Engineering framework that incorporates the fundamentals of systems engineering and formal methods into the specify-design-build-test cycle of *bio-design automation* (BDA). With Phoenix, biologists can create formal, performance-bound specifications for complex biological systems like genetic circuits, and run finite-time simulations for modular design components to identify genetic circuits with high likelihoods of satisfying the desired specification. Phoenix seamlessly connects existing and novel BDA tools to create a closed-loop, algorithm-driven design workflow for a possible solution to a complete genetic circuit specify-design-build-test cycle where the only inputs are design specifications and collected data and the only outputs are DNA assembly and genetic circuit testing instructions. The Phoenix workflow is currently being experimentally validated on some classic synthetic biology model systems including several inverter cascades and a genetic toggle switch.

2. WORKFLOW

Phoenix can be broken down into several steps corresponding to the specify, design, build, and test paradigms of BDA. Figure 1 shows a visual representation of how a user would navigate this workflow, and each step is described in more detail in the following subsections.

2.1 Specify

During the specification step, a user can provide a performance specification using *signal temporal logic* (STL) [4]. STL provides more functionality than previous Boolean methodologies in that it adds the ability to create specifications that include parameters intrinsic to genetic components, interactions with complex environments and other components, and timing of interactions and events. For users not versed in the syntax and semantics of temporal logics, Phoenix includes a canvas that can be used to graphically draw traces representing the desired behavior of the circuit. These traces are then converted into an STL formula using a *temporal logic inference* (TLI) [2] technique known as GridTLI.

In addition to the performance specifications, a user can upload structural constraint information specified using Eugene (a rule based design language) [6]. These structural specifications are checked against known design rule constraints using grammars to check the validity of the structural design.

2.2 Design

During the design phase, Phoenix builds a module tree from the specification to break the structure of each genetic module down into the abstract DNA components [7] required to build the genetic system. It then pulls from a library of genetic parts, such as SynBioHub (formerly the SBOL Stack [3]), and assigns these parts to the DNA components in the tree to generate potential candidate circuits.

2.3 Build

In the build stage, A detailed list of components (supplemented with temporary testing components) is generated along with instructions required to assemble the parts (using Raven [1]). These testing modules function like 'unit tests' for the genetic module being built. The characterized modules are used to build genetic regulatory network models with assigned DNA components using parameter estimation and *in silico* simulation, and the best ones are selected for synthesis.

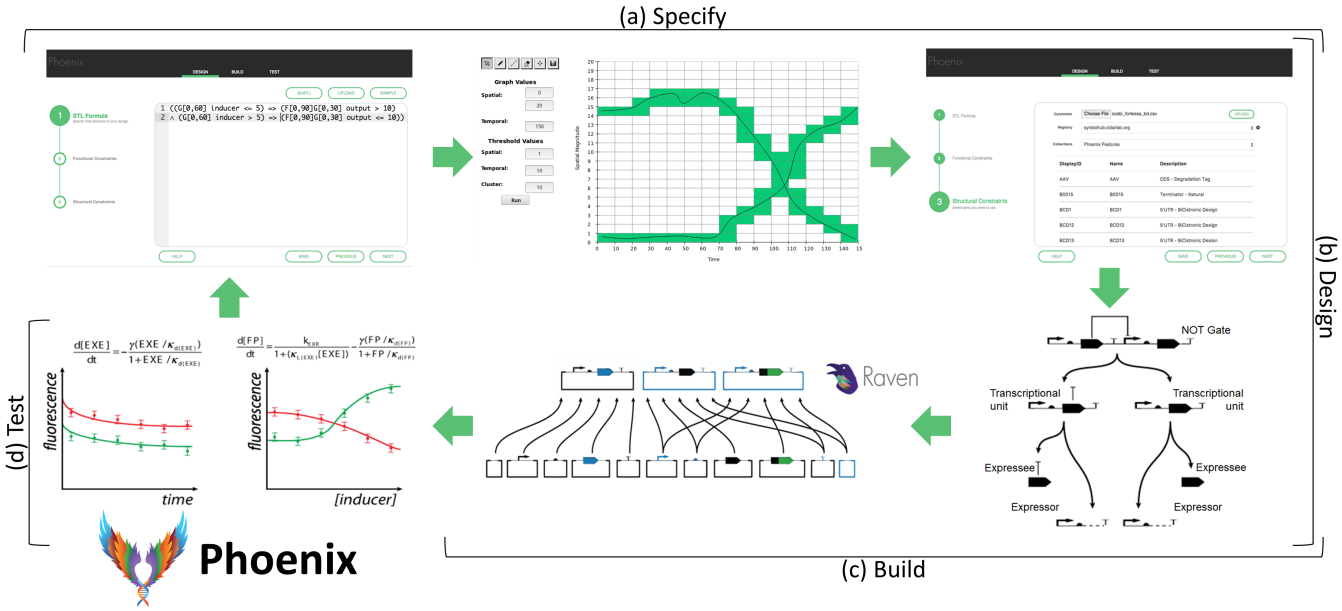


Figure 1: The Phoenix workflow. (a) A user begins on the specification page where an STL formula can be input using a text editor. Alternatively, the user can draw the desired behavior of the circuit as a collection of traces and an STL formula is inferred using GridTLI. Once a formula is provided, a user can additionally supply structural constraints as well as a parts library to be used in the design step. (b) During design, the specification is broken down into "expressors" and "expressees," and parts are selected from the library to fill these roles. (c) Next, several candidate circuit designs are generated and their assembly instructions are generated using Raven. These circuits are then simulated to verify that they produce the desired behavior. (d) The best circuits are built *in vivo*, and experimental results for the synthesized circuits are tested against the specification to determine the robustness of each circuit. This information is fed back into the specification step to improve the ability to engineer better circuits in the future.

2.4 Test

Using the assembly instructions, the circuits can be synthesized *in vivo* and experimental data can be gathered on their behavior. After post-processing the data, the resulting time series data can be verified against the performance specification of the desired genetic system to determine the robustness of each engineered genetic circuit.

3. DISCUSSION

The Phoenix workflow is being applied to the development of inverter cascades and a genetic toggle switch. In each case, an STL formula has been derived from time series data describing the behavior of the desired circuit, and a library of inverter modules is being constructed and queried to construct potential cascades and toggle switches. The circuit models whose simulation traces best match the STL specifications will be synthesized *in vivo* and the resulting experimental data will be verified against each STL formula.

