IWBDA 2016

Newcastle upon Tyne, UK August 16-18th

8th International Workshop on Bio-Design Automation
Newcastle University
August 16-18th, 2016

Foreword

Welcome to IWBDA 2016!

The IWBDA 2016 Executive Committee welcomes you to Newcastle upon Tyne, United Kingdom for the Eighth 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 22 contributed talks and 15 poster presentations. Talks are organized into seven sessions: Logic, Tools I, Pathways, Tools II, Standards, Automation and Circuits. In addition, we are very pleased to have two distinguished invited speakers: Dr. Amoolya Singh from Amyris and Prof. Natalio Krasnogor from Newcastle University.

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, Autodesk, Gen9, Twist Bioscience, ACS Synthetic Biology, DSM, Agilent Technologies, Raytheon BBN Technologies, Cytoscape, Lattice, and Minres Technologies for their support. We also thank Newcastle University and the Interdisciplinary Computing and Complex BioSystems (ICOS) research group for hosting and supporting IWBDA.

The following participants were provided financial support by our sponsors to attend IWBDA 2016

Allison Durkan	Boston University
Benjamin Lehner	Delft University of Technology
Cheng Han Hsieh	National Taiwan University
Daniel Dixon	University of Bristol
David Skelton	Newcastle University
Evan Appleton	Harvard Medical School
Hasan Baig	Technical University of Denmark
Jacob Becraft	Massachusetts Institute of Technology
James Alastair McLaughlin	Newcastle University
Johan Ospina	Boston University
Jonathan Naylor	Newcastle University
Juliano Bertozzi Silva	The University of Sheffield
Laurence Orr	Newcastle University
Luis Ortiz	Boston University
Meher Samineni	University of Utah
Mehrshad Khosraviani	Amirkabir University of Technology
Michael Zhang	University of Utah
Owen Gilfellon	Newcastle University
Prashant Vaidyanathan	Boston University
Ryan Silva	Boston University
Valentijn Broeken	Leiden University
Vishal Gupta	Universidad Politécnica de Madrid
Yury Ivanov	Boston University
Zach Zundel	University of Utah

IWBDA 2016 Sponsors

Workflow

SynBiCITE

Tool

AUTODESK.

Gen9

TWIST BIOSCIENCE

Class

ACS SyntheticBiology

DSM
BRIGHT SCIENCE. BRIGHTER LIVING.

Algorithm

Agilent Technologies

Raytheon
BBN Technologies

Cytoscape

LATTICE

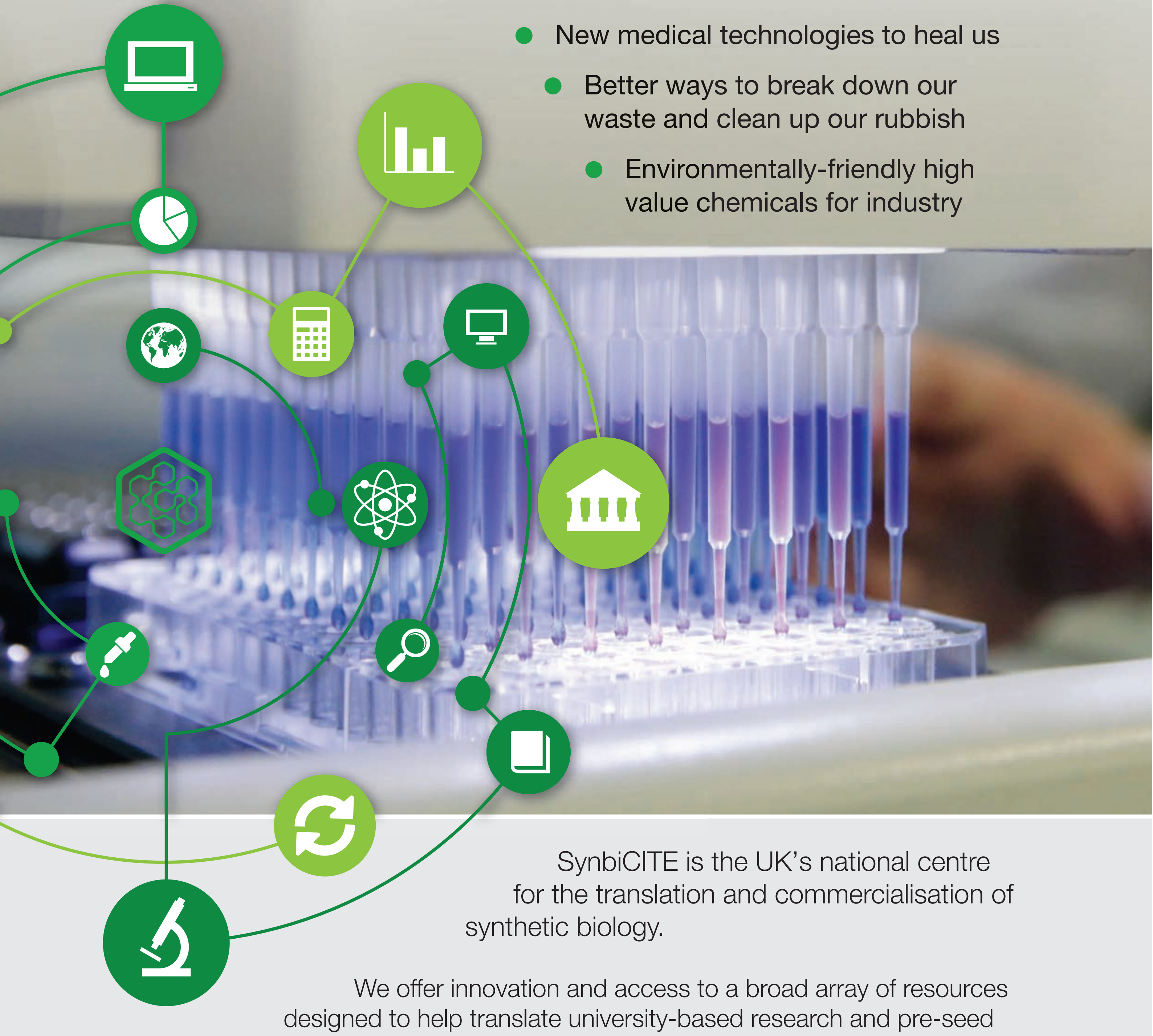
MINRES TECHNOLOGIES

Host Institute

Newcastle University

I(CO)$_2$S

Synthetic biology can improve our world



- New medical technologies to heal us
- Better ways to break down our waste and clean up our rubbish
- Environmentally-friendly high value chemicals for industry

SynbiCITE is the UK's national centre for the translation and commercialisation of synthetic biology.

We offer innovation and access to a broad array of resources designed to help translate university-based research and pre-seed concepts into commercial products, tools, processes and services.

Get in touch to find out how we can help you.

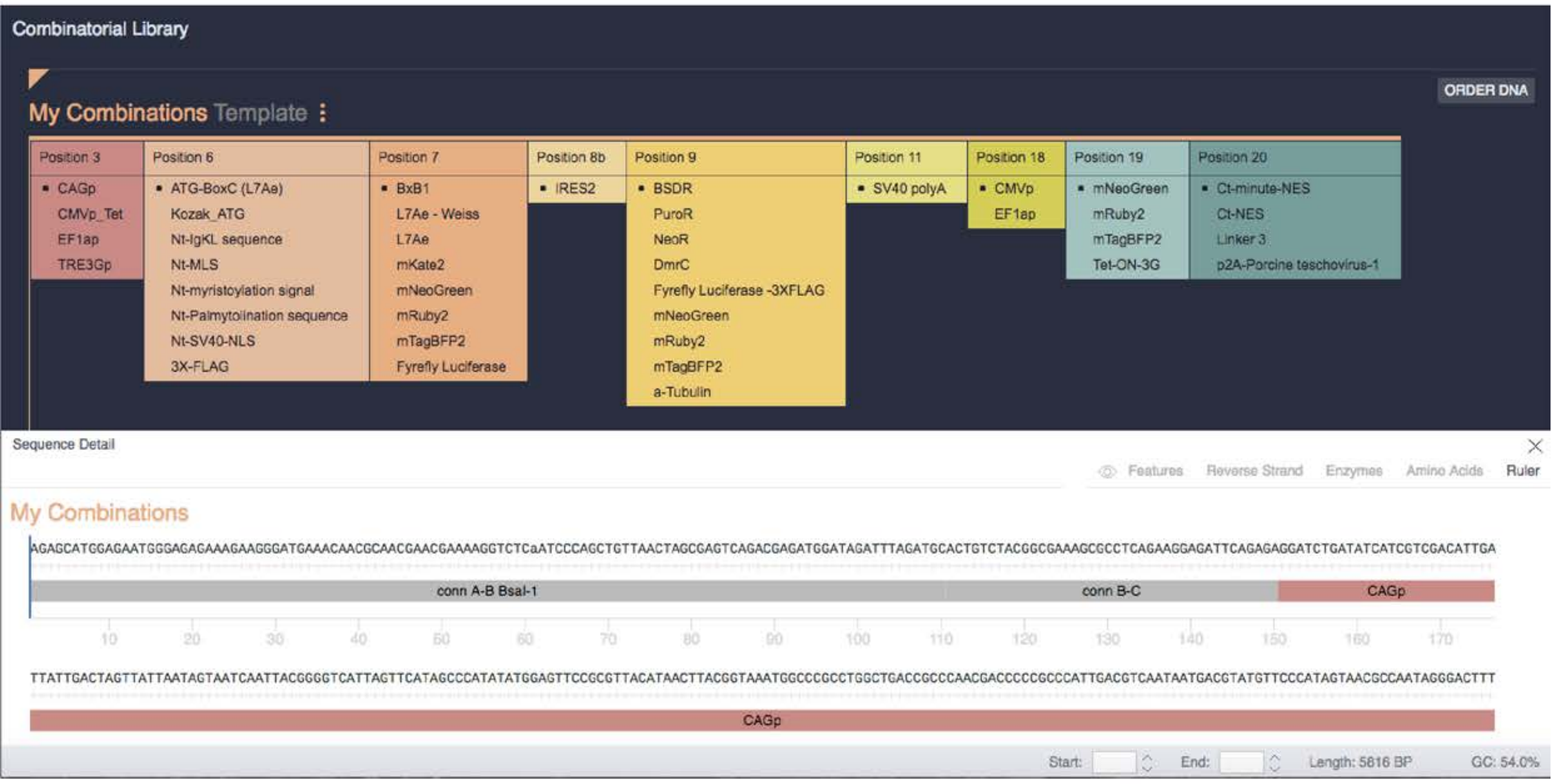
SynbiCITE

tel: +44 (0)20 7594 5910 • info@synbicite.com • www.synbicite.com

Genetic Constructor

A visually rich, open source, extensible, cloud CAD tool for biological design

Library creation made easy at www.geneticconstructor.com



My Combinations Template

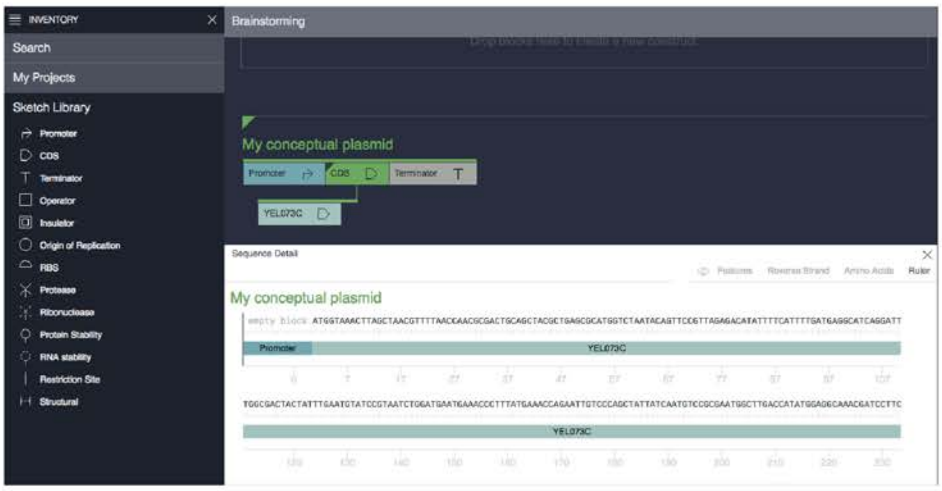
Position 3	Position 6	Position 7	Position 8b	Position 9	Position 11	Position 18	Position 19	Position 20
• CAGp	• ATG-BoxC (L7Ae)	• BxB1	• IRES2	• BSDR	• SV40 polyA	• CMVp	• mNeoGreen	• Ct-minute-NES
CMVp_Tet	Kozak_ATG	L7Ae - Weiss		PuroR		EF1ap	mRuby2	Ct-NES
EF1ap	Nt-IgKL sequence	L7Ae		NeoR			mTagBFP2	Linker 3
TRE3Gp	Nt-MLS	mKate2		DmrC			Tet-ON-3G	p2A-Porcine teschovirus-1
	Nt-myristoylation signal	mNeoGreen		Fyrefly Luciferase -3XFLAG				
	Nt-Palmytoilation sequence	mRuby2		mNeoGreen				
	Nt-SV40-NLS	mTagBFP2		mRuby2				
	3X-FLAG	Fyrefly Luciferase		mTagBFP2				
				a-Tubulin				

My Combinations

```
AGAGCATGGAGAATGGGAGAGAAAGAAGGGATGAAACAACGCAACGAACGAAAAGGTCTCaATCCCAGCTGTTAACTAGCGAGTCAGACGAGATGGATAGATTTAGATGCACTGTCTACGGCGAAAGCGCCTCAGAAGGAGATTCAGAGAGGATCTGATATCATCGTCGACATTGA
TTATTGACTAGTTATTAATAGTAATCAATTACGGGGTCATTAGTTCATAGCCCATATATGGAGTTCCGCGTTACATAACTTACGGTAAATGGCCCGCCTGGCTGACCGCCCAACGACCCCCGCCCATTGACGTCAATAATGACGTATGTTCCCATAGTAACGCCAATAGGGACTTT
```

Length: 5816 BP GC: 54.0%

Genetic constructor enables high throughout design. Use general blocks to abstract away low level base pair sequences and list blocks to quickly generate various combinations. Toggle across combinations in real time. Write a plugin to use your favorite Design of Experiment algorithm!



My conceptual plasmid

```
empty block ATGGTAAACTTAGCTAACGTTTTAACCAACGCGACTGCAGCTACGCTGAGCGCATGGTCTAATACAGTTCCGTTAGAGACATATTTTCATTTTGATGAGGCATCAGGATT
TGGCGACTACTATTTGAATGTATCCGTAATCTGGATGAATGAAACCCTTTATGAAACCAGAATTGTCCCAGCTATTATCAATGTCCGCGAATGGCTTGACCATATGGAGGCAAACGATCCTTC
```

Sketch your designs!

Throw away the paper and powerpoint slides. Brainstorm your designs and then fill the blocks with sequence from your library. No more cutting and pasting or translating designs from PowerPoint, Excel, or the laboratory white board.

AUTODESK.

Best-In-Class Design Tools

Design more DNA

Expanded Sequence Space

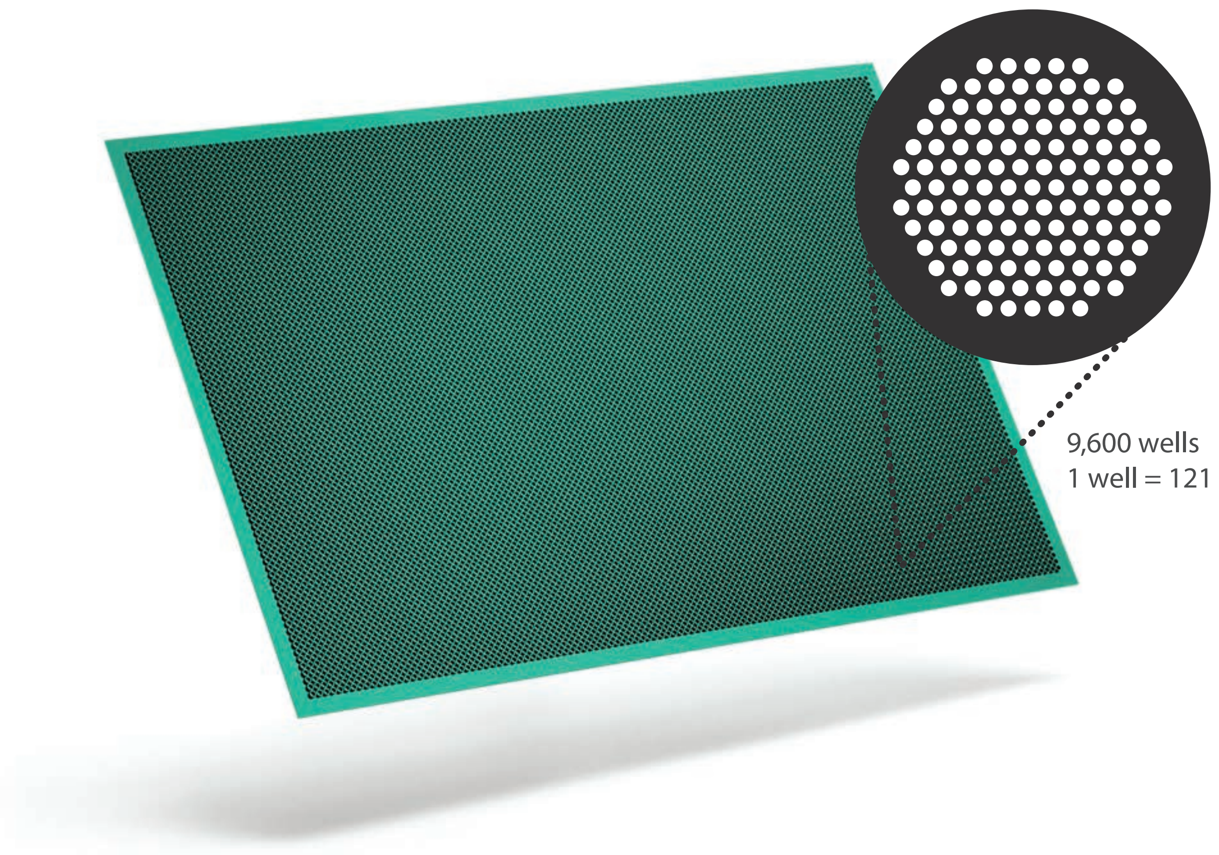
Test more hypotheses

Unparalleled Value

Discover more breakthroughs

www.gen9bio.com

Reimagine Gene Synthesis



Why clone? Let Twist Bioscience build for you

Genes

Oligo Pools

Libraries

DNA synthesis is at the core of synthetic biology, and Twist Bioscience's innovative silicon-based DNA writing technology is transforming gene synthesis. Our 9,600 nano-well semiconductor platform allows highly uniform synthesis of over a million oligonucleotides with extremely low error rates, enabling amplification-free production of sequence-perfect genes with quick turn-around times, at industry-leading prices.

The scalability of our platform makes it suitable for any size project, whether you need 1 gene or 10,000, or more. Think on a new scale, reimagine your gene designs, and accelerate your discoveries.

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

TWIST BIOSCIENCE

WWW.DSM.COM

Are societal challenges driving science?

Or is science driving societal change?

Sometimes the big issues inspire bright ideas, sometimes the big idea already exists and we can adapt it for an entirely new purpose. Whatever the catalyst, we're making bright science happen thanks to a rare and colorful mosaic of diverse competences, connections and collaborations - both inside and outside of DSM. We're able to tackle some of the very complex problems faced by the world and continue to find the answers to what we don't yet know.

HEALTH · NUTRITION · MATERIALS

DSM
BRIGHT SCIENCE. BRIGHTER LIVING.

ACS Publications
Most Trusted. Most Cited. Most Read.

ACS SyntheticBiology

SUBMIT YOUR RESEARCH TO OUR
'IWBDA 2016' SPECIAL ISSUE!

SUBMISSION DEADLINE: SEPTEMBER 30, 2016.

ACS Synthetic Biology 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
- Global Exposure
- Rapid Publication
- Highest Editorial Standards
- No Publication Charges
- New Open Access Initiatives

IMPACT FACTOR
6.076

2015 TOTAL CITATIONS
1,508

pubs.acs.org/synthbio

CHALLENGE
CONVENTION
BE PART OF THE NEXT FIRST.

Raytheon BBN Technologies has been providing advanced technology research and development for more than six decades. From the ARPANET and the first email, to GenBank and the first digital stereo mammography system, through the first network protected by quantum cryptography and our lifesaving Boomerang gunshot detection technology, BBN has consistently transitioned advanced research into innovative, practical solutions. Today, BBN scientists and engineers continue to take risks and challenge convention to create new and fundamentally better solutions.

https://jobs.raytheon.com/search-jobs/BBN

Raytheon
BBN Technologies

LATTICE AUTOMATION, INC. BOSTON, MA

LATTICE

We're Hiring

https://latticeautomation.recruiterbox.com

Abstracts - Table of Contents

Oral Presentations

Abstracts - Table of Contents

Poster Presentations

Organizing Committee

Executive Committee

General Chairs
Anil Wipat (Newcastle University) and Pietro Lio’ (University of Cambridge)

Finance Chair
Traci Haddock (iGEM Foundation)

Program Committee Chair
Chris French (University of Edinburgh)

Publication Chair
Avi Robinson-Mosher (Harvard Medical School)

Local Chairs
Goksel Misirli (Newcastle University), Harold Marc Andre Fellermann (Newcastle University), and Claire Smith (Newcastle University)

Web Chair

Aaron Adler (BBN Technologies)

Bio-Design Automation Consortium

Douglas Densmore (Boston University), President

Aaron Adler (BBN Technologies), Vice-President

Traci Haddock (Boston University), Treasurer

Natasa Miskov-Zivanov (Carnegie Mellon University), Clerk

Founders

Douglas Densmore (Boston University)

Soha Hassoun (Tufts University)

Marc Riedel (University of Minnesota)

Ron Weiss (MIT)

IWBDA 2016 Program

Monday, August 15th

09:00 - 18:30	15th SBOL Workshop

Tuesday, August 16th

09:00 - 18:00	1st BDAthlon programming contest

Wednesday, August 17th

8:00 - 8:30	Arrival, Breakfast and Registration
8:30 - 8:40	Opening Remarks Anil Wipat and Pietro Lio', Co-General Chairs
Session I: Logic	
8:40 – 9:00	Design for Improved Repression in RNA Replicons Jacob Beal and Ron Weiss
9:00 – 9:20	Noise Tolerance Analysis for Reliable Analog Computation in Living Cells Ramez Danial
9:20 – 9:40	Utilizing Signal Temporal Logic to Characterize and Compose Modules in Synthetic Biology Curtis Madsen, Prashant Vaidyanathan, Cristian-Ioan Vasile, Rachael Ivison, Junmin Wang, Calin Belta, and Douglas Densmore
9:40 – 10:00	Single Cell Analysis of RNA-engineered Logic Gates Christopher Schneider, Jascha Diemer, Leo Bronstein, Heinz Koeppl, and Beatrix Suess
10:00 – 10:30	Coffee Break
Keynote	
10:30 – 11:30	**Amoolya Singh.** Automating Design at an Industrial Biotech.
Session II: Tools I	
11:30 – 11:50	BioBlocks: A Web-Based Visual Environment for Programming Experimental Protocols in Biological Sciences Vishal Gupta, Jesús Irimia, Iván Pau, and Alfonso Rodríguez-Patón
11:50 – 12:10	An Environment for Augmented Biodesign Using Integrated Data Resources James McLaughlin, Goksel Misirli, Matthew Pocock, and Anil Wipat
Poster pitches	
12:10 – 12:30	Poster pitches: 1 minute per poster
Lunch	
12:30 – 12:40	Announcements
12:40 – 13:10	Lunch
Poster Session and Demos	
13:10 – 14:10	Poster Session

Session III: Pathways

14:10 – 14:30	extFogLight: Using Weighted Metabolic AND/OR Graph to Find Stoichiometric Balanced Pathways Mehrshad Khosraviani and Morteza Saheb Zamani
14:30 – 14:50	PathwayGenie - Pathway Design from Selection to Plasmid Neil Swainston, Pablo Carbonell, Adrian Jervis, Christopher Robinson, Mark Dunstan, and Jean-Loup Faulon
14:50 – 15:10	Integrated Predictive Genome-Scale Models to Improve the Metabolic Re-Engineering Efficiency Vishwesh Kulkarni, Lina El Menjra, Pablo Carbonell, and Jean-Loup Faulon
15:10 – 15:30	SBCDOE: a Design of Experiments-based Part Planner for Synthetic Biology Production of Chemicals Mark Dunstan, Adrian Jervis, Christopher Robinson, Neil Swainston, Jean-Loup Faulon, and Pablo Carbonell
15:30 – 15:40	Coffee Break

Evening Activities

15:40 – 15:45	Coaches leave at Centre for Life
17:00 – 18:00	Durham Castle tour
18:00 – 19:00	Durham Cathedral tour
19:30	Conference Dinner at the Durham Castle

Thursday, August 18th

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

Session IV: Tools II

8:30 – 8:50	A Web-Based Validator and Validation API for the Synthetic Biology Open Language Zach Zundel, Meher Samineni, Zhen Zhang, and Chris Myers
8:50 – 9:10	SBOLDesigner 2.0 Michael Zhang and Chris Myers
9:10 – 9:30	Bioform: an in-silico 3D Physical Modelling Platform for the Design and Analysis of Bacterial Populations Jonathan Naylor, Harold Fellermann, Waleed Mohammed, Nick Jakubovics, Joy Mukherjee, Catherine Biggs, Phillip Wright, and Natalio Krasnogor
9:30 – 9:50	TEBio - Tools for Engineering Biology James Scott-Brown, Thomas Prescott, and Antonis Papachristodoulou
9:50 – 10:20	Coffee Break

Keynote II

10:20 – 11:20	**Natalio Krasnogor.** Accelerating Synthetic Biology via Software and Hardware Advances.

Session V: Standards

11:20 – 11:40	ShortBOL: A shorthand for SBOL Matthew Pocock, Chris Taylor, Goksel Misirli, James McLaughlin, and Anil Wipat
11:40 – 12:00	A Data Model for the Description of Bioparts Iñaki Sainz de Murieta, Matthieu Bultelle, and Richard I. Kitney

Lunch
12:00 – 13:30

Session VI: Automation

13:30 – 13:50	Design and Automated Inference of Design Principles in Gene Regulatory Networks: a Multiobjective Optimization Approach Irene Otero-Muras and Julio R. Banga
13:50 – 14:10	How to Remember and Revisit Many Genetic Design Variants Automatically Nicholas Roehner and Douglas Densmore
14:10 – 14:30	Towards Automated Biosecurity: Screening of Synthetic Biology Constructs Benjamin Apra, Arthur Vigil, and James Diggans

Discussion Session

14:30 – 15:30	Discussion
15:30 – 16:00	Coffee Break

Session VII: Circuits

16:00 – 16:20	Realization of Large Logic Circuits with Long-Term Memory Using CRISPR/Cas9 Systems Tai-Yin Chiu, Cheng-Han Hsieh and Jie-Hong Roland Jiang
16:20 – 16:40	3D Printing of Microbes for Material Production Benjamin Lehner and Anne S. Meyer
16:40 – 17:00	MakerFluidics: Microfluidics for the Masses Ryan Silva, Radhakrishna Sanka, and Douglas Densmore

Closing Remarks and Awards

17:00 – 17:15	Closing remarks Anil Wipat and Pietro Lio', Co-General Chairs

Keynote Presentation

Amoolya Singh

Automating Design at an Industrial Biotech

Amyris has developed a high-throughput genetic engineering platform for designing and building custom microbes to serve as living factories. Using an industrial scale fermentation process, our microbes convert cheap sugars into a wide variety of high value target molecules, including medicines, commodity and specialty chemicals. Our end products provide low cost, high quality malaria medication and renewable substitutes for fuels and chemicals. Amyris' R&D efforts span rational & random design and construction of microbial strains, high-throughput screening and analytical chemistry, fermentation at multiple scales, and genotype/phenotype data mining. Every aspect of this work is facilitated and accelerated by quantitative science and software & hardware automation. In this talk, I will outline the computational challenges inherent in automating the design of microbial strains. To meet these challenges, we use a range of innovations including genotype specification languages and high-level functional ontologies of parts and pathways; metabolic and statistical models; literature mining algorithms; and design of experiments approaches.

Dr. Amoolya Singh is a computational biologist and Senior Scientist at Amyris. She leads Amyris R&D's Scientific Computing group, whose work includes innovations in genotype representation and data visualization, building mathematical and statistical models to analyze high-throughput, multivariate genotype and phenotype data; metabolic modeling and design-of-experiments to perturb microbial biochemical pathways and identify bottlenecks therein; and statistical process control to accelerate Amyris' design-build-test-learn cycle.

Amoolya obtained a bachelor's degree with honors at Carnegie Mellon double majoring in Biology and Computer Science; and a Ph.D. in computational biology from UC Berkeley jointly advised by Adam Arkin (Bioengineering) and Richard Karp (Math/CS). Between degrees, she worked as a software engineer at an Internet startup, a multinational wireless telecommunications firm, and a Wall Street investment bank. Prior to joining Amyris, Amoolya completed a postdoctoral fellowship at the European Molecular Biology Lab in Heidelberg, Germany (with Peer Bork) and a Computational & Life Sciences fellowship at Emory University (with Bruce Levin) in the fields of comparative genomics and metagenomics, population genetics, and experimental evolution.

Keynote Presentation

Natalio Krasnogor

Accelerating Synthetic Biology via Software and Hardware Advances

In this talk I will discuss recent work done in my lab that contributes towards accelerating the specify -> design -> model -> build -> test & iterate biological engineering cycle. This will describe advances in biological programming languages for specifying combinatorial DNA libraries, the utilisation of off-the-shelf microfluidic devices to build the DNA libraries as well as data analysis techniques to accelerate computational simulations.

Prof. Natalio Krasnogor is Professor of Computing Science and Synthetic Biology, co-directs Newcastle'sInterdisciplinary Computing and Complex BioSystems (ICOS) research group and is director of the Centre for Synthetic Biology and the Bioeconomy (CSBB). Prof. Krasnogor holds a prestigious EPSRC Leadership Fellowship in Synthetic Biology, is the overall lead in the EPSRC Synthetic Biology ROADBLOCK project involving Newcastle, Nottingham, Sheffield, Warwick and Bradford Universities that seeks to develop in silico and in vivo techniques for engineering biofilms. He leads the "Synthetic Portabolomics: Leading The Way at the Crossroads of the Digital and Bio Economies" EPSRC project.

With expertise in Synthetic Biology, Complex Systems and Machine Intelligence, Prof. Krasnogor gave several keynote talks (e.g., IEEE CEC, PPSN, GECCO); has >170 publications (H-index 37), with many of his papers in the top 0.1% and 1% for number of citations in computing science and has published also in top tier journals such as Nature Biotech, Nature Chemistry, and PNAs. He won several best papers prizes as well as Bronze, Silver and Gold awards of the American Computing Society's (ACM) HUMIES award for human-competitive results that were produced by any form of genetic and evolutionary computation and an ACM's Impact award. From 2012 to 2014 he was the Science Director of the European Centre for Living Technologies (Italy), was distinguished visiting professor at Ben Gurion University (Israel) in 2009 and Weizmann Institute of Science (Israel) in 2010, 2012, and 2013.

Allan Kuchinsky Scholarship

Nicholas Roehner

Dr. Nicholas Roehner is a postdoctoral research fellow in the Cross-Disciplinary Integration of Design Automation (CIDAR) lab of Prof. Doug Densmore at Boston University and in the MIT-Broad Foundry at the Broad Institute of MIT and Harvard. He received his Ph.D. in bioengineering from the University of Utah in 2014 working with Prof. Chris J. Myers on computational methods for genetic design automation. During this time, he also served as an editor of the Synthetic Biology Open Language (SBOL) and contributed to the development of the SBOL 2.0 data standard. He is currently a researcher on projects under the 1000 Molecules component of the DARPA Living Foundries program, including software for designing genetic libraries based on experimental designs (Double Dutch) and a database for storing and tracking changes to large combinatorial spaces of possible genetic designs (Knox). His research interests include the application of languages, games, and simulation to the development of a hierarchy of abstraction for synthetic biology, one in which experts with different specialties can effectively communicate and collaborate across abstraction barriers.

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

Previous recipients

2015	Swapnil Bhatia

Agilent Technologies Cytoscape

Design for Improved Repression in RNA Replicons

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

Ron Weiss
Massachusetts Institute of Technology
Cambridge, MA, USA
rweiss@mit.edu

1. MOTIVATION

RNA replicons are an emerging platform for synthetic biology, in which the infective capsid of a RNA virus is replaced with an engineered payload while its self-replication capability is retained [4, 3, 1, 6]. This self-replication capability allows RNA replicons entering a cell to amplify their engineered elements, providing strong expression from a low initial dose without integration into host DNA or propagation to other cells. Replicons thus offer an attractive platform for developing medical applications such as vaccines [2, 3] and stem-cell generation [7], combining both strong expression and relative genetic isolation. Development of RNA replicons to date has focused primarily on derivatives of alphaviruses, a well-characterized family of positive-strand RNA viruses, and most particularly the Sindbis and VEE vectors [4]. Protein expression from RNA replicons can be precisely predicted and controlled [1], and can support standard synthetic circuits such as cascades and toggle switches [6].

A key challenge for creating effective synthetic circuitry with RNA replicons, however, is that regulatory devices often perform less well when expressed from replicons. For example, L7Ae is a very strong RNA regulator, able to provide more than 200-fold repression when expressed from DNA plasmids, but was found to yield less than 30-fold repression when expressed from RNA replicons [6]. By examination of quantitative models derived from [1] and [6], we find that simple circuit adjustments, to exploit rather than oppose RNA replicon dynamics, should be able to reverse this problem and in fact produce significantly better circuit performance than is observed with DNA plasmids.

2. QUANTITATIVE EXPRESSION MODEL

Figure 1 shows a diagrammatic model of the interactions in a two-replicon repression circuit modeled after [6]. In this circuit, the L7Ae RNA regulator supresses expression of mVenus fluorescent protein and is in turn degraded by the small interfering RNA siRNA-FF4. When siRNA-FF4 is absent, L7Ae will not degrade and will repress mVenus, whereas when it is present L7Ae will rapidly degrade and mVenus should be high. This system may be simulated as an ODE using the following equations:

$$\frac{dR_i}{dt} = \alpha_i \cdot N \cdot R_i \tag{1}$$

$$\frac{dL}{dt} = \alpha_L \cdot A \cdot R_1 - \frac{\log 2}{t_L} \cdot \frac{1+(S/D_S)^{H_S}}{1+{K_S}^{-1}(S/D_S)^{H_S}} \cdot L \tag{2}$$

IWBDA 2016 Newcastle, UK
This work was sponsored by DARPA DSO under grant W911NF-11-054; the views and conclusions contained in this document are those of the authors and not DARPA or the U.S. Government..

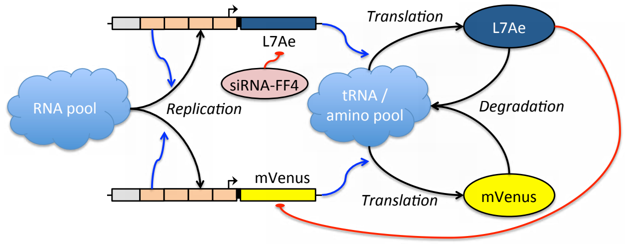


Figure 1: RNA replicon repression circuit: siRNA-FF4 degrades L7Ae, which in turn represses mVenus fluorescent protein.

$$\frac{dV}{dt} = \alpha_V \cdot A \cdot R_2 \cdot \frac{1+{K_L}^{-1}(L/D_L)^{H_L}}{1+(L/D_L)^{H_L}} - \frac{\log 2}{t_V} \cdot V \tag{3}$$

$$\frac{dN}{dt} = -\sum_i \frac{dR_i}{dt} \tag{4}$$

$$\frac{dA}{dt} = -(\frac{dL}{dt} + \frac{dV}{dt}) \tag{5}$$

where R_i is the number of copies of each replicon, N is the amount of available transcriptional resources, S is the amount of siRNA-FF4, L is the amount of L7Ae, V is the amount of mVenus, A is the amount of available translational resources, t_x is the decay half-life of species x, and α_x, K_x, D_x, and H_x are standard Hill equation coefficients. When parameterized with best-fit values derived from [1] and [6][1], the system behaves as shown in Figure 2, producing a 9-fold repression: much less than the 63-fold it predicts from plasmid DNA and an underperformance ratio equal to that observed in [6]. The model suggests that poor performance is due to the high expression of L7Ae in its "off" state and the inability of L7Ae to sufficiently repress mVenus before a significant amount has built up in the system.

3. PARAMETER OPTIMIZATION

Given the issues identified by the model and the nature of this RNA replicon circuit, there are four tuning mechanisms that offer ready means of adjusting performance. The high L7Ae "off" expression can be decreased by decreasing the relative initial dose of its expressing replicon or by decreasing per-replicon expression by decreasing the strength of its subgenomic promoter (a well-established mechanism for controlling replicon expression, e.g. [5]). Likewise, the high ini-

[1]Note that due to the insufficiency of available experimental data, some parameters are poorly constrained.

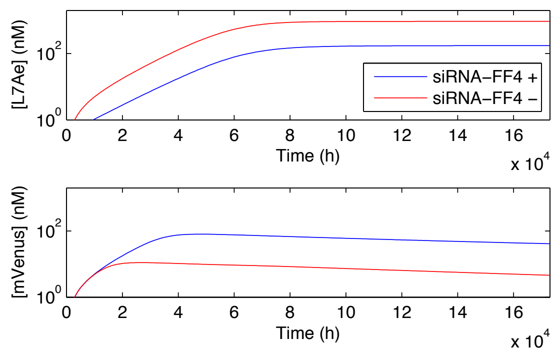


Figure 2: Unoptimized circuit has poor dynamic range due to leaky L7Ae "off" and early unrepressed expression of mVenus.

tial expression of mVenus can be decreased by decreasing the strength of its subgenomic promoter or by adding degradation tags to decrease its half-life.

To investigate the potential of these mechanisms, we performed single-parameter scans, running simulations of each adjustment across two orders of magnitude at 20 values per decade. These simulations indicate that the two L7Ae modifications have a near-equivalent effect in significantly amplifying the dynamic range of this circuit. Decreasing the half-life of mVenus can also improve dynamic range by affecting different dynamics, while adjusting mVenus promoter strength does not improve dynamic range but only shifts expression linearly.

Based on these single-parameter results, we conducted a detailed two-parameter scan for both decreasing L7Ae promoter strength and decreasing mVenus half-life. Figure 3 shows the results of this experiment, including an asymmetric region in which the combination of the two modifications is predicted to provide more than 500-fold dynamic range. The combination of decreasing L7Ae dose and decreasing mVenus half-life (not shown) produces very similar results. Intuitively, in this area decreased L7Ae expression means that unrepressed mVenus outcompetes L7Ae for resources

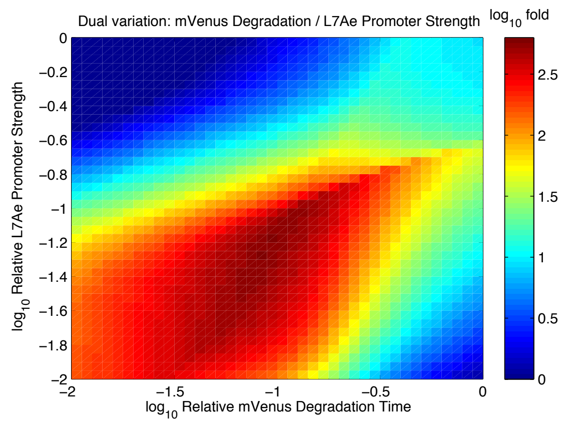


Figure 3: Decreasing L7Ae promoter strength and mVenus half-life can markedly improve the predicted dynamic range of repression.

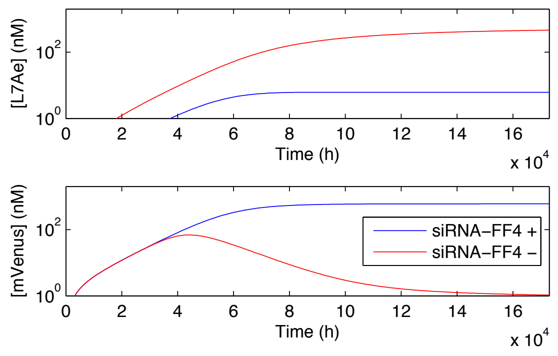


Figure 4: Optimized circuit with 7% L7Ae expression and 10% mVenus half-life has more than 50-fold improvement in predicted dynamic range.

and decreases its "off" level, while decreased mVenus half-life means that even a high initial transient can be extinguished in the repressed state. Together, these predict expression patterns such as in the example in Figure 4, predicting much greater dynamic range for both L7Ae and mVenus.