There are various tools that cover different aspects of the specify-design-build-test cycle of BDA. In most cases, each tool focuses on a very specific task and solves a very specific subproblem. To ensure reliable performance throughout the entire life cycle of a system, it is important to have an interactive and hierarchical tool that guides users with detailed instructions throughout the BDA process. Phoenix is designed to ensure that functional, performance, and structural specifications of a genetic system are well-defined, reproducible, and reliable. By capturing essential properties of engineered genetic circuits, Phoenix is able to provide synthetic biologists with an automated way to design complex, dynamic genetic circuits.

4. ACKNOWLEDGEMENTS

This work has been funded by the Office of Naval Research under Grant No. N00014-11-1-0725 and the National Science Foundation under grant CPS Frontier 1446607.

5. REFERENCES

[1] E. Appleton et al. Interactive assembly algorithms for molecular cloning. *Nat. Methods*, 11:657–662, 2014.

[2] Z. Kong et al. Temporal logic inference for classification and prediction from data. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, pages 273–282, 2014.

[3] C. Madsen et al. The Sbol Stack: A platform for storing, publishing, and sharing synthetic biology designs. *ACS Synthetic Biology*, 5(6):487–497, 2016.

[4] O. Maler et al. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

[5] A. A. Nielsen et al. Genetic circuit design automation. *Science*, 352(6281):aac7341, 2016.

[6] E. Oberortner et al. A rule-based design specification language for synthetic biology. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 11(3):25, 2014.

[7] P. Vaidyanathan et al. A framework for genetic logic synthesis. *Proceedings of the IEEE*, 103(11):2196–2207, 2015.

Poly-omic statistical methods describe cyanobacterial metabolic adaptation to fluctuating environments

Extended Abstract

Supreeta Vijayakumar
Department of Computer Science and Information Systems, Teesside University
Southfield Road, Tees Valley
Middlesbrough, United Kingdom TS1 3BX
s.vijayakumar@tees.ac.uk

Claudio Angione
Department of Computer Science and Information Systems, Teesside University
Southfield Road, Tees Valley
Middlesbrough, United Kingdom TS1 3BX
c.angione@tees.ac.uk

ABSTRACT

In this work, a genome-scale metabolic model of *Synechococcus* sp. PCC 7002 which utilizes flux balance analysis across multiple layers is analyzed to observe flux response between 23 growth conditions. This is achieved by setting reactions involved in biomass accumulation and energy production as objectives for bi-level linear optimization, thus serving to improve the characterization of mechanisms underlying these processes in photoautotrophic microalgae. Additionally, the incorporation of statistical techniques such as *k*-means clustering and principal component analysis (PCA) contribute to reducing dimensionality and inferring latent patterns.

CCS CONCEPTS

• **Applied computing** → **Systems biology**; *Biological networks*; *Bioinformatics*;

KEYWORDS

Synechococcus, phototrophic growth, multi-objective optimization, flux balance analysis, machine learning

ACM Reference format:
Supreeta Vijayakumar and Claudio Angione. 2017. Poly-omic statistical methods describe cyanobacterial metabolic adaptation to fluctuating environments. In *Proceedings of 9th International Workshop on Bio-Design Automation, Pittsburgh, Pennsylvania USA, August 2017 (IWBDA'17),* 2 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

Metabolic modelling can provide an intuitive way of monitoring the amount of change in essential biological pathways; e.g. reactions involved in cellular growth and repair, energy production, transport, etc. Genome-scale metabolic models (GSMMs) can be used to improve prediction of phenotypic outcomes through supplementing linear constraints for conducting flux balance analysis (FBA) with external data from multi-omic studies. However, this undertaking is often challenging owing to the persistent challenges of integrating multiple disparate data types [9]. The application of statistics and data mining can help to inform the inter-connectivity of these datasets when they are combined to glean meaningful information. *Synechococcus* sp. PCC 7002 is a fast-growing cyanobacterium which flourishes in both freshwater and marine environments, owing to its ability to tolerate high light intensity and a wide range of salinities. Harnessing the properties of cyanobacteria has become an imperative goal in recent years, owing to its potential for producing renewable biofuels [4]. Here, we evaluate the efficiency of *Synechococcus* sp. PCC 7002 as a chassis for biofuel production over various growth conditions, with the aim of optimizing biomass and energy production during photosynthesis.

IWBDA'17, August 2017, Pittsburgh, Pennsylvania USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

2 METHODS

We begin by calculating flux under phototrophic growth in a model of *Synechococcus* sp. PCC 7002 [4] using multi-omics flux balance analysis (FBA) [1] to obtain condition-specific flux profiles. Transcriptomic data was acquired in the form of RNA sequencing reads from a series of studies previously conducted by Ludwig and Bryant [6–8]. These data are loaded into the model using METRADE to map gene expression data to a space where each metabolic profile is associated with a different growth condition [2]. Normalised flux distributions are calculated using three pairs of objectives: (i) Biomass and ATP maintenance (ii) Biomass and Photosystem I, and (iii) Biomass and Photosystem II. The structure for bi-level linear optimization is given in (1), where FBA is carried out using the COBRA Toolbox in MATLAB. The f and g Boolean vectors weight objectives for FBA, while the $v^{\min}$ and $v^{\max}$ vectors represent lower- and upper-limits for flux rates. The product of the stoichiometric matrix of all metabolites and reactions (S) and the vector of flux rates for all reactions (v) is 0 as rates of metabolite consumption and production remain constant.

$$\begin{aligned} &\max && g^{\mathsf{T}} v \\ &\text{such that} && \max\, f^{\mathsf{T}} v, \quad Sv = 0, \\ &&& v^{\min}\varphi(\Theta) \le v \le v^{\max}\varphi(\Theta), \end{aligned} \tag{1}$$

In (2), Θ represents a vector of gene set expression values of the reactions associated with the fluxes in v, which are mapped to a coefficient for the lower- and upper-limits of the corresponding reaction by function φ, defined as:

$$\varphi(\Theta) = [\mathbf{1} + \gamma\, |log(\Theta)|]^{\mathrm{sgn}(\Theta-1)}. \tag{2}$$

PCA was conducted using the FactoMineR package in R [5] (pictured in 2) and produces a scree plot of percentage contributions to variance in the first five dimensions, as well an individual factor map where growth conditions are described by reaction fluxes

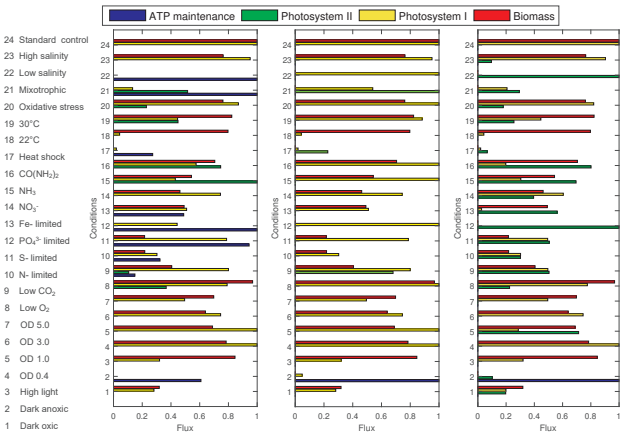


Figure 1: Flux distributions for (i) Biomass and ATP maintenance (ii) Biomass and Photosystem I and (ii) Biomass and Photosystem II, recorded across growth conditions 1-24.

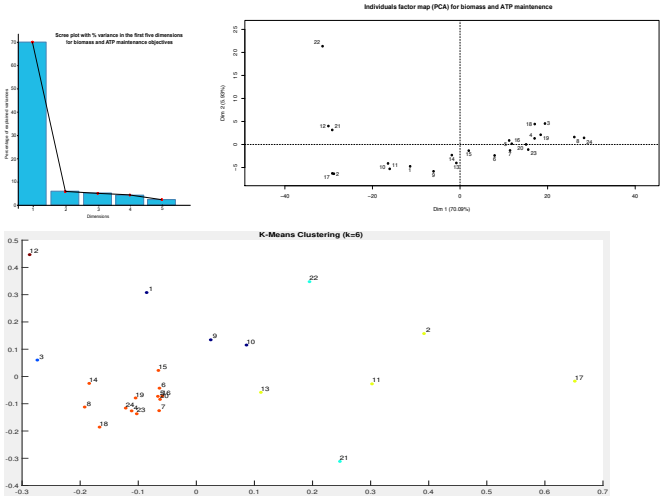


Figure 2: Percentage contribution of the first five dimensions to variance, individual factor map displaying principal component scores for 24 individuals (conditions) described by 742 variables (fluxes) on the first two principal components and *k*-means clustering performed with six clusters.

(quantitative variables) for each pair of objectives. Clustering is performed with the function *k*-means in MATLAB, using the number of clusters which returns the highest silhouette values for the majority of points (*k*=6).

3 RESULTS AND DISCUSSION

By prioritising different pairs of objectives during FBA, the mechanisms underlying each pathway become more evident. When ATP maintenance is set as the secondary objective (Fig 1), the highest fluxes occur in heat shock and growth-limiting conditions, illustrating the importance of this reaction in maintaining cellular function when growth rate or energy transfer through the photosystems is low. Absence of light and oxygen are shown to lead to a significant decrease in growth, owing to lower generation of ATP and NADPH without photoautotrophic growth; nutrient (particularly phosphate) limitation also results in low biomass flux, compared to the control. The tolerance of *Synechococcus* sp. PCC 7002 for high light intensity is evident from high flux for all three objective pairs through the biomass pathway. For the high salinity condition, fluxes through biomass and photosystem I are high for all objective pairs, but flux is only maintained in the low salinity condition for the reaction set as the secondary objective *g*. When set as objective *g*, flux through photosystem II for the low salinity condition is much higher than the high salinity condition.

Principal components analysis (PCA) was carried out across the flux distributions generated for all objective pairs to identify the conditions and/or reactions responsible for the most variance in the datasets. Fig 2 displays a scree plot with the percentage of variance explained by the first dimensions and also an individual factor map, which displays the principal component scores of 24 individuals (simulated conditions) described by 742 variables (reactions) on the first two dimensions. For all three objective pairs, more than 70% of the variance can be explained by just two dimensions i.e. two linear combinations of all fluxes (2). Low oxygen, high light intensity, high salt, and lower temperature give the highest scores for the first dimension; these conditions also yield the highest fluxes in 1 and are in concordance with experimental findings [3, 7, 10]. For the second dimension, the highest score is given by low salt, mixotrophic and phosphate-limitation conditions for the ATP objective. *k*-means also reflects the flux distributions in showing clear differentiation between conditions. In accordance with their biomass fluxes, high light intensity and phosphate limitation are isolated from all other conditions. The grouping of mixotrophic and low salt conditions is indicative of their lack of flux through the biomass reaction. Other conditions which are detrimental for growth form two separate clusters- 1, 9, 10 and 2, 11, 13, 17. In the first group of conditions, it can be noted that some growth is maintained through biomass synthesis, whereas in the second, there is higher flux through the photoexcitation reactions for photosynthesis, potentially with reliance on the ATP maintenance pathway to drive this process.

4 CONCLUSIONS

The use of a condition-specific metabolic model, which incorporates gene expression data and assesses multiple objectives, allows for prediction of significant metabolic patterns and phenotypic outcomes arising as a result of adaptation to fluctuating environmental conditions. In addition to this, statistical techniques such as PCA and clustering introduce another layer of analysis for uncovering latent patterns by re-organizing data on the basis of shared characteristics, therefore providing further insight into the maintenance of metabolic efficiency during phototrophic growth.

REFERENCES

[1] Claudio Angione, Max Conway, and Pietro Lió. 2016. *BMC Bioinformatics* 17, 4 (2016), 257.
[2] Claudio Angione and Pietro Lió. 2015. *Sci. Rep.* 5 (2015), 15147.
[3] Hans C Bernstein, Ryan S McClure, Eric A Hill, Lye Meng Markillie, William B Chrisler, Margie F Romine, Jason E McDermott, Matthew C Posewitz, Donald A Bryant, Allan E Konopka, et al. 2016. *mBio* 7, 4 (2016), e00949–16.
[4] John I Hendry, Charulata B Prasannan, Aditi Joshi, Santanu Dasgupta, and Pramod P Wangikar. 2016. *Bioresour. Technol.* 213 (2016), 190–197.
[5] Sébastien Lê, Julie Josse, and François Husson. 2008. *J. Stat. Softw.* 25, 1 (2008), 1–18.
[6] Marcus Ludwig and Donald A Bryant. 2011. *Front. Microbiol.* 2 (2011), 41.
[7] Marcus Ludwig and Donald A Bryant. 2012. *Front. Microbiol.* 3 (2012), 354.
[8] Marcus Ludwig and Donald A Bryant. 2012. *Front. Microbiol.* 3 (2012), 145.
[9] Supreeta Vijayakumar, Max Conway, Pietro Lió, and Claudio Angione. 2017. *Briefings in Bioinformatics* (2017).
[10] Qian Xiong, Jie Feng, Si-ting Li, Gui-ying Zhang, Zhi-xian Qiao, Zhuo Chen, Ying Wu, Yan Lin, Tao Li, Feng Ge, et al. 2015. *Mol Cell Proteomics* 14, 4 (2015), 1038–1053.