4. CONTRIBUTIONS AND FUTURE WORK

Having predicted modifications to markedly improve the performance of repression in replicon circuits, a clear next step is for these modifications to be implemented in the lab and tested experimentally to see whether the predicted improvements materialize (which may require combining SGP and ratio manipulation to get sufficient range). Importantly, precise quantitative prediction and design has previously been demonstrated in replicons [1] and the predicted region of high performance is fairly broad. These models may also be extended to predict a larger range of systems, including more modes of regulation and more complex replicon architectures, thereby increasing the range of replicon applications that may be more effectively engineered.

5. REFERENCES

[1] J. Beal et al. Model-driven engineering of gene expression from rna replicons. *ACS Synthetic Biology*, 4(1):48–56, 2015.

[2] A. J. Geall et al. Nonviral delivery of self-amplifying rna vaccines. *Proceedings of the National Academy of Sciences*, 2012.

[3] K. Lundstrom. Alphavirus vectors in vaccine development. *J. Vaccines Vaccin.*, 3, 2012.

[4] S. Perri et al. An alphavirus replicon particle chimera derived from venezuelan equine encephalitis and sindbis viruses is a potent gene-based vaccine delivery vector. *J. Virol.*, 77:10394–10403, 2003.

[5] R. Raju and H. V. Huang. Analysis of sindbis virus promoter recognition in vivo, using novel vectors with two subgenomic mrna promoters. *Journal of virology*, 65(5):2501–2510, 1991.

[6] L. Wroblewska et al. Mammalian synthetic circuits with rna binding proteins for rna-only delivery. *Nature Biotechnology*, 2015.

[7] N. Yoshioka et al. Efficient generation of human ipscs by a synthetic self-replicative rna. *Cell Stem Cell*, 13(2):246–254, 2013.

Noise Tolerance Analysis for Reliable Analog Computation in Living Cells

Ramez Daniel
Biomedical Engineering Department
Israel Institute of Technology
Haifa 32000, Israel
ramizda@bm.technion.ac.il

ABSTRACT

Two major computation paradigms have been implemented so far in living cells - analog paradigm that computes with a continuous set of numbers and digital paradigm that computes with two-discrete set of numbers. Here, we analyze the biophysical and technological limits of gene networks created based on analog computation in living cells. More specifically, we calculate the precision of analog systems impacted by extrinsic and intrinsic noise sources. Furthermore, an analytical description of a biophysical model recently developed for positive feedback linearization circuits and used in analog synthetic biology, is presented.

CCS Concepts

• **Applied computing~Computational biology** • **Applied computing~Biological networks** • *Applied computing~Systems biology*

Keywords

Synthetic biology, System biology, Analog computation, Digital computation, Feedback loops, Cellular noise.

1. INTRODUCTION

Early efforts at biomolecular computing have used binding and unbinding reactions to represent the "ON/OFF" or "1/0" logic states [1]. Consequently, proteins that bind to DNA or promoters and activate high levels of gene expression, represent the "1" logic state, while unbound, free proteins yield low levels of gene expression, and represent the "0" logic state. Many genetic circuits that mimic electronic digital circuits, have been constructed to perform Boolean logic gates, counter and memory devices in living cells [1].

To date, engineered artificial logic gates in living cells have been proven difficult to scale due to cellular resource limitations, a lack of orthogonal genetic devices, high leakage levels of synthetic genetic devices and the absence of suitably sharp input-to-output transfer functions [2]. Recently, novel genetic circuits have been constructed based on analog design [3]. Such gene circuits take advantage of the complex operations already naturally present in living cells, to execute sophisticated computational functions. For example, analog genetic circuits exploit positive feedback loops to implement logarithmically linear sensing, addition, division [3] and negative feedback loops while performing square-root calculations to determine chemical concentrations [3]. Analog genetic circuits involve fewer components and resources, and execute more complex operations than their digital counterparts [3]. For an in-depth analysis of the pros and cons of analog versus digital computation in living cells, readers are referred to excellent reviews on the subject [4].

In the present article, we analyze the biophysical and technological limits of large-scale gene networks created based on analog computation in living cells. The working dynamic range, noise margin, basal level of biological parts, sharpness of input-to-output transfer functions and copy number of synthesized proteins/molecules are assessed. In the second part of this paper, we analyze analog computation in living cells.

2. Accuracy of analog systems in living cells:

Computing elements in living cells that based on gen regulation can be described by an enzyme–substrate binding reaction via a Hill function:

$$z - z_0 = z_{max} \frac{(x/K_d)^n}{1+(x/K_d)^n} \qquad (1)$$

where, K_d is a dissociation constant, z_0 is the basal level of binding, z_{max} is the maximum protein concentration achieved by the system, and n is the Hill coefficient. Figure 1a describes equation 1 and it includes two regions: an analog continuous mode and a digital mode. In the analog mode, the function can be described by a log-linear transduction, while in the digital mode, it can be viewed as two discrete values ("0" and "1"). Equation 1 can be approximated at $x=K_d$ ***or*** $(y=ln(x/K_d)=0)$, using Taylor series, as:

$$z - z_0 = \frac{z_{max}}{2}\left(1 + \frac{n}{2} ln(x/K_d)\right) \qquad (2)$$

Log-linear transduction, known as Weber's Law, is widely used in natural systems, such as audition, vision and cells [5], and offers advantages over linear-linear transduction. Naturally, signals propagate through networks with random fluctuations, which can be described by a Poisson process, generating shot noise that scales as the square-root of the molecular count [6]. There are two orthogonal sources of noise in any biological system; intrinsic noise with burst size (b_{int}), generated by the system itself, and the extrinsic noise with b_{ext} size, generated by random fluctuations in the input or environmental parameters [6]. A stochastic model for a system that has a log-linear transduction is given by:

$$\#N_{level} = 1/\sqrt{\frac{8(1+b_{int})}{z_{max}} + \frac{(1+b_{ext})}{K_d} \cdot n^2} \qquad (3)$$

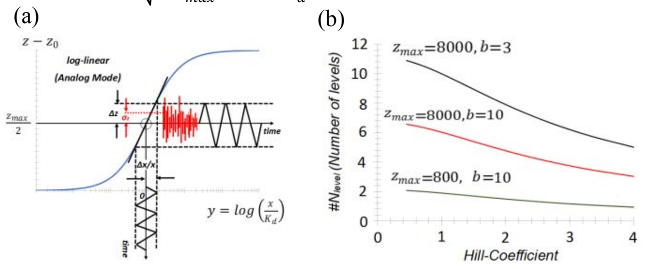


Figure 1. (a) Analog mode: input-to-output transfer function of equation 1 (blue line), log-linear function at y=0 (black line), and Noise analysis of log-linear analog systems, (b) precision of analog systems in a log-linear mode (b_{int}=b_{ext}=b)

$\#N_{level}$ is the number of levels that an analog system can distinguish in the presence of intrinsic and extrinsic noise. The burst size relies on the translation rate, number of amino acids (aa) in the synthesized protein and on half time of mRNA. Typically, in *Escherichia coli*, the translation rate ranges between 10-20 aa/sec, [7], and mRNA half time is around 3-5 min [7] and the burst size range between 3-15. Figure 1b describes the precision of log-linear analog system. To achieve a proper performance of analog systems based on protein-DNA biochemical reactions with 4-8 levels of information (2-3 bits of precision – equation 3), the effective Hill coefficient should be smaller than one, which is challenging to achieve in natural systems.

3. Analog computation in living cells:

Then, the first step toward implementation of synthetic analog computation, is to broaden the input dynamic range of genetic synthetic parts. Protein-DNA interactions typically have a narrow dynamic range, spanning 0.5 - 1 orders of magnitude. The input dynamic range of genetic parts is set by the cooperative binding of proteins to DNA and is often positive, with a Hill coefficient larger than one. *Dainal et al*. [3] implemented a positive feedback loop and decoy binding sites to shunt the proteins away from their target binding site, and achieved a Hill coefficient smaller than 1, with a very wide input dynamic range. In this article, we show a new analytical model that can explain the contribution of a shunt on an open loop and positive feedback loop. Figure 2a describes a transcription factor ***x*** (TF) that binds to ***m*** identical promoters. The ***m-1*** binding reactions act as a decoy or shunt pathway for the transcription factors. For simplicity, we assume that the Hill coefficients for all the reactions are equal to 1. The biochemical reaction model of this system is presented in Figure 2a and its solution in steady state is given by:

$$\frac{Pr_{b1}}{Pr} = \frac{(x_T - m \cdot Pr_{b1})/K_d}{1 + (x_T - m \cdot Pr_{b1})/K_d} \qquad (4)$$

where ***Pr*** is the total number of target promoters, ***Prf*** is the number of free target promoters, ***Prb*** is the number of target promoters occupied by transcription factors, $\boldsymbol{x_T}$ is the total number of transcription factors and $\boldsymbol{K_d}$ is the dissociation constant. Equation 4 can be viewed as a Michaelis–Menten with a negative feedback. The addition of decoy or shunt pathways increases the strength of the negative feedback loop and shifts the switch point of input-output transfer function to higher values (Figure 2b). If we fit the simulation results of Equation 4 to a Hill function, we find that the effective dissociation constant scales with the number of shunt reactions $\boldsymbol{K_{deff}=m \cdot K_d}$. For a very large number of shunt reactions, Equation 4 can be approximated as a linear-linear function (insert of Figure 2b), with a very weak signal.

(a) (b)

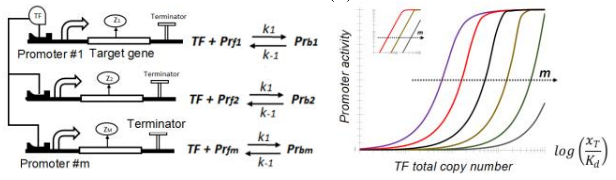


Figure 2. (a) Open loop and shunt circuit: a transcription factor binds to ***m*** identical promoters. (b) Simulation results; contribution of shunt biochemical reactions on promoter activity

To amplify the weak signal of the open loop circuit, a positive feedback loop regulating only the target promoter, was included (Figure 3a) [3]. A simple model of the circuit includes three elements: (1) a linear circuit that demonstrates the contribution of shunt reactions, (2) a positive feedback loop, and (3) inducer-transcription factor binding reaction $\boldsymbol{f(I_n)}$. Then, we can express the solution of a graded positive feedback loop and shunt circuit as:

$$z = \frac{z_0}{1 - \frac{z_{max}}{m \cdot K_d} \cdot f(I_n)} \qquad (5)$$

We can distinguish between two cases: (1) a very strong ($\boldsymbol{z_{max}/m \cdot K_d >> 1}$) positive feedback loop, which yields a sharp input-output transfer function. In this case, the inducer-output protein transfer function is set by both binding reactions: the transcription factor–promoter binding reaction and inducer-transcription factor binding reaction (Figure 3b). (2) A graded positive feedback ($\boldsymbol{z_{max}/m \cdot K_d << 1}$), which yields a log-linear transduction between input and output (Figure 3b). This can be achieved by increasing the number of shunted biochemical reactions, or by decreasing the binding efficiency of transcription factors to the promoter, or decreasing the translation/transcription rates of proteins affecting $\boldsymbol{z_{max}}$. In this case, the inducer-output transfer function is set by the inducer-transcription factor binding reaction only and is given by:

$$z \approx z_0 \cdot \left(1 + \frac{z_{max}}{m \cdot K_d} \cdot f(I_n)\right) \qquad (6)$$

Figure 3b shows that the reduction of $\boldsymbol{z_{max}/m \cdot K_d}$ broadens the input dynamic range. We can see that our analytical model fits (Equation 6) the exact model constructed based on biochemical reactions. The positive feedback loop and shunt circuit cannot widen the input dynamic range more than the dynamic range of the inducer-transcription factor binding reaction. The maximum signal that can be achieved in such a system is $\boldsymbol{z=z_0 \cdot (1+z_{max}/m \cdot Kd)}$, and therefore, the addition of shunt biochemical reactions decreases the signal output. A simple explanation was provided by *Daniel et al* [3], who suggest that the shunt creates several binding sites that delay the saturation of the transcription factor-binding site reaction at the target promoter. At the same time, as the inducer concentration increases, the positive feedback loop enables continuous production of just enough transcription factors.

(a) (b)

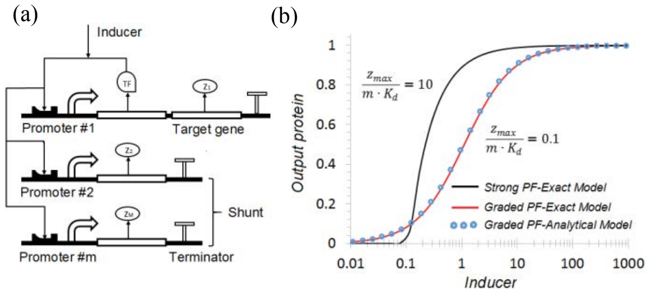


Figure 3. (a) Positive feedback and shunt circuit. (b) Simulation and analytical results showing a graded positive feedback loop.

4. REFERENCES

[1]. Uri, A. 2007. An introduction to systems biology: design principles of biological circuits. Boca Raton, FL: Chapman & Hall/CRC.

[2]. Arkin, A. P and Cardinale S. 2012. Contextualizing context for synthetic biology–identifying causes of failure of synthetic biological systems, Biotechnol. J. 7,7 (Jul 2012), 856-866. (DOI=10.1002/biot.201200085).

[3]. Daniel, R. Rubens, J. R. Sarpeshkar, R. and Lu, T. K. 2013. Synthetic analog computation in living cells. Nature, 497 (2013), 619–623. (DOI=10.1038/nature12148).

[4]. Sarpeshkar R, Analog Synthetic Biology, Phil. Trans. R. Soc. A 2014 372 (2014)

[5]. Ferrell, J.E. 2009. Signaling motifs and Weber's law. Mol. Cell. 36 (2009), 724–727. (DOI=10.1016/j.molcel.2009.11.032).

[6]. Elowitz, M. B. Levine, A. J. Siggia, E. D. Swain, P. S. 2002. Stochastic Gene Expression in a Single Cell. Science. 16, 297 (Aug 2002), 1183.

[7]. Milo, R. and Phillips. R. Cell Biology by Numbers.

Utilizing Signal Temporal Logic to Characterize and Compose Modules in Synthetic Biology

Curtis Madsen[1], Prashant Vaidyanathan[1], Cristian-Ioan Vasile[2], Rachael Ivison[3], Junmin Wang[3], Calin Belta[2,3], and Douglas Densmore[1,4]

[1]Department of Electrical & Computer Engineering, Boston University, Boston, MA
[2]Division of Systems Engineering, Boston University, Boston, MA
[3]Graduate Program in Bioinformatics, Boston University, Boston, MA
[4]Biological Design Center, Boston University, Boston, MA
{ckmadsen,prash,cvasile,rivison,dawang,cbelta,dougd}@bu.edu

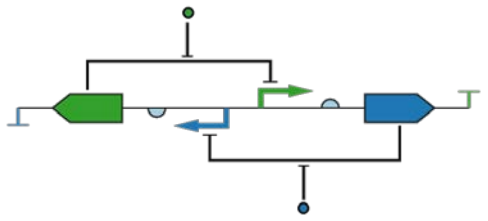
1. INTRODUCTION

The goal of synthetic biology is to allow biologists and engineers to design and build new biological systems. One way this task is achieved is though the composition of DNA segments representing genetic parts and modules. In synthetic biology, parts represent promoters, ribosome binding sites, genes, terminators, etc. while modules are comprised of these parts and include gates, switches, and oscillators. Each of these constructs has a function which can be specified in a formal way using a language such as the hardware description language Verilog. It has been shown that it is possible to reliably synthesize genetic circuits specified in this language using well established methods from logic synthesis in digital electronics [4].

Although previous approaches are very good at predicting the behavior of a designed circuit, they are Boolean in nature and do not include information about the performance of a design. Genetic circuit designs often contain real-time and real-valued constraints that can affect the dynamics of the system with varying levels of magnitude. To improve on previous approaches and include performance metrics in design specifications, temporal logics such as *signal temporal logic* (STL) [3] can be used. STL 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 example, the genetic toggle switch shown in Figure 1(a) can be described by the STL formula in Figure 1(b). This STL formula states that the toggle switch starts in a state where both TetR and aTc are above a value of 30 for 200 time units. Within 200 time units, TetR falls below 30 and stays in that state for 200 time units. At this point, IPTG is added to the system and is held at a value above 30 for 200 time units. TetR is then expected to rise above 30 within 400 time units following the introduction of IPTG.

In the work presented here, we utilize an extension to STL called STL♭ that includes syntax and semantics for composition of genetic components [6]. Using *temporal logic inference* (TLI) [2], we can use experimental data to characterize genetic modules with STL♭ specifications. Our method can then build a design space tree by trying different compositions of the characterized modules. To improve efficiency, the design space tree is automatically pruned using biological constraints for assembly rules and failure mode checks.



(a)

$[G_{[0,200)}(\text{aTc} > 30 \wedge \text{TetR} > 30)] \wedge [F_{[0,200)}G_{[0,200)}(\text{TetR} \leq 30)] \wedge [G_{[400,600)}(\text{IPTG} > 30)] \wedge [F_{[400,800)}(\text{TetR} > 30)]$

(b)

Figure 1: The genetic toggle switch. (a) A physical realization of the genetic toggle switch. (b) An STL specification for the genetic toggle switch.

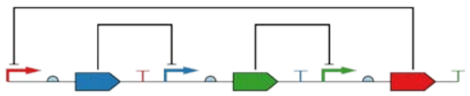
2. WORKFLOW

Given a library of modules and some experimental data, the methodology presented here can be used to characterize the modules with STL♭ specifications. These modules can be composed using a tree-based search and prune design space exploration technique to produce a genetic circuit that implements a desired performance specification given in STL.

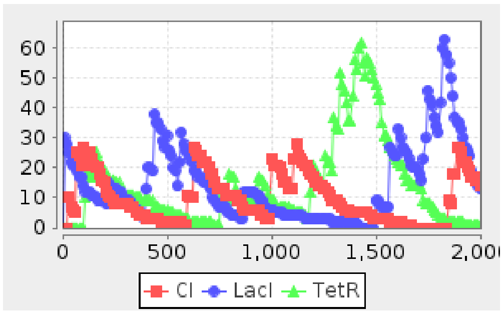
2.1 Characterization of Modules

Our method uses experimental characterization data along with the structural specification of the genetic module to construct a mathematical model representing its function. This mathematical model is simulated to produce traces representing possible behaviors of the system. These traces are passed through TLI to produce an STL♭ specification that captures the behavior of the module. However, TLI requires not only a set of traces for the desired behavior of a system but also requires a set of undesirable or unachievable traces. To address this problem, we have devised an automated method that is capable of producing this set by perturbing the set of traces produced during simulation.

For example, consider the repressilator module shown in Figure 2(a). Using experimental data, a mathematical model for this module can be constructed and simulated resulting in the simulation traces shown in Figure 2(b). This plot shows how LacI, TetR, and λCI oscillate due to the repression ring relationship they have with each other. Using TLI, the SLT♭ specification for the repressilator shown in



(a)



(b)

$$[G_{[0,400)}F_{[0,900)}(\text{LacI} > 15)] \wedge [G_{[0,1200)}F_{[0,900)}(\text{LacI} \leq 15)] \wedge$$
$$[\neg G_{[0,400)}F_{[0,700)}(\text{LacI} > 15)] \wedge [\neg G_{[0,1200)}F_{[0,700)}(\text{LacI} \leq 15)] \wedge$$
$$[G_{[0,400)}F_{[0,900)}(\text{TetR} > 15)] \wedge [G_{[0,1200)}F_{[0,900)}(\text{TetR} \leq 15)] \wedge$$
$$[\neg G_{[0,400)}F_{[0,700)}(\text{TetR} > 15)] \wedge [\neg G_{[0,1200)}F_{[0,700)}(\text{TetR} \leq 15)] \wedge$$
$$[G_{[0,400)}F_{[0,900)}(\lambda\text{CI} > 15)] \wedge [G_{[0,1200)}F_{[0,900)}(\lambda\text{CI} \leq 15)] \wedge$$
$$[\neg G_{[0,400)}F_{[0,700)}(\lambda\text{CI} > 15)] \wedge [\neg G_{[0,1200)}F_{[0,700)}(\lambda\text{CI} \leq 15)]$$

(c)

Figure 2: A diagram showing the steps involved in characterizing a module for the repressilator. (a) A genetic module representing the physical realization of the repressilator. (b) Time series data for the repressilator showing oscillations in the three signals. (c) The STL♭ specification resulting from applying TLI to the data in (b).

Figure 2(c) is generated. This specification can be read as: each signal (LacI, TetR, and λCI) will always eventually rise above a value of 15 within 800 time units and will always eventually fall below a value of 15 within 800 time units.

2.2 Design Space Exploration

The genetic modules in our library can easily be composed using the STL♭ specifications obtained from our characterization method. However, genetic modules may not behave as expected due to physical properties of genetic systems being hard to quantify [5]. Genetic components can also fail due to unanticipated nonmodularity that arises when genetic components are used in new genetic and environmental contexts [1]. To help catalog these scenarios, we have developed grammars for known failure modes, and after each iteration of testing assigned modules *in vivo*, the results of both successful and unsuccessful tests are used to fine-tune a set of rules we use to prune out undesirable or impossible combinations of modules. To name a few pruning rules, our grammars are able to eliminate module combinations that introduce cross-talk, that introduce secondary structures, and that are prone to undesirable homologous recombinations. They additionally consider different ways that modules can be combined and how these different combinations can lead to failure modes such as terminators on one strand of DNA affecting transcription on the other strand.

Figure 3 shows an example of a possible design space tree generated from a set of three modules. In this example, including m_1 and m_2 in the same design would lead to cross-talk as they both produce the same protein. Branches in the design space tree that include both of these modules are pruned and no more exploration is done on these paths. In the worst case, the pruning algorithm is unable to remove any branches; however, in this example, it is able to cut the design space in half by reducing a 15 node tree to 8 nodes.

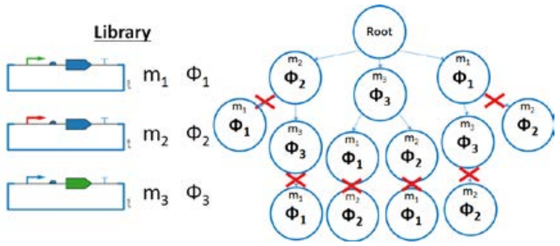


Figure 3: An example of a design space tree that could be generated from a library of three modules and their characterizations (m_1, ϕ_1 through m_3, ϕ_3). In this example, m_1 and m_2 produce the same protein, and therefore, they cannot be composed together in the same design due to problems with cross-talk. As such, branches that would contain both m_1 and m_2 are pruned indicated by the red X's on the design space tree.

3. DISCUSSION

The workflow presented here can be used to: 1) characterize genetic modules with STL♭ specifications, and 2) efficiently explore the design space of an STL specification given a library of characterized modules. Once a set of designs are found, they can be compared against a desired specification using the distance metric found in [6], and the best design can be synthesized in the wet-lab. With this methodology, synthetic biologists will be able to convert physical modules that are currently being stored in a fridge in their laboratory into STL specifications. They will then be able to use these modules to automatically explore the design space of and construct more complex genetic circuit designs.

4. ACKNOWLEDGEMENTS

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

5. REFERENCES

[1] J. A. Brophy et al. Principles of genetic circuit design. *Nature methods*, 11(5):508–520, 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] 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.

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

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

[6] C.-I. Vasile et al. Compositional signal temporal logic with applications to synthetic biology. In *IEEE Conference on Decision and Control (CDC)*, 2016 (Submitted).

Single-Cell Analysis of RNA-engineered Logic Gates

Christopher Schneider*, Jascha Diemer$, Leo Bronstein$, Heinz Koeppl$ and Beatrix Suess*

* Dept. of Biology, Synthetic Genetic Circuits, TU Darmstadt, Schnittspahnstrasse 10, 64287 Darmstadt

$ Dept. of Electrical Engineering, Bioinspired Communication, TU Darmstadt, Rundeturmstrasse 12, 64283 Darmstadt

***bsuess@bio.tu-darmstadt.de $ heinz.koeppl@bcs.tu-darmstadt.de**

ABSTRACT

Here we propose a working strategy to efficiently analyze biological population heterogeneities. We generate data of yeast single-cells by time-resolved flow cytometry and fluorescence microscopy in a microfluidic environment. Stochastic kinetic models are used to compute design regimes for the implementation of reliable logic gates and small genetic circuits.

Keywords

Riboswitch; Modeling; Microfluidics

1. INTRODUCTION

During the past years synthetic biologists supplied this research field with various RNA- and protein-based regulators to control gene expression and to start building artificial genetic circuits *de novo*. There is a vast set of well-working regulatory elements available such as promotors and terminators, transcription factors, riboswitches and reporter genes [2–6]. Although their individual functionality could be shown in a given assay, it is poorly understood how to combine several different genetic parts to set up functional higher-order circuitries from scratch [1]. It is therefore sought to determine and characterize intrinsic and extrinsic parameters that interfere with the construction of genetic circuits. Thus, existing engineered riboswitches and transcription factors are selected and coupled to set up logic entities that allow for the study of simple but versatile fluorescent reporter constructs *in vivo* (Figure 1).

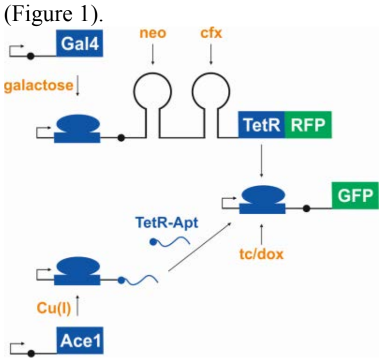


Figure 1. Example of a small genetic circuit operating in *Saccharomyces cerevisiae*. Expression of the RFP-coupled Tet-Repressor is driven by a GAL4 activated promotor and further controlled by neomycin- and ciprofloxacin-sensitive riboswitches constituting a NOR gate. Expression of the ultimate reporter GFP is influenced by the activity of TetR. TetR may be inactivated by the small molecules tetracycline or doxycycline or captured by the inducibly transcribed TetR aptamer.

To infer the influence of genotypic and phenotypic noise imposed by environmental factors and system-inherent kinetic parameters on rationally designed genetic systems, it is mandatory to make use of analytical techniques that operate on a single-cell level. We use flow cytometry and fluorescence microscopy on a microfluidic chip to gain insight into a given cell population and resolve time-dependent changes of the fluorescence signal, respectively. Here, biological CAD is employed in a feed-forward manner to identify those parameters that, once adjusted, may enhance the performance of the engineered circuitry. To validate the functionality of these re-designed genetic switchboards, different reporter cassettes and newly selected riboswitches are inserted.

2. *IN VIVO* EXPERIMENTS

The genetic modules designed and implemented in this study consist of a promotor, riboregulator, terminator and a fluorescent reporter gene. Depending on the context, a variety of logic gates can be realized by a differential coupling of these parts. All measurements are performed with baker's yeast and the fluorescence signal is recorded.

2.1 Bulk Measurements

For prototyping reasons, modules are first assembled on plasmids and analyzed as bulk. To exemplify the design of a module, a constructed NOR gate is displayed in figure 2.

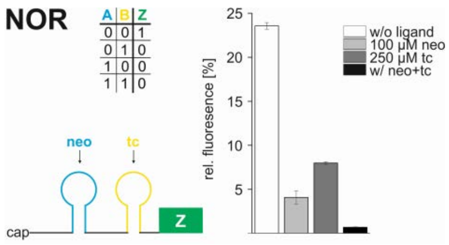


Figure 2. NOR gate assembled from a neomycin- and tetracycline-sensitive riboswitch and analyzed for GFP fluorescence levels (Z). Without input A or B about 23% of the fluorescence generated by a construct without NOR gate may be reached. Ligand addition reduces the fluorescence signal accordingly.

2.2 Single-Cell Measurements

Cells assayed for bulk fluorescence are further subjected to single-cell measurements by flow-cytometry (Figure 3). From these data a vast phenotypic heterogeneity can be deduced. Since a primary goal of this study is to identify, model and tune parameters responsible for cell-to-cell variability, this is a major issue to be addressed and may not only be solved by a genomic integration of the genetic modules, thus decreasing genotypic noise by a copy number reduction. A detailed modeling of kinetic parameters that include synthesis and degradation rates of protein and mRNA as well as reaction constants for the riboswitch-ligand interaction are to be assessed to reveal further target sites.

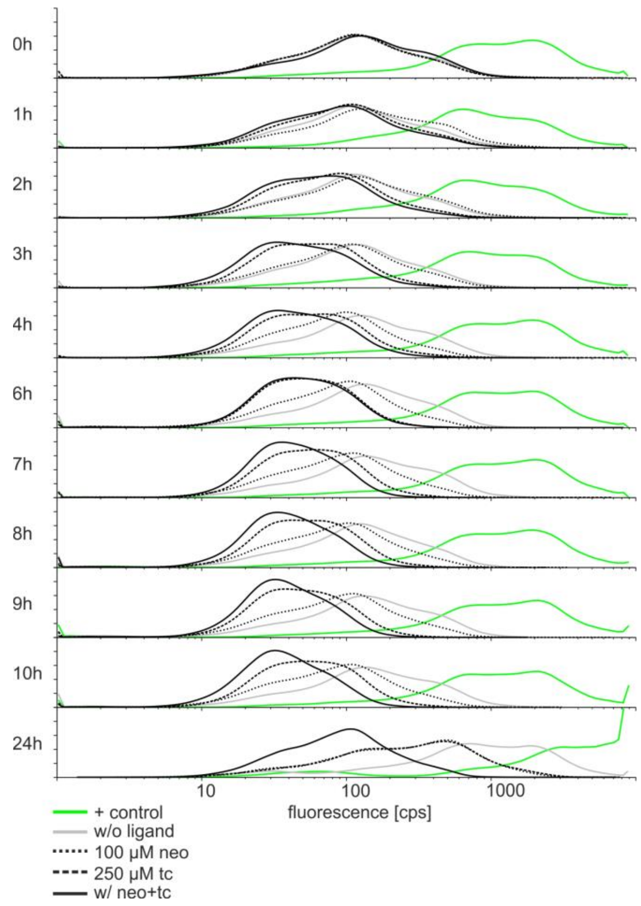


Figure 3. Histogram of NOR gate from figure 2 expressed in yeast and measured by time-resolved flow cytometry. Samples were drawn from a continuous yeast culture during exponential growth (0-10h) and steady state (24h) phase.

3. MICROFLUIDIC SYSTEM

In order to capture cell-to-cell heterogeneity we utilize microfluidic chips for single-cell analysis. Yeast cells are trapped by PDMS structures and cultivated inside the chip.

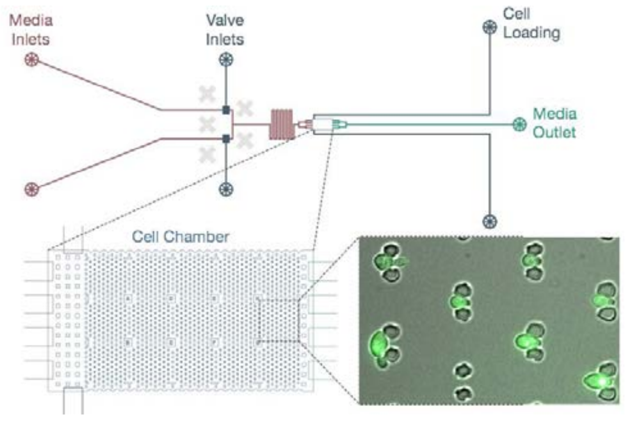


Figure 4. PDMS chip as microfluidic device. Yeast cells are trapped and tracked over time.

These structures are flow-optimized to trap single cells, are bypassed if occupied and allow the washout of daughter cells (Figure 4). The mother cell is recorded with a microscope up to days and is supplied with new nutrients. We adapted the principle of pulse width modulation from electrical engineering to change the concentration of ligands within seconds with valves directly embedded in the chip. The microfluidic device provides good control over cell growth and ligand concentration and helps to automatize experiments and their analysis. Current work includes optimization of the experimental setup and an image processing pipeline to extract changes in fluorescence for each individual cell over time.

4. MODELING

Stochastic kinetics models of the logic gate variants accounting for population heterogeneity are built. In particular, extrinsic noise is captured through cell specific kinetic parameters (e.g. translation rate, ligand concentration) and through plasmid copy number variations. We adopt the moment-based approach of Zechner et al. to allow fast simulation and testing of different parameter sets and riboswitch architectures [7]. Presented time-lapsed flow-cytometry data is used to calibrate extrinsic and intrinsic model parameters using Markov chain Monte Carlo techniques. Based on the obtained parameter posterior distribution, average sensitivity coefficients for all experimentally accessible parameters are computed. The resulting sensitivity scores are used to rank candidate experimental redesigns of the gates. Through the incorporation of extrinsic parameters, the computational analysis also pinpoints to the most effective redesign options for reducing the large observed cell-to-cell variability- thus making the gate more reliable on the single-cell level.

5. REFERENCES

[1] Berens, C. and Suess, B. 2015. Riboswitch engineering - making the all-important second and third steps. *Current opinion in biotechnology*. 31C, (Feb. 2015), 10–15.

[2] Callura, J.M. et al. 2012. Genetic switchboard for synthetic biology applications. *Proceedings of the National Academy of Sciences of the United States of America*. 109, 15 (Apr. 2012), 5850–5.

[3] Chang, A.L. et al. 2012. Synthetic RNA switches as a tool for temporal and spatial control over gene expression. *Current opinion in biotechnology*. 23, 5 (Oct. 2012), 679–88.

[4] Groher, F. and Suess, B. 2014. Synthetic riboswitches - A tool comes of age. *Biochimica et biophysica acta*. 1839, 10 (Oct. 2014), 964–973.

[5] Lee, S. et al. 2013. Improved blue, green, and red fluorescent protein tagging vectors for S. cerevisiae. *PloS one*. 8, 7 (Jan. 2013), e67902.

[6] Pardee, K. et al. 2014. Paper-Based Synthetic Gene Networks. *Cell*. 159, 4 (Oct. 2014), 940–54.

[7] Zechner, C. et al. 2012. Moment-based inference predicts bimodality in transient gene expression. 109, 21 (2012).

BioBlocks: A web-based visual environment for programming experimental protocols in biological sciences

Vishal Gupta
vishal.gupta@upm.es

Jesús Irimia
j.irimia@alumnos.upm.es

Iván Pau
ivan.pau@upm.es

Alfonso Rodríguez-Patón
arpaton@fi.upm.es

Laboratorio de Inteligencia Artificial
Universidad Politécnica de Madrid

ABSTRACT

The method of research and experimentation in biological sciences is evolving very fast. Fluid handling robots and on-demand biology enterprises are automating biological experiments as an alternative to traditional manual protocol execution. There are various new high-level languages like Autoprotocols and Antha which allow for automation of protocol execution. These languages offer various benefits to the user like high-throughput experimentation, rapid prototyping and improved reproducibility of results. However, learning these new languages to automate protocols is not a trivial task for a non-computer scientist and using multiple languages can burden the user, as different languages are compatible with proprietary hardware platforms. To overcome this, we have developed an open-source web-based visual editor called BioBlocks for describing experimental protocols in biology. It is a visual high-level programming language which requires little or no programming knowledge to automate the execution of protocols locally or remotely i.e. in the cloud. The experiments are automatically translated to different robotic languages making BioBlocks a useful upper layer to robotic GUIs. BioBlocks allows the users to define complex experiments such as turbidostats and chemostats in a modular fashion. The experiments can saved, modified and shared between multiple users to execute on compatible platforms hence improving the reproducibility of their research.

Keywords

Lab Automation, Lab Protocols, Blockly, BioBlocks, Rapid prototyping, High-level programming language, Reproducibility in Biology.

1. INTRODUCTION

Reproducibility of experimental results has long been the elephant in the room plaguing biological sciences. With the cost of research and development increasing, the inability to reproduce the results of biological research has become a critical issue to address because of its economic and scientific impact[1]. There are many factors that contribute to the problem of reproducibility; the ambiguity introduced by natural languages (English) when describing methods, the person-to-person variability while carrying out experiments, inadequate data sharing, etc.

Many interesting approaches have been proposed to tackle these problems such as use of programming languages for the description of biological protocols to reduce ambiguity[2], automation of experiments via robotic execution to reduce human error[3], improved data sharing and representation[4] and applying quality control procedures[5]. Programming languages allow the description of biological methods in an unambiguous manner which reduces the possibility of misinterpretation. Further, the execution of protocols described using programming languages can be automated as their description (code) is machine-readable. There have been many efforts in this direction like BioCoder[2] and Puppeteer (for protocol description)[6] and AquaCore[7] and Par-Par[3] (for protocol automation). However, this approach has not been entirely successful because it requires the user (biologist) to have prior knowledge of programming[8,9].

2. RESULTS

Here, we present an open-source web-based tool for describing experimental protocols in biology. It is based on Google's Blockly[10], an open-source framework for building visual programming editors. Like other similar tools[11], it consists of a toolbox of jigsaw-like blocks which can be linked together to generate code in multiple languages. We have used it to develop a new visual programming environment with a new library of blocks called 'BioBlocks' which allows for description of experimental protocols by linking blocks in a simple drag-and-drop manner. The logic of BioBlocks is largely based on Autoprotocol[9], a language developed by Transcriptic[12] for specifying experimental protocols in biology.

The BioBlocks can be roughly divided into two-types of blocks namely, 'container blocks' which represent containers like 96-well plates, tubes and beakers and 'operation blocks' which contain common procedures (actions) carried out during experimentation like pipetting, measuring, electrophoresis etc. The blocks have been customized in a manner that prohibits linking of two incompatible blocks (the blocks snap away). The protocols described using BioBlocks are automatically translated in real-time to simultaneously generate three different types of output. The first output is a natural language (English) description of the protocol to aid in verification. The output is in the conventional format consisting of step-wise description of the protocol. The second output is the representation of the protocol as a workflow. It is based on mathematical graphs, where the nodes and edges represent the containers and the action performed over the containers respectively. This provides the user more insight into planning and executing the protocol. The last type of output is a machine-readable code of the protocol for its automated execution. As a proof of principle, the output code is generated using JSON syntax in two modes. The first mode is fully compatible with Autoprotocol, potentially allowing for remote execution of the described protocols at Transcriptics which is a commercial lab-in-a-cloud company that uses Autoprotocol syntax. The second mode is an extension of Autoprotocol which allows the description of protocols requiring feedback during execution e.g. continuous culture devices like turbidostats and chemostats. The users have the option to customize the blocks to generate code specific to their robotic platforms which would then allow automatic translation and execution of their protocols.

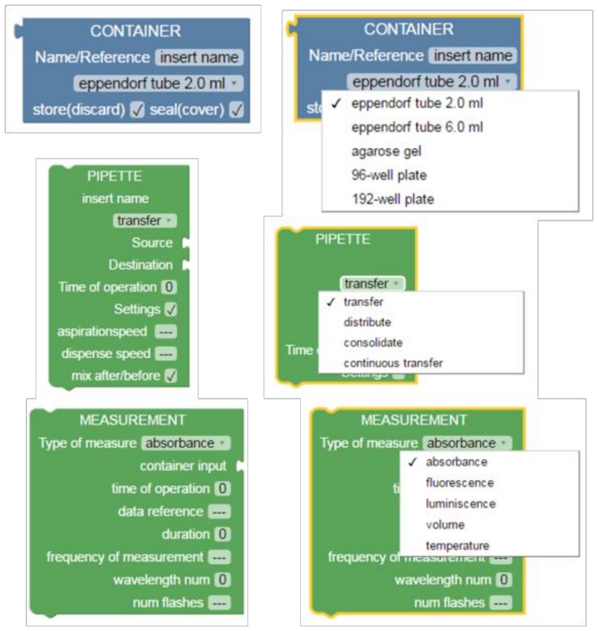


Figure 1 BioBlocks: A few blocks from the BioBlocks library are shown. The container blocks (blue) can be linked to the operation blocks (green) which in turn can be linked to each other to form complex protocols.

The definition of constraints in the design of the blocks helps avoid syntactic and logical errors.. Syntactic errors are avoided because the code is generated in an automated manner and also due to the domain-specific customization of blocks i.e. the experimental biology domain. For example, certain operations like electrophoresis are compatible only with a specific type of container like agarose gels. Therefore, a user is not permitted to link containers representing multi-well plates or tubes to an electrophoresis block. Logical errors like overdrawing and under drawing fluid volumes can also be avoided. The constraints are system-wide encoded in the blocks but since the software is open-source the user can create new blocks with different functionalities with a novel set of constraints or reuse/modify the existing constraints.

BioBlocks allow non-programmers to describe complex conditional protocols with little effort. Experiments such as chemostats, turbidostats etc. which are being increasingly used for continuous evolution of genetic circuits and orthogonal sensors can be easily described. The real-time feedback and control of experiments, enables the user to guide the experiment based on real-time data, to obtain the best results. The protocols in BioBlocks are modular in nature and can be re-used in combination with other protocols by simply linking them together, to create new complex protocols. Care has been taken to ensure that visual manipulation of large protocol is easy; protocols described in steps/smaller modules can be collapsed to allow for easy navigation between different parts of a protocol. They can be saved, retrieved, modified and shared between multiple users. As the DIY community of making open and 3D printable lab[13,14] machines grows, BioBlocks would be very helpful for biologists to use it to operate the machines.