Re-wiring plant regulatory networks to enhance stress tolerance

IULIA GHERMAN*, MATHIAS FOO*, DAVID WILD[$], DECLAN BATES*, KATHERINE DENBY[‡]

* Warwick Integrative Synthetic Biology Centre, School of Engineering, University of Warwick, Coventry, CV4 7AL
[$] Department of Statistics and Systems Biology Center, University of Warwick, Coventry CV4 7AL, UK
[‡] Department of Biology, University of York, York YO10 5DD
* I.Gherman@warwick.ac.uk

1 INTRODUCTION

Synthetic biology is an emerging discipline that adopts an engineering approach to biological systems. We are interested in engineering complex phenotypes, such as enhanced disease or stress responses. Using a systems biology and control engineering approach, this project models the stress response in the plant *Arabidopsis* so that we can engineer more resilient plants by controlling the expression of transcription factors (TFs). This is the same strategy used by pathogens, whereby both fungi and bacteria can secrete effectors that manipulate TFs through interactions with key proteins that results in the downregulation of the plants immune response that subsequently undermines the defence system [1]. In addition, our network optimization approach is applicable to other scenarios, given appropriate inputs in the form of time series data and an optimization strategy.

Engineering the plant stress response

To genetically engineer a single gene in crops may confer a reduction in biotic stresses known as quantitative resistance but this is unlikely to prevent disease altogether [2]. However, if the gene in question is a TF, its manipulation can result in the differential expression of all the target genes it regulates, thus giving a wider reach than is possible by modifying non-TF genes [3]. Additionally, if the engineered TF targets other TFs, further effects can be seen at the downstream levels. Transcriptional reprogramming is a common feature in both biotic (e.g. pathogen infection), and abiotic stresses, (e.g. drought, high light and senescence) [4][5]. It is a change in gene expression of large numbers of genes, compared to the usual state of the plant and is carried out by TFs.

The choice of genes whose expressions need to be optimised is done based on knock-out and overexpression studies. These determine whether a gene gives a positive phenotype (in the case of infection, this means increased resistance), a negative phenotype (decreased resistance) or wildtype (no difference). Optimisation to a set point needs to be achieved with minimum amount of modifications and impact to the rest of the network other than pertinent nodes.

2 RESULTS

2.1 Network Inference

The modelling component of this project consists of inferring the regulatory network between *Arabidopsis* TFs which are responsible for transcriptional reprogramming following a particular stress. Several of the best-performing network inference algorithms [6] (GENIE3 [7], Inferelator [8], GRENITS [9], TIGRESS [10], CSI [11]) were used to infer the structure of *Arabidopsis* TF networks subject to infection with *Botrytis cinerea* or *Pseudomonas syringae* pathogens, senescence, drought or high light, using time series transcriptome data taken from *Arabidopsis* leaves. The resulting network models from these algorithms are ranked using several methods, such as the use of yeast-1-hybrid data, ChIP-seq (chromatin immunoprecipitation with sequencing) data, and knockout mutant transcriptome data combined with mathematical approaches in using Bayesian Information Criterion (BIC) score based on marginal likelihoods derived from the fitting of Gaussian processes to the data. From the different methods of ranking these models, we conclude that no one algorithm yields the best performance. Nonetheless, the performance of the consensus model is robust and consistently good, matching the findings of [6] and [12]. Therefore, this is our model of choice, with ChIP-seq and yeast-1-hybrid data included as prior information to construct a high confidence TF network for each stress response.

2.2 Network modelling and re-wiring

Using the consensus model obtained above, we turn our focus on enhancing the defence response to stress induced by *Botrytis cinerea* and *Pseudomonas syringae* pathogens, through different re-wiring simulation scenarios that will be used to inform our experimental set-up. The size of the network is first reduced by thresholding to remove edges and looking at subnetworks containing genes with phenotypes of interest.

The reduced models are then parameterised with a linear Ordinary Differential Equation (ODE) system for simulation purposes using system identification techniques [13][14]. Typical equations of the kind present in our modelling are shown below. Using nodes N1 and N2 as examples, Eqn. 1 shows the ODEs of the two nodes without re-wiring, while Eqn. 2 shows the ODE when node N1 is re-wired to include the incoming nodes of N2.

$$\frac{dN_1}{dt} = \theta_1 R_1 + \theta_2 R_2 + W_1 - \gamma_1 N_1$$
$$\frac{dN_2}{dt} = \theta_5 R_3 + \theta_6 R_4 + W_2 - \gamma_2 N_2 \quad (1)$$

$$\frac{dN_1}{dt} = \theta_1 R_1 + \theta_2 R_2 + \theta_5 R_3 + \theta_6 R_4 + W_1 - \gamma_1 N_1 \quad (2)$$

where R_x denote the gene expression of a regulator, θ_x represents parameters describing the strength and type (inhibitory or activating) of regulation, W_x captures the unknown/missing regulation and γ_x represent the degradation rates. This way of re-wiring is the simplest type that can be implemented experimentally as it involves the transformation of the wildtype plant with a plasmid containing the TF of N1 under the control of the promoter from N2.

Modelling of one such Arabidopsis subnetwork formed of 9 TFs is pictured below.

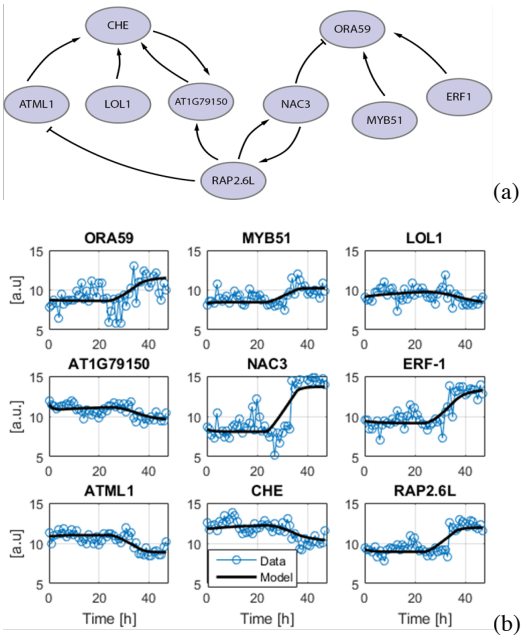


Figure 1. Modelling a subnetwork of TFs during the Arabidopsis stress response to *Botrytis*. (a) The inferred consensus network. (b) Simulations of our ODE model (black) fitted to actual gene expression data (blue).