3. CONCLUSION

Here, we present a web-based visual programming environment that addresses the problem of reproducibility by reducing ambiguity and minimizing human error using automation. On the front end, it is a visual programming interface, which allows for the precise description of biological protocols in a simple and unambiguous manner. On the back end, the software system allows for the automated execution of the entire experiment on a compatible hardware platform hence allowing for rapid prototyping of biological experiments. This work is an attempt to make it easier for biologists to automate their experiments, without the user requiring little or no programming knowledge, allowing them to connect to multiple academic and commercial solutions.

4. SOFTWARE

Software available soon on our webpage-
http://www.lia.upm.es/index.php/software/Bioblocks

5. ACKNOWLEDGMENTS

This work was partially funded by EU FP7–FET-Proactive Project 610730 EVOPROG grant, Spanish National Project TIN2012 - 36992 grant.

6. REFERENCES

1. Freedman, L. P., Cockburn, I. M. & Simcoe, T. S. The Economics of Reproducibility in Preclinical Research. *PLOS Biol*. **13,** e1002165 (2015).
2. Ananthanarayanan, V. & Thies, W. Biocoder: A programming language for standardizing and automating biology protocols. *J. Biol. Eng*. **4,** 13 (2010).
3. Linshiz, G. *et al*. PaR-PaR laboratory automation platform. *ACS Synth. Biol*. **2,** 216–222 (2013).
4. Soldatova, L. N., Aubrey, W., King, R. D. & Clare, A. The EXACT description of biomedical protocols. *Bioinformatics* **24,** 295–303 (2008).
5. Sadowski, M. I., Grant, C. & Fell, T. S. Harnessing QbD, Programming Languages, and Automation for Reproducible Biology. *Trends Biotechnol*. **xx,** 1–14 (2015).
6. Yaman, F., Bhatia, S., Adler, A., Densmore, D. & Beal, J. Automated selection of synthetic biology parts for genetic regulatory networks. *ACS Synth. Biol*. **1,** 332–344 (2012).
7. Amin, A., Thottethodi, M., Vijaykumar, T., Wereley, S. & Jacobson, S. C. Aquacore: a programmable architecture for microfluidics. *Proc. 34th Annu. Int. Symp. Comput. Archit.* 254–265 (2007). doi:10.1145/1250662.1250694
8. Antha. at <https://www.antha-lang.org/>
9. Autoprotocol. at <http://autoprotocol.org/>
10. Blockly. at <https://developers.google.com/blockly/>
11. MIT Scratch. at <https://scratch.mit.edu/>
12. Transcriptic. at <https://www.transcriptic.com/>
13. OpenTrons. at <http://www.opentrons.com/>
14. Takahashi, C. N., Miller, A. W., Ekness, F., Dunham, M. J. & Klavins, E. A Low Cost, Customizable Turbidostat for Use in Synthetic Circuit Characterization. *ACS Synth. Biol*. (2014).

An Environment for Augmented Biodesign Using Integrated Data Resources

James Alastair McLaughlin
ICOS, School of Computing Science
Newcastle University, UK
j.a.mclaughlin@ncl.ac.uk

Göksel Mısırlı
ICOS, School of Computing Science
Newcastle University,UK
goksel.misirli@ncl.ac.uk

Matthew Pocock
Turing Ate My Hamster Ltd, UK
turingatemyhamster@gmail.com

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

1. INTRODUCTION

Systematic design is fundamental to synthetic biology. The traditional bottom-up approach to synthetic biology design process involves the designer selecting and assembling the genetic parts that they think will encode the desired behaviour of the intended system at a genetic level. This process relies heavily on the users knowledge of the parts behaviour and understanding of the biological context (i.e. chassis) in which the synthetic system will be deployed. The designer has the task of trying to enhance their knowledge from the wide of resources that are available in the numerous databases, websites and scientific literature reports. This is an arduous task and a rate limiting step in the design of biological systems.

Understanding novel data in the context of external, existing data to gain knowledge has been one of the major features of bioinformatics data analysis for many years. The approach has been to integrate data from disparate, heterogeneous and heterologous datasets to provide an integrated view of a certain area of biology. The integration is carried out using a variety of approaches, but two major methods have emerged as the most common; that of data warehousing where all data is gathered from multiple data sources into one database that can be mined to gain knowledge, and data federation, where limited datasets are drawn from multiple, remote data databases and integrated on demand in response to a particular query. Each of these approaches have advantages and disadvantages. For example, the data warehousing approach requires that the integrated dataset is updated periodically as the source data changes and the federated approach can suffer from reduced performance due to the need to query multiple, remote, data sources simultaneously. Once integrated these data help a scientist access data that enhances their knowledge of a biological domain and helps further insights to be gained from the analysis of novel datasets. The use of integrated data to help synthetic biologists design biological systems is a promising approach that has still to be exploited and research in this area has been very limited to date.

Recently, we have developed two systems that provide key components in the infrastructure necessary to allow a user to access data that will enhance the design process, so called data augmented biodesign (Figure 1). Firstly, we have developed an ontology for synthetic biology (SyBiOnt) [3] that allows both the entities and the relationships between those entities captured in a computationally tractable format. This ontology provides a uniform data model into which multiple heterogeneous data sources can be mapped and integrated to produce a data model for a data warehouse. Secondly, we have also produced a data warehouse, termed the SBOL Stack [2], which stores synthetic biology part definitions that have been mapped into the SynBiOnt data format but, in addition, stores a variety of different data sources in an integrated format. This data warehouse can be used not only to store data about synthetic biology parts, but also to store data that can be used to provide additional information about those parts and their behaviour, and also data that can be mined to produce new parts.

Here we describe a Web-based portal, AmBiT (Augmented Biodesign Environment), that provides an environ-

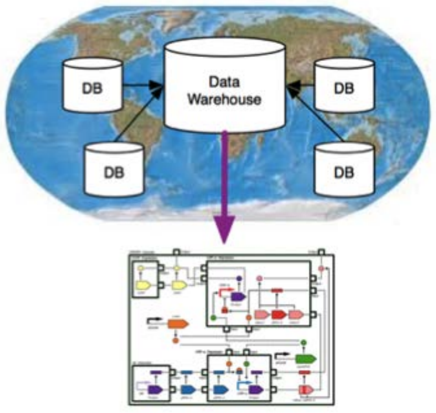


Figure 1: Data augmented biodesign. Data from distributed databases is gathered, integrated and used to aid synthetic biology systems design.

*To whom correspondance should be addressed

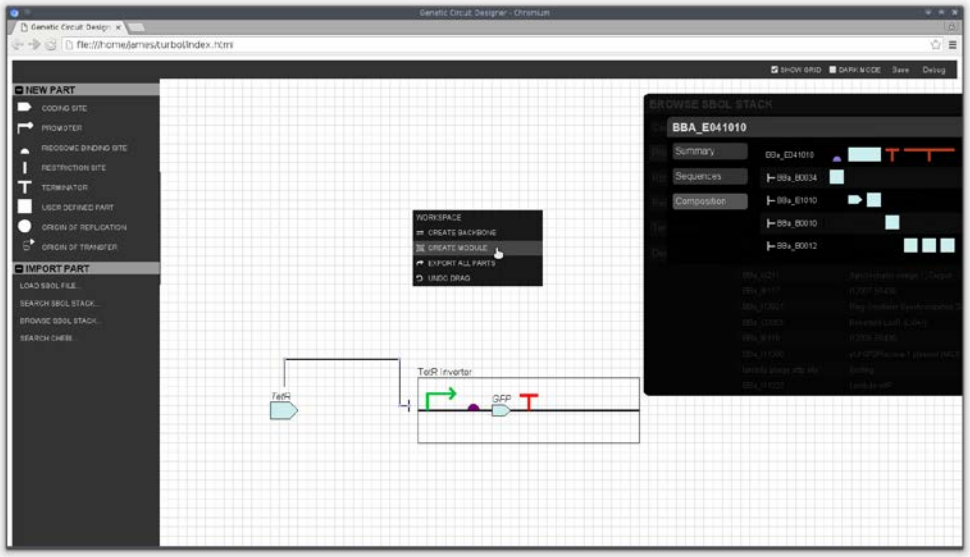


Figure 2: The AmBiT design environment. Systems can be designed at a genetic level and additional data is provided to augment the process. In this example, information about the interaction of TetR with a TetR regulated promoter is automatically supplied from the KEGG database.

ment for the design of biological systems where the user can access a wealth of data, drawn from multiple data sources, to aid them in the design of their system by augmenting their existing knowledge with new data.

2. THE AMBIT SYSTEM

The AmBiT system provides a Web based environment that allows the genetic design of a system using the familiar diagrammatic drag and drop approach that has become a well accepted approach employed by many CAD tools. In a similar fashion to many other systems, genetic parts can be specified *de novo* or selected from a palette of existing parts from the menu provided.

However, AmBiT is linked to a data warehouse in the form of the SBOL Stack which provides a large library of thousands of genetic parts and, importantly, integrated data about those parts drawn from a large range of online data sources. This means that the genetic designs produced can be enhanced by a rich linked data model complete with sequence information, annotations, and cross-references to databases. As the user adds parts and devices to the design the system automatically mines the AmBiT repository to provide additional information about the design to the user. For example, proteins that are encoded by a coding sequence can be added automatically since they are stored in the integrated databases (drawn from GenBank and UniProt), and interactions of those proteins with other entities in the design can automatically be provided. The resulting design can be stored back in the AmBiT SBOL stack or exported in the SBOL2.0 [1] format to share with other computational tools. The provenance of the added information can be added to the SBOL2.0 in the form of annotations. The system also has the capability of importing a basic SBOL2.0 formatted design and adding further interactions and annotations to enhance this design, before re-exporting the enriched synthetic system, again in SBOL2.0 format.

The current version of AmBiT includes parts and devices mined from the *Bacillus subtilis* and *Escherichia coli* genomes. Also included are data from 10 disparate data sources, together with over 20,000 parts and devices from the current (2015) iGEM parts registry.

3. CONCLUSIONS

The provision of relevant data to a user has the potential to drastically improve the design process, potentially enhancing both the accuracy and scale of the genetic systems produced. To meet this challenge data needs to be automatically mined and integrated from multiple data sources and presented to the user in a friendly and intuitive environment. The AmBiT system provides such an environment by utilising a custom built data warehouse implemented in the SBOL Stack and by utilising the SyBiOnt ontology to mediate the data integration process.

4. ACKNOWLEDGMENTS

The Engineering and Physical Sciences Research Council grant EP/J02175X/1 (to A.W. and G.M.).

5. REFERENCES

[1] B. Bartley et al. Synthetic Biology Open Language (SBOL) Version 2.0.0. *J Integr Bioinform*, 12(2):272, 2015.
[2] C. Madsen et al. Sbol stack: The one-stop-shop for storing and publishing synthetic biology designs. In *7th International Workshop on Bio-Design Automation*, 2015.
[3] G. Misirli et al. Data integration and mining for synthetic biology design, submitted. *ACS Synth Biol*, 2016.

extFogLight: Using Weighted Metabolic AND/OR Graph to Find Stoichiometrically Balanced Pathways

Mehrshad Khosraviani, Morteza Saheb Zamani

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

{ mkhosraviani, szamani }@aut.ac.ir

ABSTRACT

Using a specialized AND/OR graph based on the stoichiometric principle, a novel representation of biochemical reactions is proposed. This enables us to extend the basic version of FogLight, as a pathfinding approach, so that it can deal concurrently with the stoichiometries and the process of pathfinding. This approach stoichiometrically guarantees the balance of found metabolic pathways without significant time/space cost.

Categories and Subject Descriptors

• **Computing methodologies → Modeling and simulation → Model development and analysis → Modeling methodologies**

• **Applied computing → Life and medical sciences → Bioinformatics**

Keywords

Synthetic biology; metabolic networks; enzymatic metabolic pathways; weighted AND/OR graph, stoichiometrically balanced pathway.

1. INTRODUCTION

Synthetic biology is an evolving research field that combines the investigative nature of biology with the constructive nature of engineering. It treats living systems as a hierarchy of functional modules one of which is metabolic networks [1]. The engineering of these networks is defined as the optimization of its enzymatic pathways to ensure that their product compounds are produced at a high yield and in high titers by cell factories [2]. Many scientists are interested in finding the pathways if they exist.

Two complementary approaches have been used to answer the above question: constraint- and graph theory-based methods [3]. The graph-based pathfinding approaches focus on discovering pathways without considering fundamental principles of the living factories, such as stoichiometric principles. However, some of the principles can be considered after doing the process of pathfinding. The algorithm proposed in [4], namely FogLight, used this strategy to verify the found metabolic pathways according to stoichiometry constraints with the aim of satisfying steady-state condition. For ensuring the steady-state condition, a system of linear equations with n equations and n variables (where n is the number of reactions) is solved as a post-process step. The result of this procedure is a sequence of biochemical reactions which produce a compound from another.

For the purpose of finding balanced pathways, we have extended FogLight to take stoichiometries into account during the metabolic pathfinding process rather than after it. The advantage of the extended version of FogLight, namely extFogLight, is the ability of finding balanced pathways with concurrent consideration of the steady-state constraints and the process of metabolic pathfinding, even though FogLight leads to the same results through a post-process step. We describe the reformed modules of extFogLight in the following section. To this end, the AND/OR graph model of the metabolic network utilized by FogLight, must also be generalized based on the stoichiometric principles to be used in extFogLight.

As discussed in [4], a metabolic network is visualized as a set of Boolean functions consisting of AND/OR operations with two or more variable. As a simple example, in the biochemical reaction $m_1 + m_2 \rightarrow m_3$, the metabolite m_3 is produced if the metabolites m_1 and m_2 are both present. That is, the relation between these three metabolites is given by the Boolean function $m_3 = m_1 \text{ AND } m_2$. Conversely, considering two reactions $m_1 \rightarrow m_3$ and $m_2 \rightarrow m_3$, the relation between the corresponding metabolites can be interpreted as the Boolean function $m_3 = m_1 \text{ OR } m_2$, which means the production of m_3 depends on the existence of m_1 or m_2. In this paper, to find stoichiometrically balanced pathways, we generalized the original model of metabolic AND/OR graph to include stoichiometries, a piece of quantitative information, and finally provide an appropriate procedure to assess the elemental balances. The generalized form of the model associates another label (weight) with every edge in the graph which will be described in detail.

2. MATERIAL AND METHOD

2.1 Generalized Representation

Each metabolic network can be represented by a circuit graph. Here, we extend our mathematical definition of the metabolic AND/OR circuit graph $G = (\mathbb{V}, \mathbb{A}, \mathbb{L}_{\mathbb{V}}, \mathbb{L}_{\mathbb{A}})$, provided in [1], to 5-tuple $G = (\mathbb{V}, \mathbb{A}, \mathbb{L}_{\mathbb{V}}, \mathbb{L}_{\mathbb{A}}, \mathbb{W}_{\mathbb{A}})$ as follows:

- $\mathbb{V}$ and $\mathbb{A} = \{(v_i, v_j) : v_i, v_j \in \mathbb{V} \textit{ and } (v_i, v_j) \neq (v_j, v_i)\}$ are nonempty sets of vertices (metabolites) and arcs (biochemical reactions), respectively.
- $\mathbb{L}_V$ and $\mathbb{L}_A$ are two disjoint nonempty sets of the unique identifiers for labeling the vertices and the arcs within G, respectively, based on the name of the metabolites and their related reactions.
- $\mathbb{W}_{\mathbb{A}}$ describes a nonempty set of weights (denoted by $\mathcal{W}(v_i, v_j)$), consisting of the ordered pairs $(w_i : w_j)$ associated with each arc $(v_i, v_j) \in \mathbb{A}$, in which the integer numbers w_i and w_j respectively represent stoichiometric coefficients of the vertices (metabolites) $v_i \in \mathbb{V}$ and $v_j \in \mathbb{V}$ participating in reaction $\ell_e \in \mathbb{L}_{\mathbb{A}}$.

2.2 extFogLight: An approach based on the weighted metabolic AND/OR graph

Elementary biochemical reaction $\mathtt{R_i}$, the $\mathtt{i}$-th step of a given metabolic pathway, can be represented by the equation of the following form:

$$w_{s1}S_1 + w_{s2}S_2 + \cdots + w_{sm}S_m \rightarrow w_{p1}P_1 + w_{p2}P_2 + \cdots + w_{sn}P_n$$

Considering our proposed weighted AND/OR graph in Section 2.1, $\mathcal{W}(S_i^{\mathtt{R_i}}, P_j^{\mathtt{R_i}}) = (w_{si}^{\mathtt{R_i}}, w_{pj}^{\mathtt{R_i}})$ for $1 \le i \le m$ and $1 \le j \le n$, in which $w_{si}^{\mathtt{R_i}}$ and $w_{pj}^{\mathtt{R_i}}$ represent stoichiometric coefficients of substrate S_i and product P_j in bioreaction $\mathtt{R_i}$, respectively.

The stoichiometric imbalance of a sequence of elementary biochemical reactions can be due to the less molar production of a main compound in bioreaction $\mathtt{R_i}$ compared to the consumption of the same compound in the next reaction, *i.e.*, $\mathtt{R_{i+1}}$. In other words, based on the aforementioned equation, we would have no balanced pathway if condition $w_{pj}^{\mathtt{R_i}} < w_{si}^{\mathtt{R_{i+1}}}$, for $P_j = S_i$ is satisfied. Algorithm 1, Lines 8-9, 17-18 and 21-22, depict our solution to this problem by using $sc = w_{si}^{\mathtt{R_{i+1}}} / w_{pj}^{\mathtt{R_i}}$, for $P_j = S_i$, resulted from the mentioned inequality condition. There is an easy, but efficient, solution that helps us to simply overcome the imbalance problem. To this end, coefficient $sc > 1$ is multiplied by the stoichiometric values of the reactants of reaction $\mathtt{R_i}$.

Since extFogLight is the extended version of FogLight, the process of finding the metabolic pathways is retrosynthetically (*i.e.*, starting from a target product $v \in \mathbb{V}_H \subset \mathbb{V}_G$ to a source) done through the subgraph H of a graph G. Accordingly, Line 2 of Algorithm 1 shows this procedure which is fully explained in [4].

As stated before, the search space was modeled by a generalized AND/OR graph in which 'AND' and 'OR' (inspired from the two familiar Boolean gates) determine the types of searching process steps that should be followed through. Lines 6-7 and 20 show how extFogLight deals with the multiple different inputs of the now- and new-processing 'AND' gates. As illustrated in Algorithm 1, output of extFogLight will be a set of the different pathways which each stack (denoted by σ) includes one of them. If the label of a new-processing input $(u,v) \in \mathbb{A}_H \subset \mathbb{A}_G$ currently existed in stack σ_{front}, it is pushed. Otherwise, the label $\ell_{(u,v)} \in \mathbb{L}_{\mathbb{A}}$ is compared with the labels of the arc on the top of the last stack (σ_{rear}) and then it is placed on the top of it if they are identical.

On the other hand, Lines 12-16 focus on handling new 'OR' gate. As each of the inputted arc to an 'OR' gate may be resulted in discovering a new pathway, this arc and copy of the other arcs saved in the now-processing stack must be processed separately.

Algorithm 1 extFogLight; an extended version of FogLight

```
     Input: $G = (\mathbb{V}_G, \mathbb{A}_G, \mathbb{L}_{\mathbb{V}}, \mathbb{L}_{\mathbb{A}}, \mathbb{W}_{\mathbb{A}})$, $v = target$, $\mathtt{i} = 23$
     Output: List of the found balanced pathways $Sol[\,]$
1:   begin
2:   $H :=$ PRUNE_METABOLNET($G$)
3:   while ( queue of stacks $\sigma$ is not empty ) do
4:       for each arc $(u,v)$ do
5:           $(u',v') :=$ TOP ($\sigma_{front}$)
6:           if ( $v \ne v'$ or $\ell_{(u,v)} = \ell_{(u',v')}$ ) then
7:               PUSH (($u, v$), $\sigma_{front}$)
8:               if ($w_{u'}^{\mathtt{R_{i+1}}} > w_v^{\mathtt{R_i}}$) then
9:                   $\mathcal{W}(u,v) := w_{u'}^{\mathtt{R_{i+1}}} \cdot \mathcal{W}(u,v) / w_v^{\mathtt{R_i}}$
10:          else
11:              $(x,y) :=$ TOP ($\sigma_{rear}$)
12:              if ( $\ell_{(u,v)} \ne \ell_{(x,y)}$ ) then
13:                  $\sigma_{rear+1} := \sigma_{front}$
14:                  POP ( [ ] $\rightarrow v$, $\sigma_{rear+1}$)
15:                  $(z,t) :=$ TOP ($\sigma_{front}$)
16:                  PUSH (($u, v$), $\sigma_{rear+1}$)
17:                  if ( $w_z^{\mathtt{R_{i+1}}} > w_v^{\mathtt{R_i}}$ ) then
18:                      $\mathcal{W}(u,v) := w_z^{\mathtt{R_{i+1}}} \cdot \mathcal{W}(u,v) / w_v^{\mathtt{R_i}}$
19:              else
20:                  PUSH (($u, v$), $\sigma_{rear}$)
21:                  if ($w_x^{\mathtt{R_{i+1}}} > w_v^{\mathtt{R_i}}$) then
22:                      $\mathcal{W}(u,v) := w_x^{\mathtt{R_{i+1}}} \cdot \mathcal{W}(u,v) / w_v^{\mathtt{R_i}}$
23:          if ( stack $\sigma_{front}$ includes a wrong pathway ) then
24:              EMPTY ($\sigma_{front}$)
25:          else
26:              while ( arcs with a common target is not observed ) do
27:                  $Sol_{front} :=$ POP ($\sigma_{front}$)
28:          if ( $\sigma_{front} = \emptyset$ ) then $front := front + 1$
```

3. RESULTS AND DISCUSSION

Using extFogLight, we were able to find some balanced metabolic pathways from glucose to valine and AMP with no significant time/space overhead in comparison with FogLight. Moreover, as extFogLight inherits the solution space (a reduced space [4]) from its predecessor, it can find stoichiometrically balanced pathways without missing any admissible ones.

Details of the shortest pathway found by extFogLight within the KEGG compound network related to conversion from glucose (`C00031`) to valine (`C00183`) have been shown below:

$\mathtt{C00031} \rightarrow \mathtt{C00198} \rightarrow \mathtt{C00257} \rightarrow \mathtt{C00204} \xrightarrow{\mathtt{R08570}} \mathtt{C00022} \xrightarrow{\mathtt{R00006}} \mathtt{C00900} \rightarrow \mathtt{C04039} \rightarrow \mathtt{C00141} \rightarrow \mathtt{C00183}$

While in bioreaction `R00006` two moles of `C00022` disappear for appearance of two moles `C00900`, in `R08570` only one mole of `C00022` is produced and this amount is not sufficient for `R00006`. This inequality in production and consumption of an intermediate reactant causes imbalance problem of the pathway which it can be solved by increasing rates of the first four reactions twice the rates of the last four reactions. As a result, the flux vector of the above pathway will be $[2,2,2,2,1,1,1,1]^T$ obtained by extFogLight.

Glucose/AMP is another pair of source/target metabolites between which extFogLight looks for the balanced pathways. The results, including the pathways and their rate vectors, are as following:

Pathway 1, $[2,2,2,2,2,2,1]^T$: `C00031` → `C00668` → `C00103` → `C00029` → `C00015` → `C00009` → `C00008` → `C00020`

Pathway 2, $[1,1,1,1,1,1,1]^T$: `C00031` → `C00668` → `C00103` → `C00029` → `C00015` → `C00046` → `C00002` → `C00020`

4. REFERENCES

[1] Crook, N. and Hal, S.A. 2013. Model-based design of synthetic, biological systems. *Chemical Engineering Science*. 103, 2-11.

[2] Nielsen, J. and Keasling, J.D. 2011. Synergies between synthetic biology and metabolic engineering. *Nature Biotechnology*. 29, 8, 693-695.

[3] Pitkänen, E., Jouhten, P. and Rousu, J. 2009. Inferring branching pathways in genome-scale metabolic networks. *BMC Systems Biology*. 3, 1, 1-22.

[4] 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 (Feb. 2016), 398-408. DOI=http://dx.doi.org/10.1093/bioinformatics/btv578.

PathwayGenie - pathway design from selection to plasmid

Neil Swainston
Manchester Centre for Synthetic Biology of Fine and Speciality Chemicals (SYNBIOCHEM)
University of Manchester
Manchester M1 7DN
United Kingdom
+44 161 306 5131
neil.swainston@manchester.ac.uk

Pablo Carbonell
Manchester Centre for Synthetic Biology of Fine and Speciality Chemicals (SYNBIOCHEM)
University of Manchester
Manchester M1 7DN
United Kingdom
+44 161 306 5131
pablo.carbonell@manchester.ac.uk

Adrian Jervis
Manchester Centre for Synthetic Biology of Fine and Speciality Chemicals (SYNBIOCHEM)
University of Manchester
Manchester M1 7DN
United Kingdom
+44 161 306 5131
adrian.jervis@manchester.ac.uk

Christopher Robinson
Manchester Centre for Synthetic Biology of Fine and Speciality Chemicals (SYNBIOCHEM)
University of Manchester
Manchester M1 7DN
United Kingdom
+44 161 306 5131
christopher.robinson@manchester.ac.uk

Mark Dunstan
Manchester Centre for Synthetic Biology of Fine and Speciality Chemicals (SYNBIOCHEM)
University of Manchester
Manchester M1 7DN
United Kingdom
+44 161 306 5131
mark.dunstan@manchester.ac.uk

Jean-Loup Faulon
Manchester Centre for Synthetic Biology of Fine and Speciality Chemicals (SYNBIOCHEM)
University of Manchester
Manchester M1 7DN
United Kingdom
+44 161 306 5131
jean-loup.faulon@manchester.ac.uk

ABSTRACT

PathwayGenie is a web-based tool to allow wet-lab synthetic biologists to design libraries of plasmids to express synthetic metabolic pathways.

The application integrates a number of tools to provide a full design pipeline, from pathway and enzyme selection, through ribosome binding site and codon usage optimization, to design of experiments and the support of DNA assembly. Community standards such as SBML and SBOL are supported.

PathwayGenie has been applied to a number of pathway designs, and the experimental implementation of such designs is in progress.

CCS Concepts

- **Applied computing**
- **Life and medical sciences**
- **Computational biology**

Keywords

Synthetic biology; metabolic engineering; metabolism; design; pathway; enzymes; molecular biology.

1. INTRODUCTION

Metabolic engineering is an interdisciplinary process requiring numerous computational and experimental steps in order to produce target chemicals at a desired yield and titre.

The Manchester Centre for Synthetic Biology of Fine and Speciality Chemicals (SYNBIOCHEM) adopts an iterative DESIGN-BUILD-TEST-LEARN cycle involving pathway selection, plasmid design and assembly, numerous screening approaches including mass spectrometry based metabolomics approaches, and machine learning for results interpretation and iterative optimization of the process.

Key to these efforts is the development of computational design tools to facilitate the process. PathwayGenie integrates a number of novel and existing tools to allow wet-lab synthetic biologists to design pathways of metabolic enzymes and collections of plasmids to express such pathways. Such a pipeline involves pathway and enzyme selection, ribosome binding site (RBS) and coding sequence (CDS) optimization, search space reduction through design of experiments (DoE) and consideration of assembly methods.

2. MATERIALS AND METHODS

PathwayGenie is a web application, with a server written in Python using the Flask web framework (http://flask.pocoo.org/) and a Javascript front-end using jQuery (https://jquery.com/) and Bootstrap (http://getbootstrap.com/).

Freely available community standards such as SBML [1] and SBOL [2] are used for definitions of pathways and genetic constructs respectively. Custom Python scripts have been written to enable required operations on SBOL documents, such as concatenation and restriction digestion. The data repository ICE [3] is used for storage of SBOL documents.

3. RESULTS AND DISCUSSION

The PathwayGenie pipeline is illustrated in Figure 1. Individual tools are described below.

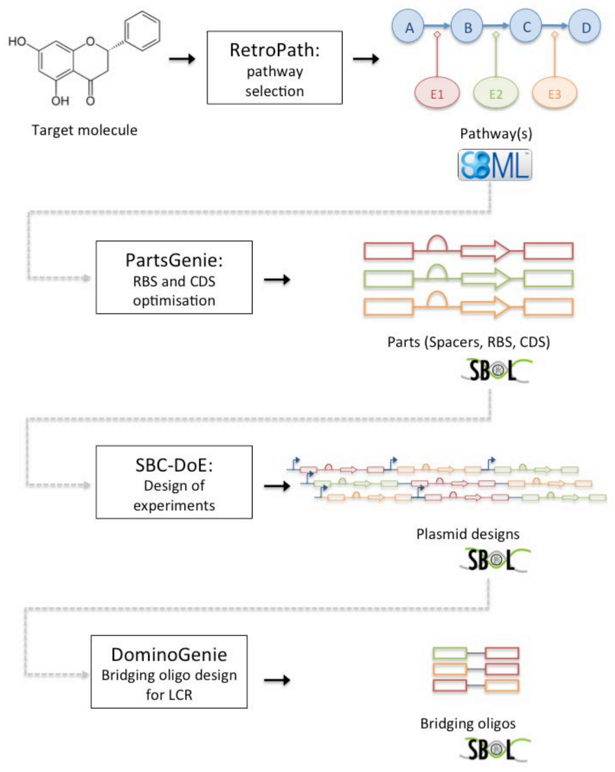

Target molecule
RetroPath: pathway selection
Pathway(s)
SBML
PartsGenie: RBS and CDS optimisation
Parts (Spacers, RBS, CDS)
SBOL
SBC-DoE: Design of experiments
Plasmid designs
SBOL
DominoGenie Bridging oligo design for LCR
Bridging oligos
SBOL

Figure 1: Representation of the PathwayGenie workflow, indicating the steps from chemical target input to design of parts and bridging oligos, from which plasmids may be assembled.

3.1 Tools

3.1.1 RetroPath

The retrosyntheis tool RetroPath [4] is utilized for pathway and enzyme selection. RetroPath explores and enumerates metabolic pathways connecting the endogenous metabolites of a chassis cell to the target compound, considering multiple factors including maximizing yield, mitigating the accumulation of toxic metabolic intermediates and suitability of enzymes for a given host.

3.1.2 PartsGenie

Once enzymes have been selected, PartsGenie optimizes the design of each enzymatic part, typically a construct including an RBS, a CDS and spacer sequences to enable assembly. Designing enzymatic parts individually allows for their reuse and for combinatorial assembly methods to be implemented. PartsGenie implements a simulated annealing algorithm to simultaneously design RBSs of a desired translation initiation rate with RBSCalculator [5] while simultaneously optimizing codon usage for a given host organism. Both single enzymes and pools of homologous enzymes can be designed, to further support combinatorial pathway assembly methods.

3.1.3 SBC-DoE

Rather than designing a single plasmid to express a desired metabolic pathway, SYNBIOCHEM generates variant libraries of plasmids, allowing screening and subsequent learning to be performed, allowing for iterative improvements of designs. Variable factors to consider include host selection, vector selection, the order of the enzymatic parts within the plasmid, and the strength (or absence) of promoters between enzymatic parts.

It is clear that implementing all combinations of these factors could quickly become prohibitively expensive. Intelligent sampling of such a multifactorial search space is provided by the design of experiments (DoE) module, which uses statistical principles to sample a feasible set of combinations to be implemented experimentally.

3.1.4 DominoGenie

Assembly of individual enzymatic parts into plasmids expressing a synthetic metabolic pathway can be performed with a number of molecular biology techniques. SYNBIOCHEM utilises the ligase chain reaction (LCR) [6] for assembly, a process that uses bridging oligos (“dominoes”) to link individual parts together. DominoGenie takes the output of the DoE and designs such bridging oligos, a process involving the calculation of consistent melting temperatures and the mitigation of undesired misannealing events.

3.2 Usage

By combining enzymatic parts and appropriate pools of bridging oligos, variant libraries of a given metabolic pathway can be assembled, expressed and screened.

PathwayGenie has been applied to the design of variant libraries of three distinct enzymatic pathways. The experimental assembly of these designs is in progress.

Subsequent work will involve the integration of PathwayGenie’s design methods with robotic platforms to enable automated assembly.

4. ACKNOWLEDGMENTS

All authors acknowledge the funding from the BBSRC under grant BB/M017702/1, “Centre for synthetic biology of fine and speciality chemicals (SYNBIOCHEM)”, BB/K019783/1.

This is a contribution from the Manchester Centre for Synthetic Biology of Fine and Speciality Chemicals (SYNBIOCHEM).

5. REFERENCES

[1] Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*. 2003, 19(4):524-31.

[2] Galdzicki M, Clancy KP, Oberortner E, Pocock M, Quinn JY, et al. The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nat Biotechnol*. 2014, 32(6):545-50.

[3] Ham TS, Dmytriv Z, Plahar H, Chen J, Hillson NJ, Keasling JD. Design, implementation and practice of JBEI-ICE: an open source biological part registry platform and tools. *Nucleic Acids Res*. 2012, 40(18):e141.

[4] Carbonell P, Parutto P, Baudier C, Junot C, Faulon JL. Retropath: automated pipeline for embedded metabolic circuits. *ACS Synth Biol*. 2014, 3(8):565-77.

[5] Salis HM. The ribosome binding site calculator. *Methods Enzymol*. 2011, 498:19-42.

[6] Wiedmann M, Wilson WJ, Czajka J, Luo J, Barany F, Batt CA. Ligase chain reaction (LCR)--overview and applications. *PCR Methods Appl*. 1994, 3(4):S51-64.

Integrated Predictive Genome-Scale Models to Improve the Metabolic Re-Engineering Efficiency

[Extended Abstract]

Vishwesh V. Kulkarni
University of Warwick
School of Engineering
Coventry, CV4 7AL
V.Kulkarni@warwick.ac.uk

Lina El Menjra
University of Warwick
School of Engineering
Coventry, CV4 7AL
L.el-Menjra@warwick.ac.uk

Pablo Carbonell
University of Manchester
SYNBIOCHEM
Manchester, M1 7DN
pablo.carbonell@manchester.ac.uk

Jean-Loup Faulon
University of Manchester
SYNBIOCHEM
Manchester, M1 7DN
jean-loup.faulon@manchester.ac.uk

Introduction

One of the most common applications of metabolic circuits is to produce a desired chemical in a chassis organism, such as the *Escherichia coli* (E. coli), by importing heterologous genes encoding for the enzymes that participate in the biosynthetic pathway. Recently, an automated pipeline named *RetroPath* was developed to synthesise embedded metabolic circuits [1]. These circuits are to be embedded in *E. coli* for a wide range of applications such as regulating biomass productions, sensing specific molecules, processing specific molecules, and releasing specific molecules. In RetroPath, the available circuit design space, constrained by the set of design specifications, is searched and a set of *optimal* circuits is obtained. In this process, the basic steps are as follows:

1. **Step 1:** Define the input set of metabolites $\mathbf{S}$, the output (target metabolite set) $\mathbf{T}$, and the metabolic space, i.e., the set of all possible metabolites and chemical reactions that can be generated *in vivo*, $\mathbf{M}$;
2. **Step 2:** Define the specifications of the desired circuit;
3. **Step 3:** Compute the set of enzymes involved in at least one minimal pathway that converts $\mathbf{S}$ into $\mathbf{T}$;
4. **Step 4:** Enumerate all metabolic pathways converting $\mathbf{S}$ into $\mathbf{T}$ (see [2]); and
5. **Step 5:** Solve a nonlinear optimization problem wherein the objective function comprises the expected reaction efficiencies, inhibition effects, and perturbation effects. Here, the *flux balance analysis* (FBA) is used.

The FBA (see [4]–[6]) predicts metabolic flux distributions at a steady state by solving the linear programming problem

$$\text{maximize} \;\; w^T v \quad \text{subject to} \;\; Sv = 0 \;\; \text{and} \;\; \alpha \le v \le \beta,$$

where $w^T v$ denotes the biomass, v denotes the vector of metabolic fluxes, S is the stoichiometric matrix, and (α, β) are the *a priori* known bounds on the fluxes. Thus, it is inherently assumed in RetroPath that the chassis organism has reached a steady state following the insertion of the heterologous gene. As shown in [4], such an assumption of optimality may not be valid for genetically engineered knockouts and bacterial strains that were not exposed to long-term evolutionary pressure. Following [4], we show that Step 5 can indeed be executed more efficiently by assuming that the metabolic fluxes undergo a minimal redistribution with respect to the flux configuration of the wild type. This *minimization of metabolic adjustment* (MOMA) is computed by solving the quadratic programming problem

$$\text{minimize} \;\; \frac{1}{2}\|v\|^2 + v^T v^* \quad \text{s. t.} \;\; Sv = 0 \;\; \text{and} \;\; \alpha \le v \le \beta,$$

where v^* denotes the flux distribution predicted by the FBA.

We next show that the efficiency of RetroPath can be improved if transcriptomic data is available. This is achieved by replacing the use of FBA in Step 5 first with the PROM algorithm derived in [3] and then with an extension PROM-E derived by us. PROM refines the upper bounds and the lower bounds in the metabolic flux constraints of the FBA by using Bayesian estimation on the available transcriptomic and metabolomic datasets. Here, the probabilities on the gene-TF interactions are empirically determined using available datasets; the more the number of datasets, the better is the expected performance. Effectively, PROM implements a slight modification of the FBA using linear programming and achieves an improvement in not only the capacity to generate larger genome-scale models but also in the accu-

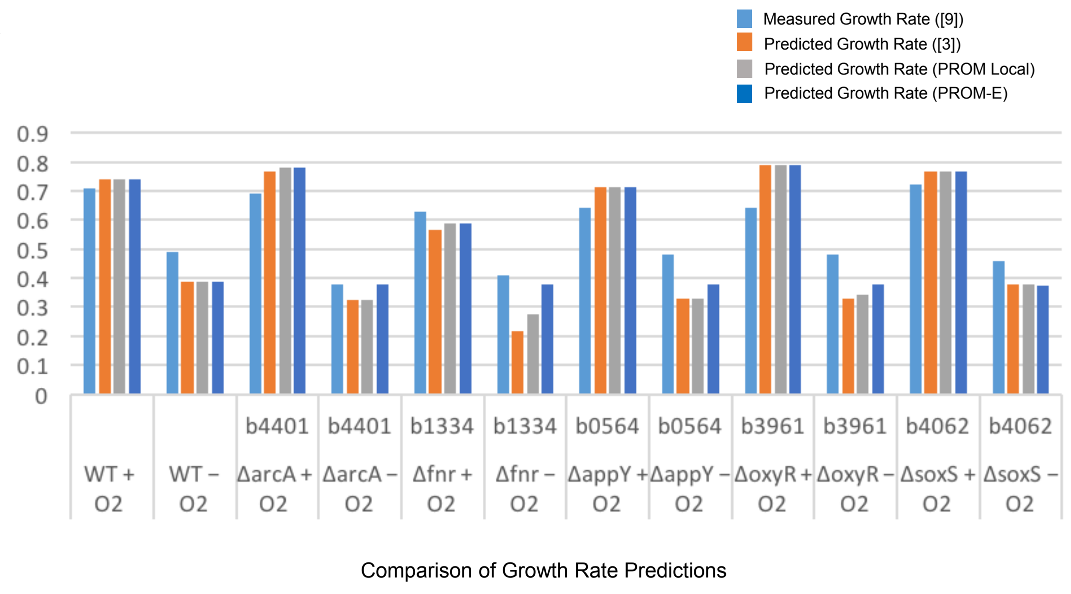


Figure 1: Our proposed algorithm PROM-E predicts the growth rates better than PROM [3] and RFBA [8] on the datasets of [8] across a variety of anaerobic conditions. For aerobic conditions, the PROM-E and PROM perform equally well and slightly better than RFBA.

racy of predicting the flux states. We then show how further improvement are obtained using our extension PROM-E.

Results

We now illustrate how our extension PROM-E of PROM estimates the flux rates more accurately than PROM on the datasets of [8] wherein growth phenotypes from *A Systematic Annotation Package* (ASAP) are predicted for community analysis of genomes database. The ASAP database has growth phenotypes of several *E. coli* gene KOs under various conditions. In [3], 15 TFs for which growth phenotypes under different 125 conditions are considered and it shown that PROM predicts the growth phenotypes more accurately than RFBA [8]. In [3], six strains with KOs of key transcriptional regulators in the oxygen response (ΔarcA, ΔappY, Δfnr, ΔoxyR, ΔsoxS, and the double KO ΔarcAΔfnr) were constructed and then the growth rates were measured in aerobic and anaerobic glucose minimal medium conditions. As Fig. 1 shows, our proposed PROM-E algorithm predicts the growth rate more accurately than PROM in anaerobic conditions and equally well in aerobic conditions. Furthermore, the standard deviation of the prediction error is significantly lower in PROM-E compared to PROM. We obtained these results in MATLAB R2015b interfaced with COBRA Toolbox 2.0 [7] and Gurobi Optimizer 6.5. As the cost of collecting omics datasets is reducing at Moore's law, we expect that our approach will soon be useful in practical contexts.