All possible re-wirings are simulated, and for each one, the difference between the observed data and the desired gene expression of the network is calculated. The re-wirings that result in genes with a positive phenotype being expressed earlier or at higher levels are desirable. In this case, adding the edges from nodes regulating *ora59* to *che* or edges from nodes regulating *rap2.6l* to *atlm1* will increase the levels of *che*, a node which has a positive effect on the defence response to *Botrytis* (unpublished data), while having minimum impact on the rest of the network.

3 ONGOING WORK

The validation of the conclusions of our re-wiring simulations is being carried out in a protoplast system with plasmids containing target genes re-wired with different promoters. Protoplasts are plant cells which have had their cell wall removed through enzymatic means, making them easy to transiently transform with plasmid constructs [15]. Chitin, a complex carbohydrate found in fungal cell walls, is used to induce the immune response in lieu of *Botrytis* infection, and flg22, a peptide fragment derived from *P. syringae* is used as an inducer to mimic *P. syringae* infection in protoplasts. Currently, we are measuring levels of reporter genes to determine the best concentration/length of induction. These reporter genes are obtained from literature search (such as *frk1*) and our network models (such as *che*) and indicate the strength of the *Arabidopsis* defence response. This is measured by coupling the promoter of the reporter genes to GFP or luciferase; a change in these readings will indicate a change in the levels of the reporter gene. The re-wirings found to produce the desired enhanced defence response based on our *in silico* predictions and the reporter gene expression system in protoplasts will be used to generate stable constructs in plants and infected with *Botrytis* or *Pseudomonas* to confirm our results.

ACKNOWLEDGMENTS

Research supported by EPSRC/BBSRC Oxford/Warwick/Bristol Centre for Doctoral Training in Synthetic Biology (SynBio CDT) via research grant EP/L016494/1.

REFERENCES

[1] L. Lo Presti, D. Lanver, G. Schweizer, S. Tanaka, L. Liang, M. Tollot, A. Zuccaro, S. Reissmann, R. Kahmann, Fungal effectors and plant susceptibility, Annual review of plant biology 66 (2015) 513–545.

[2] D. A. St. Clair, Quantitative disease resistance and quantitative resistance loci in breeding, Annual review of phytopathology 48 (2010): 247-268.

[3] P. Bhatnagar-Mathur, V. Vadez, K. K. Sharma, Transgenic approaches for abiotic stress tolerance in plants: retrospect and prospects, Plant cell reports 27 (3) (2008) 411–424.

[4] O. Windram, P. Madhou, S. McHattie, C. Hill, R. Hickman, E. Cooke, D. J. Jenkins, C. A. Penfold, L. Baxter, E. Breeze, et al., Arabidopsis defense against botrytis cinerea: chronology and regulation deciphered by high-resolution temporal transcriptomic analysis, The Plant Cell 24 (9) (2012) 3530–3557.

[5] E. Breeze, E. Harrison, S. McHattie, L. Hughes, R. Hickman, C. Hill, S. Kiddle, Y.-s. Kim, C. A. Penfold, D. Jenkins, et al., High-resolution temporal profiling of transcripts during arabidopsis leaf senescence reveals a distinct chronology of processes and regulation, The Plant Cell 23 (3) (2011) 873–894.

[6] D. Marbach, J. C. Costello, R. Kuffner, N. M. Vega, R. J. Prill, D. M. Camacho, K. R. Allison, M. Kellis, J. J. Collins, G. Stolovitzky, et al., Wisdom of crowds for robust gene network inference, Nature methods 9 (8) (2012) 796–804.

[7] A. Irrthum, L. Wehenkel, P. Geurts, et al., Inferring regulatory networks from expression data using tree-based methods, PloS one 5 (9) (2010) e12776.

[8] R. Bonneau, D. J. Reiss, P. Shannon, M. Facciotti, L. Hood, N. S. Baliga, V. Thorsson, The inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo, Genome biology 7 (5) (2006) 1.

[9] E. R. Morrissey, M. A. Ju´arez, K. J. Denby, N. J. Burroughs, On reverse engineering of gene interaction networks using time course data with repeated measurements, Bioinformatics 26 (18) (2010) 2305–2312.

[10] A.-C. Haury, F. Mordelet, P. Vera-Licona, J.-P. Vert, Tigress: trustful inference of gene regulation using stability selection, BMC systems biology 6 (1) (2012) 1.

[11] C. A. Penfold, D. L. Wild, How to infer gene networks from expression profiles, revisited, Interface focus 1 (6) (2011) 857–870.

[12] T. Schaffter, D. Marbach, D. Floreano, Genenetweaver: in silico benchmark generation and performance profiling of network inference methods, Bioinformatics 27 (16) (2011) 2263–2270.

[13] L. Ljung. System identification: theory for the user. 2nd Edition, Prentice-Hall, Upper Saddle River, NJ 1999

[14] T.S. Gardner, D. di Bernardo, D. Lorenz, and J.J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. Science, 301:102–105, 2003.

[15] S. D. Yoo, Sang-Dong, Y.-H. Cho, and J. Sheen, Arabidopsis mesophyll protoplasts: a versatile cell system for transient gene expression analysis, Nature protocols 2.7 (2007): 1565-1572.

Software and Automation Enabled NGS Pipelines at Ginkgo Bioworks

Yaoyu Yang, Dawn Thompson, Teryn Citino, Zach Neuschaefer, Lina Faller, Chris Mitchell, Benjie Chen, Jamie Cho, Kristen Tran, Elaine Shapland, Barry Canton
Ginkgo Bioworks
{yaoyu,dawnt,tcitino,neuschaefer,lina,chris,benjie,jamie,kristen,elaine,barry}@ginkgobioworks.com

ABSTRACT

Ginkgo Bioworks has developed high-throughput, low-cost next-generation sequencing (NGS) pipelines that support plasmid, amplicon, and genome sequencing through a unique combination of state of the art automation, miniaturization through acoustic liquid handling, and custom software.