Acknowledgments

Research supported by BBSRC under grant BB/M017702/1, *Centre for Synthetic Biology of Fine and Speciality Chemicals* (SYNBIOCHEM).

1. REFERENCES

[1] Carbonell, P., Parutto, P., Baudier, C., Junot, C. and Faulon, J-L.: ‘Retropath: Automated Pipeline for Embedded Metabolic Circuits’, ACS Synthetic Biology, vol. 3, pp. 565–577, 2014.

[2] Carbonell, P., Fichera, D., Pandit, S. and Faulon, J-L.: ‘Enumerating Metabolic Pathways for the Production of Heterologous Target Chemicals in Chassis Organisms’, BMC Systems Biology, 6(10), pp. 1–18, 2012.

[3] Chandrasekaran, S. and Price, N.: ‘Probabilistic Integrative Modeling of Genome-Scale Metabolic and Regulatory Networks in Escherichia Coli and Mycobacterium Tuberculosis’, PNAS, 107 (41), pp. 17845–50, 2010.

[4] Segrè, D., Vitkup, D. and Church, G.: ‘Analysis of Optimality in Natural and Perturbed Metabolic Networks’, PNAS, 99(23), pp. 15112–17, 2002.

[5] Papoutsakis, E.: ‘Equations and Calculations for Fermentations of Butyric Acid Bacteria’, Biotechnology and Bioengineering, 26(2), pp. 174–187, 1984.

[6] Watson, M.: ‘Metabolic Maps for the Apple II’, Biochemical Society Transactions, 12(6), pp. 1093–1094, 1984.

[7] Schellenberger, J., *et al.*: ‘Quantitative Prediction of Cellular Metabolism with Constraint-Based Models: The COBRA Toolbox v2.0’, Nature Protocols, 6(9), pp. 1290–1307, 2011.

[8] Covert, M., Knight, E., Reed, J., Herrgard, M. and Palsson, B.: ‘Integrating High-Throughput and Computational Data Elucidates Bacterial Networks’, Nature, 429(6987), pp. 92–96, May 2004.

SBCDOE: a Design of Experiments-based Part Planner for Synthetic Biology Production of Chemicals

[Extended Abstract]

Mark Dunstan
SYNBIOCHEM, Manchester Institute of Biotechnology, University of Manchester
Manchester M13 9PL, UK
mark.dunstan@manchester.ac.uk

Adrian Jervis
SYNBIOCHEM, Manchester Institute of Biotechnology, University of Manchester
Manchester M13 9PL, UK
adrian.jervis@manchester.ac.uk

Christopher Robinson
SYNBIOCHEM, Manchester Institute of Biotechnology, University of Manchester
Manchester M13 9PL, UK
chistopher.robinson@manchester.ac.uk

Neil Swainston
SYNBIOCHEM, Manchester Institute of Biotechnology, University of Manchester
Manchester M13 9PL, UK
neil.swainston@manchester.ac.uk

Jean-Loup Faulon
SYNBIOCHEM, Manchester Institute of Biotechnology, University of Manchester
Manchester M13 9PL, UK
jean-loup.faulon@manchester.ac.uk

Pablo Carbonell
SYNBIOCHEM, Manchester Institute of Biotechnology, University of Manchester
Manchester M13 9PL, UK
pablo.carbonell@manchester.ac.uk

ABSTRACT

Application of the design-build-test-learn cycle to synthetic biology allows the development of increasingly efficient cell factories for bio-based production of a wide range of valuable chemicals. To fully exploit lab throughput capabilities under technological constraints, more focus on experimental design is needed. At each iteration of the cycle, predictive models for biological parts, systems and devices should guide next target constructs to build and test through automated protocols, whereas learning strategies should determine ways to efficiently explore the design space. Here, we present a full-fledged toolbox for optimal experimental design focused on synthetic biology production of chemicals.

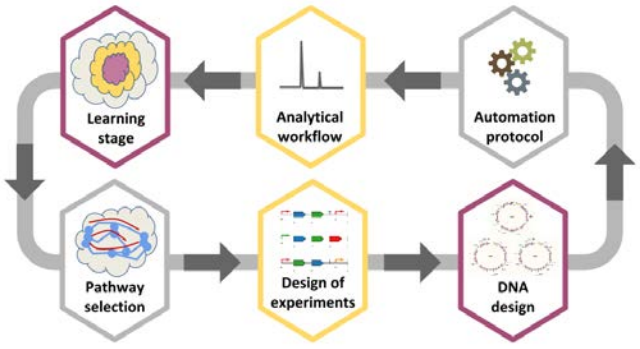


Figure 1: Design of experiments as part of the workflow of the design-build-test-learn cycle of synthetic biology.

CCS Concepts

•**Applied computing** → **Bioinformatics;** *Systems biology;* •**General and reference** → *Experimentation;* •**Theory of computation** → Active learning;

Keywords

synthetic biology; design; build; test; learn; design of experiments

1. INTRODUCTION

Experimental design is a piece of precision engineering central to synthetic biology for efficient chemical production. Current synthetic biology demands increasing efforts on experimental design in order to fully exploit design capabilities under technological constraints and lab throughput [6], [7], [10], [11]. As shown in Figure 1, design of experiments (DoE) should be performed within the design/build/test/learn cycle after the learning stage, once the next design target has been identified [9]. Observations at the end of the iteration of the cycle, including negative data from failures, should inform the next iteration. Active learning approaches from the field of machine learning should guide hypothesis-driven design of experiments by defining the factors and constraints of next-iteration design space [8].

DoE applies statistical principles in order to optimize the number of experimental runs in function of existing factors. For instance, in [1], more than 10 factors are listed for synthetic biology designs operating at different levels and time/space scales, such as transcriptional, translational or post-translational. In order to automate DoE at each iteration of the cycle, we have developed SynBioChem-Design Of Experiments (SBCDOE), a SynBio toolbox for design of experiments.

IWBDA16 August 16–18, 2016, Newcastle, UK
© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2. MATERIALS AND METHODS

2.1 Combinatorial library generation

Our tool generates combinatorial libraries to explore the design space determined by a target chemical-producer pathway that will be imported into a chassis organism. A pathway construct is initially defined by the sequential composition of genetic parts available in the parts repository. From the point of view of DoE, each part is considered as a factor and the number of levels associated with the factor is given by the number of possible variations of the design parameter. We used two DoE approaches: a) regular fractional factorial design by means of the `planor` R package [5]; b) orthogonal arrays, which are a generalized form of mutually orthogonal latin squares, by means of the `DoE.base` R package [4]. We considered an additional factor given by the variation of the positional order of genes. This factor can be used in order to perform permutations of the desired n genes within the construct and combinations can be reduced to n by using a latin square. The tool accepts SBOL v2 [2], taking advantage of its ability to define constraints between components. Each resulting construct in the library is processed through design and optimisation algorithms of DNA parts (RBS and CDSs) and plasmids that are tailored to the chosen assembly method. The tool is wrapped into a biologist-friendly web interface for ease of future application.

2.2 Automated assembly

Our Hamilton robotics platforms have been optimised for the effect assembly of DNA pathways and the analysis of downstream combinatorial libraries. Equipped with both 8 channel and 96 head liquid handling capabilities and integrated PCR (Trobot) allows for the fast and effect assembly of DoE-based generated pathway libraries. Automated worklist packages generated from DoE-based libraries are feed directly to the platform for pathway assembly.

Platforms are equipped with a solid phase extraction vacuum manifold allowing efficient and reproducible extraction of target compounds, coupled with a cytomat storage unit and inbuilt plate reader with a throughput upwards of 12000 samples per day. Our best performers are cherry picked for downstream quantitative mass spectrometry (MS) analysis.

3. RESULTS AND DISCUSSION

As shown in Figure 2, our DoE approach was applied to the design of combinatorial libraries for pathway optimization. Here, we considered as factors the following parts: a) enzyme sequence; b) promoter strength; c) positional order of genes. In order to select parts in pathways we applied our pathway design pipeline RetroPath [3]. This pipeline was recently shown to successfully select optimal pathways and enzymes for the production of flavonoids, and is easily extensible to the production of other fine chemicals. Next, a collection of plasmids and promoters were chosen. In terms of DoE, each enzymatic step in the pathway enters as a factor with as many levels as sequence candidates. Promoter strength and plasmid copy number are additional factors. Furthermore, an additional factor was considered regarding the sequential order of genes in the plasmid. The approach allowed streamlining the interface between design and build, notably reducing the number of experimental runs to test in order to inform the learning stage, achieving in that way a synthetic biology workflow harboring an optimal balance between our experimental throughput and design capabilities.

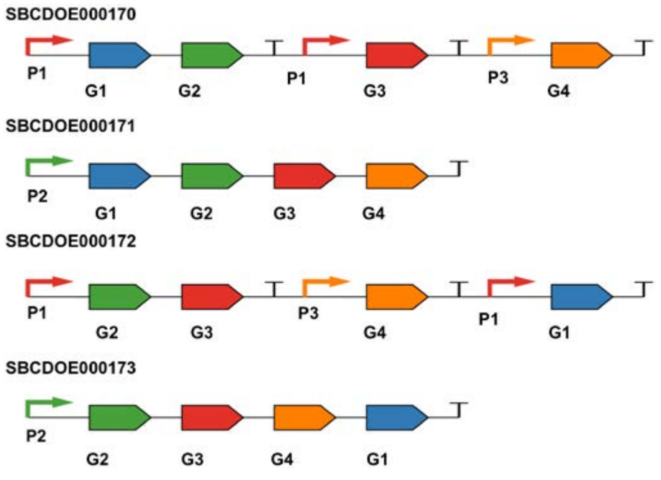


Figure 2: Detail of a DoE-based library composed of 4 genes, 3-level promoters and 2 different plasmids. Images were generated using *Pigeon* CAD tool.

4. ACKNOWLEDGMENTS

All authors acknowledge the funding from the BBSRC under Grant BB/M017702/1, "Centre for synthetic biology of fine and speciality chemicals".

5. REFERENCES

[1] J. A. J. Arpino, et al., Tuning the dials of synthetic biology. *Microbiology*, 159(Pt 7):1236–1253, 2013.

[2] B. Bartley, et al., Synthetic biology open language (SBOL) version 2.0.0. *J Integr Bioinfo*, 12:272, 2015.

[3] P. Carbonell, et al., Retropath: Automated pipeline for embedded metabolic circuits. *ACS Synth Biol*, 3:565–577, Aug. 2014.

[4] U. Groemping, B. Amarov, and H. Xu. *DoE.base: Full Factorials, Orthogonal Arrays and Base Utilities for DoE Packages*. R package version 0.28.

[5] A. Kobilinsky, A. Bouvier, and H. Monod. *PLANOR: an R package for the automatic generation of regular fractional factorial designs*. R package version 0.2-4.

[6] V. Kumar, A. Bhalla, and A. S. Rathore. Design of experiments applications in bioprocessing: concepts and approach. *Biotechnol Progr*, 30:86–99, 2014.

[7] C.-F. F. Mandenius and A. Brundin. Bioprocess optimization using design-of-experiments methodology. *Biotechnol Progr*, 24:1191–1203, 2008.

[8] J. Nielsen and J. D. Keasling. Engineering cellular metabolism. *Cell*, 164:1185–1197, 2016.

[9] C. J. Petzold, et al., Analytics for metabolic engineering. *Front Bioeng Biotechnol*, 3, 2015.

[10] N. Roehner, et al., Double Dutch: A tool for designing combinatorial libraries of biological systems *ACS Synth Biol*, 2016.

[11] H. Zhou, et al., Algorithmic co-optimization of genetic constructs and growth conditions: application to 6-ACA, a potential nylon-6 precursor. *Nucleic Acids Res*, gkv1071+, 2015.

A Web-Based Validator and Validation API for the Synthetic Biology Open Language

Zach Zundel
Dept. of Bioengineering
zach.zundel@utah.edu

Meher Samineni
School of Computing
m.samineni@utah.edu

Zhen Zhang
Dept. of Elec. and Comp. Eng.
zhen.zhang@utah.edu

Chris J. Myers
Dept. of Elec. and Comp. Eng.
myers@ece.utah.edu

1. INTRODUCTION

The *Synthetic Biology Open Language* (SBOL) [1] is an emerging standard for the expression of both structural and functional data regarding biological constructs. The data is encoded in an RDF-XML format that allows for hierarchical representation of the construct. As with all data standards, files encoding SBOL data must be validated according to a set of validation rules that describe correct formatting and encoding, as well as a minimum acceptable set of information which represents a complete and correct construct. Furthermore, it is useful to be able to convert between files in the SBOL standard and files that other tools are able to use and interpret. Our validator has functionality to convert between SBOL versions 1.1 and 2.0, as well as between SBOL and GenBank, a commonly used format for annotated DNA sequences. Our validation software embeds the libSBOLj [2] validation routines to allow validation of SBOL 2.0 files through web access to validation, as well as RESTful API calls for validation. This software allows for a universal standard for validation that existing tools (such as iBioSim [3] and Cello [4]) can use for validation without requiring tool developers to create, test, and maintain validation routines.

2. VALIDATION REQUIREMENTS

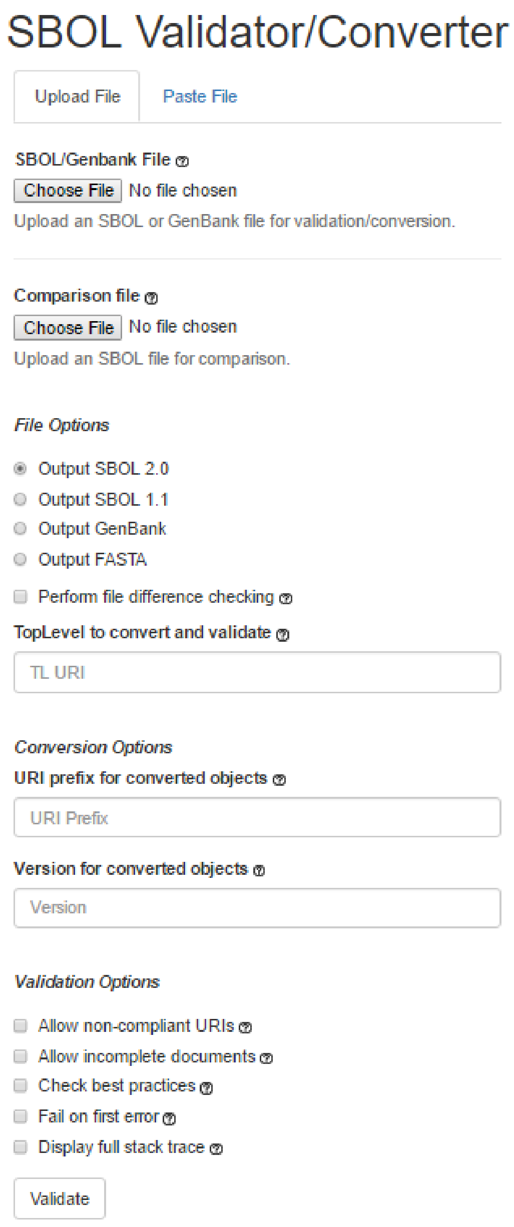
Proper validation of a file has several aspects. SBOL's validation rules are divided into two distinct categories. The first group is the *required validation rules*, and the second is the *best practice validation rules*. Required validation rules are a set of rules set in the SBOL specification that define precise relationships and requirements for valid SBOL [1]. These rules define things such as the number and nature of relationships between specific SBOL objects, attributes, and formats for these objects, and proper encoding and representation of the objects. A file is valid SBOL, if and only if it complies with all of these rules, so they are the most important to check. These rules are developed to be as *machine-checkable* and *unambiguous* as possible. Best practices validation rules describe a set of rules in the SBOL specification that define guidelines that help to ensure sensible and meaningful SBOL. These rules are, by nature, often less *machine-checkable*, though it is still desirable that these retain this trait so that they can be programatically validated. A file can still be valid SBOL if it does not pass these rules, but its sensibility, usefulness, and exchangability may not be the same level as one that passes both validation and best practices checks.

3. THE SBOL VALIDATOR

Our validator is backed by libSBOLj, a Java library for reading, writing, and encoding data in the SBOL standard and includes several validation routines that allow for checking of both required and best practice validation rules. The library performs certain validation checks on reading the RDF/XML file, and a second set of checks once the file has been read into the internal data model. Currently, the standard is version 2.0, which represents a significant departure from both the model and amount of data encoded in its 1.1 version. libSBOLj also provides several methods for converting the SBOL data model to/from other formats. In particular, it allows for the conversion between SBOL versions and between SBOL and the GenBank format.

The web-based validator, shown in Fig. 1, is essentially a PHP wrapper around the Java library. Files can be uploaded or copied into the validation form, and all validation options available in the library are available in the validator. Furthermore, the validator is designed to be able to return any converted files to the user so that all functionality of libSBOLj's validation routines are available. The validator's form processing utility creates several PHP objects that are all components of an overarching ValidationRequeset object. This object uses its component field objects to generate the Java command for validation and execute it. The resulting string is sent to a ValidationResult object for processing and then returned to the user via a web interface.

Our validator also has a RESTful API [5] endpoint to allow programmatic validation of SBOL files through a web service. The validation API extends the framework for the web-based validator. First, its endpoint processing utility accepts a JSON object from the POST request headers and checks to ensure that all necessary fields of the JSON object are included and properly formatted. A properly-formatted JSON object first requires a sub-object which contains Boolean switches and strings for each of the validation options given through the web validator. All of these options must be sent, even if they are empty or false – no values are assumed from missing elements. Secondly, the user must specify whether or not they would like a file returned to them. Finally, all required files must be encoded as separate strings. The endpoint utility accepts and validates all JSON requests, and


SBOL Validator/Converter

Upload File | Paste File

SBOL/Genbank File
Choose File No file chosen
Upload an SBOL or GenBank file for validation/conversion.

Comparison file
Choose File No file chosen
Upload an SBOL file for comparison.

File Options

[✓ Output SBOL 2.0]
Output SBOL 1.1
Output GenBank
Output FASTA

Perform file difference checking

TopLevel to convert and validate
TL URI

Conversion Options

URI prefix for converted objects
URI Prefix

Version for converted objects
Version

Validation Options

Allow non-compliant URIs
Allow incomplete documents
Check best practices
Fail on first error
Display full stack trace

Validate

Figure 1: Screenshot for the web validator interface.

builds an APIRequest object (which is simply an extension of the ValidationRequest class) using this JSON object. The APIRequest is then used to build and execute the validation command. The result of the command is passed to the requester in the form of another JSON object containing the result of the validation.

4. DISCUSSION

Though any application that wishes to perform validation on any SBOL files can simply include the libSBOLj library as a dependency and build any validation commands itself, providing a centralized service for validation of these files allows for a standard and universal definition of validity that can be applied, developed, and debugged independent of the development of each of the specific libraries. Furthermore, a single validator that is accessible through the web helps to ensure that competing validation routines do not emerge, avoiding risk of multiple definitions of validity that may have slight variance or discrepancies.

At time of writing, no SBOL libraries except libSBOLj implement validation for file conversion. pySBOL, libSBOL, and libSBOLjs are the three other major libraries currently under development that could benefit from the API validation – it shifts the onus of writing and maintaining validation routines to a single source. Therefore, library developers can simply write methods to access the RESTful API, a process that is simple, well-documented, and extensible in many languages. Additionally, many tools currently support reading and writing SBOL documents and more are under development. If these tools read and write SBOL without using one of the four main libraries (libSBOLj, libSBOL, pySBOL, libSBOLjs) they will also be able to validate their output to ensure full compliance with the specification using the web API.

Finally, the developers of the SBOL standard can use the validator as part of the compliance certification process. After dictating a set of constructs to be created in SBOL, one phase of validation of a tool's output would be validating their results, while another would be comparing the output of the tools to a premade file using the validator. A workflow such as this allows the community to define what is and what is not SBOL specification compliance, addressing the emerging issue of tools claiming compliance without the ability to read, write, and interpret valid SBOL documents.

Our validator is currently accessible online at `http://www.async.ece.utah.edu/sbol-validator`, with the API accepting requests at `http://www.async.ece.utah.edu/sbol-validator/endpoint.php`.

5. ACKNOWLEDGEMENTS

This work is funded by the National Science Foundation under Grants 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.

6. REFERENCES

[1] B. Bartley *et al.*, "Synthetic Biology Open Language (SBOL) Version 2.0.0," *J Integr Bioinform*, vol. 12, no. 2, p. 272, 2015.
[2] Z. Zhang *et al.*, "libSBOLj 2.0, A Java Library to Support SBOL 2.0," *IEEE Life Sciences Letters*, 2015.
[3] C. Madsen *et al.*, "Design and test of genetic circuits using iBioSim," *IEEE Design and Test*, pp. 32–39, 2012.
[4] A. A. K. Nielsen *et al.*, "Genetic circuit design automation," *Science*, vol. 352, no. 6281, 2016.
[5] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," in *Proceedings of the 22nd International Conference on Software Engineering*, ICSE '00, (New York, NY, USA), pp. 407–416, ACM, 2000.

SBOLDesigner 2.0

Michael Zhang
Dept. of Electrical and Computer Engineering
michael13162@gmail.com

Chris J. Myers
Dept. of Electrical and Computer Engineering
myers@ece.utah.edu

1. INTRODUCTION

Synthetic biology is an engineering discipline where biological components are assembled to form devices or systems with more complex functions. A workflow, such as the one shown in Fig. 1, is necessary to advance the field of synthetic biology by giving biologists the ability to abstract their designs and use automated software to ease the development process [9]. In this workflow, DNA sequences for known components are obtained from databases, such as the SBOL Stack, the iGem Registry, and the JBEI-ICE repository. These DNA sequences can then be edited and manipulated inside a sequence *computer-aided design* (CAD) tool, such as **SBOLDesigner** [3, 7], to create a complete structural design of genetic circuit components. Next, circuit *computer-aided engineering* (CAE) tools, such as **Cello** [6] and **iBioSim** [5], can be utilized to compose genetic circuit components into complete genetic circuit designs, including the addition of functional design information for simulation and analysis. Finally, the complete genetic circuit can be archived in the part repositories, completing the cycle.

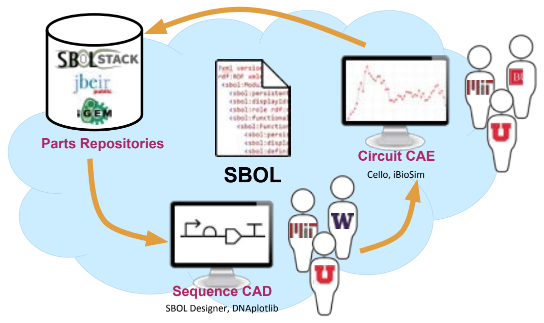


Figure 1: The SBOL enabled workflow for designing genetic circuits.

The *Synthetic Biology Open Language* (SBOL) facilitates communication between these tools and services [4]. SBOL is a standardized digital format that allows biologists to share genetic designs stored in a principled medium. The SBOL standard facilitates communication between experimental biologists, computational biologists, genetic engineers, and their computer tools. The latest version, SBOL 2.0, introduces the specification of generalized genetic components, enhances means to annotate and constrain sequence features, and enables the description of behavioral aspects of a biological design [1].

SBOLDesigner is a simple, biologist-friendly CAD software tool for creating and manipulating the sequences of genetic constructs using SBOL natively. This abstract describes our update of **SBOLDesigner** to support SBOL 2.0, as well as the enhancements that this conversion enables.

2. SBOLDESIGNER

SBOLDesigner has an simple user interface that allows biologists to visualize and edit the details of their creation. It supports hierarchical or nested assembly and offers generic, user-defined parts to ease fabrication from partial sequences to complete genetic constructs. Additionally, the user can flip the orientation of parts and view or edit their names and descriptions. Throughout the design process, SBOL Visual [8] symbols, a system of schematic glyphs, provide standardized visualizations of individual parts.

The original version of **SBOLDesigner** has been updated to SBOL 2.0. While the user interface remains largely the same, the transition from SBOL 1.1 to SBOL 2.0 as the backend data model required re-implementing features using the libSBOLj 2.0 Java library [10]. Fig. 2 shows the main parts of the SBOL 2.0 data model that are used. SBOL 2.0 separates **SequenceAnnotations** from SBOL 1.1 into **Components**, **SequenceAnnotations**, and **SequenceConstraints**, which requires a fundamental change in the representation of parts in **SBOLDesigner**. Overall, adapting **SBOLDesigner** to SBOL 2.0 enables the workflow described above and maintains the relevance of this CAD tool.

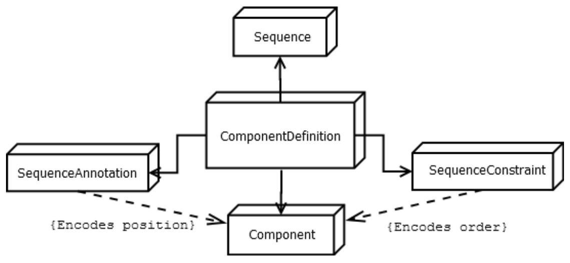


Figure 2: A simplified view of the structural portion of the SBOL 2.0 data model.

Fig. 3 shows the user interface of **SBOLDesigner** with part of the genetic toggle switch circuit on the canvas. This canvas represents a **ComponentDefinition** that brings together information on the design's **Sequence**, its **Components**, and their organization. Below the canvas is a row of genetic elements that can be added to the design. When placed on the canvas, each element represents a **Component**. These **Components** are organized by **SequenceAnnotations** and **SequenceConstraints**. **SequenceAnnotations** specify the precise **Location** and *orientation* (position and direction) of a **Component**'s **Sequence**. **SequenceConstraints** encode the information on how **Components** are ordered. Clicking on the "focus in" button expands a **Component** to expose its **ComponentDefinition**, generating a nested canvas. This feature allows the user to create hierarchically defined designs.

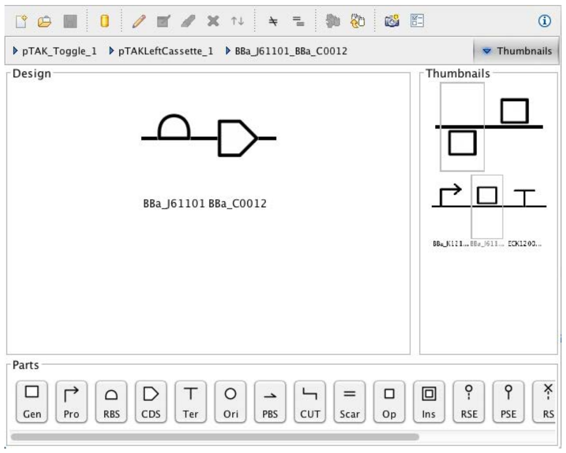
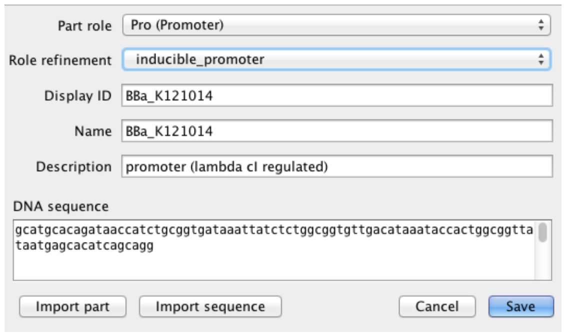


Figure 3: SBOLDesigner's user interface showing a hierarchical view of a genetic toggle switch circuit. This design is composed of two genetic inverters, and each inverter is composed of a promoter, ribosome binding site, coding sequence, and terminator.

Fig. 4 shows the menu for specifying and editing a **ComponentDefinition**'s *role*, *display ID*, *name*, *description*, and *sequence*. The role of the **Component** can be promoter, ribosome binding site, coding sequence, terminator, etc. A new feature is the ability to specify a more specific refinement role from the Sequence Ontology Project [2]. Additionally, **SBOLDesigner 2.0** supports importing **ComponentDefinitions** and **Sequences** from external SBOL, GenBank, and FASTA files.



Part role	Pro (Promoter)
Role refinement	inducible_promoter
Display ID	BBa_K121014
Name	BBa_K121014
Description	promoter (lambda cI regulated)

DNA sequence

```
gcatgcacagataaccatctgcggtgataaattatctctggcggtgttgacataaataccactggcggtta
taatgagcacatcagcagg
```

Figure 4: SBOLDesigner's window for editing a ComponentDefinition for a promoter.

When the design is complete and ready to export, **SBOLDesigner** stitches together all of the specified **Sequences** to form a composite root **Sequence** that is attached to the root **ComponentDefinition**. If desired, this root sequence can then be sent to a DNA synthesis service for fabrication.

3. DISCUSSION

SBOLDesigner completes a workflow for users of *genetic design automation* tools. It combines a simple user interface with the power of the SBOL standard and serves as a launchpad for more detailed designs involving simulations and experiments. Some new features include SBOL Stack integration, the ability to import and write GenBank and FASTA files, extended ontology support, the ability to partially open designs with multiple root **ComponentDefinitions**, and backwards compatibility with SBOL 1.1. Support for RNA and protein parts, more general SequenceConstraints, user annotations, saving into existing designs, and automated **Sequence** optimization are being added. With sequence CAD tools like **SBOLDesinger**, genetic design software like **iBioSim** and **Cello**, and parts repositories like the SBOL Stack, biologists have a wide array of tools for prototyping and automating design of genetic circuits. These tools stimulate advancement of synthetic biology and allow biologists to easily approach genetic design.

4. ACKNOWLEDGMENTS

We thank Evren Sirin (Complexible), Michal Galdzicki (Arzeda), Bryan Bartley (U. of Washington), and John Gennari (U. of Washington) for their work on the original version of **SBOLDesigner**. This work is funded by the National Science Foundation under Grants 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.

5. REFERENCES

[1] B. Bartley et al. Synthetic biology open language (SBOL) version 2.0.0. *Journal of Int. Bioinfo.*, 2015.
[2] K. Eilbeck et al. The Sequence Ontology: a tool for the unification of genome annotations. *GenBio.*, 2005.
[3] M. Galdzicki, B. A. Bartley, S. C. Sleight, E. Sirin, and J. H. Gennari. Version control for synthetic biology. *IWBDA2013*, pages 19–20, 2013.
[4] M. Galdzicki et al. The synthetic biology open language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nature Biotechnology*, 32:545–550, 2014.
[5] C. Madsen et al. Design and test of genetic circuits using iBioSim. *IEEE D. & T.*, pages 32–39, 2012.
[6] A. Nielsen et al. Genetic circuit design automation. *Science*, page 53, 2016.
[7] C. Olsen, K. Qaadri, H. Shearman, and H. Miller. Synthetic biology open language designer. *IWBDA2014*, pages 60–61, 2014.
[8] J. Quinn et al. SBOL visual: A graphical language for genetic designs. *PLOS Biology*, 2015.
[9] N. Roehner et al. Sharing structure and function in biological design with SBOL 2.0. *ACS SynBio*, 2016.
[10] Z. Zhang et al. libSBOLj 2.0: A java library to support SBOL 2.0. *IEEE Life Sciences Letters*, 2016.

Bioform: an *in-silico* 3D physical modelling platform for the design and analysis of bacterial populations

Jonathan Naylor
School of Computing Science
Newcastle University, UK
j.r.d.naylor@ncl.ac.uk

Harold Fellermann
School of Computing Science
Newcastle University, UK
harold.fellermann@ncl.ac.uk

Waleed Mohammed, Nick Jakubovics
School of Dental Sciences
Newcastle University, UK
{w.k.mohammed, nick.jakubovics}@ncl.ac.uk

Joy Mukherjee, Catherine Biggs
Department of Chemical and Biological Engineering
Sheffield University, UK
{j.mukherjee, c.biggs}@sheffield.ac.uk

Phillip Wright
Faculty of Science, Agriculture and Engineering
Newcastle University, UK
phillip.wright@ncl.ac.uk

Natalio Krasnogor*
School of Computing Science
Newcastle University, UK
natalio.krasnogor@ncl.ac.uk

ABSTRACT

We present a spatially explicit modeling platform, *Bioform*, which allows for the design, simulation and analysis of synthetic bacterial populations. The design of biological systems in Bioform is flexible, enabling one to represent constructs in a variety of forms appropriate for the desired level of abstraction. A virtual lab provides the modeller with equivalent devices to those typically used in wetlab experiments.

Allowing for the representation of a variety of cellular and environmental properties, a wide range of bacterial populations can be simulated, from planktonic cells and colonies, to biofilm formation and development. An extendable and modular framework allows for the platform to be updated as scientific knowledge advances, coupled with an intuitive user interface to allow for model definitions with minimal programming experience.

1. INTRODUCTION

Synthetic biology aims to repurpose biological components for novel applications. The design, synthesis and analysis of such systems is time and resource intensive, typically involving multiple iterations around the workflow.

To catalyse this process *in-silico* models have been developed, providing insights into the dynamics of proposed systems allowing verification of their feasability prior to synthesis. Such models typically focus on genetic circuits and the behaviour of single cells, however the dynamics and self-organising capabilities exhibited by populations of interacting cells is in need of renewed effort.

The development of distributed multicellular devices allow for mixed cohorts of bacteria with each species programmed to carry out simpler tasks. A major challenge in the development of these systems resides in their scalability and robustness, thus models of population dynamics are essential in the realisation of such systems.

2. IMPLEMENTATION

Developed in Java on a modified version of the Cortex3D platform [1], Bioform is a multiscale agent-based modelling environment for specifying custom model behaviour, parallelisable on distributed CPU computing. Agents are modelled as physical entities with biological processes, embedded in chemical fields. The platform allows for processes that occur at a range of spatial and temporal scales to be defined and integrated.

The platform is designed with a modular architecture, allowing for model features to be represented as discrete components which can be readily added, removed and modified for the specific modelling application. This is achieved via a three component architecture consisting of a simulation core, a modelling library and a modelling interface. This *plug n' play* framework allows for rapid prototyping of models and reiterative designs for the reification of models.

2.1 Framework architecture

The core of Bioform is the computational engine, integrating all model defined processes and scheduling their execution for parallel execution.

The modelling library contains discrete submodels describ-

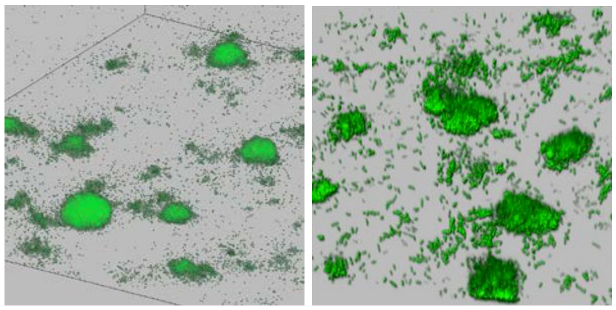


Figure 1: Model of biofilm (left) and confocal microscopy image of actual biofilm [2] (right)

ing specific model behaviour, such as physical, chemical and biological processes as well as environmental properties.

The modelling interface allows for the composition of models via attaching library submodels to the simulation core.

2.2 Model features

Newtonian physics are implemented with Langevin dynamics dictating agent movement. Basic cell morpholgy, pressure sensing, surface charge, cell shoving, physically bonded geometries and environmental forces can be represented.

A diffusion-reaction system implemented as a 3D grid allows for custom definition of chemical species, including diffusion, degradation and reaction rates. Chemicals can exist in the extracellular space or within cells and can be transported across membranes via a variety of mechanisms.

A wide range of biological processes are implemented, including cell growth and division, quorum-sensing, conjugation, cell-cell and cell-surface adhesion as well as gene regulatory networks. Processes may be represented at a variety of abstractions, such as gene networks represented as boolean networks or sets of ordinary differential equations.

2.3 Analysis features

A virtual lab is implemented for model analysis, offering typical wetlab instruments and mathematical analysis features. This includes a simulated spectrophotometer to obtain optical density measurements. Mathematical tools include measurements of the mean squared displacement and velocity autocorrelation function of bacteria, as well as detailed data gathering regarding cell interactions, gene expression and spatially distributed biomass concentrations.

3. CASE STUDIES

3.1 Synthetic biofilm formation

Modelling biofilm formation of synthetic *Escherichia coli*, simulating cell growth, motility, basic gene regulation and secretion of extracellular polymeric substances. We aim to understand the mechanisms responsible for architectural differences in the biofilms formed by mutant strains of *E. coli*. Figure 1 shows an example of a modelled biofilm. Callibration of the model on experimental data ensured model validity, and model feedback has insofar directed further experimental data gathering and aided in the formalisation of hypotheses about the system.

3.2 Bacterial coaggregation

We model the coaggregation of bacterial strains found in the mouth, namely *Streptococcus gordonii* and *Actinomyces oris*. Cell-cell interacitons include specific interactions due to cell surface adhesins and receptors, as well as through non-specific mechanisms such as van der Waals forces and electrostatic repulsion. Optical density measurements decrease as aggregates form. Figure 2 expands on the details of this study.

4. CONCLUSIONS AND FUTURE WORK

We have presented an overview of Bioform and example case studies illustrating its potential for computer aided design in the synthetic biology workflow. Future developments include an expansion of the modelling library and virtual lab, development of a graphical user interface, alongside optimisation of the simulation core and integration with the

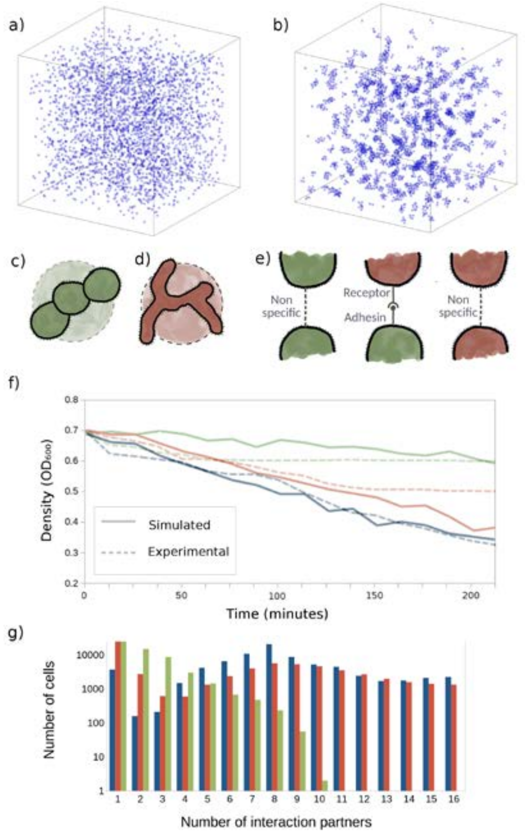


Figure 2: (a) Intial mixed population. (b) Aggregated population. (c) *Streptococcus gordonii* (represented as a sphere). (d) *Actinomyces oris* (represented as a sphere). (e) Adhesion interactions for *S. gordonii* and *A. oris*. (f) Real (solid) and simulated (dashed) optical density measurements. Green: *S. gordonii*, Red: *A. oris*, Blue: *Mixed population* (g) Number of interaction partners per cell.

Infobiotics Workbench 2.0 [3]. We plan to implement parsers for interfacing with SBML, SBOL and MATLAB. Development of an image processor to allow for model initialisation from microscopy images will ensure a seemless modelling process for experimentalists.

5. ACKNOWLEDGMENTS

This work was supported by EPSRC (EP/I031642/2, EP/J004111/2, EP/L00149012/2, EP/N031962/1).

References

[1] F. Zubler, “A framework for modeling the growth and development of neurons and networks,” *Front. Comput. Neurosci.*, vol. 3, 2009.

[2] http://www.botinst.uzh.ch/en/research/microbiology/eberl/research.html. Accessed: 2016-04-1.

[3] J. Blakes, J. Twycross, F. J. Romero-Campero, and N. Krasnogor, “The Infobiotics Workbench: an integrated in silico modelling platform for systems and synthetic biology,” *Bioinformatics*, vol. 27, pp. 3323–3324, oct 2011.

TEBio - Tools for Engineering Biology

James Scott-Brown Thomas P. Prescott Antonis Papachristodoulou

{james.scott-brown, thomas.prescott, antonis}@eng.ox.ac.uk
Department of Engineering Science, University of Oxford, Parks Road, Oxford, OX1 3PJ, United Kingdom

1. INTRODUCTION

The Tools for Engineering Biology (TEBio) project is aimed at developing algorithms and software for the automated design of multi-cellular genetic circuits.

In general, there are many possible solutions to a design problem, and so when designing a circuit it is necessary to make comparisons between candidate designs. Our initial work has therefore focused on the *comparison* of models and systems, rather than directly on design.

We have also focused on the use of visualisation to support comparisons, as we believe that there is considerable potential for improved visualisation in synthetic biology.

This abstract describes two initial contributions: a tool for visually representing a single model in SBML format, or visually comparing multiple models (SBML-diff); and a visualisation tool for the results of Bayesian parameter estimation and model comparison using Approximate Bayesian Computation (TEBio-Fit).