1 INTRODUCTION

Ginkgo Bioworks's mission is to make biology easy to engineer; to this end, we have built the Bioworks 1 and Bioworks 2 foundries in an effort to scale the process of engineering biology using software and hardware automation. Engineering biology at Ginkgo Bioworks involves designing and synthesizing DNA, transforming that DNA into cells, and testing the phenotypes of the resulting systems to determine what effects that DNA has on the cell. DNA sequencing is essential to verify that the DNA built and transformed into cells matches the design.

Current approaches to sequencing involve a combination of Sanger sequencing and next-generation sequencing (NGS). Due to its fast turnaround time and low cost, Sanger is typically used for verifying constructs such as plasmids and PCR amplicons. However, the low throughput nature of Sanger makes it inadequate for genome scale sequencing projects and would not allow the identification of other DNA species such as contaminants. NGS sequencing can provide a more unbiased, higher resolution profile of genome scale projects. To drive down the cost of NGS, we developed pipelines to achieve high-throughput, low-cost sequencing through a unique combination of state of the art automation, miniaturization through acoustic liquid handling methods using the Labcyte Echo machine, and custom software. This enables the pipeline to process a large number of user requests, track samples, organize complex experimental workflows, aggregate and process data in an automated fashion, and deliver results back to the requester.

Currently there are two sequencing pipelines running in production mode at Ginkgo Bioworks: Plasmid and Amplicon Sequencing (PandA) and Genome Sequencing (GeNOME). The PandA pipeline takes in purified DNA plasmid or amplicon samples from foundry users, performs library prep and pooling for loading on an Illumina Miseq, and delivers demultiplexed NGS datasets and summary tables through an automated computational pipeline. The PandA pipeline is capable of processing a total of 1536 samples from multiple foundry users in each run and delivers results to the users in an average of 31 hours. The GeNOME pipeline takes in purified genomic DNA samples from foundry users, performs library prep and pooling for loading on an Illumina Nextseq 500, and delivers demultiplexed NGS datasets and summary tables through the same automated computational pipeline. The GeNOME pipeline is capable of processing a total of 96 samples per run and delivers results in an average of 5 days.

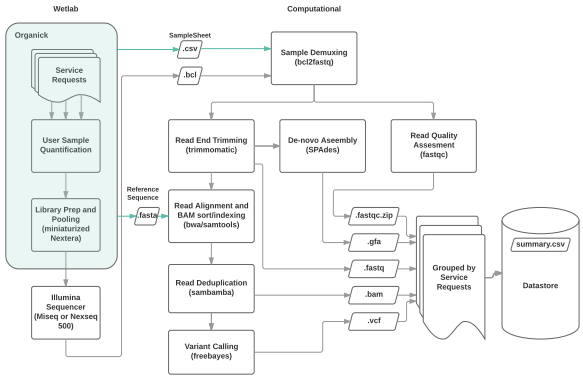


Figure 1: PandA and GeNOME pipelines flowchart

2 IMPLEMENTATION

Both the PandA and GeNOME pipelines are implemented through an assortment of software, hardware, and wetware. Here we describe the implementation of these two pipelines in three parts: the customer interface, wetlab workflows, and automated computational pipeline.

2.1 Customer Interface

The customer interface is implemented using three different software tools: Organick, Datastore, and Slack.

Organick [5] is a workflow management software developed and used internally at Ginkgo Bioworks that helps foundry users and operators plan protocols, track samples, and request foundry services. Both the PandA and GeNOME pipelines are implemented in Organick as a foundry Service, and users can submit a list of Service Requests in Organick to request the PandA Service or the GeNOME Service. Each Service Request indicates the samples that the user wants the Service to process.

Datastore is a web-based application that aggregates raw data generated from various instruments and serves as an automated, reproducible analytical platform for data analysis at Ginkgo Bioworks. Datastore analyzes the raw data with instrument and user specific methods, and provides an user interface (UI) for users to view and

download analysis results. Datastore provides a persistent data storage and a customer-facing UI for the output datasets produced by the automated computational pipeline.

Slack (slack.com) is a cloud-based tool for team communication and is used for real-time group and 1:1 communication at Ginkgo Bioworks. We utilize the Slack API to notify individual users when their Service Requests have been completed and send a link to relevant analyses in Datastore for the user to view and download.

2.2 Wetlab Workflows

The wetlab steps are organized by two Organick workflows: Quantification and Library Prep. The quantification workflow batches samples from the user submitted Service Requests and measures the DNA concentration. High-throughput, high-speed sample processing is achieved using acoustic liquid handling. Quantified samples that are in the accepted concentration range are queued for the Library Prep workflow. The Library Prep workflow performs a miniaturized Nextera library preparation similar to what developed by Shapland et al [9] using acoustic liquid handling before loading the samples on an Illumina Sequencer. The Illumina Miseq is used for PandA and Illumina Nextseq 500 is used for GeNOME.

Each Organick workflow is generated automatically with the click of a button, eliminating the need for the operator to do any complex setup and preventing human errors. The workflow consists of steps for operators to follow and execute. Steps that involve liquid transfers will generate a robot.csv file that describes each liquid transfer in a row and can be loaded onto the liquid handling robots through the foundry's automation platform. Organick keeps track of sample information such as content, volume, and lineage in the workflows. Most importantly, Organick enables a single operator to manage and execute the complex NGS library prep workflows with thousands of samples in a reliable and reproducible manner.

2.3 Automated Computational Pipeline

The automated computational pipeline is triggered when the end of a sequencing run has been detected. It retrieves the raw sequencing data (.bcl file) from the Illumina sequencer and performs the computational steps shown in Fig 1. Finally, the datasets are uploaded to Datastore and a Slack notification is sent to the users, instructing them how to view and download the results.

At the start of the pipeline, the Library Prep workflow and sample information from Organick are retrieved to prepare a SampleSheet.csv file that contains sample information and the associated barcodes. This SampleSheet.csv file is used by Illumina's bcl2fastq conversion software to demultiplex the raw .bcl image files to .fastq files. After .fastq files have been produced, FastQC [2] is used to perform read quality assessment. Trimmomatic [4] is used to remove low-quality bases from the ends of the reads, bwa [7] is then used on the trimmed .fastq files to perform read alignment against a reference sequence (.fasta) retrieved from Organick, followed by samtools [8] to sort and index the .bam files produced by bwa. The sorted .bam files are further processed by sambamba [10] to perform read deduplication. Then freebayes [6] is used to perform variant calling to produce variant calling files (.vcf). To detect the presence of unwanted DNA species, such as recircularized plasmids and contaminants, de novo assembly is performed on the trimmed fastq files using SPAdes [3].