2. VISUALLY COMPARING SBML MODELS

2.1 Existing tools

There are several existing tools which are capable of visually representing a single SBML model as a bipartite reaction graph. Several of these are large GUI packages with many other features (*e.g.* CellDesigner [2], iBioSim [5]), such that the visualisation component is difficult to script or incorporate into other tools. There are also a few freestanding tools that can generate a DOT representation of a reaction network from an SBML file (*e.g.* The Systems Biology Format Converter's SBML2DOT Java module [6], or the python library VisualiseSBML [3]). A recent paper [7] provides a tool for comparing SBML models; it is intended for a slightly different purpose (to help track the history of a model as it changes over time), cannot directly compare more than 2 models, and has fewer options for controlling the graphical output, but supports CellML as well as SBML, and can produce reports in Markdown or HTML format.

2.2 Implementation

Our goal was to create a freestanding (but easily composable) tool that can be used to quickly visualise a single model, or to compare a set of models. It reads in SBML, and produces output in DOT that can be converted to an image by GraphViz, or by alternative software.

Whilst it is possible to compare a set of models by simply juxtaposing an independently created visualisation of each, this has the disadvantage of requiring the user to actively compare the diagrams to 'spot the differences', a task made harder by the fact that unstable network layout algorithms may assign equivalent nodes to very different spatial locations when visualising different models. In contrast, we use a pre-attentive cue (colour) to highlight the differences between models without the need for search; colours were chosen to be distinguishable by colourblind users. We provide the option of either producing the DOT code for a combined diagram superimposing all of the models, or separate diagrams for each model (but with the invisible features from all other models influencing the layout to ensure consistent node placement).

Each reaction is represented by a rectangular node; each species is represented by an elliptical node. Directed edges with solid lines join each reactant species to the corresponding reactions, and each reaction to the corresponding product species. Dashed lines join species to reactions whose rate they modify, but in which they are not reactants; the corresponding `kineticLaw` for the reaction is analysed to determine whether this influence is activatory or inhibitory, and the arrowhead shape is set accordingly. Rules of type `AssignmentRule` or `RateRule` that set concentrations of particular species are represented as parallelograms, with dotted lines indicating directed away from the species they depend on, and towards the species that they affect. Reactions or rules that appear in more than one model, but with different kinetic laws, are displayed with a grey background. Colouring is used to indicate whether each node and edge is common to all models (grey), some but not all models (black), or a single model (a colour specific to that model). A commandline option allows the user to choose between labelling reaction nodes with the corresponding reaction name, kinetic law, or name and rate law, or leaving them unlabelled.

3. VISUALISING RESULTS OF PARAMETER ESTIMATION

In systems and synthetic biology, it is common to use Markov chains as models. However, Bayesian analysis of such models is difficult due to the unavailability of a likelihood function. One approach is to sample from the (approximate) posterior distribution using Approximate Bayesian Computation.

TEBio-Fit is an interactive tool for visualising the results of applying ABC methods to parameter estimation and model selection. The interactive graphs are implemented in Javascript using the D3.js library[1], and work in recent versions of Firefox or Chrome, without requiring the installation of any additional software or plug-ins.

Currently, it uses ABC-SysBio [4] as a backend, but could be extended to support additional systems and methods (*e.g.* conventional MCMC for models with a likelihood function).

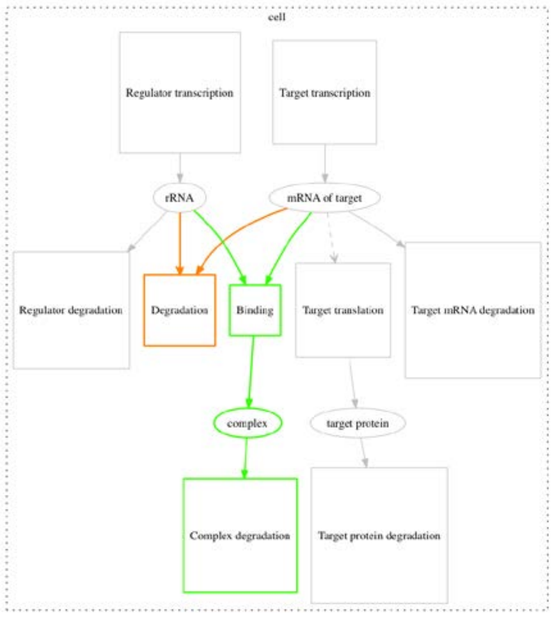


Figure 1: Example output from SBML-diff, comparing two models for how a gene might be regulated by a regulatory RNA. Options allow the labels for reaction nodes to be removed, or replaced by the corresponding `kineticLaw`

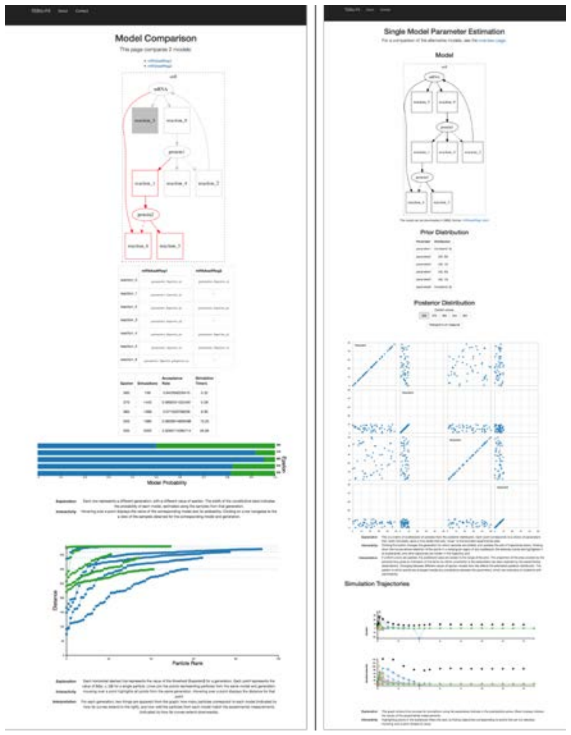


Figure 2: Two screenshots of TEBio-Fit, showing the model comparison view (left) and the single-model detail view (right). This example is accessible as an interactive demo at http://sysos.eng.ox.ac.uk/tebio

.

The results are presented as a webpage, grouping together information that would otherwise be scattered across multiple files (numerical results from output files, model details from input SBML files, settings from configuration files, explanations from manuals). This is a self-contained interactive report that can be presented to third parties, and provides explanations of each figure at the point of need.

An overview page provides a visual comparison of the models being compared (produced using SBML-diff) and a table of the kinetic laws for each reaction, a table of acceptance rates and simulation time at each generation, a stacked bar chart of the model probabilities, and a dot-plot showing the closeness of fit for each sample. A single-model page shows the model's details, the prior distribution specified for its parameters, a matrix of scatterplots of the samples, and time-series of simulations using the corresponding parameter values. Selecting a rectangular region in any scatterplot highlights the corresponding points in all scatterplots, and hides other trajectories from the time-series plot.

Interactivity fulfils several roles in this system. It allows the user to see all the data, yet also focus only on what is important to them at a particular moment, by fading or hiding the representations of other data-points. It provides precise numerical details and context, in the form of a tooltip when the user mouses over specific elements. It allows direct navigation between related views of the same data (*e.g.* clicking on the stacked bar chart navigates to the single-model view with samples from the corresponding generation plotted; by selecting particular regions of parameter space, the user can see the shape of the corresponding simulation trajectories).

Links to code and example output are available at http://sysos.eng.ox.ac.uk/tebio/

4. ACKNOWLEDGMENTS

J.S-B acknowledges funding through the EPSRC & BBSRC Centre for Doctoral Training in Synthetic Biology, EP/L016494/1.

5. REFERENCES

[1] M. Bostock. D3.js: Data-driven documents. *https://d3js.org/*.

[2] A. Funahashi, M. Morohashi, H. Kitano, and N. Tanimura. CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. *BIOSILICO*, 1(5):159 – 162, 2003.

[3] C. S. Gillespie, D. J. Wilkinson, C. J. Proctor, D. P. Shanley, R. J. Boys, and T. B. L. Kirkwood. Tools for the SBML community. *Bioinformatics*, 22(5):628–629, 2006.

[4] J. Liepe, C. Barnes, E. Cule, K. Erguler, P. Kirk, T. Toni, and M. P. Stumpf. ABC-SysBio - approximate Bayesian computation in python with GPU support. *Bioinformatics*, 26(14):1797–1799, 2010.

[5] C. J. Myers, N. Barker, K. Jones, H. Kuwahara, C. Madsen, and N.-P. D. Nguyen. iBioSim. *Bioinformatics*, 25(21):2848–2849, Nov. 2009.

[6] N. Rodriguez, J.-B. Pettit, P. Dalle Pezze, L. Li, A. Henry, P. M. van Iersel, G. Jalowicki, M. Kutmon, K. N. Natarajan, D. Tolnay, I. M. Stefan, C. T. Evelo, and N. Le Novère. The systems biology format converter. *BMC Bioinformatics*, 17(1):1–7, 2016.

[7] W. D. Scharm M., Wolkenhauer O. An algorithm to detect and communicate the differences in computational models describing biological systems. *Bioinformatics*, 32:563–570, 2016.

ShortBOL: A Shorthand for SBOL

Matthew Pocock[*]
Turing Ate My Hamster Ltd
turingatemyhamster@gmail.com

Chris Taylor
School of Computing Science,
Newcastle University
c.p.d.taylor@newcastle.ac.uk

Göksel Mısırlı
ICOS, School of Computing
Science, Newcastle University
goksel.misirli@ncl.ac.uk

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

Anil Wipat[*]
ICOS, School of Computing
Science, Newcastle University
anil.wipat@ncl.ac.uk

1. INTRODUCTION

Synthetic Biology Open Language (SBOL) version 2 [1] is one of the emerging synthetic biology (synbio) data standards. It facilitates the computational design and exchange of novel, reproducible and composable designed biological systems. SBOL is defined as a data model and RDF/XML serialization. While the data model is flexible in its ability to be generic and extensible as well as very explicit, it can sometimes be too verbose. A single biological design concept may be captured through multiple SBOL entities, and this level of normalization can make it difficult for a person to edit SBOL data manually. While well-suited for precise machine communication, SBOL RDF/XML is too verbose and complex for humans to directly edit non-trivial designs.

Software tools and libraries are emerging to manipulate SBOL. For example, libSBOLj [5] can be linked to other software, enabling them to read, write and manipulate SBOL data. While libSBOLj supports tool developers, it does not directly help synthetic biologists work with SBOL. CAD and visualisation tools have been developed to visualise designs and make the designs easier for humans to communicate [4, 3, 2]. However, visual design is an essentially manual process. We have identified a need for a light-weight SBOL scripting language that bridges the gap between manual, visusal design, and software development.

Here, we present ShortBOL, a human readable/writable shorthand language for SBOL. This language is developed for synthetic biologists who wish to rapidly sketch synthetic biology designs using a simple, text based scripting language. The terminology used is designed to be much closer to the concepts synthetic biologists have when describing designs. Using this language, design information can be captured more easily and quickly, without limiting the functionality that SBOL already provides.

2. THE SHORTBOL LANGUAGE

ShortBOL is designed to be easy to use for synthetic biologists who may not have much software development training. The language is text based, has a simple syntax that uses white spacing to demarcate blocks, as in popular programming languages such as Python, rather than punctuation, as in RDF/Turtle or JSON. A standard template library is provided, covering the SBOL data model and its use for capturing genetic designs. Power users can create new templates to capture abstractions common within their designs or their synbio domain. If these libraries are made available on the Web, they can then be imported and used by others.

The ShortBOL shorthand is built around a minimal selection of language constructs. A typical shorthand document is a list of imports, property assignments and template applications. Template libraries are pulled into a ShortBOL document using import statements. These libraries are themselves written in shorthand, and declare new templates. Custom templates can be used to provide simple aliases, application-specific syntax, access to common terminologies, and can even be used to model complex parameterised multi-component designs. Assignment statements associate a value with an identifier, using the equals (`=`) operator. For example, `repressor = tetR` associates the value `tetR` with the identifier `repressor`. This can be used to set up aliases to provide more natural local names for remotely defined terms and design components.

SBOL entities are created within the shorthand by using the colon (`:`) operator to apply a template (Figure 1A). For example, `lacI_cds : CDS` introduces a new identifier `lacI_cds` that will hold the value of expanding the `CDS` template. In this case, the `CDS` template expands to a `SBOL:ComponentDefinition` template with the type set to the `DnaRegion` BioPAX term and role set to the CDS (`SO000316`) Sequence Ontology term, as recommended in the SBOL best practices for encoding a CDS using SBOL (Figure 1B). Some templates are parameterised by one or more arguments. For example, the `DNASequence` template expects a single argument, containing a DNA string. When the template is expanded, the `elements` property of the resulting `SBOL:Sequence` is set to be equal to the supplied argument. This mechanism allows common design and composition patterns to be captured relatively easily within templates, without requiring a full programming language.

Template applications can be directly followed by an indented block of ShortBOL expressions. These are used to declare additional properties and their values. For example, the template application `lacI_cds : CDS` may be followed by an indented block containing the assignment `description = "The lacI CDS"`.

2.1 Expansion

Shorthand documents go through an expansion pipeline,

[*]To whom correspondance should be addressed

A)

```
import shorthand:sbol2

lacI_cds : CDS
   description = "The lacI CDS"
   name = "lacI"
   sequence = lacI_seq

lacI_seq : DNASequence("atggtgaatgt")
```

B) ↓ *Template Expansion*

```
lacI_seq : Sequence
   encoding = <SBOL:IUPACDNA>
   displayId = "lacI_seq"
   elements = "atggtgaatgt"

lacI_cds : ComponentDefinition
   role = <SBOL:CDS>
   type = <SBOL:DNA>
   displayId = "lacI_cds"
   description = "The lacI CDS"
   name = "lacI"
   sequence = lacI_seq
```

C) ↓ *Rendering to SBOL RDF/XML*

```
<sbol:ComponentDefinition rdf:about="http://partsregistry.org/
    cd/lacI_cds">
  <sbol:persistentIdentity rdf:resource="http://partsregistry.
      org/cd/lacI_cds"/>
  <sbol:displayId>lacI_cds</sbol:displayId>
  <dcterms:title>lacI</dcterms:title>
  <dcterms:description>lacI CDS</dcterms:description>
  <sbol:type rdf:resource="http://www.biopax.org/release/
      biopax-level3.owl#DnaRegion"/>
  <sbol:role rdf:resource="http://identifiers.org/so/
      SO:0000316"/>
  <sbol:sequence rdf:resource="http://partsregistry.org/seq/
      lacI_seq"/>
</sbol:ComponentDefinition>
<sbol:Sequence rdf:about="http://partsregistry.org/seq/seq/
    lacI_seq">
  <sbol:persistentIdentity rdf:resource="http://partsregistry.
      org/seq/seq/lacI_seq"/>
  <sbol:displayId>lacI_seq</sbol:displayId>
  <sbol:elements>atggtgaatgt</sbol:elements>
  <sbol:encoding rdf:resource="http://www.chem.qmul.ac.uk/
      iubmb/misc/naseq.html"/>
</sbol:Sequence>
```

Figure 1: Rendering SBOL documents using ShortBOL. A genetic circuit representation in ShortBOL is recursively rendered using templates until standard SBOL documents are produced. A) Shorthand representation of a CDS component. B) This shorthand representation is recursively expanded into a version that includes no reference to a template. C) Standard SBOL representation of the same component is produced.

being re-written until they become RDF/XML documents. The steps are as follows:

- *Import processing:* Import URIs are resolved to ShortBOL documents. These are then interpreted and the declared assignments and templates made available to the current shorthand script.
- *Variable assignment:* Assigned values are associated with their alias, and made available for value substitution.
- *Template registration:* Templates are associated with their identifier, and made available for future application.
- *Variable substitution:* Identifiers are looked up in the current assignment dictionary, and if found, replaced with the corresponding value.
- *Template expansion:* If the name of a template application matches a registered template, expand that template and set all the nested properties.
- *Hoisting:* If a property's value is set to a complex object where a reference was expected (e.g. the sequence property of a ComponentDefinition holds as value a Sequence instance), hoist the value up to the top level and replace the value with a reference.
- *Assigning identifiers:* Mint identifiers (URIs) for any instances that are anonymous in shorthand but need identifiers in an RDF rendering.
- *Transliteration to RDF/XML:* Generate XML/RDF from the processed shorthand script. This is a direct transliteration of the URIs associated with identifiers, property names and remaining template applications to RDF resources, predicates and classes.

3. CONCLUSIONS

ShortBOL fulfills the need for an SBOL shorthand. It is designed to be easy for biologists to read and write, allowing the rapid creation and exchanging of synbio designs without heavyweight computational tools or the need for a mediating GUI. ShortBOL comes with a formal syntax and semantics, so is also suitable for machine exchange. ShortBOL is not intended to replace SBOL, which can represent complex design information with user-defined semantics. Moreover, SBOL is based on RDF and can benefit from existing Semantic Web tooling. The ShortBOL syntax simplifies the creation of SBOL documents. As a textual language, with a defined syntax, it has the advantage of describing design information unambiguously for machines, compared to visual languages which are for human consumption.

Development of a fully on-line editor and expansion pipeline is ongoing, supporting while-you-type integration with other SBOL tooling, including VisBOL ([4]). We hope that the open nature of ShortBOL template libraries will support rapid development of SBOL extensions and domain-specific design terminologies[1]. Moreover, we envisage community-driven development of template libraries to intuitively design biological systems according to the needs of different labs.

4. ACKNOWLEDGMENTS

The Engineering and Physical Sciences Research Council grant EP/J02175X/1 (to A.W. and G.M.). JM is supported by FUJIFILM DioSynth biotechnologies.

5. REFERENCES

[1] B. Bartley et al. Synthetic Biology Open Language (SBOL) Version 2.0.0. *J Integr Bioinform*, 12(2):272, 2015.

[2] L. Clark & Parsia. SBOL design tool powered by semantic technologies. http://clarkparsia.github.io/sbol/.

[3] J. A. McLaughlin et al. An environment for augmented biodesign using integrated data resources, submitted. In *8th International Workshop on Bio-Design Automation*, 2016.

[4] J. A. McLaughlin et al. Visbol: Web-based tools for synthetic biology design visualization. *ACS Synth Biol*, 2016.

[5] Z. Zhang et al. libsbolj 2.0: A java library to support sbol 2.0. *IEEE Life Sci Lett*, 1(4):34–37, Dec 2016.

[1]Examples are available online at https://github.com/drdozer/shortbol

A Data Model for the Description of Biopart Datasheets

Iñaki Sainz de Murieta
Imperial College London
RSM 4.01
South Kensington Campus
London SW7 2AZ
i.sainzdemurieta@ic.ac.uk

Matthieu Bultelle
Imperial College London
RSM 4.01
South Kensington Campus
London SW7 2AZ
m.bultelle97@imperial.ac.uk

Richard I. Kitney
Imperial College London
RSM 3.16
South Kensington Campus
London SW7 2AZ
r.kitney@imperial.ac.uk

CCS Concepts
Information systems → Information systems applications → Enterprise information systems. Applied computing → Life and medical sciences → Genomics. Life and Medical Sciences → Biology-related information processing

Keywords
Synthetic Biology, Datasheet, Standardisation, Characterisation, Data Model.

INTRODUCTION
Systematic design, also known as design-build-test cycle, has become instrumental in the development of synthetic biology as an engineering discipline. An important part of the systematic design approach is to define the behaviour of synthetic biology parts, within a particular host, such that it is repeatable.

A common concept in much of engineering is that systems can be produced from the combination of standardised components. For this to be accomplishable in synthetic biology, there is an imperative to develop fully characterised parts that can be made available in public catalogues or registries, so they can be picked and reused following a bottom-up design approach. Taking the example of the design of a device, the first step in the process is to define the device specifications –then, on the basis of the specifications, to develop the design. The modularisation approach means that the device will comprise a number of parts connected together.

It is important to note that, as with the same approach in other fields, the interfacing of the parts may be non-trivial. However, this is offset by the fundamental power of the approach, which is that every device does not need to be designed and built from scratch. Previous devices (either originating from nature, or human designed, possibly by other teams in different locations) can be reused and altered with standard parts drawn from repositories. When there is a requirement for new parts to be designed and built they need to go through a process of characterisation, which may also include host characterisation and DNA assembly.

An important part of the systematic design approach is to define the behaviour of synthetic biology parts, within a particular host, such that it is repeatable. This process is called characterisation [3], which consists of describing the part's behaviour and performance by two sets of information – the characterisation data (which is the record of the part's behaviour within the host cell) and the metadata (data on the experimental and other conditions). These sets of data should be the result of a characterisation experiment performed specifically for the part under study. Systematic design and modularisation require that the properties of parts and their functional behaviour are extremely well characterised [3].

Different biopart repositories have been created to date, such as the iGEM Registry of Standard Biological Parts [5], the Virtual Parts Repository [8], the Standard European Vector Arquitecture (SEVA) Repository [4], or the JBEI Inventory of Composable Elements (ICE) [2], to name a few. Although they are taking advantage of standards such as FASTA, GenBank or SBOL[1] for the representation of genetic constructs, when it comes to the representation of other quantitative or qualitative features, they all implement their own data models. It is our opinion that the absence of a common data model for the representation of bioparts, which goes beyond the sequence annotations and includes quantitative and qualitative features of a biopart, is hindering the development of all these repositories into standard tools that are commonly used by synthetic biologists in conjunction with bioCAD tools.

DATA MODEL
Aiming for standardisation of biopart datasheets, present raw data measurements in an efficient manner and to facilitate their storage and retrieval, we have developed the first data acquisition standard for synthetic biology: DICOM-SB [6]. It is based on the highly successful Digital Imaging and Communications in Medicine (DICOM). Its data model comprises not only the representation of the raw data produced by the experiment, but, also, all the metadata related to the experimental context (e.g. chassis information, biopart sequence, experimental protocols, experiment date, modality settings, etc). DICOM-SB been specifically tailored for synthetic biology, such that it is modular, optimised for the storage of large amounts of data and empowers data interoperability.

Once standard experimental results are available, the data undergo a process of curation (involving e.g. normalisation and data analysis) to produce the final set of features available in a biopart datasheet. The present work focuses on expanding the data model previously presented by the authors [5], such that it can be used to encode any kind of biopart datasheet at any repository, following a unified model.

The UML diagram in Figure 1 represents the basic information entities required in the curation of a datasheet: an Analysis Protocol is executed, taking as input the experimental results from the wet lab, and producing a set of curated data (normalised and analysed). Input and output data can be modelled as DICOM-SB Series, which currently allows the representation of raw data coming from plate reader or flow cytometer modalities. Raw data formatted in several series can be used to generate different metrics (e.g. relative promoter units / RPU, doubling time, growth rate) as properties of a Metrics table. Properties may change depending on the type of biopart under analysis, equipment used, etc. The protocol may (or not) be related to a Model that describes the processing steps (Model already existing in SBOL 2.0).

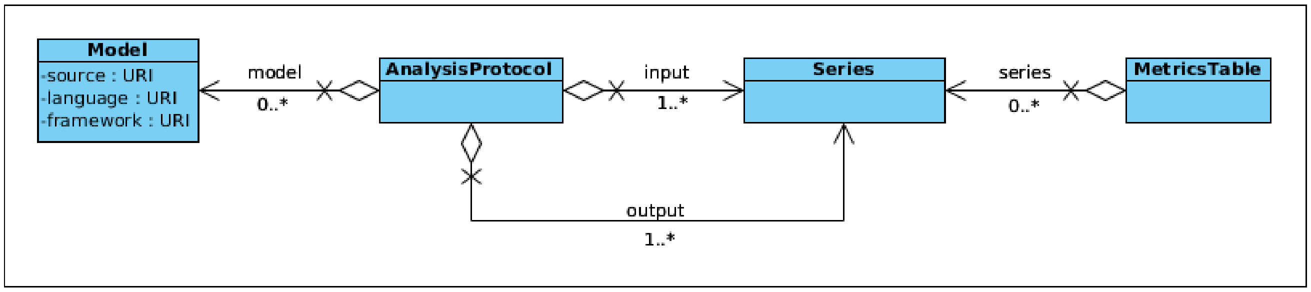


Figure 1. Producing metrics from experimental data.

Upon completion of the data analysis, all the Metrics Tables are grouped into a Datasheet (see Figure 2). This should be the entity to be referenced when talking about the implementation of a specific biopart within a host. It should contain at least one reference to a metrics table from the "metrics" property, as well as references to the institution where the component has been characterised, person / team running the experiments, person / team processing data, etc.

PRESENT AND FUTURE WORKS

The Synthetic Biology Information System (SynBIS) developed at the Imperial College Centre for Synthetic Biology and Innovation (CSynBI) is using the data model here introduced to disseminate the data obtained from our automatised characterisation pipeline. Examples of datasheets encoded using a preliminary version of the model proposed here can be found at the SynBIS website [7], where the reader can find pdf descriptions of several datasheets, as well as an API offering XML and SBOL serializations. The CSynBI is also working with other institutions aiming to better understand the data they produce, enhance the present model accordingly (and potentially produce new official DICOM-SB and SBOL extensions) and automate the incorporation of the data they produce into SynBIS.

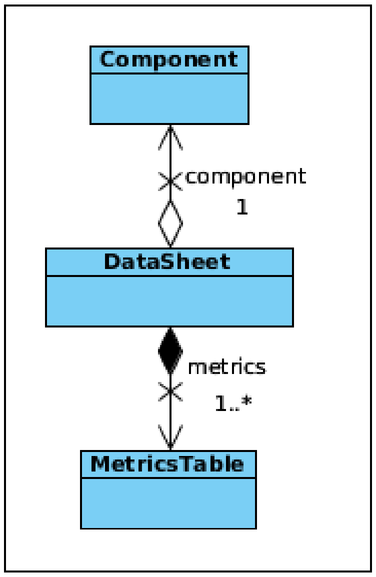


Figure 2: Producing datasheets from metrics.

REFERENCES

[1] Bartley, B. et al. 2015. Synthetic Biology Open Language (SBOL) Version 2.0.0. *Journal of Integrative Bioinformatics*. 12, 2 (2015), 272.

[2] JBEI Inventory of Composable Elements (ICE): *https://public-registry.jbei.org*.

[3] Kitney, R. and Freemont, P. 2012. Synthetic biology – the state of play. *FEBS Letters*. 586, 15 (Jul. 2012), 2029–2036.

[4] Martínez-García, E. et al. 2014. SEVA 2.0: an update of the Standard European Vector Architecture for de-/re-construction of bacterial functionalities. *Nucleic Acids Research*. (Nov. 2014), gku1114.

[5] Registry of Standard Biological Parts: *http://parts.igem.org*.

[6] Sainz de Murieta, I. et al. 2016. Toward the First Data Acquisition Standard in Synthetic Biology. *ACS Synthetic Biology*. 0, 0 (2016), null.

[7] Synthetic Biology Information System (SynBIS): 2014. *http://synbis.bg.ic.ac.uk*.

[8] Virtual Parts Repository: *http://sbol.ncl.ac.uk:8081*.

DOI: http://dx.doi.org/10.1145/12345.67890

Design and automated inference of design principles in gene regulatory networks: a multiobjective optimization approach

[Extended Abstract]

Irene Otero-Muras
BioProcess Engineering Group, IIM-CSIC
Spanish National Research Council
Vigo, Spain
ireneotero@iim.csic.es

Julio R. Banga
BioProcess Engineering Group, IIM-CSIC
Spanish National Research Council
Vigo, Spain
julio@iim.csic.es

ABSTRACT

In order both to uncover design principles of gene regulatory networks and implement synthetic circuits with increasing levels of complexity, advanced tools for optimal automated design of biocircuits need to take into account more sophisticated setups [1]. Here we present a computational tool for automated design of biocircuits with predefined performance specifications that exploits the efficiency of Mixed Integer Nonlinear Programming solvers to handle high levels of complexity and, on the other hand, incorporates multiple criteria in the design.

From the perspective of forward engineering, our multiobjective modular approach allows designing gene circuits that, in addition to predefined performance specifications, can mimic additional desirable properties of natural circuits. In a reverse engineering mode, the design space is explored to automatically infer structure-functionality relationships of gene networks from Pareto solutions. We use a mixed integer modeling framework based on ordinary differential equations (ODEs), complementing methods recently developed in the field [1] relying on rule base stochastic modeling. We show examples of forward design (automated design of switches and oscillators from a library of components) and reverse engineering (exploring design principles of biological oscillators, switches and stripe forming motifs).

CCS Concepts

•Applied computing → Computational biology; Biological networks;

Keywords

Global Optimization; Multi-objective optimization; Pareto optimality; Gene Regulatory Network; Design principles; Synthetic Biology

IWBDA '16 Newcastle, UK
© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

1. METHODS, RESULTS AND DISCUSSION

We encode the dynamics of gene regulation in a mixed integer framework. A gene circuit is characterized by two vectors x and y of real and integer variables respectively, and the dynamics are given by a set of ordinary differential equations of the form:

$$\dot{z} = f(z, x, y, k), \quad z(0) = z_0 \tag{1}$$

where z is the vector of state variables containing the levels of the species involved, x is the vector of real variables containing the set of tunable parameters, y is the vector of integer variables encoding the circuit structure and k is a vector of fixed parameters. This mixed integer description can accommodate different kinetics (mass action, Hill, etc) and levels of detail (granularity).

The design targets are encoded in a set of objective functions of the form: $J_i(\dot{z}, z, x, y, k)$, $i = 1, \dots, S$, and the automated design is formulated as finding the vector x of continuous variables and the vector y of integer variables which minimize the vector $J = (J_1, J_2, \dots, J_S)$, subject to the dynamics (1) and to additional constraints.

Our method combines an $\varepsilon-$constraint strategy with state of the art MINLP solvers to obtain the Pareto front of optimal solutions to the multiobjective problem. The software implementation is available upon request.

Multiobjective Forward-Engineering Design. In the *forward* mode, we start from a library of parts or components and search for circuits among all possible combinations of devices. Circuit structures are defined by a vector of binary variables y where each entry indicates whether the corresponding device (promoter-repressor pair) is active (1) or not (0).

Switch-like behaviour in response to inducers. Circuits with switch-like behaviour in response to different inducers have been found by [2] using a single objective optimization approach. By introducing as additional design criterion the cost of protein production (protein burden), we find non intuitive configurations with practically the same performance [4], as illustrated in Fig 1 A. In this way we show how multiobjective optimization leads to well defined design problems, providing additional information to select better circuit candidates for lab-implementation.

Sustained endogenous oscillations. Starting from a library of components [5] and encoding the desired oscillatory be-

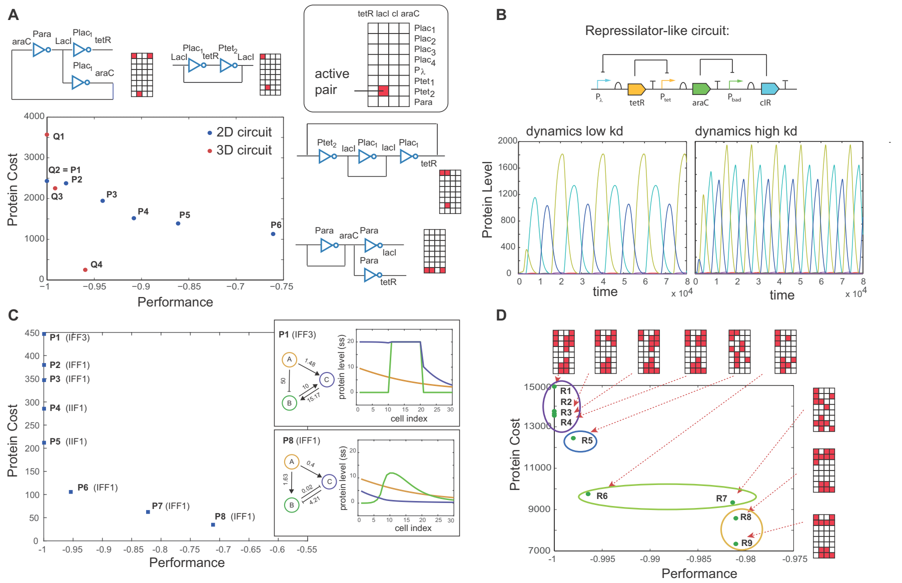


Figure 1: Examples of multiobjective design problems. A) Pareto front of 2D-3D circuits with switch-like behaviour (performance vs protein production cost or *protein burden*). Structure matrix with active promoter-transcript pairs is indicated. B) Endogenous Repressilator-like oscillator with optimal stability and tunability, and its corresponding dynamics for low and high degradation constant (kd). C) Pareto front of 3D stripe forming circuits (performance vs protein production cost). Stripes generated by both Pareto extremals (P1 and P8) are shown. D) Pareto front of circuits with switch-like response upon induction and unconstrained number of devices (performance vs protein production cost).

haviour in a suitable objective function we find a number of oscillators with a Repressilator-like structure. Using the multiobjective approach we select those circuits that, in addition to sustained oscillations satisfy the tradeoff between stability of the limit cycle oscillator (robustness against perturbations in the trajectory), and tunability of the period (variability of the period with respect to a degradation constant in the system). Period tunability has been postulated as evolutionary advantageous in a wide range of natural oscillators. We depict in Fig 1 B one of the circuits fulfilling the tradeoff.

Automated inference of design principles. In *reverse mode* our method allows exploring design principles of gene regulatory networks. We start from biologically verified models encoded in the mixed integer framework (1). We select the decision variables and a priori objectives, solving the resultant multiobjective problem to obtain the Pareto front of nondominated circuits. Solutions are further analyzed to get insight into the design principles of regulatory circuits, including what behavioural targets are in a tradeoff, as well as potential relationships between structure-parameter patterns and functionality.

Stripe forming motifs. Feed-forward IFF1 and IFF3 motifs are found as core structures for stripe formation in 1D tissues in [3] by combining exhaustive exploration and analytical methods. Using optimization, we automatize the single objective search ensuring computational efficiency and optimality (circuit with best stripe formation performance). Adding the cost of protein production as an opposing objective we obtain the Pareto front in Fig 1 C. We can conclude that IFF3 and IFF1 are stripe forming motifs, with structures evolving from IFF3 to the IFF1 as we move along the Pareto front (performance vs. cost).

Switches. In order to automatize the analysis of the solutions we develop a clustering method to automatically detect structure-parameter patterns along the Pareto front. This is illustrated in Fig 1 D where we find four clusters along the front of non dominated circuits.

Oscillators. Starting from the same 3-gene model, we search for sustained oscillators finding that CFF4 and IFF3 not only exhibit oscillatory behaviour but also fulfill the tradeoff between oscillation stability and period tunability.

Acknowledgments: We acknowledge funding from MINECO project SYNBIOFACTORY (DPI2014-55276-C5-2-R).

2. REFERENCES

[1] H. Cao, F. Romero-Campero, S. Heeb, M. Camara, and N. Krasnogor. Evolving cell models for systems and synthetic biology. *Syst Synth Biol*, 4(1):55–84, 2010.

[2] M. S. Dasika and C. D. Maranas. Optcircuit: An optimization based method for computational design of genetic circuits. *BMC Syst Biol*, 2:24, 2008.

[3] A. Munteanu, J. Cotterell, R. Sole, and J. Sharpe. Design principles of stripe-forming motifs:the role of positive feedback. *Sci Rep*, 4:5003, 2014.

[4] I. Otero-Muras and J. R. Banga. Multicriteria global optimization for biocircuit design. *BMC Syst Biol*, 8:113, 2014.

[5] M. Pedersen and A. Phillips. Towards programming languages for genetic engineering of living cells. *J R Soc Interface*, 6:S437–S450, 2009.

How to Remember and Revisit Many Genetic Design Variants Automatically

Nicholas Roehner
Boston University, US
nicholasroehner@gmail.com

Douglas Densmore
Boston University, US
dougd@bu.edu

1. INTRODUCTION

Consider the design of a small bacterial gene cluster containing up to four genes. Even under the assumption that all of the cluster genes must have the same orientation and that the library of available parts for controlling these genes' expression contains only four constitutive promoters, four ribosome binding sites, and four terminators, there are still 684,544 different cluster design variants to choose from. Encoding and storing these design variants in most modern genetic part/design repositories [3] requires a significant amount of memory, over 6 gigabytes if one assumes an average of 10 kilobytes per design variant. As genetic design scales from larger gene clusters [5] to genomes, it is clear that storing each individual design variant is not a practical solution to specifying the space of all possible designs and tracking how different versions of this space change over time. Both of these concepts–design specification and version control–are critical to the support of tools for learning what makes a genetic design work.

Previous rule-based [1] and grammar-based [2] approaches to genetic design have been successful in specifying the composition constraints and patterns of DNA components making up design variants, but they have done so without a firm theoretical basis for comparing these specifications. Without such a basis, it is difficult for a machine to merge similar specifications and calculate the differences between them, two key features of modern version control systems. To meet this need, we have developed an approach to genetic version control and implemented it in a tool called Knox. Our approach builds on the formal framework of a language/tool called Finch[1], in which genetic design variants are specified, compared, and composed as "genetic design spaces."

2. GENETIC DESIGN SPACES

A genetic design space is a graph that implicitly represents many genetic designs by encoding rules for composing sets of DNA components. As shown in Figure 1, each design space consists of a set of nodes and a set of directed edges between these nodes. In general, each edge can be labeled with a set of DNA components to choose from, but here we label each edge with a single DNA component for simplicity. Regardless, each path from a start node (green) to a stop node (red) represents a linear composition of one DNA component from each edge into a genetic design (see Figure 1C). Designs inferred in this manner are "correct by construction," adhering to the rules encoded by the graph.

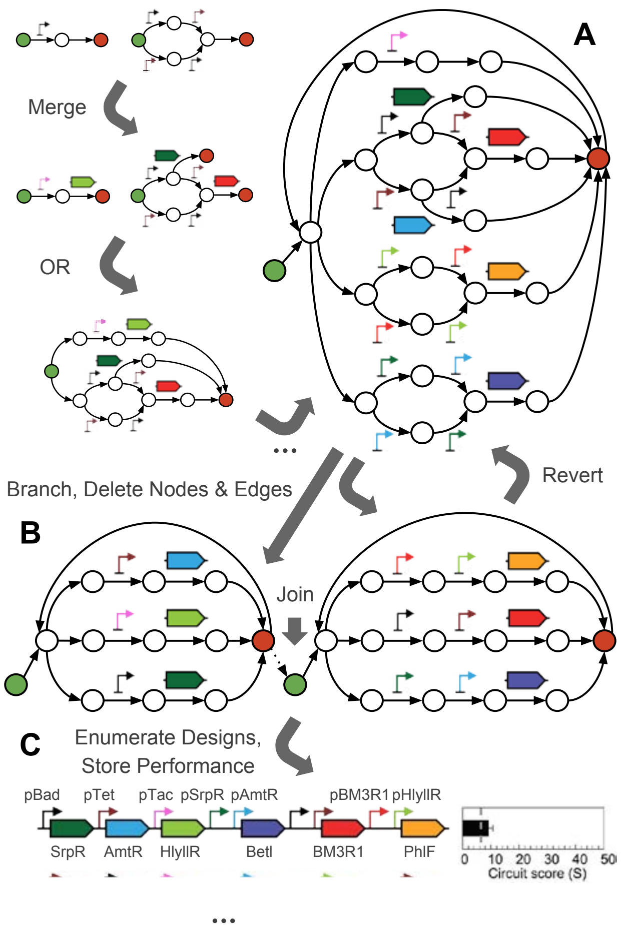


Figure 1: Version control of a Cello-designed Majority circuit with Knox. (A) Building up the genetic design space for the Majority circuit. Each gray arrow represents the application of an operator from Knox. (B) Pruning the design space and later reverting to its original form. (C) Enumerating designs from the pruned design space and storing performance data. See [4] for a definition of circuit score. The dashed line indicates the worst circuit score found.

[1] www.synbiotools.org

Besides storing large numbers of genetic design variants more compactly, the design space formalism has the added benefit that new design rules can be incorporated via straightforward graph operations. Knox provides a variety of such operations for editing and combining design spaces, including Join, OR, AND, Merge, Repeat, and Delete. In Figure 1A, Merge, OR, and Repeat are used to build up a design space that captures all gene and promoter orders for a Majority transcriptional logic circuit designed with Cello [4].

3. GENETIC VERSION CONTROL

As a version control system, Knox provides operators similar to those used in Git, including Branch, Checkout, Commit, Merge, and Revert. Unlike their Git equivalents, the Knox version control operators take the semantics of design spaces into account, which enables Knox to record version histories in terms of changes to a graph rather than changes to text. In this way, biodesign automation (BDA) tools can more readily interpret these changes and act upon them for applications such as machine learning. Changes to a graph are computed by merging the paths from its start node with those of its previous versions and removing the common edges.

Figure 1 provides an example of how Knox can aid in the refinement of the Majority circuit. Since every Knox operator that combines two design spaces also combines their version histories, building up the initial design space for the Majority circuit from the design spaces for its individual genes in Figure 1A also creates a record of which design spaces were combined and in what order (see Figure 2).

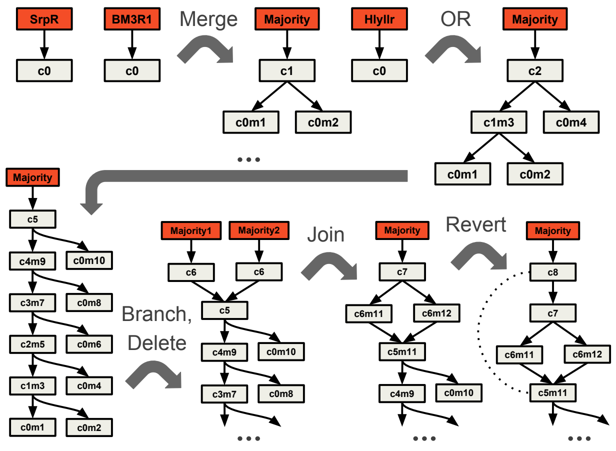


Figure 2: The changing version history of the genetic design space for the Majority circuit. Each white box represents saved changes to the design space (a commit), while each red box represents a different version/branch of the design space and points to the latest commit on that branch. Each gray arrow represents an operation in Knox and corresponds with an arrow in Figure 1. The dotted line indicates that two commits are identical.

In Figure 1B, the Majority design space is pruned so that it only captures designs in which all one-promoter genes precede all two-promoter genes. In particular, the Branch operator is used to create two different versions/branches of the Majority design space. The Delete operator is then used to prune these branches so that they only contain one-promoter and two-promoter genes, respectively. As shown in Figure 2, these changes are saved/committed to the separate branches before being combined using the Join operator. All combinational operators in Knox can take two design spaces or two branches of a single design space as inputs.

Finally, as shown in Figure 1C, circuit designs are enumerated from the pruned design space and data are collected on their performance and scored. Since these designs are not much better than the worst designs tested so far, one might use the Revert operator to undo the changes to the design space shown in Figure 1B. Like its Git equivalent, the Revert operator creates a new commit that is a copy of a previous commit, rather than destroy the intervening commits (see Figure 2). In doing so, a complete record of all changes made to the branches of the design space is maintained, such that one may revisit any previous state of the design space.

4. CONCLUSION

Knox is not just a database for efficiently storing genetic design variants. It is a genetic version control system that enables BDA tools to track and revert changes to genetic design spaces as new data on design performance are gathered and new design rules are learned. As genetic design scales to larger systems containing many sequence feature variants, version control will play a critical role in supporting applications ranging from statistical design to machine learning. This support could involve further extension of the genetic design space formalism to incorporate quantitative data on system parameters and conditional probabilities.

5. ACKNOWLEDGMENTS

This work is supported by the DARPA Living Foundries award HR0011-15-C-0084. We thank Swapnil Bhatia (Boston University) for his work on the genetic design space formalism leveraged by Knox. We also thank Christopher Voigt (MIT) and D. Benjamin Gordon (MIT-Broad Foundry) for helpful discussions related to the presentation and applications of Knox.

6. REFERENCES

[1] 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.

[2] 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.

[3] T. S. Ham et al. Design, implementation and practice of JBEI-ICE: an open source biological part registry platform and tools. *Nucleic Acids Res.*, 40(18):e141, 2012.

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

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

Towards automated biosecurity: screening of synthetic biology constructs

Benjamin Apra
Twist Bioscience
455 Mission Bay Blvd. South
San Francisco, CA 94158
1-415-216-8966
bapra@twistbioscience.com

Arthur Vigil
Twist Bioscience
455 Mission Bay Blvd. South
San Francisco, CA 94158
1-415-216-8966
avigil@twistbioscience.com

James Diggans, PhD
Twist Bioscience
455 Mission Bay Blvd. South
San Francisco, CA 94158
1-415-216-8966
jdiggans@twistbioscience.com

ABSTRACT

The growth rate in our collective knowledge about individual proteins and biological systems capable of posing potential threats to public safety and/or the environment is tremendous. This knowledge, however, is widely distributed across diverse research communities, institutions and even journals. No centralized information source focuses on annotating the potential for a given protein to cause harm and in what context this harm can arise. Here we introduce two key tools to address this challenge. A web-based curation system allows experts to curate sequences of concern. A companion web-based screening system aids synthetic biologists in screening sequences against a curated collection of potential threats. Together these tools can lower the bar to effective biosecurity screening and increase the safety of synthetic biology research.

CCS Concepts

• Applied computing → Life and medical sciences

• Applied computing → Life and medical sciences → Bioinformatics

• Applied computing → Life and medical sciences → Bioinformatics

Keywords

Biosecurity; synthetic biology; synthetic DNA; security; pathogenicity; biosafety; screening; algorithms; gene ontology.

1. INTRODUCTION

With the rapid growth in design capability in synthetic biology, it is now possible to create large numbers of constructs often using heavily mutated sequence that does not directly resemble the reference sequence from which it was originally derived. At the same time, scientific advances in the understanding of the processes behind pathogenicity (in a variety of hosts and biological contexts) are rapidly creating new knowledge of protein sequences that, in context-dependent ways, can cause harm to human beings, specific plants or animals, or to the environment more broadly.

Regulatory regimes in the US, EU and other countries recognize this potential for harm in biological agents known to naturally express these proteins. The pace of regulation, however, cannot keep pace with science; an increasing number of recently discovered organisms [1] and viruses [2] are known to pose a threat, but access to these organisms or materials is often not directly regulated even years after their initial discovery.

These factors, taken together, have created an environment in which ethical, responsible synthetic biologists may unwittingly create constructs capable of causing harm, but be unable to predict or understand that capability prior to instantiating synthetic designs in living systems. As predicting function from primary sequence alone is not feasible [3], these scientists would be well-served by having access to 1) a repository of metadata on what sequences can cause harm along with regulatory status and 2) an effective screening system for checking DNA or protein sequences against that metadata and alerting the user to any potential concern. In addition, a screening system capable of addressing these needs must itself be amenable to automation so as to fit seamlessly into high-throughput design/build/test workflows.

The current burden for sequence screening falls on the companies that synthesize gene-length double-stranded DNA. The US government, in 2010, provided guidance [4] to these companies, recommending a set of basic screening practices to ensure that sequences from pathogens or toxins falling under current biosecurity or export control regulation were not inadvertently sequenced or exported.

In 2009, major synthesis companies formed the International Gene Synthesis Consortium (IGSC), an industry body dedicated to developing best practices in biosecurity screening and assisting governments in preventing misuse of gene synthesis technology. The IGSC developed a harmonized screening protocol [5], spelling out the process by which member companies screen ordered sequences. IGSC member companies screen using the IGSC Regulated Pathogen Database, an internally curated set of sequences that fall under the control of one or more governmental regulations. IGSC does not make this database or any screening tools available to the public, limiting the utility of their screening expertise only to sequences ordered commercially by synthetic biologists. In addition, the Regulated Pathogen Database by its very nature is updated only as the regulatory environment changes; it is not intended to keep up with scientific knowledge but rather to establish a baseline for safety that parallels government regulation.

Here we demonstrate a pair of software tools to address both the lack of publicly available protein-level metadata on pathogenicity as well as the lack of open source tools for effective screening.

2. SEQUENCE ANNOTATION

Knowledge about the capacity of any single sequence to cause some type of harm is extremely distributed. Individual communities of researchers focus on widely varying aspects of pathogenicity including the ability of organisms to infiltrate host cells, hijack host cellular machinery, hide from the host immune system and even to enhance the host immune response.

The distributed nature of this knowledge makes centralization challenging but not impossible: witness the Gene Ontology consortium [6], which has done exactly this centralization for more general function and location-based categorization of genes. SNPedia [7] is another example of open knowledge created and maintained by a community. For each polymorphism, SNPedia provides a description and links to relevant journal articles. SNPedia even offers a application, *Promethease*, which leverages SNPedia contents to analyze individual variability.

We have created a Mediawiki-based user interface in which interested parties can submit sequences along with tag-based annotation of roles in pathogenicity. The interface is available at *seqshield.com*. Users are encouraged to submit several tags for each sequence to describe the general patterns of harm associated with a given sequence modeled as

Host + Context = Outcome + Level of Concern

The present system takes a tag-based approach so as not a priori to impose a single controlled vocabulary. The collection of tags resulting from community annotation could form the basis of such a controlled vocabulary over the longer term.

As each sequence is uploaded, users are asked to add tags in each of four categories. Tagging 'Host' and 'Level of Concern' are mandatory; adding tags for 'Context' and 'Outcome' are optional given the additional complexity and domain knowledge required.

As an example, a sequence encoding the toxin ricin might be tagged by a user as:

Tag	Values
Host	human
Context	ingestion, inhalation
Outcome	fever, cough, respiratory_failure, death
Level of Concern	extreme

The goal is accumulation of metadata over time more than universal completeness. The system is centrally hosted and offers the entire set of curated sequences (or subsets based on queries by tag) for download as FASTA or BLAST Database for use in screening.

Data quality and public participation can both be concerns associated with publicly available databases. To maximize immediate utility, we have carried out an initial curation process adding many pathogenic proteins to the database in an attempt to include most potentially regulated sequences or other sequences known to be harmful. We have also curated a white list of NCBI GI identifiers corresponding to proteins we consider harmless. That white list is also open to curation.

We use CAPTCHAs to prevent unauthorized bot-driven curation and require user registration before creating or editing pages. GI identifiers are periodically verified (for existence) and records tagged for human review on failure. Users can also flag records to request community or administrator review.

3. SCREENING TOOL

Constructing a screening system capable of determining whether a given sequence poses a biosecurity risk requires a degree of investment in time and expertise not available to all synthetic biologists or even to all synthetic biology companies. Even assuming one has access to a database of dangerous sequences, basic parameterization of an aligner and result processing (including culling alignment counts to similar regions so as not to hide homology to shorter regions) requires domain expertise.

Here we introduce *bioseqscreen*, a Python-based reference implementation of a screening system available at *github.com/twistbioscience/bioseqscreen*. Given a query nucleotide sequence, the tool compares that sequence (via BLAST [8]) to the set of protein sequences derived from the annotated collection produced by the interface discussed in the previous section.

Results can be filtered by the degree of homology, E-score and alignment length. Passing hits are summarized by the distribution of tags associated with those sequences and the regions of the query found problematic. Links to the originating database entries are provided so that users can follow-up in more detail. In compliance with IGSC guidance, the algorithm is 100% sensitive and reports can be downloaded for archival use. Screening short (<200 bp) sequences will result in a large number of false positive findings. Effective screening of oligo-length sequences requires a unique algorithmic approach not (yet) addressed by this tool.

The screening system sits atop a database and includes a RESTful API for screen request submission and result retrieval as well as a graphical user interface. The application is easy to install and operate on a laptop computer, and scales reasonably well to high-throughput use via API calls.

4. CONCLUSION

We have introduced two software tools enabling a new approach to effective biosecurity based on community knowledge and participation. It is our hope that the annotation tool will help the synthetic biology community to track emerging science on the link between individual proteins and negative outcomes. We also hope the screening tool enables the community to broaden both interest in- and effective practice of biosecurity so that practitioners are empowered to evaluate the safety of their designs during the design phase rather than waiting until synthesis or even expression.

5. REFERENCES

[1] Fisher, M.C. et al. 2009. Global emergence of Batrachochytrium dendrobatidis and amphibian chytridiomycosis in space, time, and host. *Annual review of microbiology*. 63, (Jan. 2009), 291–310.

[2] Rasmussen, S.A. et al. 2016. Zika Virus and Birth Defects — Reviewing the Evidence for Causality. *New England Journal of Medicine*. (Apr. 2016), NEJMsr1604338.

[3] National Research Council, 2010. *Sequence-Based Classification of Select Agents: A Brighter Line*. DOI = http://dels.nas.edu/Report/Sequence-Based-Classification-Select-Agents/12970

[4] Department of Health and Human Services, 2010. *Screening Framework Guidance for Providers of Synthetic Double-Stranded DNA*. DOI = http://www.phe.gov/Preparedness/legal/guidance/syndna/Documents/syndna-guidance.pdf

[5] IGSC, 2009. *Harmonized Screening Protocol*. DOI = https://customer.sgidna.com/files/IGSC Harmonized Screening Protocol.pdf

[6] Ashburner, M. et al. 2000. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature genetics*. 25, 1 (May 2000), 25–9.

[7] Cariaso, M. and Lennon, G. 2012. SNPedia: a wiki supporting personal genome annotation, interpretation and analysis. *Nucleic acids research*. 40, Database issue (Jan. 2012), D1308–12.

[8] Altschul, S.F. et al. 1990. Basic local alignment search tool. *Journal of molecular biology*. 215, 3 (Oct. 1990), 403–10

Realization of Large Logic Circuits with Long-Term Memory Using CRISPR/Cas9 Systems

Tai-Yin Chiu[1], Cheng-Han Hsieh[2], and Jie-Hong R. Jiang[1,2]
[1]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
[2]Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
{b99202046, b01901080, jhjiang}@ntu.edu.tw

1. INTRODUCTION

Synthetic biological circuits in living cells are commonly designed with certain intended logic functions that make cells respond to specific environment stimuli or even change their growth and cellular development. When it comes to designing logic circuits in a cell, in addition to functional correctness one may further require the circuit configuration changes to be stored and propagated to its descendant cells. In recent work [1], a scheme for constructing synthetic cellular circuits with integrated logic and memory was proposed, where recombinases such as *Bxb1* and *phiC31* were used to implement various two-input logic gates. The so implemented circuits were built and tested in *Escherichia coli* cells and they showed a long-term memory for at least 90 cell generations. This approach, however, is primarily limited by the available recombinases and may not be scaled to implement large circuits. In this paper, we overcome this limitation by using the CRISPR/Cas9 system for large logic circuit implementation.

Before introducing our method, we illustrate the inner working of the scheme proposed in [1] with a two-input AND gate example shown in Figure 1. Let chemicals AHL and aTc be the inputs to the AND gate. Inducers AHL and aTc first activate the expression of recombinases *Bxb1* and *phiC31*, respectively. These recombinases will then irreversibly invert (flip) the DNA sequences flanked by their recognition sites. The DNA sequences being flanked can be a promoter, a transcription terminator, or a reporter, say, a green fluorescent protein (GFP). Inverting these DNA sequences will alter the output gene expression. In Figure 1, two terminators were flanked by the recognition sites of recombinases *Bxb1* and *phiC31*, and the output green fluorescent reporter is highly expressed only when both inducers AHL and aTc are in high concentration to flip, and thus disabling, both terminators. Therefore, the circuit of Figure 1 effectively implements a two-input AND gate. Note that such DNA sequence changes will survive cell division and can be inherited by the descendant cells. Hence the logic function with a long-term memory is achieved.

To construct a circuit containing m gates each with n inputs, the prior method may require up to mn different types of recombinases. The construction is not feasible when the number of available recombinases is less than mn. Because CRISPR/Cas9 systems [2] are able to induce irreversible DNA sequence inversions [3, 4, 5] and target millions of different sites according to its single guide RNA (sgRNA) sequence, it can potentially be used to implement large logic circuits with a long-term memory. In this paper, we demonstrate how the CRISPR/Cas9 system can be used to implement complex logic gates and large logic circuits. Unlike prior work [6] using Cas9 to repress gene expression without its nuclease activity, our method utilizes the Cas9 nuclease as recombinases.

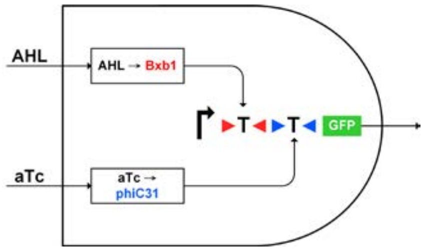


Figure 1: Implementation of an AND gate using recombinases. The right-turn arrow represents a promoter; the red and blue triangles are the targeting sites of recombinases *Bxb1* and *phiC31*, respectively; the letter T's flanked by the targeting sites are transcription terminators; the green box represents the gene encoding the green fluorescent protein.

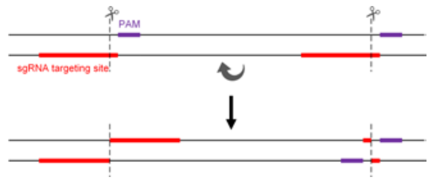


Figure 2: Schematic illustration of DNA sequences inversion using Cas9 nuclease.

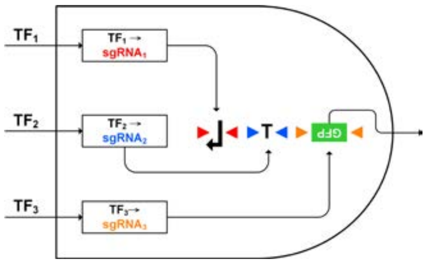


Figure 3: Implementation of a 3-input AND gate using Cas9 nucleases. The red, blue, and orange triangles denote the targeting sites of Cas9-sgRNA$_i, i = 1, 2, 3$, respectively.

Similar to recombinases, Cas9 nuclease is also capable of irreversibly inverting DNA sequences. Figure 2 shows a schematic illustration how Cas9 proteins induce irreversible inversion. First, the Cas9-sgRNA complex specifically binds to its recognition sites composed of a short DNA sequence called protospacer adjacent motif (PAM) and a DNA sequence complementary to sgRNA. After targeting, Cas9 nuclease cleaves the DNA strands and generates two double-strand breaks (DSBs). Finally, it inverts the DNA sequences between the two DSBs. Since the recognition sites are destroyed after the inversion, Cas9 nuclease cannot bind and invert the sequence again.

2. METHODS

With the ability to induce DNA sequence inversion, Cas9-

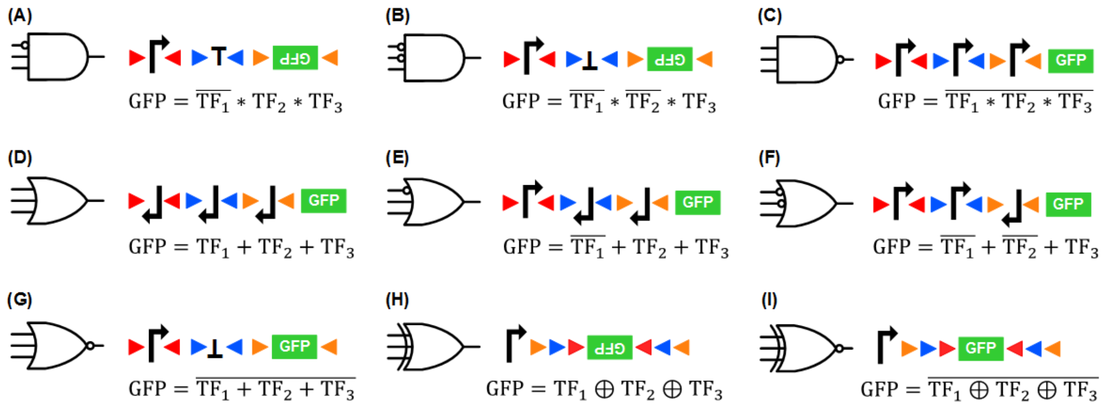

(A)
GFP = $\overline{TF_1} * TF_2 * TF_3$
(B)
GFP = $\overline{TF_1} * \overline{TF_2} * TF_3$
(C)
GFP = $\overline{TF_1 * TF_2 * TF_3}$
(D)
GFP = $TF_1 + TF_2 + TF_3$
(E)
GFP = $\overline{TF_1} + TF_2 + TF_3$
(F)
GFP = $\overline{TF_1} + \overline{TF_2} + TF_3$
(G)
GFP = $\overline{TF_1 + TF_2 + TF_3}$
(H)
GFP = $TF_1 \oplus TF_2 \oplus TF_3$
(I)
GFP = $\overline{TF_1 \oplus TF_2 \oplus TF_3}$
GFP

Figure 4: Implementation of basic 3-input logic gates using Cas9 nucleases. The inputs of each gate from top to down are TF1, TF2, and TF3, respectively; TF_i activates the expression of Cas9-sgRNA$_i$; the red, blue, and orange triangles denote the targeting sites of Cas9-sgRNA$_i, i = 1, 2, 3$, respectively.

sgRNA complex can replace recombinases to fulfill gate operations. Since Cas9 nucleases with different sgRNAs have different recognition sites and there are billions of different sgRNA designs, there are equivalently billions of different recombinases made of Cas9-sgRNA complex, which makes multi-input cellular logic gates feasible. As an example, Figure 3 show a realization of a 3-input AND gate using Cas9 nucleases. The logic gate takes three different transcription factors (TFs) as inputs. Let TF_i activate the expression of sgRNA$_i$, for $i = 1, 2, 3$ along with Cas9 nuclease. Then sgRNA$_i$ and Cas9 nuclease together form a complex to induce the inversion of the corresponding DNA sequence. In order to express GFP in this gate, first we require Cas9-sgRNA$_1$ to invert the inverted promoter so that the GFP gene can possibly be expressed. Second, Cas9-sgRNA$_2$ is needed to flip the terminator to avoid the termination of transcription before reaching the GFP gene. Third, Cas9-sgRNA$_3$ is demanded to upright the GFP gene. Collectively, to have GFP highly expressed all TF_i's must exist, which makes this circuit a 3-input AND gate.

In Figure 4 we present nine other basic 3-input gates implemented with Cas9 nucleases. One special implementation is the XOR gate in (H). In this gate existence of one or three input TFs results in one or three times of GFP gene inversion and makes the upside down gene become upright, while existence of two input TFs makes GFP gene flip twice and remain upside down. Similar situations happen in the XNOR gate in (I).

Since Cas9 nucleases can be applied to the implementations of multi-input gates, we are not constrained to only 3-bit gates and basic gate types such as AND, OR, NAND, NOR, XOR, and XNOR gates. Rather, we can construct complex logic gates with more inputs. Figure 5(A) shows an example of a 4-input logic circuit

$$O = (A_1 + \bar{A}_2 \oplus A_3)\bar{A}_4,$$

which can be directly realized by a single 4-input complex logic gate, instead of cascading multiple two-bit gates.

Another advantage of constructing genetic gates using Cas9 nucleases is that more logic gates can be built in DNA strands of a given length using Cas9 nucleases than recombinases. It is because the length of the targeting site of a Cas9 nuclease (20 nt) is shorter than that of a recombinase (about 50 nt for *Bxb1* and 40 nt for *phiC31*). This also makes Cas9 nucleases more appropriate for the implementation of large logic circuits.

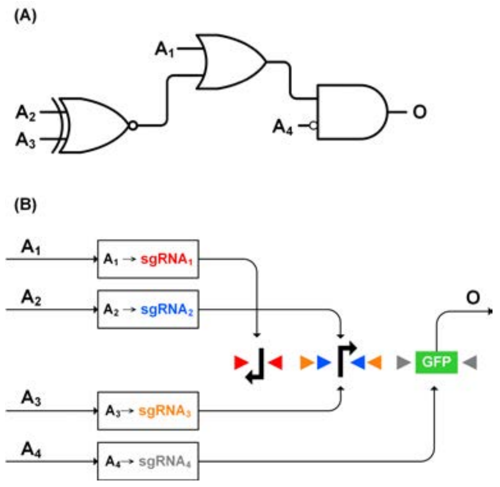

(A)
A_1
A_2
A_3
A_4
O
(B)
A_1
$A_1 \rightarrow$ sgRNA$_1$
A_2
$A_2 \rightarrow$ sgRNA$_2$
A_3
$A_3 \rightarrow$ sgRNA$_3$
A_4
$A_4 \rightarrow$ sgRNA$_4$
O
GFP

Figure 5: (A) Schematic illustration of a 4-bit non-basic logic function $O = (A_1 + \bar{A}_2 \oplus A_3)\bar{A}_4$ (B) Corresponding implementation using Cas9 nucleases.

3. CONCLUSION

We showed the possibility of using CRISPR/Cas9 systems to implement large logic circuits with long-term memory. We gave a few examples of multi-input gate implementations, which can be composed to form large circuits; a systematic construction of general complex logic gates is to be detailed in a technical report. Also we discussed its several advantages in terms of scalability and DNA length efficiency. We envision its wide bioengineering applications.

4. REFERENCES

[1] P. Siuti *et al.* Synthetic circuits integrating logic and memory in living cells. *Nature Biotechnology*, 31(5):448–452, 2013.

[2] L. Cong *et al.* Multiplex genome engineering using crispr/cas systems. *Science*, 339(6121):819–823, 2013.

[3] R. B. Blasco *et al.* Simple and rapid in vivo generation of chromosomal rearrangements using crispr/cas9 technology. *Cell Reports*, 9(4):1219–1227, 2014.

[4] P. S. Choi and M. Meyerson. Targeted genomic rearrangements using crispr/cas technology. *Nature Communications*, 5:3728, 2014.

[5] J. Li *et al.* Efficient inversions and duplications of mammalian regulatory dna elements and gene clusters by crispr/cas9. *Journal of Molecular Cell Biology*, 7(4):284–98, 2015.

[6] A. A. Nielsen and C. A. Voigt. Multi-input crispr/cas genetic circuits that interface host regulatory networks. *Molecular Systems Biology*, 10(11):763, 2014.

3D printing of microbes for material production

Benjamin Lehner
TU Delft Bionanoscience
Lorentzweg 1
2628 CJ Delft (NL)
+31 15-278-6091
b.lehner@tudelft.nl

iGEM Team 2015
TU Delft Bionanoscience
Lorentzweg 1
2628 CJ Delft (NL)

Anne S. Meyer
TU Delft Bionanoscience
Lorentzweg 1
2628 CJ Delft (NL)
+31 15-278-9249
a.s.meyer@tudelft.nl

ABSTRACT

The development of techniques that allow environmentally-friendly, large-scale production of materials is critically important for today's economy and society. In this paper we describe the first steps towards 3-dimensional printing of bacterial cultures and their possible application for the production of different materials. To achieve this, a commercial 3D printer was purchased and modified for bacterial systems. Printing temperature, printing speed, spatial resolution, and alginate-based bio-ink chemistry were all adapted. As a proof of principle, our 3-dimensional printing technique was applied to the production of spatially structured, microbially reduced graphene oxide (mrGO) using *Shewanella oneidensis*. Our combination of 3D printing technology with biologically engineered systems enables a sustainable approach for the production of numerous new materials.

CCS Concepts

• Applied Computing → Life and medical science → Systems Biology

Keywords

3D printing, synthetic biology, *Shewanella oneidensis*, materials production, graphene, microbial reduction

1. INTRODUCTION

Tissue (bio-)printing is an emerging technique with multiple applications in medicine and biology, including visualization, education and transplantation. [1][2][3] To date, the prices of these techniques are tremendously high and not yet adapted for bacteria. Current efforts on bacterial printing suffer limitations of poor spatial resolution [4] or require laborious clean-room fabrication of microstructures that shape the printed bacteria [5].

We have developed a "microbial 3D printer" that can deposit bacteria cells in specific three-dimensional patterns using simple devices and chemistries to produce materials. Our focus was to achieve a high spatial resolution as well as to print reproducible samples. As a demonstration, the technique was applied to the production of graphene. This two-dimensional carbon material combines excellent mechanical strength, high flexibility, good optical transparency, high carrier mobility, and maximal surface area. [6] [7] Microbial reduction of graphene oxide (GO) by the bacteria *S. oneidensis* is an extremely promising environmentally-friendly large-scale method of microbially reduced graphene (mrGO) production. [8] [9]

Herein, these material-modifying properties of the bacteria were connected to our new 3D printing technique. The combination of multiple methodologies to print 3D microbial structures with post-processing methods allowed us a high-resolution deposition of bacteria and the fabrication of spatially-patterned materials.

2. MATERIAL AND METHODS

2.1 Printer ink

To obtain 500 µL of bio-ink containing *Escherichia coli*, 1 mL of bacterial cells was spun down at 4000 rpm for 3 minutes and the supernatant discarded. The cells were resuspended in 100 µL of sterile Lysogenic Broth (LB) (Sigma Aldrich), and 400 µL of 1% w/v sodium alginate were added to the solution, followed by vortexing. The printing surface was prepared by the equally distributed application of 100 µL 0,1 M $CaCl_2$.

To obtain 500 µL of bio-ink containing *Shewanella oneidensis*, the same protocol was followed except that Tryptic Soy Broth (TSB) was used instead of LB, and graphene oxide powder was added at 0.005g/mL, 0.01g/mL, or 0.02g/mL.

2.2 Printing system

The extruder and heater of a standard 3D printer (DIY CoLiDo from ReprapWorld) were replaced by a syringe tip and a self-built peristaltic pump (Fig. 1). [11] Silicon tubing (VWR DENE 3100103/25) with an inner diameter of 1 mm and an outer diameter of 3 mm connects the syringe with a continuous stirred ink reservoir, and the pump is used to regulate the speed of bio-ink extrusion. Printed objects were drawn in the free, online CAD program Tinkercad and implemented for printing using CoLiDo Software.

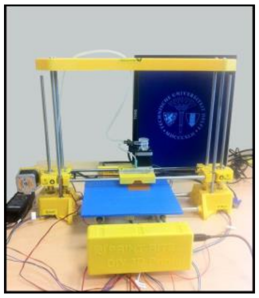


Fig. 1. Modified 3D printer

2.3 Graphene Oxide synthesis

A modified Hummer and Offeman method [8] [9] was used to chemically synthesize GO. In brief, 0,5 g graphite (Pure graphite flakes NGS Trading & Consulting GmbH with an average flake size of 45 µm (Ma -399,5 RG)) was mixed on an ice bath under continuous stirring with 20 mL H_2SO_4 and 5 mL HNO_3. The mixture was stirred for 30 min, then 3 g KMnO4 were added (still on an ice bath). The mixture was stirred again for 30 minutes and incubated on the ice bath for one hour. The sample was heated to 35°C for 3 hours and diluted with 40 mL ultrapure water. The mixture was incubated at 35°C for 2 hours, and then 100 mL ultrapure water was added. Finally, 3 mL H_2O_2 (30%) was slowly added, and the mixture was washed, centrifuged (1500rpm), and sonicated (2h).

2.4 Bacterial strains, media, and imaging

Shewanella oneidensis MR-1 (ATCC® 700550™) was cultured in TSB media, and *E. coli* TOP10 cells constitutively expressing GFP or RFP were aerobically cultured in Luria broth media at 37°C with continuous shaking (250 rpm). Spinning disk confocal microscopy was performed using excitation at 488 and 561 nm to acquire 74 z-stack images over a height of 37.5 µm.

2.5 Analysis of the produced material

A thin structural element was transferred from the 3D printed material onto a MICA sheet (Sigma Aldrich), after multiple washing steps using ultrapure water. The surface topography was studied via AFM (Bruker Multimode AFM with Nanoscope IIIa controller) in tapping mode. Raman Spectrometry (inVia model from Renishaw) was performed using an excitation line of 632,8 nm, provided by a He-Neon laser.

3. RESULTS & DISCUSSION

In order to create a bacterial 3-D printer, the extruder of a standard 3D printer was removed and replaced with a modified print-head, to prevent excessive heating of the bacteria. Addition of a custom-built peristaltic pump allowed for adaptation of the pumping velocity to the low viscosity of our samples. Custom bio-inks were developed in which bacteria were mixed with dissolved alginate. The addition of calcium ions during the printing process triggered rapid cross-linking of the alginate molecules, forming a stable, biocompatible scaffold to support the bacteria. This gel retained its 3D structure on the order of hours to days but was fully removable upon vacuum drying and washing. Initial printing was performed using bio-ink containing fluorescent *E. coli* to demonstrate the spatial resolution of our multi-layer (3-dimensional) bacterial printing.

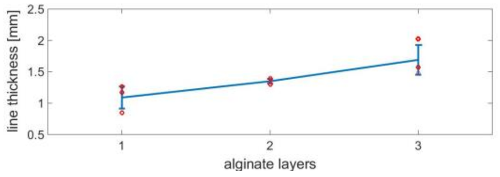


Fig. 2. High-resolution 3D printing of alginate-based bio-ink

Specific automated control of the pump and the print-head was implemented, resulting in the printing of high-resolution lines of bio-ink with a thickness of approximately 1 ± 0,2 mm (Fig. 2). This resolution is limited by the duration of bio-ink gelation and partially uncontrolled diffusion processes. Printing of multiple layers on top of each other resulted in an increase in line thickness of approximately 0,25 ± 0,05 mm per layer (Fig. 2). This minimal spreading can likely be further improved via additional tuning of the bio-ink composition and the printing parameters.

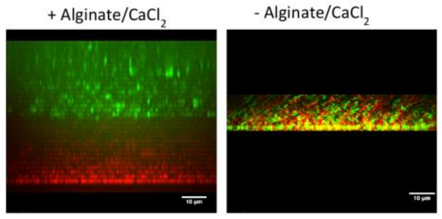


Fig. 3. Bio-ink prevents mixing between bacterial layers

Inspection of the interior structure of stacked layers of printed bio-ink was performed by alternating the printing of fluorescent *E. coli* bacteria of two different wavelengths (green and red). Spinning disk fluorescent confocal microscopy indicated that bacterial mixing was limited to a region with thickness of < 10 µm. In the absence of alginate, the bacteria printed in stacked layers were extensively co-mingled (yellow) (Fig. 3).

Experiments to produce spatially patterned mrGO will be shown at the conference. In short, the printing of bio-ink containing *Shewanella* and graphene oxide allows the microbial reduction of GO to occur within the spatially patterned ink. Our preliminary results using *E. coli* ink demonstrate that we have developed the technology to print bacteria three-dimensionally in a cost-efficient way and in a stable matrix. Connecting these novel bacteria printing techniques with approaches of synthetic biology will further improve its value as a "green" material production process and patterning methodology.

4. ACKNOWLEDGMENTS

Our thanks to Roland Kieffer for helping us to develop the 3D printer. This work was supported by the Netherlands Organization for Scientific Research (NWO/OCW), as part of the Frontier of Nanoscience program.

5. REFERENCES

[1] Murphy, S.V and Atala, A., 2014 *3D bioprinting of tissues and organs*. Nature Biotechnology 2014, doi:10.1038/nbt.2958

[2] Pati, F., Ha, D.H., Jang. J., Ha, H.H., Rhie, J.W. and Cho, D.W. 2015, *Biomimetic 3D tissue printing for soft tissue regeneration*. Biomaterials 62 (2015) http:// dx.doi.org/10.1016/j.biomaterials.2015.05.043.

[3] Pati, F., Jang. J., Ha, D.H., Kim, S.W., Rhie, J.W., Shim, J.H., Kim, D.H., and Cho, D.W. 2014, *Printing three-dimensional tissue analogues with decellularized extracellular matrix bioink*. Nature Communications 2014, DOI: 10.1038/ncomms4935

[4] Connell, J.L., Ritschdorff, E.T., Whiteley, M., and Shear, J.B. 2013. 3D printing of microscopic bacterial communities. PNAS vol.110 no. 46, doi: 10.1073/pnas.1309729110

[5] Dosier, G.K., 2011. Methods for Making Construction Material Using Enzyme Producing Bacteria US 20110262640 A1, http://www.google.com/patents/US20110262640

[6] Palmero, V., Kinloch, I. A., Ligi, S. and Pugno, N.M. 2016. *Nanoscale Mechanics of Graphene and Graphene Oxide in Composites: A Scientific and Technological Perspective*. Advanced Materials 2016, DOI: 10.1002/adma 201505469 http://onlinelibrary.wiley.com/doi/10.1002/adma.201505469/abstract

[7] Dreyer, R.D., Park, S., Bielawski, C.W. and Ruoff, R.S. 2009 *The chemistry of graphene oxide*. Chem. Soc. Rev., 2010,39, 228-240, DOI: 10.1039/B917103G

[8] Liu, G., Zhang, X., Zhou J., Wang, A., Wang, J., Jin, R., Lv, H. 2013. *Quinone-mediated microbial synthesis of reduced graphene oxide with peroxidase-like activity*. Bioresource Technology 149 (2013) 503-508

[9] Wang, G., Qian, F., Saltikov, C. W., Jiao Y. and Li, Y. 2011. *Microbial Reduction of Graphene Oxide by Shewanella*. Nano Res. 2011, 563-570, DOI 10.1007/s12274-011-0112-2

[10] Faley, S., Baer, B., Richardson, M., Larsen, T., and Bellan, L.M., 2015, DIY peristaltic pump http://blogs.rsc.org/chipsandtips/2015/06/26/diy-peristaltic-pump

MakerFluidics: Microfluidics for the Masses

Ryan Silva
Department of Electrical and Computer Engineering
Boston University
rjsilva@bu.edu

Radhakrishna Sanka
Department of Electrical and Computer Engineering
Boston University
sanka@bu.edu

Douglas Densmore
Department of Electrical and Computer Engineering
Boston University
dougd@bu.edu

ABSTRACT

MakerFluidics is a microfluidic fabrication and control paradigm that operates within a set of constraints guided by the ideals of automation and the modern "maker movement". This paradigm integrates into the larger microfluidic design flow shown in Figure 1, but it can also be employed on its own. MakerFluidics accepts physical and temporal valve control requirements and a microfluidic device layout as inputs and generates all of the necessary control software, manufacturing files and assembly information required to build a self-contained microfluidic device and control infrastructure.

1. INTRODUCTION

A significant aim of the modern maker movement is to make technology and technological "know-how" accessible to the masses. One way to accomplish this goal is to devise solutions using resources that are flexible and ubiquitous[5]. A significant criticism often levied against the field of microfluidics is its high barrier to entry often as a result of the need for highly specialized fabrication and control equipment as well as expertise that is typically found only in labs dedicated to microfabrication[7]. This work seeks to create a design-to-device, automated microfluidic work-flow constrained by the ideals espoused in maker culture.

2. CONSTRAINTS

An important goal of the maker movement is to increase technology's accessibility. This ideal is levied on MakerFluidics in the form of a series of constraints, the first of which is that all fabrication equipment must be sourced through ubiquitous consumer and retail product outlets such as Amazon.com in the Unites States, or Amazon.ca, .co.uk, .jp, etc. internationally, and each individual piece of equipment required for fabrication and control must cost less than \$100. A desktop CNC mill and 3D printer are excepted from this constraint on the basis that they are common maker-space tools with a wide variety of uses extending well beyond the field of microfluidics; the cost of the CNC mill (Othermill, Othermachine Co.) and 3D printer (Ultimaker 2, Ultimaker B.V.) used in our lab are each less than \$2,500. Additionally, all elements of the complete software tool chain must be free and/or open-source.

To further facilitate microfluidic accessibility, all fabrication and control protocols must be accomplished without specialized infrastructure beyond a wall electrical outlet. This excludes fume hoods, clean room facilities, tank storage, vacuum lines and corona/plasma bonders. The process for designing, fabricating and controlling programmable (i.e., valved) microfluidics within the specified constraints is explored. Each stage of the process includes a comparison of current methods to methods developed or adopted by the MakerFluidics framework.

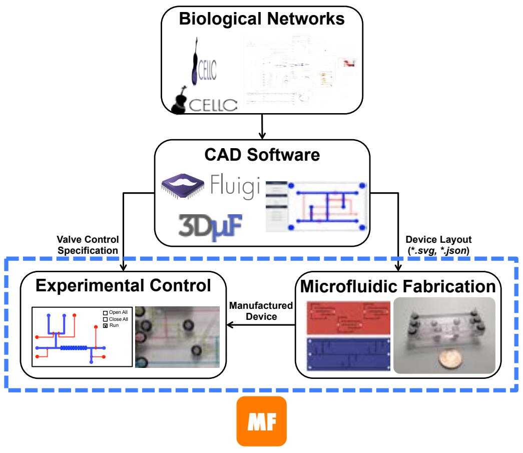


Figure 1: MakerFluidics is part of a larger microfluidic design flow that spans from biological specification to microfluidic fabrication and control

3. MICROFLUIDIC FABRICATION

The fabrication of a microfluidic device typically has two main steps: pattern geometries and seal layers[4].

Channel and valve geometries are etched in thermoplastic polymers using a desktop CNC mill. This stands in sharp contrast from conventional methods of microfluidic fabrication, namely photolithography and wet etching, which require the use of clean room facilities and highly specialized equipment. The CNC approach is well-suited for integrating valving technologies such as monolithic membrane valves [1] (Figure 2) and centrifugal capillary valves [3]. A significant trade-off for the relative ease of CNC milling thermoplastics using a desktop (i.e., not industrial-grade) CNC mill is that the maximum resolution is 25μm with an exponential increase in reliability seen at feature sizes greater than 250μm[2], whereas microfluidic geometries using conventional methods such as photolithography can reliably achieve features smaller than 1μm[4].

Once geometries are etched into the desired substrate, sealing these channels becomes the next challenge. Polydimethylsiloxane (PDMS) is a common material for fabri-

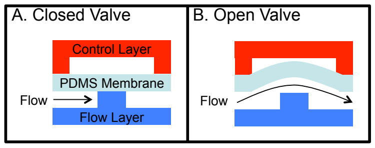


Figure 2: Normally-closed monolithic membrane valves [1] are realized by introducing discontinuities in the flow layer (blue) and a corresponding pneumatic cavity in the control layer (red). These two layers are separated by a PDMS membrane. To open the valve a vacuum is introduced into the cavity in the control layer.

cating microfluidics[4] and is also commonly used to encapsulate solar panels and outdoor lighting. It is because of the latter property that PDMS (Sylgard 184, Dow Corning) is available through retail outlets, such as Amazon, and, thus, falls within the constraints for adoption by the MakerFluidics fabrication paradigm. PDMS can be sealed irreversibly through modifications to its surface chemistry via plasma or corona exposure or sealed reversibly simply using the material's inherent Van der Waals attraction to various materials including itself, glass and thermoplastics[4]. Since irreversible sealing through surface treatments involves specialized machinery, MakerFluidics employs the latter method via Van der Waals force. The trade-off being that the reversible seal cannot withstand pressures greater than 5psi[4]. Using these techniques we have fabricated a number of continuous flow devices inluding the device shown in Figure 1 as well as centrifugal microfluidic systems.