The files produced by the computational pipeline are grouped based on the lists of Service Requests from users and uploaded to Datastore. Datastore also includes a summary.csv file that compiles summary information such as total reads sequenced, average sequencing depth, number of single nucleotide polymorphisms (SNPs), etc, for each sample in a grouped dataset by further analyzing the .bam and .vcf files. Finally, a Slack message is sent to each user with a link to their datasets and the corresponding summary.csv file.

The computational pipeline is written in Python, and uses Celery as a distributed task queue to process chained tasks. The pipeline is deployed using Docker on an in-house computational cluster managed by Rancher [1].

3 RESULTS

The PandA pipeline was launched in January of 2017. It is capable of sequencing 1536 plasmid and amplicon samples in a single run and the cost per sample is similar to Sanger sequencing. As of June 2017, we have used it to sequence around 18,000 samples from 36 different foundry users with an average turnaround time of 31 hrs. Measured by the data from inline controls, the PandA service has generated over 30X coverage for 97.5% of samples.

The GeNOME pipeline was launched in May 2017. It is capable of sequencing 96 genomic DNA samples in a single run with an average turnaround time of 5 days. As of June 2017, it has sequenced around 200 genomic DNA samples from over 10 foundry users.

ACKNOWLEDGMENTS

The software and automation enabled NGS pipelines are the result of significant contributions by all current and past employees of Ginkgo Bioworks.

REFERENCES

[1] *Rancher: Platform for operating Docker in production.* https://github.com/rancher/rancher.
[2] Simon Andrews. 2010. *FastQC: a quality control tool for high throughput sequence data.* http://www.bioinformatics.babraham.ac.uk/projects/fastqc.
[3] Anton Bankevich, Sergey Nurk, Dmitry Antipov, Alexey A Gurevich, Mikhail Dvorkin, Alexander S Kulikov, Valery M Lesin, Sergey I Nikolenko, Son Pham, Andrey D Prjibelski, et al. 2012. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of computational biology* 19, 5 (2012), 455–477.
[4] Anthony M Bolger, Marc Lohse, and Bjoern Usadel. 2014. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics* (2014), btu170.
[5] Benjie Chen, Dan Cahoon, Barry Canton, and Austin Che. 2015. Software for Engineering Biology in a Multi-Purpose Foundry. (August 2015).
[6] Erik Garrison and Gabor Marth. 2012. Haplotype-based variant detection from short-read sequencing. *arXiv preprint arXiv:1207.3907* (2012).
[7] Heng Li and Richard Durbin. 2009. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics* 25, 14 (2009), 1754–1760.
[8] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, et al. 2009. The sequence alignment/map format and SAMtools. *Bioinformatics* 25, 16 (2009), 2078–2079.
[9] Elaine B Shapland, Victor Holmes, Christopher D Reeves, Elena Sorokin, Maxime Durot, Darren Platt, Christopher Allen, Jed Dean, Zach Serber, Jack Newman, et al. 2015. Low-cost, high-throughput sequencing of dna assemblies using a highly multiplexed nextera process. *ACS synthetic biology* 4, 7 (2015), 860–866.
[10] Artem Tarasov, Albert J Vilella, Edwin Cuppen, Isaac J Nijman, and Pjotr Prins. 2015. Sambamba: fast processing of NGS alignment formats. *Bioinformatics* 31, 12 (2015), 2032–2034.

Towards Computer-Assisted Genetic Design

Yury V. Ivanov, Douglas Densmore
Department of Electrical and Computer Engineering
Boston University
Boston, MA 02215, USA
yvi1@bu.edu; dougd@bu.edu

Keywords

CAD; design templates; Eugene; CIDAR; part library.

1. INTRODUCTION

Genetic Design is a process in which a designer starts with a concept in mind and turns it into a DNA-level design. Whether this concept is a genetic circuit, bio-synthetic pathway, or even a multi-domain protein, designer needs to choose appropriate libraries of DNA parts and put those parts together in a particular order and structural orientation. This top-down design takes a lot of time, when done manually. Genetic design automation software SBOLDesigner 2 [5] utilizes a canvas-based user interface where designers can put parts together into a genetic construct. While SBOLDesigner 2 provides some great functionalities and an intuitive way to work with genetic constructs, the process of fetching parts and stitching parts together into a genetic construct has to be repeated by users for each of their designs. To achieve high throughput and automation, Computer-Assisted Genetic Design would greatly benefit from more formalized design abstractions of the desired part library and constraints on their composition. This motivated creation of the synthetic-biology computer-language Eugene [2].

Eugene is a computer language for Synthetic Biology that enables forward-engineering of DNA-level designs by expressing design abstractions [4]. While Eugene provides a powerful constrain-based system for expressing design strategies, the designer still have to learn this language in order to write specification files. This is particularly challenging for experimentalists that typically lack programming skills. To leverage this, we provide design templates, written in Eugene, to abstract some of the design strategies that experimentalists employ in their research. Each template specifies a structural design strategy, commonly used by molecular biology, to express gene(s) of interest. These templates autogenerate rule-based combinatorial designs of variable length and part composition without a need to re-write the specification each time for similar designs.

2. DESIGN TEMPLATES

Each design template is a function that takes a part library as an argument and returns rule-compliant combinatorial assemblies (Fig. 1).

2.1 Specification

Design template specifications are written in Eugene language [2] and provide instructions for a computer to put parts together, in a correct order and structural orientation (Fig. 1).