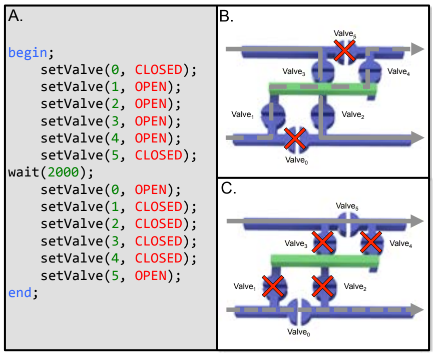
4. EXPERIMENTAL CONTROL

MakerFluidics views experimental conditions as a sequence of temporally-specified valve conditions. This necessitates a data structure consisting of an enumerated array of valve objects and a sequence of temporal specifications regarding their state. This data structure is automatically generated by microfluidic CAD software developed in CIDAR Lab, but it can also be created manually as a series of wait statements and valve conditions shown in Figure 3. These valve objects are linked to the microfluidic layout provided to the fabrication software through the use of a JSON object for display in a graphical user interface (GUI).

Pneumatics are provided through an array of 3D printed syringe pumps controlled by custom firmware on an Arduino microcontroller. This microcontroller receives serial commands derived from the GUI running on a host computer. The MakerFluidics control GUI and firmware is extensible and interoperable with a conventional, solenoid-driven control infrastructure.

5. CONCLUSIONS

The field of microfluidics is a promising area for expanding the boundaries of bio-design automation. Unfortunately this field carries a high barrier to entry in terms of expertise and equipment. The field of physical manufacturing and computing once had similarly high barriers; the cost of computer-controlled precision machinery could only be borne by industry. Time brought down the cost of the machinery but that alone was not enough to bring physical manufacturing to the consumer. It was only through



```
begin;
    setValve(0, CLOSED);
    setValve(1, OPEN);
    setValve(2, OPEN);
    setValve(3, OPEN);
    setValve(4, OPEN);
    setValve(5, CLOSED);
wait(2000);
    setValve(0, OPEN);
    setValve(1, CLOSED);
    setValve(2, CLOSED);
    setValve(3, CLOSED);
    setValve(4, CLOSED);
    setValve(5, OPEN);
end;
```

Figure 3: The temporal specification (A)[6] for a device containing six valves (B, C). The specification in (A) dictates the set of conditions in (B) to begin the assay. After 2000ms the valves change state to that shown in (C), thus affecting the movement of fluid through the device.

low-cost techniques paired with an enthusiastic community that created a market for accessible manufacturing techniques, which ultimately led to the rise of consumer-grade 3D printers and CNC mills in addition to low-cost, ubiquitous computing platforms such as microcontrollers (Arduino, MSP430, etc.) and single-board computers (Raspberry Pi, BeagleBoard, etc.). This work aims to lower the barrier to entry into microfluidics by developing microfluidic solutions able to be employed by the masses. This is accomplished by viewing accessibility as a constraint to development. It is our hope that with enough eyes, microfluidics can open up new avenues for automation within the bio-design community.

6. REFERENCES

[1] W. H. Grover, A. M. Skelley, C. N. Liu, E. T. Lagally, and R. A. Mathies. Monolithic membrane valves and diaphragm pumps for practical large-scale integration into glass microfluidic devices. *Sensors and Actuators B: Chemical*, 89(3):315–323, 2003.

[2] 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.

[3] M. J. Madou and G. J. Kellogg. Labcd: a centrifuge-based microfluidic platform for diagnostics. In *BiOS'98 International Biomedical Optics Symposium*, pages 80–93. International Society for Optics and Photonics, 1998.

[4] J. C. McDonald and G. M. Whitesides. Poly (dimethylsiloxane) as a material for fabricating microfluidic devices. *Accounts of chemical research*, 35(7):491–499, 2002.

[5] A. R. Schrock. Education in disguise: Culture of a hacker and maker space. *InterActions: UCLA Journal of Education and Information Studies*, 10(1), 2014.

[6] W. Thies, J. P. Urbanski, T. Thorsen, and S. Amarasinghe. Abstraction layers for scalable microfluidic biocomputing. *Natural Computing*, 7(2):255–275, 2008.

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

Bacterial detoxification of Martian soil coupled to oxygen production

The 2016 iGEM team of Leiden University
Sylviusweg 72
2333 BE Leiden, the Netherlands
igem@science.leidenuniv.nl

ABSTRACT

Perchlorate (ClO_4^-) contamination of groundwater, surface water and food supplies is a widely spread hazard for the environment and our health on earth. However, it also poses a challenge for our future aims to colonize the planet Mars. Martian soil contains 0.5 - 1% perchlorate, which therefore needs to be remediated in order to cultivate edible crops. Using synthetic biology approaches, the Leiden iGEM team will equip the bacterium *Escherichia coli* with the tools to convert the toxic compound perchlorate into chloride and oxygen. To this end, we will introduce a set of codon-optimized genes from *Dechloromonas aromatica*, encoding for the perchlorate reductase complex, which reduces perchlorate to chlorite (ClO_2^-), and chlorite dismutase that then splits chlorite to chloride (Cl^-) and oxygen. By designing this system as re-usable modules, called BioBricks, we pave the way for many future applications. Altogether, our system is widely applicable to remove perchlorate from contaminated soils on earth, while also being useful for future Mars expeditions.

Keywords

iGEM; perchlorate; soil remediation, Mars; Martian soil; *Escherichia coli*; *Dechloromonas aromatica*; perchlorate reductase; chlorite dismutase; simulated partial gravity; microgravity; Random Positioning Machine; BioBrick.

1. INTRODUCTION

Perchlorate is a highly soluble anion (ClO_4^-) that is toxic for mammals, including humans. The compound interferes with the iodine uptake in the thyroid gland, thereby affecting our metabolism as well as physical and mental development [1].

Most of the perchlorate present on earth is synthetically produced for various industrial applications, and is present in fireworks, in rocket fuel and in oils [5]. Also nitrogen fertilizers contribute to the perchlorate contamination of soil, resulting in a large-scale investigation at request of the European Commission last year. The biggest problem lies in the high solubility of perchlorate, making that this compound easily leaches. Therefore it accumulates into the groundwater, thus contaminating our drinking water sources[6][7]. Notably, recent studies indicated that Martian soil contains 0.5 - 1% perchlorate, which potentially poses a challenge in face of the intended missions to this planet [8].

Apart from the toxic effects of touching or ingesting perchlorate directly, plants grown on perchlorate-contaminated soils accumulate this toxic compound, making them inedible [6].

One potential method to decontaminate perchlorate from soil is by using bacteria that can convert perchlorate by reduction. Due to the high reduction potential of perchlorate (ClO_4^-/Cl^- $E^o = 1.287$ V), some bacteria can use this compound as an electron acceptor. Such bacteria typically utilize the enzymes perchlorate reductase and chlorite dismutase to convert perchlorate into chloride and oxygen [4]. In this project, we will apply a synthetic biology approach to equip the model bacterium *Escherichia coli* with the necessary tools to convert perchlorate. Not only will our design be useful for soil remediation on earth, but also for future expeditions to Mars.

2. OUR PROJECT

2.1 The perchlorate reduction system from *Dechloromonas aromatica*

Perchlorate reducing bacteria are found in different environments, such as pristine and hydrocarbon-contaminated soils, aquatic sediments, paper mill waste sludges, and farm animal waste lagoons. Most species belong to the genera *Azospira* and *Dechloromonas*[4]. The genes encoding for the perchlorate reducing system are contained on a conserved gene cluster [12], which will be transferred to *E. coli* as four modular BioBricks (Fig. 1)..

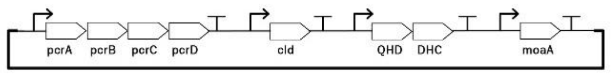


Figure 1. Schematic overview of the conserved gene cluster used, encoding the perchlorate reducing system. Codon-optimized genes will be transferred to *E. coli* as four modular BioBricks, each expressed from its own promoter (arrows). Terminators are indicated as a T.

The actual conversion of perchlorate to chlorite is carried out by the perchlorate reductase complex[8], which contains the PcrA, PcrB, and PcrC subunits. PcrD, which is also essential for perchlorate reduction, is a chaperone protein required for the assembly of the PcrAB complex prior to translocation via the Tat secretion pathway. The perchlorate reductase complex resides in the periplasm[2] and is tethered to the inner membrane via components of the electron transport chain (QDH, DHC; Fig. 2) [12]. The product of the perchlorate reductase complex is chlorite (ClO_2^-), which is further degraded by the enzyme chlorite dismutase (Cld). This periplasmic homotetramer converts chlorite into chloride (Cl^-) and a molecule of oxygen [14]. Perchlorate reductase and chlorite dismutase only properly function in anaerobic or micro-aerobic environments [3].

Codon-optimized versions of the genes encoding the proteins necessary for the reduction of perchlorate are currently being synthesized and will subsequently be introduced in *E. coli* to provide this model chassis with perchlorate reducing capability.

2.2 Characterization of *E. coli* under Martian gravity conditions

Previous studies have indicated that *E. coli* seems to grow faster in space than on Earth [10]. To further explore these differences, we will characterize the physiological response of *E. coli* cells grown in an environment with a simulated gravity of Mars. For this, we will use a so-called Random Positioning Machine. Using RNA sequencing and validation by qPCR, we will identify the genes that are differentially expressed at reduced gravity. In addition to a better understanding of the physiological response of the cells, this approach will also enable us to identify promoters that can be used for the expression of our BioBricks. This will ensure that our system is expressed under conditions of reduced gravity.

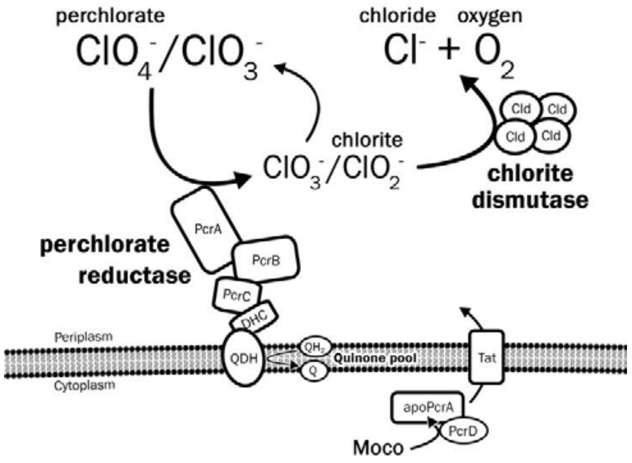


Figure 2. Schematic representation of the perchlorate reductase, which converts perchlorate to chlorite, and which is subsequently being converted into chloride and oxygen by chlorite dismutase. Mature PcrA contains a Molybdopterin cofactor (Moco). The insertion of this cofactor into apoPcrA is facilitated by PcrD, prior to secretion via the Tat translocation pathway (Based on Bender 2005 and Melnyk 2013).

2.3 Modelling the metabolic fluxes and bioreactor functions

The effects on the growth of *E. coli* upon introducing the perchlorate reductase will be modelled *in silico*. We will employ Elementary Flux Mode analysis based on stoichiometric data such as the *E. coli* core model by Orth, Fleming and Palsson [13]. This should help us to optimize *E. coli* for perchlorate reduction in an otherwise opaque metabolism. We will also model how the biological detoxification by our design would function in a contained environment such as a bioreactor. The use of a bioreactor would prevent release of the genetically modified organism into the environment and creates the needed micro-aerobic environment.

3. THE LEIDEN iGEM-TEAM

Our team consist of thirteen highly motivated bachelor and master students (Fig. 3), with backgrounds ranging from biology to physics and mathematics. By participating in the 2016 international Genetically Engineered Machine competition (iGEM), we would like to contribute to the community of synthetic biology and hope to meet many inspiring people. The results of the project will be presented during the Giant Jamboree in Boston (October 2016).

Figure 3. Team picture of the enthusiastic Leiden iGEM team.

4. REFERENCES

[1] Bardiya, N, *et al.,* 2011. Dissimilatory perchlorate reduction: A review. *Microbiological Research,* **166** (January 2011), 237–254.

[2] Bender, K. S. *et al.* 2005. Identification, Characterization, and Classification of Genes Encoding Perchlorate Reductase, *J Bacteriol.* **187** (April 2005), 5090–5096.

[3] Chaudhuri, S. K., *et al*. 2002. Environmental factors that control microbial perchlorate reduction. *Appl. Environ. Microbiol.* **68** (September 2002), 4425–4430.

[4] Coates, J. D. *et al*. 1999. The ubiquity and diversity of dissimilatory (per)chlorate-reducing bacteria. *Appl. Environ. Microbiol.* **65** (December 1999), 5234–5241.

[5] Collette, T. W. *et al*. 2003. Analysis of hydroponic fertilizer matrixes for perchlorate: comparison of analytical techniques. *Analyst* **128** (August 2003), 88–97.

[6] EPA 2014. *Technical Fact Sheet – Perchlorate*, Environmental Protection Agency, US.

[7] EFSA Panel on Contaminants in the Food Chain, 2014. Scientific Opinion on the risks to public health related to the presence of perchlorate in food, in particular fruits and vegetables, *EFSA Journal* **12** (September 2014), 3869.

[8] Hecht M. H. *et al.* 2009. Detection of Perchlorate and the Soluble Chemistry of Martian Soil at the Phoenix Lander Site, *Science* **325** (July 2009), 5936.

[9] Kengen, S. W. M. *et al.* 1999. Purification and characteriza-tion of (per)chlorate reductase from the chlorate-respiring strain GR-1. *J. Bacteriol.* **181** (November 1999), 6706–6711.

[10] Kotakonda A., *et al.* 2013. Effect of Simulated Microgravity on E. coli K12 MG1655 Growth and Gene Expression, *Plos One* **8** (March 2013), e57860.

[11] Melnyk R.A. *et al.* 2014. Transposon and deletion mutagenesis of genes involved in perchlorate reduction in Azospira suillum PS. *MBio* **5** (January 2014), e00769-13.

[12] Melnyk, R. A. *et al.* 2011. Identification of a Perchlorate Reduction Genomic Island with Novel Regulatory and Metabolic Genes, *Appl Environ Microbiol* **77** (October 2011), 7401–7404.

[13] Orth, J.D. *et al.* 2010. Reconstruction and Use of Microbial Metabolic Networks: the Core Escherichia coli Metabolic Model as an Educational Guide. *EcoSal Plus* **10** (September 2010), 1128.

[14] Van Ginkel, C. G *et al.* 1996. Purification and characterization of a chlorite dismutase: a novel oxygen generating enzyme. *Arch. Microbiol*. **166** (1996), 321–32.

Business Process Management of Synthetic Biology Workflows

Dr Christopher Reynolds[1]
Jocelyne Holdbrook[1]
Imperial College London
RSM 4.01
South Kensington Campus
London SW7 2AZ
+44 (0)7903127870
cr308@imperial.ac.uk

Kealan Exley
Imperial College London
k.exley11@imperial.ac.uk

Professor Richard Kitney
Imperial College London
RSM 3.16
South Kensington Campus
London SW7 2AZ
+44 (0)20 7594 6226
r.kitney@imperial.ac.uk

ABSTRACT

In this paper, we describe methods for using Business Process Management systems to improve the efficiency of Synthetic Biology workflows. The systems of Business Process Management and its associated notation are detailed, and their implementation into a synthetic biology workflow are described. A system is discussed whereby Business Process Management software is integrated with other software packages and laboratory robotics in order to standardize and increase the efficiency of laboratory workflows.

Keywords

Business Process Management; Business Process Model Notation; Databases; Enterprise computing; Synthetic Biology.

1. INTRODUCTION

Synthetic biology is intended to bring engineering principles to the discipline of genetic engineering. Key engineering principles are the use of standardized parts and workflow processes, both of which should be consistent and replicable. Modelling the processes requires a method of specifying tasks and corresponding notation, a defined set of user interfaces, and the ability to integrate with existing standards and technologies. The methods should be constructed to maximize workflow efficiency, data collection and provide opportunities for the automation of tasks.

This project aims to develop an open-source Business Process Management (BPM) system aimed at the needs of synthetic biologists. Business Process Management software is used to model workflow processes, which can then be uploaded to a server which will generate a workflow and accompanying data model that records task timings, allows data entry and facilitates the integration of processes into external applications. End users of the workflow are able to access the data remotely through a web interface.

2. BUSINESS PROCESS MANAGEMENT

2.1 Business Process Model and Notation

Business Process Model and Notation (BPMN) is a widely accepted format developed for specifying processes in graphical form [1]. BPMN is the core of BPM and provides a standard form of communication. It is currently the internationally accepted standard for process modelling [2]. As industrialization is an end goal of synthetic biology [3], the use of industry-standard tools such as BPM will increase synergy between industries and academia.

The benefits of using BPM include:

- Ease of constructing and altering processes with minimal programming.
- Ability to define simple user interfaces.
- Visibility of all sub-processes and individual tasks that form a process.
- Reproducibility of processes due to design of processes in standard BPM notation.
- Standardised representation of protocols.

2.2 Modelling a Process

Figure 1 shows a Synthetic Biology workflow process modelled with BPMN. The process is divided into two lanes, one assigned to a user and one for a robot, where the task triggers an automated process. Tasks in each lane are to be completed by their assignee, and as the process of automation continues, tasks can flow from the user to the robot lane.

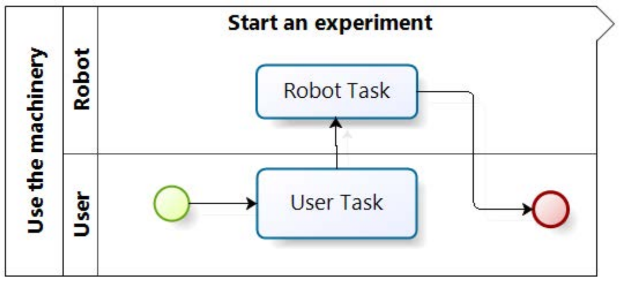


Figure 1. An example of a synthetic biology workflow process in BPMN modelled in Bizagi Process Modeler [2]. The circles symbolize the start (green circle) and end (red circle) of the task. The blue rectangles symbolize tasks. The workflow is divided into two lanes, for user tasks (bottom lane) and robot tasks (top lane).

[1] These authors should be treated as joint first authors.

2.3 Existing Business Process Management Tools

Other Business Process Management Suites are offered by major companies such as Oracle, IBM and Pega. Bizagi is being used as a basis for this project because of its data model is contained in an accessible SQL Server backend and its free status for small-scale enterprises.

3. INTEGRATING TECHNOLOGIES

3.1 Integration of Software

Business Process Management (BPM) encompasses a field of techniques used to optimize workflows. In this study, BPM methods are used to model synthetic biology processes to standardize experiments, increase the ease of automation and allow analysis of workflow data. This study demonstrates an implementation of BPM for synthetic biology, with the data structure stored in a SQL Server Database, and custom ASP.NET websites used for additional data handling and analyzing.

As part of this integration, the workflow processes were first modelled using industry-standard BPMN. The processes were then automated and combined to produce an application that could be accessed by an end user via a web-browser. Tasks in the process can be undertaken by an experimentalist or interface with other technologies.

Figure 2 shows the modelling of the data structure within Bizagi. These are translated these into SQL tables in SQL Server Express, which can be accessed with Object Linking and Embedding through ASP.NET, allowing users to complete the workflows either through the Bizagi interface or a custom web interface.

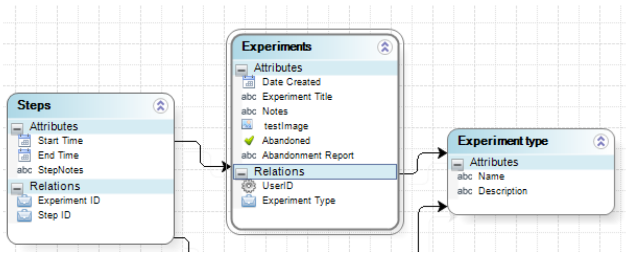


Figure 2. Data modelling in Bizagi Studio. Data is modelled as a series of entities that relate to each other.

Figure 3 shows the custom ASP.NET interface designed for laboratory technicians to keep track of their workflow. Each step of a workflow can be associated with an image and instructions, along with a timer indicating a minimum and maximum time that each step should take. As the user steps through the process, the time taken for each step is recorded to the database, allowing future process analysis. The user can also record notes at each step, and customized input fields can be added for each step.

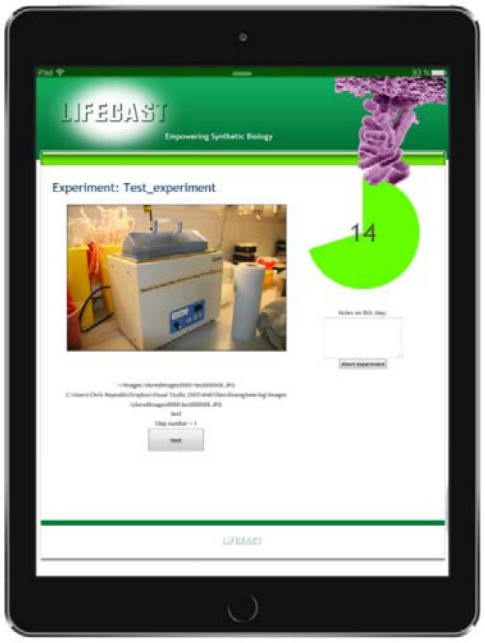


Figure 3. Mockup of the enterprise system running on a mobile device.

3.2 Integration with Robotics

A language which translates protocols into instructions for a variety of liquid handling platforms can be incorporated into the application. This would enable the automated implementation of a single synthetic biology protocol across research centers with different types of liquid-handling robots and instrumentation. Software such as the BioCAD suite [4] and the stack for robotic execution of protocols developed by Vasilev et al [5] would allow process in the workflow to be viewed, analyzed and controlled remotely via one common web application. With the exception of a few unavoidable manual tasks (such as the preparation of samples), this would lead to a total automation of the workflow.

4. REFERENCES

[1] Object Management Group, Business Process Model and Notation, http://www.bpmn.org

[2] Bizagi, http://www.bizagi.com

[3] Kitney, R. and Freemont, P. (2012) Synthetic Biology – the State of Play, *FEBS Letters* 586, 2029-2036.

[4] Schreck, B. Babb, J., Weiss, R. (2014) Better Scheduling Software and User Interface for Liquid-Handling Robots, *IWBDA*, 36-37.

[5] Vasilev, V., Liu, C., Haddock, T., Bhatia, S., Adler, A., Yaman, F., Beal, J., Babb, J., Weiss, R. and Densmore, D. (2011) A software stack for specification and robotic execution of protocols for synthetic biological engineering. *International workshop on Bio-Design Automation.*

CVLTool: Oligonucleotide Design Software for assembly of Orthogonal Codon Variant Gene Libraries

Surya Teja Chinta
Computer Science Department
The College of New Jersey
Ewing, NJ 08628, USA
+1 609 771 2268
chintas1@tcnj.edu

Dimitris Papamichail
Department of Computer Science
The College of New Jersey
Ewing, NJ 08628, USA
+1 609 771 2268
papamicd@tcnj.edu

ABSTRACT

Advances in large-scale de novo DNA synthesis technologies provide access to thousands of custom-designed sequence variants at low cost. Building on our algorithms for the design of diverse coding sequence libraries, we have developed a web-based tool to generate orthogonal codon variant genes. It is our hope that this tool will reduce the cost of gene library synthesis and enable research into the precise effects of codon utilization in gene expression in a variety of organisms.

Categories and Subject Descriptors

D.2.2 [**Algorithms**]: Nonnumerical Algorithms and Problems – *computations on discrete structures*

General Terms

Algorithms, Experimentation.

Keywords

Gene design, genomic libraries, synthetic biology.

1. INTRODUCTION

Gene synthesis is a process during which oligonucleotides are combined into larger DNA sequences, ranging from hundreds to hundreds of thousands of nucleotides in length. Techniques such as the *in-vitro isothermal assembly* [2] have been used to assemble sequences of varying lengths, including mega-base genomes [1]. In [5] the same technique was used to create a combinatorial library of biochemical pathways, containing 144 combinations of 3 promoters and 4 gene variants. This feat demonstrated the effective use of assembly methods to accurately construct combinatorial libraries and, more importantly, rationally designed sets.

Traditionally, due to the complexity of designing genomic sequences with well-controlled attributes and the gap of knowledge on the effect of these attributes, large-scale gene design experiments have relied on random synonymous mutations to generate the gene libraries. Welsh et al. [8] synthesized 72 variants and chimeric combinations of genes encoding commercially valuable proteins and showed that variation in expression is highly correlated to codon usage in *Escherichia coli*, pinpointing 5-6 codons as most critical for expression. In constrast, a paper from the Plotkin lab [3] claimed that most of the variance in expression results from the amount of secondary structure in the 5' end of the gene, after testing 154 variants of the GFP protein carrying random synonymous mutations. Further analysis of Plotkin's dataset by Supec and Mac [7] identified 5 specific codons from 4 amino acids to contribute almost all of the variation in expression levels attributable to codon usage. These codons were different than the ones identified by Welsh *et al*. Additional findings from the Plotkin lab [4] indicate complex relationships between codon selection, translation initiation and elongation, misfolding of proteins and autocorrelation.

DOI: http://dx.doi.org/10.1145/12345.67890

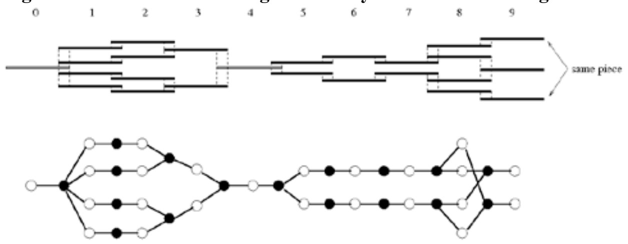
Currently it is not clear, even in well-studied model organisms, which rules one should follow to design genes for optimized gene expression. A common belief, emphasized by Plotkin's group in [3] and by Super and Mac in [7], is that the mechanisms which determine gene expression can be established only by further large-scale experimentation. Toward that goal we have developed a web-based tool that enables the design of synthetic protein coding gene libraries to address the needs for large-scale *in-vitro* experimentation, while minimizing the cost of synthesis.

2. METHODS AND RESULTS

In our previous work [6] we investigated and solved the problem of minimizing the number of genomic fragments needed to synthesize a library of gene variants, all coding for the same amino acid sequence, but each having a unique codon distribution. For testing 4 different frequency levels (such as low, medium low, medium high and high) of the usage of a codon, 4 designs would be needed, each utilizing the codon at one of the 4 levels. For examining the effects of 5 different codons at 4 levels each, one would need to synthesize 1024 (4^5) different genes, to account for all possible combinations. The number of individual gene variants increases exponentially with the number of codons that we wish to investigate, as does the cost of synthesizing all the different genes.

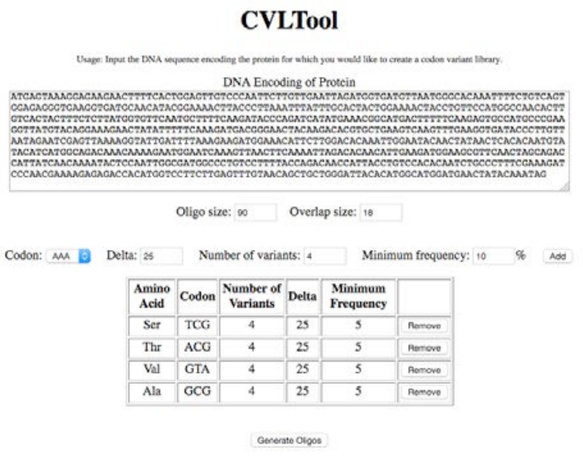
Our algorithm minimizes the number of sequence fragments required to construct gene variants, by sharing common fragments among variants. Let us examine a gene that can be assembled using 10 overlapping segments. We define as 1x coverage the total worth of sequence required to synthesize one gene variant. For each codon whose frequency we intend to alter, our algorithm examines consecutive DNA fragments to identify groups (intervals) containing enough corresponding amino acids to allow a 'step' between the frequency levels we wish to achieve. Assuming that we would like to create constructs that vary the usage of a particular codon according to the frequencies (.05, .30, .55, .80), we would identify consecutive fragments containing at least 25% (the step) of the corresponding amino acid's occurrences in the gene.

Figure 1: Combinatorial design of library that varies the usage of two



codons, with 4 frequency values per codon, 16 gene variants. All neighboring oligos share overlaps. The graph representation uses white nodes for internal oligo regions and black nodes for overlaps. Edges connect nodes (regions) that are part of the same oligo.

Then the algorithm identifies disjoint groups of consecutive fragments which, when combined, can produce the desired constructs, each with a unique frequency of the codon under consideration. An example varying the occurrences of two codons is shown in Fig. 1. Instead of ordering 160 fragments to assemble the 16 desired gene variants, 24 segments suffice, with 24 being also the minimum number to realize this library design.



CVLTool

Usage: Input the DNA sequence encoding the protein for which you would like to create a codon variant library.

DNA Encoding of Protein

```
ATGAGTAAAGGAGAAGAACTTTTCACTGGAGTTGTCCCAATTCTTGTTGAATTAGATGGTGATGTTAATGGGCACAAATTTTCTGTCAGT
GGAGAGGGTGAAGGTGATGCAACATACGGAAAACTTACCCTTAAATTTATTTGCACTACTGGAAAACTACCTGTTCCATGGCCAACACTT
GTCACTACTTTCTCTTATGGTGTTCAATGCTTTTCAAGATACCCAGATCATATGAAACGGCATGACTTTTTCAAGAGTGCCATGCCCGAA
GGTTATGTACAGGAAAGAACTATATTTTTCAAAGATGACGGGAACTACAAGACACGTGCTGAAGTCAAGTTTGAAGGTGATACCCTTGTT
AATAGAATCGAGTTAAAAGGTATTGATTTTAAAGAAGATGGAAACATTCTTGGACACAAATTGGAATACAACTATAACTCACACAATGTA
TACATCATGGCAGACAAACAAAAGAATGGAATCAAAGTTAACTTCAAAATTAGACACAACATTGAAGATGGAAGCGTTCAACTAGCAGAC
CATTATCAACAAAATACTCCAATTGGCGATGGCCCTGTCCTTTTACCAGACAACCATTACCTGTCCACACAATCTGCCCTTTCGAAAGAT
CCCAACGAAAAGAGAGACCACATGGTCCTTCTTGAGTTTGTAACAGCTGCTGGGATTACACATGGCATGGATGAACTATACAAATAG
```

Oligo size: 90 Overlap size: 18

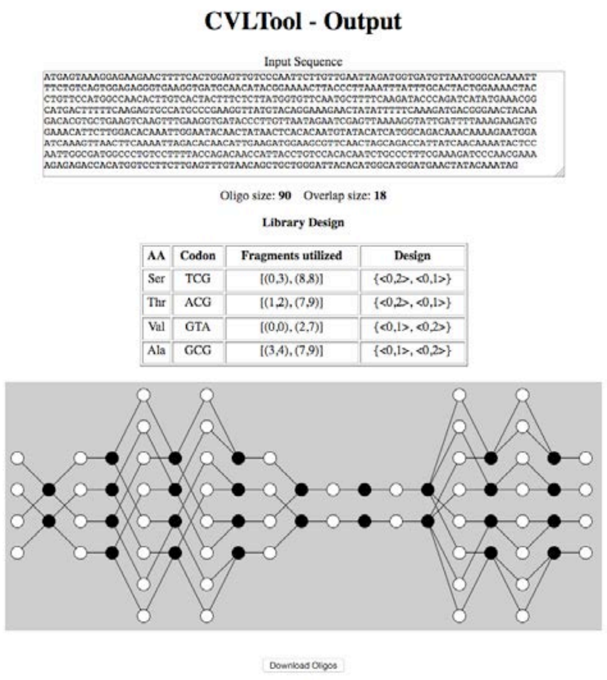
Codon: AAA Delta: 25 Number of variants: 4 Minimum frequency: 10 %

Amino Acid	Codon	Number of Variants	Delta	Minimum Frequency	
Ser	TCG	4	25	5	Remove
Thr	ACG	4	25	5	Remove
Val	GTA	4	25	5	Remove
Ala	GCG	4	25	5	Remove

Figure 2: Input interface of CVLTool. In this example we input the DNA sequence encoding the GFP protein, and we request a design that varies 4 codons at 4 frequency values each. The "Delta" parameter signifies the difference between varying codon frequencies.

We have now developed a gene library design software, the *Codon Variant Library Tool* (CVLTool), whose input and output screen are shown in Fig. 2 and 3 respectively. The example shown involves the design of a GFP library varying 4 codons from amino acids (S, T, V, A) at 4 frequency levels (.05, .30, .55, .80). The GFP protein has a length of 238 amino acids, and with an oligo size of 90bp and overlap length of 18bp we need 10 overlapping fragments to cover the whole coding sequence. The resulting library design has 7.2x coverage, and demands ordering only 2.8% of the amount of sequence the 256 separate genes would require. The CVLTool displays information about the library design, and the corresponding graph demonstrating the assembly of the oligos. While the number of occurrences of codons not involved in the design remains globally unaltered, a number of synonymous codons are swapped in and between certain oligos to avoid unintended overlaps. Once the library design is realized, the oligo DNA sequences can be downloaded in text format.

The CVLTool will be accessible at http://algo.tcnj.edu/cvltool.



CVLTool - Output

Input Sequence

```
ATGAGTAAAGGAGAAGAACTTTTCACTGGAGTTGTCCCAATTCTTGTTGAATTAGATGGTGATGTTAATGGGCACAAATT
TTCTGTCAGTGGAGAGGGTGAAGGTGATGCAACATACGGAAAACTTACCCTTAAATTTATTTGCACTACTGGAAAACTAC
CTGTTCCATGGCCAACACTTGTCACTACTTTCTCTTATGGTGTTCAATGCTTTTCAAGATACCCAGATCATATGAAACGG
CATGACTTTTTCAAGAGTGCCATGCCCGAAGGTTATGTACAGGAAAGAACTATATTTTTCAAAGATGACGGGAACTACAA
GACACGTGCTGAAGTCAAGTTTGAAGGTGATACCCTTGTTAATAGAATCGAGTTAAAAGGTATTGATTTTAAAGAAGATG
GAAACATTCTTGGACACAAATTGGAATACAACTATAACTCACACAATGTATACATCATGGCAGACAAACAAAAGAATGGA
ATCAAAGTTAACTTCAAAATTAGACACAACATTGAAGATGGAAGCGTTCAACTAGCAGACCATTATCAACAAAATACTCC
AATTGGCGATGGCCCTGTCCTTTTACCAGACAACCATTACCTGTCCACACAATCTGCCCTTTCGAAAGATCCCAACGAAA
AGAGAGACCACATGGTCCTTCTTGAGTTTGTAACAGCTGCTGGGATTACACATGGCATGGATGAACTATACAAATAG
```

Oligo size: **90** Overlap size: **18**

Library Design

AA	Codon	Fragments utilized	Design
Ser	TCG	[(0,3), (8,8)]	{<0,2>, <0,1>}
Thr	ACG	[(1,2), (7,9)]	{<0,2>, <0,1>}
Val	GTA	[(0,0), (2,7)]	{<0,1>, <0,2>}
Ala	GCG	[(3,4), (7,9)]	{<0,1>, <0,2>}

Figure 3: Output screen of CVLTool. Displays library design and gene variant graph. "Download Oligos" button enables the retrieval of oligo DNA sequences in text format. "Fragments utilized" and "Design" information is described in detail in [6].

3. REFERENCES

[1] Gibson, D.G. et al. 2008. Complete chemical synthesis, assembly, and cloning of a Mycoplasma genitalium genome. *Science*. 319, 5867 (2008), 1215–1220.

[2] Gibson, D.G., Young, L., Chuang, R.Y., Venter, J.C., Hutchison 3rd, C.A. and Smith, H.O. 2009. Enzymatic assembly of DNA molecules up to several hundred kilobases. *Nat Methods*. 6, 5 (2009), 343–345.

[3] Kudla, G., Murray, A.W., Tollervey, D. and Plotkin, J.B. 2009. Coding-sequence determinants of gene expression in Escherichia coli. *Science*. 324, 5924 (2009), 255–258.

[4] Plotkin, J.B. and Kudla, G. 2010. Synonymous but not the same: the causes and consequences of codon bias. *Nat Rev Genet*. 12, 1 (2010), 32–42.

[5] Ramon, A. and Smith, H.O. Single-step linker-based combinatorial assembly of promoter and gene cassettes for pathway engineering. *Biotechnol Lett*. 33, 3, 549–555.

[6] Ryan, D. and Papamichail, D. 2013. Rational design of orthogonal libraries of protein coding genes. *ACS Synthetic Biology*. 2, 5 (2013), 237–244.

[7] Supek, F. and Smuc, T. On relevance of codon usage to expression of synthetic and natural genes in Escherichia coli. *Genetics*. 185, 3, 1129–1134.

[8] Welch, M., Govindarajan, S., Ness, J.E., Villalobos, A., Gurney, A., Minshull, J. and Gustafsson, C. 2009. Design parameters to control synthetic gene expression in Escherichia coli. *PLoS One*. 4, 9 (2009), e7002.

Genetic Systems Engineering

Prashant Vaidyanathan[1], Evan Appleton[2,4], Curtis Madsen[1], Cristian-Ioan Vasile[3], Alan Pacheco[2,4], Iman Haghighi[3], Nicholas Roehner[1], Rachael Ivison[4], Junmin Wang[4], Yash Agarwal[2], Zachary Chapasko[1], Calin Belta[3,4] and Douglas Densmore[1,2,4]

[1]Department of Electrical & Computer Engineering, Boston University, Boston, MA
[2]Biological Design Center, Boston University, Boston, MA
[3]Division of Systems Engineering, Boston University, Boston, MA
[4]Graduate Program in Bioinformatics, Boston University, Boston, MA
{prash,eapple,ckmadsen, cvasile,pachecoa,haghighi,nroehner,rivison,dawang,yash1214,ztc,cbelta,dougd}@bu.edu

1. INTRODUCTION

Systems engineering is an interdisciplinary field that focuses on building and maintaining complex engineering systems over their entire life cycles. Applying standards and core concepts of related sub-fields such as performance engineering (which focuses on ensuring that a system meets its expected performance requirements throughout its life cycle) and reliability engineering (which focuses on making sure that a system does not fail more than expected during its life cycle) 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.

Recent work has demonstrated that genetic logic circuits can be specified, synthesized, and built using a hardware description language (like Verilog) [3]. Although this was a vital stepping stone in creating a reliable framework to synthesize genetic regulatory networks, Boolean functions lack the functional richness required to capture the finer details of molecular biology. Furthermore, sparse or poorly defined characterization of genetic modules makes creating robust genetic circuits difficult [2]. As biologists try to build more complicated genetic systems, where the performance specification of the desired system includes additional parameters that are intrinsic to genetic components and their interactions with the complex environment they operate in, a more holistic approach is required. To accomplish this goal, we have designed a software tool that accounts for many types of decisions made throughout a complete specify-design-build-test-learn work flow to enable iterative design of complex genetic systems.

2. IMPLEMENTATION AND WORKFLOW

We introduce Phoenix as a design environment that can implement an end to end implementation of systems engineering for synthetic biology.

- **Specification**: Any genetic system would require 3 specifications:
  - **Functional specification**: which describes the overall expected behavior of a system as well as the interaction with environmental context
  - **Performance specification**: which describes the temporal logic and timing constraints
  - **Structural specification**: which describes the orientation of DNA strands as well as the combination and relative positions of DNA components.

  These specifications are available as parameterized *Signal temporal logic*(STL) function templates for an inverter, oscillator or toggle switch. A user can specify a system consisting of one of these modules or a combination of these modules. The user can also upload annotated DNA sequence data (multi-part genbank files) which is stored in a database using Clotho (a database management tool for synthetic biology). Structural orientation of the DNA components are specified using Eugene (a rule based design language) [4]. The structural specifications are checked against known design rule constraints using grammar files (written in ANTLR) to check the validity of the structural design.

- **Design**: The tool then builds a module tree to break down the structure of the genetic module [5] down to the abstract DNA components [5] required to build the genetic system. 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 can function like 'unit tests' for the genetic module being built.

- **Build and Test** Once these components have been built and tested in vivo, the raw cytometry data is processed by a data analysis script using the *Bioconductor* library in R. These characterized modules are then used to synthesize a genetic regulatory network with assigned DNA components using parameter estimation and in silico simulation. The assignment algorithm uses various design space pruning methods along with stochastic methods (like simulated annealing). These assigned modules are checked against the STL specification for the system and scores for each assignment is assigned based on the closeness to the STL formula [6].