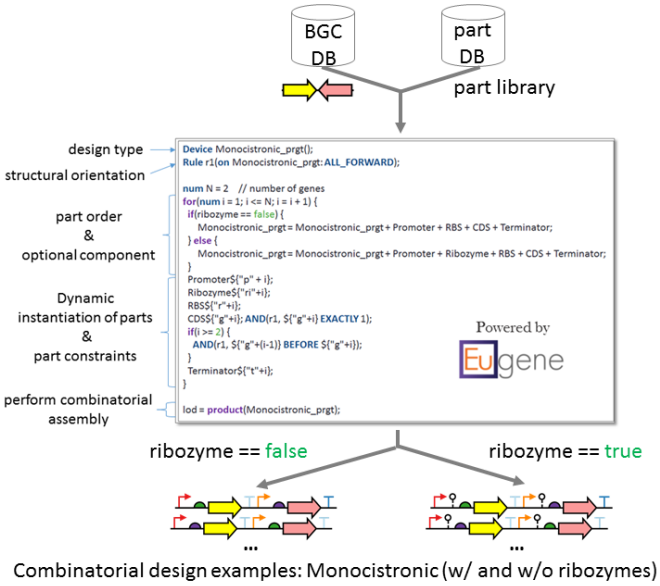


Figure 1: Example of a design template written in Eugene language. This template can be applied onto a Bio-synthetic Gene Cluster (BGC) of a variable length and part library.

2.1.1 *Optional Features*

Figure 1 contains an optional component ribozyme (a self-cleavable RNA). Designers can specify their own components using Eugene and create custom part performances values and constraints.

2.1.2 *Integration of Templates*

We are extending capabilities of Owl [1] software to create, test, store, and invoke genetic design templates for various applications. Owl exposes a set of design templates for various applications of Synthetic Biology.

2.2 Use-Case Scenario

Figure 2 summarizes an application of design templates in target molecule production, or gene cluster design. Described workflow can be integrated vertically with the downstream applications for further analyses, such as but not

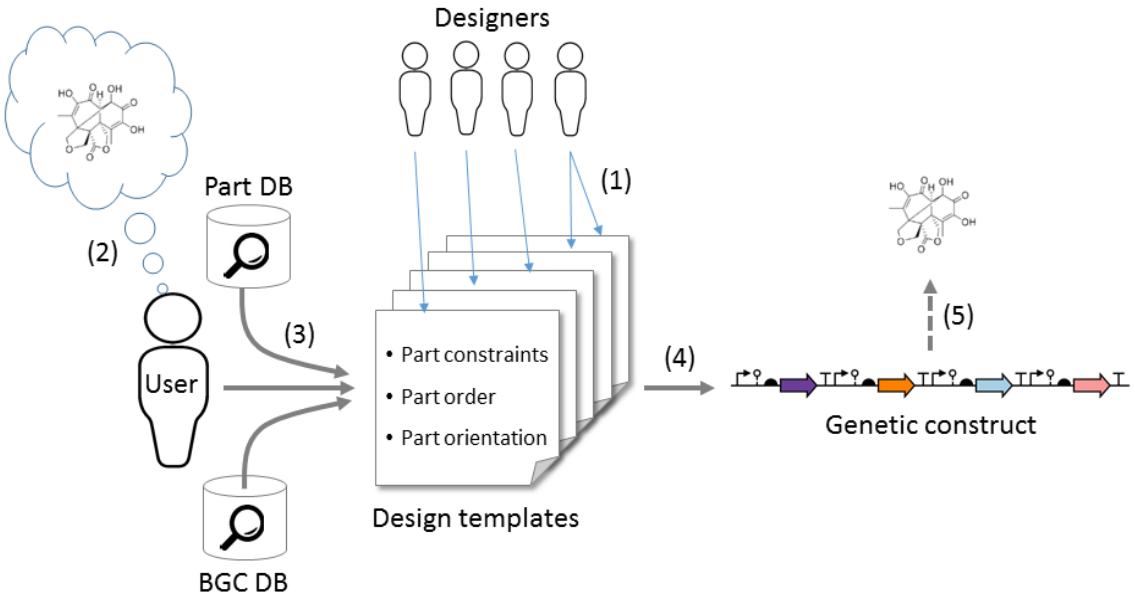


Figure 2: Data workflow and use-case scenario. (1) Designers encode abstractions of their genetic designs, in Eugene language. (2) A user has a concept in mind of a target molecule to produce. (3) The user searches Bio-synthetic Gene Cluster (BGC) database, for gene cluster that is either known or predicted to produce that target molecule, and part repository, to select part library (regulatory sequences). (4) User invokes particular design template to produce genetic constructs. (5) Genetic construct is ready for downstream applications and subsequent tests for target molecule production.

limited to: codon and part optimizations, Design of Experiments, part refactoring, chemical assembly and cloning, and Machine Learning.

2.3 Applications

2.3.1 *Gene Cluster Design*

Gene Cluster Design use-case scenario is explained on Figure 2.

2.3.2 *Genetic Circuit Design*

Eugene specification for genetic circuit design is already used by Cello [3]. Cello invokes Eugene for constrained combinatorial enumerations of genetic constructs within genetic circuits.

2.3.3 *Protein Design*

Design templates can contain protein domains (tags, fusion proteins, catalytic sites, other structural and functional domains) as parts and, therefore, do constraint-based protein design as well.

2.3.4 *CRISPR design*

Design templates could also be used to specify spacer compositions within the CRISPR arrays, where spacers are represented as parts for combinatorial rule-based assembly.

2.3.5 *RNA Design*

Same as for Protein Design, RNA motifs can be treated as parts for specification purposes. Information about motifs can be retrieved from RNA structural databases, such as the RNA 3D Motif Atlas[1].

3. ACKNOWLEDGEMENTS

We thank Ernst Oberortner, Ben Gordon, Fang Chen, Alex Cristofaro, and Dmitry Khazanovich for useful discussions.

4. REFERENCES

[1] E. Appleton, J. Tao, F. C. Wheatley, D. H. Desai, T. M. Lozanoski, P. D. Shah, J. A. Awtry, S. S. Jin, T. L. Haddock, and D. M. Densmore. Owl: Electronic datasheet generator. *ACS Synthetic Biology*, 3(12):966–968, 2014. PMID: 25524100.

[2] 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.

[3] A. A. K. Nielsen, B. S. Der, J. Shin, P. Vaidyanathan, V. Paralanov, E. A. Strychalski, D. Ross, D. Densmore, and C. A. Voigt. Genetic circuit design automation. *Science*, 352(6281), 2016.

[4] E. Oberortner, H. Huang, S. Bhatia, and D. Densmore. Eugene 2.0: A domain-specific language to specify constraint synthetic biological devices. poster presented at the SynBERC Spring Retreat, University of California-Berkeley, March 2012.

[5] M. Zhang, J. A. McLaughlin, A. Wipat, and C. J. Myers. Sboldesigner 2: An intuitive tool for structural genetic design. *ACS Synthetic Biology*, 2017.

[1]http://rna.bgsu.edu/rna3dhub/motifs