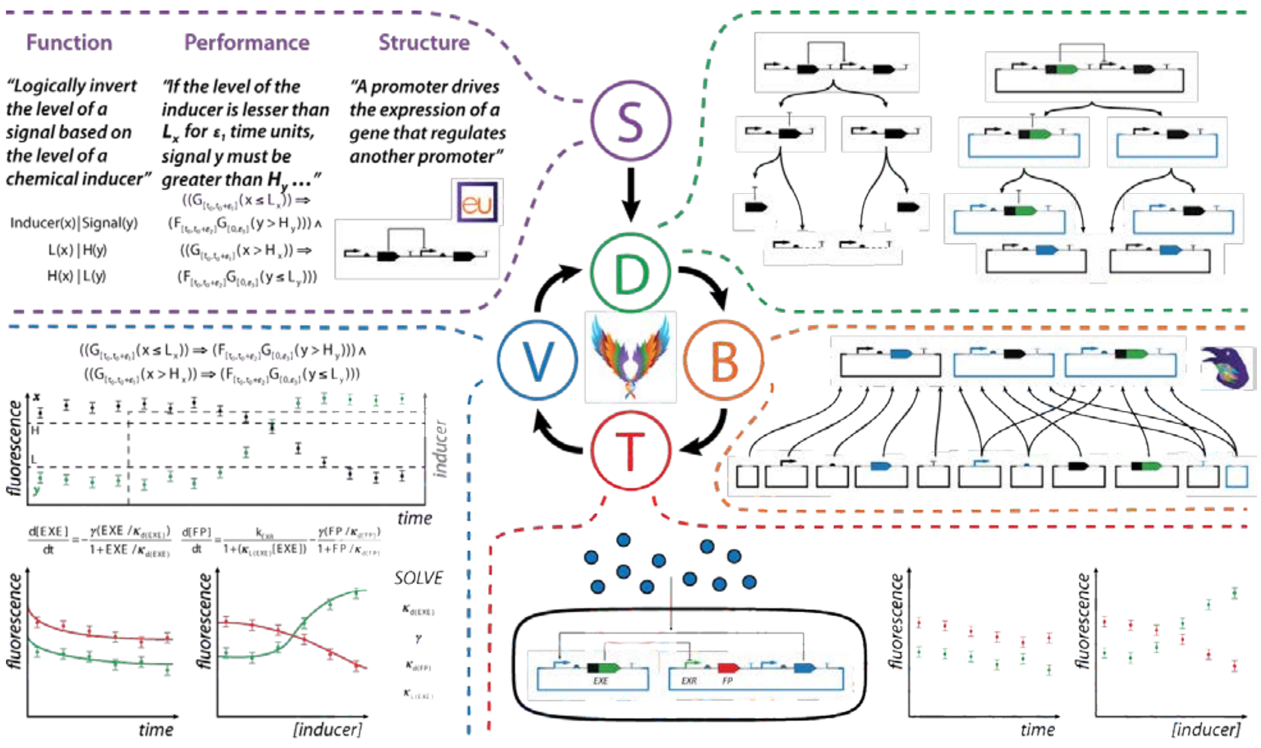

Function
Performance
Structure
"Logically invert the level of a signal based on the level of a chemical inducer"
"If the level of the inducer is lesser than L_x for ϵ_1 time units, signal y must be greater than H_y ..."
"A promoter drives the expression of a gene that regulates another promoter"
Inducer(x)|Signal(y)
L(x) | H(y)
H(x) | L(y)
S
D
B
T
V
fluorescence
inducer
time
SOLVE
[inducer]
EXE
EXR
FP

Figure 1: *Phoenix* tool map and flow. **S**pecify: To build robust systems, functional, performance and structural specification is required. This example describes a genetic inverter. **D**esign: The design step creates a module tree based on the structural and functional specifications. The DNA components(augmented with testing modules) required to build the genetic system as well as assignments for the genetic regulatory network are synthesized. **B**uild: The build step generates assembly instructions to build these genetic components and modules in vivo. **T**est: In the test phase, raw cytometry data is gathered for the systems built in the Build step. **V**erify: These results are then processed and verified against the performance specification for the genetic system to compute the robustness of the system.

The assignments with the best scores are then generated as the output of that iteration along with assembly instructions (using Raven) to build them in vivo. These assignments are tested and verified against the performance specification of the genetic system.

- **Verify and Learn**: The results of both successful and unsuccessful assignments are cataloged and used to fine-tune and improve the synthesis results for successive iterations using principles of machine learning.

3. CONTRIBUTIONS

Synthetic biology has various tools in the Specify(GEC, Eugene), Design(Cello, SBROME), Build(Pr Pr, Puppeteer), and Test/Verify(iBioSim, Tinkercell) phases of BDA. Each tool focuses on a very specific task and solves a very specific sub problem. However, 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. The framework is designed to ensure that the functional, performance and structural specification of the genetic system is well defined, reproducible and reliable. The iterative process will use principles of reinforcement learning to improve the performance of the synthesis and assignment algorithms for successive iterations.

4. ACKNOWLEDGEMENTS

The authors would like to thank Dr. Ernst Oberortner and Melissa Rhoads, Dr. Nikolas Pontikos, Dr. Sonya Iverson, Dr. Traci Haddock, and Diego Cuerda, for helpful conversations and Addgene for providing DNA samples used to build genetic regulatory networks. This work has been funded by the Office of Naval Research under Grant No. N00014-11-1-0725 and National Science Foundation Award 1446607.

5. REFERENCES

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

[2] J. Beal. Signal-to-noise ratio measures efficacy of biological computing devices and circuits. *Frontiers in bioengineering and biotechnology*, 3, 2015.

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

[4] 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.

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

[6] C.-I. Vasile et al. Compositional signal temporal logic with applications to synthetic biology. Submitted.

Improving controllability of biosynthesis of gold nanoparticles by *Shewanella oneidensis* through genetic manipulation

Juliano Bertozzi Silva
Chemical and Biological Engineering
The University of Sheffield
Mappin Street
Sheffield S1 3JD
+44 (0)791 403 5621
juliano@sheffield.ac.uk

Gregory Fowler
Chemical and Biological Engineering
The University of Sheffield
Mappin Street
Sheffield S1 3JD
+44 (0)114 222 7500
g.fowler@sheffield.ac.uk

Phillip C. Wright
Newcastle University
Newcastle upon Tyne
NE1 7RU
+44 (0)191 208 5556
phillip.wright@ncl.ac.uk

1. INTRODUCTION

The application of scientific knowledge for the manipulation of matter at the nanoscale is an emerging field of research that is generating a large degree of interest not only to the scientific community but also to the public in general. This field is known as nanotechnology. Nanoparticles, small agglomerates of atoms, play a central role in the field. They are important because they have the potential to compose the building blocks for the construction of nanodevices and because materials reduced to the nanoscale frequently have properties different from the bulk parts. Gold nanoparticles (AuNPs) are probably the most studied nanoparticles and can be considered the model entities for the field. One of the reasons why AuNPs are so researched is that they display the surface plasmon resonance (SPR) phenomenon when at nanoscale. SPR happens when the mean free path of the conduction electrons in the bulk material is bigger than the particle radius and is characterized by a standing oscillation of the free electrons in the metal when stimulated by photons at certain wavelengths [1]. AuNPs have been widely studied for applications in a range of fields, such as electronics, medicine and catalysis.

Overall, the methods for synthesizing metallic nanoparticles can be grouped into three categories: chemical, physical and biological methods. Chemical synthesis is the preferred method. In general, this approach is simple and has good controllability, however it frequently raises concerns for personal and environmental safety because of the necessary handling of harsh chemicals. Physical methods are more commonly used for the top-down manufacturing of nanodevices than for the synthesis of nanoparticles. The main reason for this is the high cost of equipment in comparison to other methods. Biological processes are better explained as a chemical reduction performed by biomolecules. This approach is gaining increased attention due to the lack of requirement of strong chemicals, high temperatures or pressures, the low cost involved, and the simplicity of the method.

Several different organisms have been demonstrated to be able to synthesize metallic nanoparticles. These include species of bacteria, algae, fungi and even plants. Although extensive work has been done for the determination of new species capable of producing metallic nanoparticles, very little research has been directed towards the mechanisms involved in nanoparticle biosynthesis. What is known so far is that the process starts with adsorption and/or absorption of metallic ions by the organisms, followed by biochemical reduction [2].

Shewanella oneidensis is a facultatively aerobic Gram-negative bacterium isolated from the Lake Oneida in New York [3]. This strain is peculiar for having an advanced protein system that allows it to utilize a range of inorganic compounds as terminal electron acceptors. This system, called the Mtr pathway, is comprised of a group of *c*-type cytochromes and proteins that shuttle electrons from the quinone pool to outside the cell. A representation of the Mtr pathway can be found in Figure 1. It can be seen in Figure 1 that the electron, coming from the cytoplasmic membrane, is transferred from CymA, a tetraheme cytochrome, to the PEC, a decaheme cythochrome that is composed of one of four paralogs, MtrA, MtrD, DmsE and SO4360. β-barrel, a protein that can be either MtrB, MtrE, DmsF or SO4359, facilitates the direct transfer of electrons from PEC to OMC, a decaheme cythochrome made of one of three paralogs, MtrC, MtrF and OmcA, that is then responsible for reducing the inorganic compounds extracellularly.

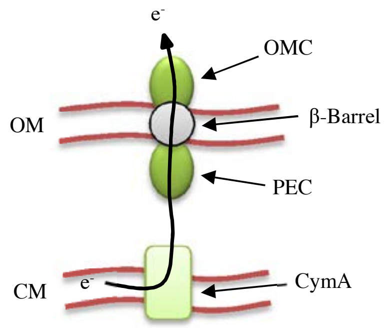


Figure 1: Mtr pathway of *S. oneidensis* MR-1 (Adapted from [4]). OMC stands for outer-membrane cytochrome, PEC for periplasmic electron carrier, OM for outer membrane and CM for cytoplasmic membrane.

This special system in *S. oneidensis* makes it attractive for the biomanufacturing of metallic nanoparticles, as it is reasonable to assume that the respiratory mechanisms of the organisms, if are not fully responsible, might have at least an influence in the reduction of metallic ions in solution. In fact, previous studies have already used this strain for the biosynthesis of different kinds of metallic nanoparticles, including AuNPs [5].

Here, we investigate the effect of OMC in the biosynthesis of AuNPs. We report the formation of gold nanoparticles by several mutants of *S. oneidensis* containing all the combinations of gene deletions of the OMC paralogs. The parameters monitored were surface plasmon band peaks, gold ions adsorption capability, quantity of nanoparticles produced and characteristics of the particles made (size and shape). These experiments provide some insights into the factors that influence nanoparticle formation and improved controllability of the AuNPs formed. Ultimately we are interested in isolating the parts of the MTR pathway that are responsible for the formation of particles of specific sizes and shapes. Developing a model, based on empirical results, describing the biosynthesis of the particles is also a goal of the project.

2. SOME INITIAL FINDINGS

Figure 2 contains the peaks of the normalized wavelength scan readings, obtained via spectrophotometry, of *S. oneidensis* wild-type (WT) and some of the mutants tested. The scans were normalized to obtain a comparison of the surface plasmon bands generated. The values correspond to the differences between the measurements taken after 48h of incubation in gold solution and the measurements taken at time zero.

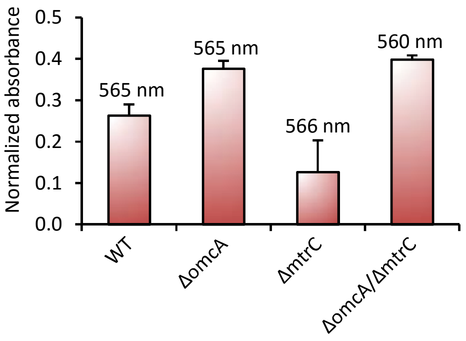


Figure 2: Wavelength scan measurements of *S. oneidensis* WT and mutants during incubation in gold solution for 48h. The results are the peaks of the normalized spectra. The numbers on top of the bars correspond to the wavelengths of the peaks. Error bars indicate the standard deviation of three replicates.

Figure 2 shows clearly that the deletion of genes that contribute to extracelullar respiration impacts on the biosynthesis of AuNPs. The differences in the peaks were statistically significant in some cases. These results mean that some gene products favor the production of nanoparticles in such a way that the plasmon band formed is bigger or smaller than others. Various factors influence plasmon band intensity, such as size and shape of the particles, as well as the amount of nanoparticles made, among other things. Several different analyses can be carried out to better determine the factors that were influenced by the mutations. One of them, transmission electron microscopy (TEM), was used in this study for the same strains. The results can be found in Figure 3.

The TEM results in Figure 3 also show differences in the nanoparticles synthesized by the strains. For instance, it is clear that the AuNPs made by the Δ*mtrC* mutant are, in general, much smaller than those made by the other strains. At the same time, it can be seen that the peak for this mutant in Figure 2 is also smaller in comparison to the other strains. In the case of the Δ*omcA* mutant, the particles made were bigger than the ones made by the other strains. This result also correlates to Figure 2, where this mutant had an overall high peak. The mutant containing deletions in the *mtrC* and *omcA* genes had the highest plasmon band peak, but didn't present the largest particles in Figure 3. However, it can be seen that this strain had a higher amount of particles in comparison to WT and Δ*omcA* mutant.

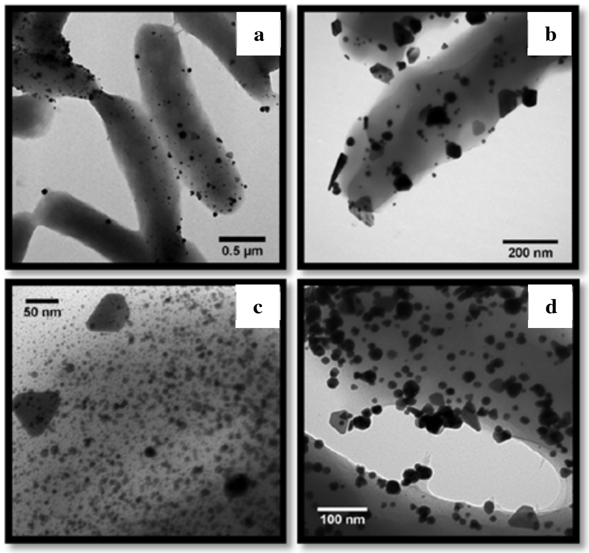


Figure 3: Sample TEM pictures of *S. oneidensis* (WT and mutants) with the biosynthesized AuNPs. a) WT, b) Δ*omcA* mutant, c) Δ*mtrC* mutant, d) Δ*omcA*/Δ*mtrC* mutant.

These results combined show that both particle size and quantity made influenced the peaks. More importantly, it is clear that certain genes have a direct influence on the size of the particles. That is the case, for instance, of *omcA*. The particles with size under 10 nm were made only when this cytochrome was present. It can be argued that *omcA* is also contained in the WT strain, but it should be noted that *mtrC* is the preferential paralog for OMCs [4].

These are only some of the results obtained in this project, much more findings were achieved and we look forward to present them at IWBDA.

3. ACKNOWLEDGMENTS

This project is supported by EPSRC and The University of Sheffield Studentship. We also acknowledge Dr. Jeff Gralnick for providing the mutant strains.

4. REFERENCES

[1] Eustis, S. and El-Sayed, M. A. 2006. Why gold nanoparticles are more precious than pretty gold: Noble metal surface plasmon resonance and its enhancement of the radiative and nonradiative properties of nanocrystals of different shapes. *Chemical Society Reviews*, 35, 209-217.

[2] Li, X., Xu, H., Chen, Z.-S. and Chen, G. 2011. Biosynthesis of nanoparticles by microorganisms and their applications. *Journal of Nanomaterials*, 1-16.

[3] Myers, C. R. and Nealson, K. H. 1988. Bacterial manganese reduction and growth with manganese oxide as the sole electron acceptor. *Science*, 240(4857), 1319-1321.

[4] Coursolle, D. and Gralnick, J. A. 2012. Reconstruction of extracellular respiratory pathways for iron(III) reduction in *Shewanella oneidensis* strain MR-1. *Frontiers in Microbiology*, 3(56), 1-11.

[5] Suresh et al. 2011. Biofabrication of discrete spherical gold nanoparticles using the metal-reducing bacterium *Shewanella oneidensis*. *Acta Biomaterialia*, 7, 2148-2152.

Load Capacity Improvements in Transcriptional Systems Using Discrete-Time L1-Adaptive Control

[Extended Abstract]

Hamidreza Jafarnejadsani
University of Illinois Urbana Champaign
Urbana, IL 61801, USA
hamid.jafarnejad@gmail.com

Jongmin Kim
Wyss Institute
Harvard University
Cambridge, MA 02115
Jongmin.Kim@wyss.harvard.edu

Naira Hovakimyan
University of Illinois Urbana Champaign
Urbana, IL 61801, USA
nhovakim@illinois.edu

Vishwesh V. Kulkarni*
University of Warwick
School of Engineering
Coventry, CV4 7AL
V.Kulkarni@warwick.ac.uk

1. INTRODUCTION

DNA-based circuits relying on predictable thermodynamics and kinetics of DNA strand interactions impart flexibility in synthesizing synthetic biological constructs and in coupling these circuits to *in vivo* processes [1, 2, 6, 7]. Here, we focus on the synthetic Kim-Winfree oscillator network, illustrated in Fig. 1(i), which is a simple but effective coupled oscillator system in which two DNA switches SW1 and SW2 are coupled through activator and inhibitor blocks realized by RNA signals and auxiliary DNA species (see [3]). A typical experimental realization is *closed* in the sense that once the operation starts, we do not either add any chemicals, especially NTP fuel, externally into the wet-lab apparatus or remove any chemicals, especially waste products, from the apparatus. Within the closed system, the oscillations are bound to die out sooner or later — diminishing NTP fuel eventually stops supporting the production of RNA signals and accumulating waste products clog down the toeholds and, as a result, adversely affect the signal propagation. Furthermore, the oxidation effects and the pH variations tend to deactivate the enzymes. Loading poses an additional challenge since it increases the order and the uncertainty of the system — indeed, these oscillators have recently been used in [8] to drive conformational changes of a DNA nanomechanical device called DNA tweezers. As Fig. 1(ii) shows, the oscillator performance degrades sharply under loading. We propose to improve the loading capacity of such transcriptional devices by adopting a partially open architecture and by using a discrete-time *in silico* controller, a block diagram of which is illustrated in Fig. 2, which is to be coupled to the wet-lab apparatus.

A light switching *in silico* controller, implementing a combination of a Kalman filter and a model predictive controller, was recently reported in [10]. In [4], a continuous-time $\mathcal{L}_1$ adaptive controller was proposed to use a DNA/RNA strand based actuation that faciliates a much greater control channel bandwidth than the one provided by a light-based actuation. In the $\mathcal{L}_1$ adaptive control architecture, the estimation loop is decoupled from the control law. This decoupling allows for the use of fast estimation rates, leading to uniform performance bounds and guaranteed robustness in the presence of bounded nonlinearities and system uncertainties. Hence, the closed-loop system converges to a reference system with partial compensation of uncertainties, which is linear, and hence has a scalable, repeatable, and predictable response. Our discrete-time $\mathcal{L}_1$ adaptive controller builds on the theory developed in [5] which ensures stability for any sampling time. This controller is optimized using a numerically efficient convex optimization method and is well suited for many such bioengineering applications since the frequencies of the reference inputs encountered are slow enough, the biological processes evolve slowly enough, and the wet-lab measurements are at discrete time intervals.

2. MAIN RESULTS

We adopt a partially open architecture analogous to a microchemostat (see [9]) so as to inject control inputs without increasing the reaction volume. In this continuous flow reactor, DNA species, enzymes, and NTP fuel flow in at a low rate, while the outflow removes a portion of reaction mixture, keeping the accumulation of waste products in check. We choose the switch outputs of the main reaction chamber to be the state variables that are to track the desired periodic waveforms. The DNA switches are tagged with fluorophores and the auxiliary DNA activators, A1 and A2, are tagged by quenchers such that the binding of an activator to its target switch reduces fluorescence signal. The two switches, SW1 and SW2, have different fluorophores, allowing for a

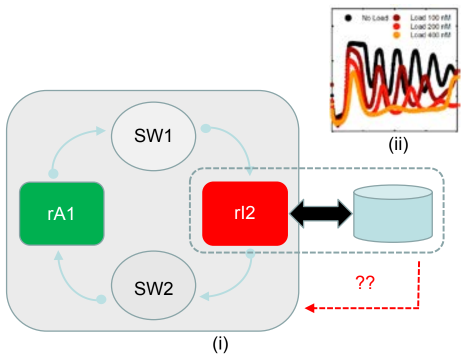


Figure 1: (i) The Kim-Winfree oscillator network comprises two switches (SW1 and SW2) connected through an activator rA_1 and an inhibitor rI_2 block. In [8], it is used to drive DNA tweezers. Due to the loading effects and a closed-system design, this network is unable to drive even moderate loads: the plot (ii), taken from [8], illustrates the loss of oscillations as the load increases from 0 nM to 400 nM. This highlights the need for more sophisticated controllers and for a more open design.

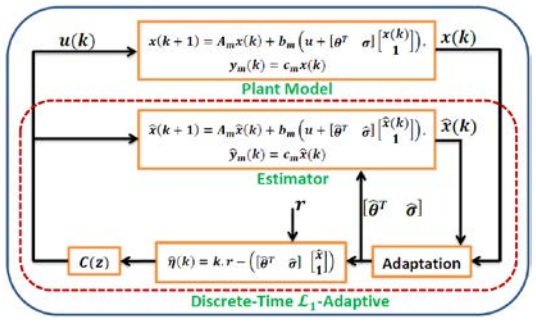


Figure 2: Our discrete-time $\mathcal{L}_1$ adaptive controller. This controller is implemented *in silico* in a computer outside the wet-lab apparatus and the interfaced with the apparatus.

real-time measurement of the switch outputs as fluorescence signals. The target waveforms are generated internally as the reference signals in an *in silico* controller implemented inside a computer — the commands of *in silico* controller controls, in discrete-time, the concentration of inhibitor and activator strands to track the reference signals. We characterize the expected disturbance and modeling uncertainty, obtain a discrete-time model of the overall system to be controlled, and then synthesize a discrete-time $\mathcal{L}_1$ adaptive feedback controller to achieve the desired performance. As Fig. 3 illustrates, a significant improvement in the tunability and loading capacity of oscillator is obtained. This approach can easily be adopted to improve the robustness and the loading capacity of a wide range of wet-lab devices.

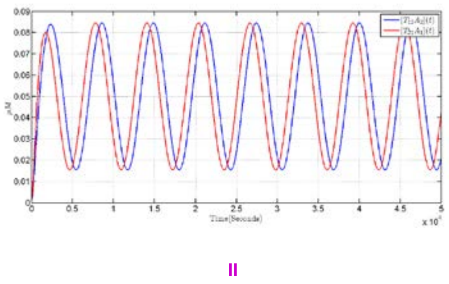


Figure 3: Simulation results for the models of DNA tweezers loaded with 200nM. If the $\mathcal{L}_1$ adaptive controller is not used then the free running system fails to generate any oscillations at all; this has been observed experimentally as well for high load cases (see [8]). However, if our $\mathcal{L}_1$ adaptive controller is used then the desired oscillations are obtained.

3. REFERENCES

[1] Zhang, D.Y. and Seelig, G.: 'Dynamic DNA Nanotechnology using Strand-Displacement Reactions', Nature Chemistry, 2011, 3, (2), pp. 103–113

[2] Padirac, A., Fujii, T., and Rondelez, Y.: 'Nucleic acids for the Rational Design of Reaction Circuits', Current Opinion in Biotechnology, 2012, http://dx.doi.org/10.1016/j.copbio.2012.11.011

[3] Kim, J. and Winfree, E.: 'Synthetic in vitro transcriptional oscillators', Molecular Systems Biology, 2011, 7, 465

[4] Kulkarni, V., Kharisov, E., Hovakimyan, N., and Kim, J.: 'Load Capacity Improvements in Nucleic Acid Based Systems Using Partially Open Feedback Control', ACS Synthetic Biology 3, 617-626.

[5] Jafarnejadsani, H. and Hovakimyan, N.: 'Optimal Filter Design for a Discrete-Time Formulation of L1 Adaptive Control', AIAA InfoTech at Aerospace, Kissimmee, FL, January 2015.

[6] Chen, Y., and Smolke, C.: 'From DNA to Targeted Therapeutics: Bringing Synthetic Biology to the Clinic'. Science Translational Medicine, 2011, 3, 106

[7] Montagne, K., Plasson, R., Sakai, Y., Fujii, T., and Rondelez, Y.: 'Programming an *in vitro* DNA Oscillator using a Molecular Networking Strategy', Molecular Systems Biology, 2011, 7, 466

[8] Franco, E., Friedrichs, E., Kim, J., Jungmann, R., Murray, R., Winfree, E., Simmel, F.: 'Timing Molecular Motion and Production with a Synthetic Transcriptional Clock', PNAS, 2011, 108, E784-E793

[9] Balagaddé, F.K., You, L., Hansen, C.L., Arnold, F.H., Quake, S.R.: 'Long-term Monitoring of Bacteria Undergoing Programmed Population Control in a Microchemostat', Science, 2005, 309, 137-140

[10] Milias-Argeitis, A., Summers, S., Stewart-Ornstein, J., Zuleta, I., Pincus, D., El-Samad, H., Khammash, M., and Lygeros, J.: '*In silico* Feedback for *in vivo* Regulation of a Gene Expression Circuit', Nature Biotechnology, 2011, 29, 12, pp. 1114-1116

Logic and Timing Analysis of Genetic Logic Circuits using D-VASim

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

1. INTRODUCTION

In microelectronics, timing analysis is a very crucial requirement for ensuring the correct operation of a logic circuit. For the genetic logic circuits, the timing analysis may likely become an essential design characteristic. In digital electronics, circuits are made up of digital logic gates, which themselves are composed of transistors [1]. The transistors, typically used in the composition of digital logic gates, have well-defined threshold voltage value [1], which categorizes the logic level-0 and 1. To date, the timing characteristics, like propagation delay, hold time, setup time etc., are all well characterized. This is however not the case for genetic logic gates, where each of them are composed of different species and promoters, which may result in different threshold concentration values. Furthermore, digital logic gates have the same physical quantity, i.e., voltage, as their input and output. On the contrary, genetic logic gates usually have different species, which act as input to control the regulation of other specie, which acts as output. Besides this, the signals in electronic circuits propagate in separate wires, which does not directly interfere with each other. However in genetic circuits, signals are molecules, drifting in the same volume of the cell, and hence easily merge with the concentration of other compounds, resulting in crosstalk with the neighbouring circuit components. These facts make the timing analysis of genetic circuits quite challenging.

Similar challenges were encountered when the field of microelectronic circuits design was immature in its early days. Today, some of these challenges are solved through the enhancement in the fabrication process; others have been addressed through the development of advanced electronic design automation (EDA) tools. The advancement in the genetic design automation (GDA) tools can also help addressing these challenges, which may result in the reduction of design complexity of genetic logic circuits. D-VASim (Dynamic Virtual Analyzer and Simulator) is one such tool, which is developed for the simulation and analysis of genetic logic circuits [2]. A new feature of logic and timing analysis has been introduced in D-VASim. This feature is introduced to help users in verifying the logic function of a genetic logic circuit by extracting the observed logic function from the simulation data. During the analysis, a user may apply all the possible input combinations to find out the correct logic behavior of a circuit. The key challenge in determining the correct Boolean logic function from the analog simulation data is to categorize the input concentrations levels into logic-0 and logic-1. As mentioned earlier, this is similar to digital electronic circuits in which a certain threshold value of input voltage differentiates the logic levels 0 and 1 [1]. The threshold value for the concentration of input species must also be identified, which significantly effects the concentration of output specie of a genetic logic circuit.

2. METHODOLOGY

We performed the preliminary timing analysis on the genetic logic circuit models developed by Myers [3] using iBioSim [4]. Fig. 1 shows the results from running the stochastic simulation of the genetic NAND gate, one (Fig. 1(a)) and fifty times (Fig. 1(b) and (c)), respectively. The unit of species' concentration used for these models is the "number of molecules" [3]. Fig. 1(a) shows that both of the inputs are triggered to 11 molecules, TetR after 1000 time units (s) and LacI after 2000 time units (s), and that the output is highly stochastic, which makes it difficult to determine the input threshold value. A smooth output curve is obtained by plotting the average of 50 runs, as shown in Fig. 1(b) and (c).

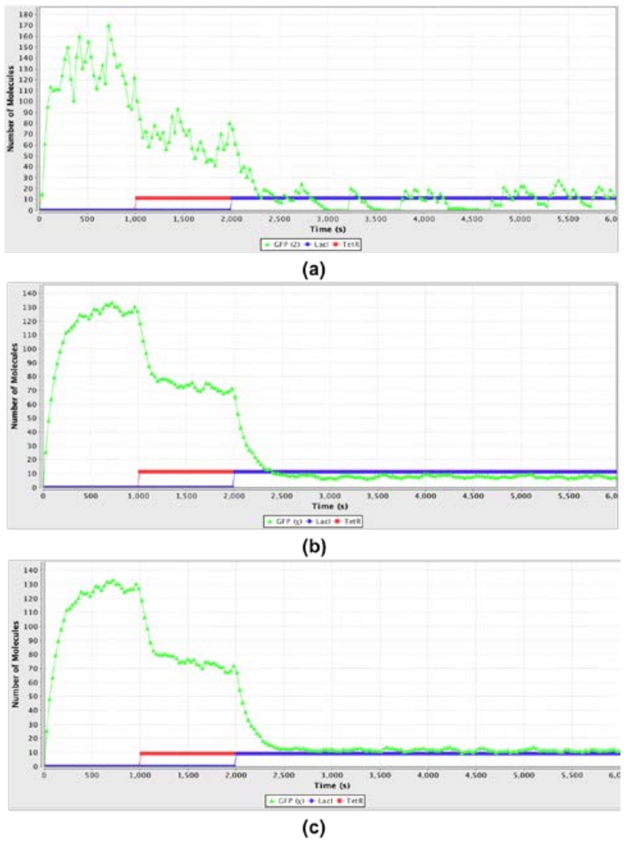


Fig. 1. Preliminary analysis of a threshold value for the genetic NAND gate using iBioSim.

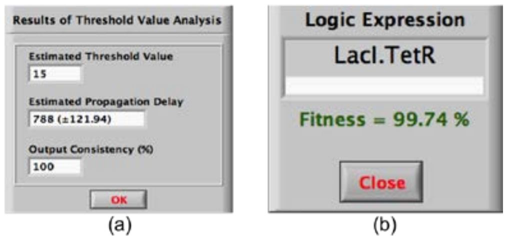
In Fig. 1(b), it can be observed that keeping the input concentration to 11 molecules causes the average output concentration to cross the level of input concentration. If we reduce the input concentration further to 10 molecules, the average output concentration stays above the level of input concentration, as depicted in Fig. 1(c). We performed the same analysis with different concentration levels on different logic circuits. Based on this analysis we define the threshold value as "*the minimum concentration of input species, which causes the average production of output specie to cross the level of threshold* *input concentration*". D-VASim obtains the threshold value of a genetic logic circuit iteratively; by automating this process of gradually increasing the level of input concentrations to a next user-defined level and observe if it effects the concentration of output specie. Another important parameter in obtaining the correct logic function of a genetic circuit is the *input-output propagation delay*. The input-output propagation delay can be defined as "*the time from when the input concentration reaches its threshold value until the time that corresponds to the instant when the output concentration crosses the same threshold value*". For instance, assume that the initial concentrations of the input species, of a two-input genetic AND gate, are zero (logic-0). The input-output propagation delay is the time between the instant when both of these input species are triggered to a significant concentration level (threshold level, logic-1) and the instant when the concentration of output specie reaches to the same significant level of concentration (threshold level, logic-1). Therefore, for the single-gate genetic circuits, the input-output propagation delay can also be termed as a *gate delay*. Similar to electronic circuits, where the multiple transitions of inputs go undetected when they occur either in one clock cycle or in a time less than the gate delay; the genetic circuits may also be expected to have such behavior. Hence, in order to obtain the correct behaviour of a genetic circuit, all the possible input combinations must be applied each after this propagation delay. The propagation delay in Fig. 1(b) is about 400 time units.

3. EXPERIMENTATION AND RESULTS

D-VASim requires some user-defined parameters to perform threshold value and propagation delay analysis. It begins by asking *Start at* value, which specifies the concentration of input specie(s), from which the tool should start its threshold analysis. The user is also required to specify *Increment of* and *Stop at* parameters. *Increment of* denotes the value with which the input concentration is increased for each iteration. *Stop at* value specifies the input concentration at which the algorithm should stop the analysis of the threshold value. When D-VASim estimates the possible threshold value, it verifies this value by iterating the model for the user-defined *No. of iterations*. During this iterative verification process, D-VASim obtains the average propagation delay. It also identifies how consistent the average output for the estimated threshold value is.

The algorithm requires an initial assumption of the propagation delay value. It is already mentioned earlier that the propagation delay value is critical in extracting the correct logic behavior of a circuit model. Thus, it is necessary for D-VASim to wait until this time value has lapsed before switching on to the next combination of inputs while searching for an appropriate threshold level. Since the propagation delay value is unknown for the automatic analysis, D-VASim begins the analysis with an assumed value and later estimates the approximate one. Assuming a higher value increases the estimation time but gives a more accurate estimation of the threshold value and propagation delay.

Fig. 2(a) shows the threshold value analysis results obtained by D-VASim for the genetic AND gate [3]. D-VASim also estimates the standard deviation (enclosed in braces in the *Estimated Propagation Delay* value box) of propagation delay for all number of iterations. These results indicate that an input concentration value of 15 (default units obtained from SBML model) is a threshold value for this model. Any value below is considered as logic-0 and above as logic-1. When the concentration of both the inputs are kept above 15 units, the output specie is supposed to be triggered approximately after 788 (±121.94) time units. The *output consistency* specifies that the average output, during iterative analysis, remained 100% consistent for the estimated threshold value and propagation delay.


Results of Threshold Value Analysis

Estimated Threshold Value
15

Estimated Propagation Delay
788 (±121.94)

Output Consistency (%)
100

OK

Logic Expression

LacI.TetR

Fitness = 99.74 %

Close

Fig. 2. Outcomes of D-VASim automated analysis. (a) Results of threshold value and propagation delay analysis (b) Behavior of a model in terms of Boolean expression.

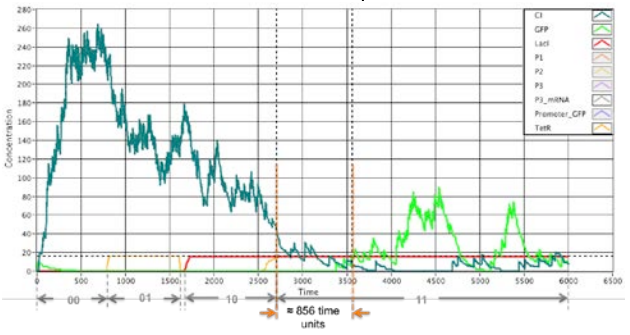


Fig. 3. Simulation traces for genetic AND gate in D-VASim.

These results can be verified in the simulation traces shown in Fig. 3. It can be observed clearly in this figure that when both the inputs, LacI and TetR (see [3] for circuit details) are triggered to logic-1 (threshold concentration level ≥ 15 molecules at ~2700 time units), the output specie (GFP) turns to logic-1 (crosses this threshold value) approximately after 856 time units, which lies under the standard deviation calculated by D-VASim.

When all the possible input combinations are applied with an estimated propagation delay, a logic verification option can be executed. Fig. 2(b) shows the Boolean expression obtained for this genetic circuit model from the simulation data. D-VASim also calculates the percentage fitness of an obtained Boolean expression in the entire simulation data. This tells that the extracted Boolean expression of this model satisfy 99.74% of the simulation data. The rest 0.26% are those instants where the output of an AND gate is low when both of its inputs are high (see Fig. 3 after 3500 time units).

4. SUMMARY

The capability of timing and logic analysis of genetic circuits may help users to characterize the timing constraints of genetic components. With D-VASim, it is also possible to perform the timing and logic analysis on the intermediate components of complex genetic circuits. We will explore this feature further by analysing the timings on recently published genetic circuits [5].

REFERENCES

[1] Balch Mark. Complete Digital Design: A comprehensive guide to digital electronic and computer system architecture. McGraw-Hill Professional. **2003**.

[2] Hasan Baig and Jan Madsen, "D-VASim: Dynamic Virtual Analyzer and Simulator for Genetic Circuits", 7th IWBDA, **2015**.

[3] Chris J. Myers. Engineering Genetic Circuits. Chapman & Hall/CRC Press, July **2009**.

[4] C. J. Myers, N. Barker, K. Jones, H. Kuwahara, C. Madsen, and N.-P. D. Nguyen, ''iBioSim: A tool for the analysis and design of genetic circuits,'' Bioinformatics. **2009**, *vol. 25, no. 21, pp. 2848–2849*.

[5] Alec A. K. Nielsen *et al*., "Genetic circuit design automation", Science 352, **2016**.

MINT

A MIcrofluidic NetlisT Format

Radhakrishna Sanka[1], Haiyao Huang[1], Ryan Silva[1], and Douglas Densmore[1]

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

{sanka,rjsilva,dougd}@bu.edu

1. INTRODUCTION

Fluigi [6] is a microfluidic design framework that allows researchers to realize abstract descriptions of liquid flow relationships automatically as physical devices and corresponding control software. Its goal is to provide synthetic biology researchers with the tools to use microfluidics for novel computation, discovery, and test applications. A critical component of this work-flow is MINT, a format for describing the microfluidic components and the connectivity of the control and flow layers in the microfluidic device. This work describes MINT and where it falls in the larger Fluigi software flow.

Figure 1 illustrates the two work-flows that are a part of Fluigi. In particular, it demonstrates how Fluigi generates designs from both MINT design files and "Functional Descriptions" using tools like Cello [8]. The output generated by Fluigi can be either a photomask drawing for traditional soft layer photolithography [4] or a vector drawing output for the Makerfluidics [10] toolchain to fabricate the microfluidic devices. Using MINT, designers can share their designs with the rest of the community and make use of Fluigi to fabricate their designs.

2. MINT

MINT is based on the microfluidic netlist abstraction introduced in [7]. The microfluidic component descriptions have been expanded to include additional microfluidic components from the scientific and microfluidic literature [6] and has syntax and semantics extended to create a self contained Microfluidic Hardware Description Language with the ability to describe constructs required to perform common microfluidic operations.

The MINT microfluidic constructs are primarily described as either Primitives [7] or Modules [6] where the Modules are a collection of Primitives which are treated as a single entity. All of the MINT constructs can be customized to fit specific design needs by modifying their parameters. The primitives consist of 'PORT', 'MIXER'[11], 'CHANNEL', 'VALVE'[12], 'NET' and 'CELL TRAP'[9] and are the fundamental building blocks. The MINT modules currently supported by Fluigi are 'BANK', 'MUX'[12], 'TREE', 'LOGIC ARRAY', 'GRADIENT GENERATOR'[3], 'DROPLET GENERATOR' [11] and 'ROTARY PUMP'[13].

In a MINT device description, the microfluidic constructs are declared in either the "FLOW" or the "CONTROL" layer by declaring them within the respective 'LAYER' blocks. In addition to the primitives and the standard modules, MINT also allows the designer to import devices created by the user and instantiate them. MINT lets designers tag any of the 'PORT' components present in their design as a 'TERMINAL'. Once tagged the device can be instantiated and be treated any like modules. By instantiating them before the layer blocks, the designer can then make connections to the instance 'TERMINAL'. An example of this importing capability has been demonstrated in Figure 2. A detailed documentation and a tutorial of the MINT example shown in Figure 2 is available in the MINT Wiki [1].

3. INTEGRATION WITH FLUIGI

MINT allows the designer to use Fluigi as and end-to-end microfluidic design tool. The Fluigi software tool consists of a parser for the MINT, a place and route layout system based on [2] and [5] algorithms, a control sequence generator, and an output generator for drawing photomask and vector design files of the device layout. It can use the MINT design files to place and route the device layouts. It is also capable of importing and instantiating designs to generate complex hierarchical designs as shown in Figure 2. In addition, it is also capable of generating microfluidic devices for genetic circuits and their corresponding control sequences [7].

4. CONCLUSION

Using MINT designers can now share their microfluidic designs and create complex Lab on Chip (LoC) or Microfluidic Large Scale Integration (MLSI) systems utilizing the "sandbox" created by the combination of MINT, Fluigi and Makerfluidics.

The integration of synthetic biology, novel microfluidic fabrication techniques, and embedded electronics has the potential to transform the landscape of engineering biological systems. Since applications in cell-based therapeutics and biosensors expand on the idea of distributed biological computation, they could be greatly benefited with advances in CAD tools that not only automate the design process for Flow Microfluidics but also take advantage of novel fabrication techniques.

5. REFERENCES

[1] MINT Github Wiki. https://github.com/CIDARLAB/mint/wiki, 2016.

[2] V. Betz and J. Rose. VPR: A new packing, placement, and routing tool for FPGA research. In *International Workshop on Field Programmable Logic and Appliation*, 1997.

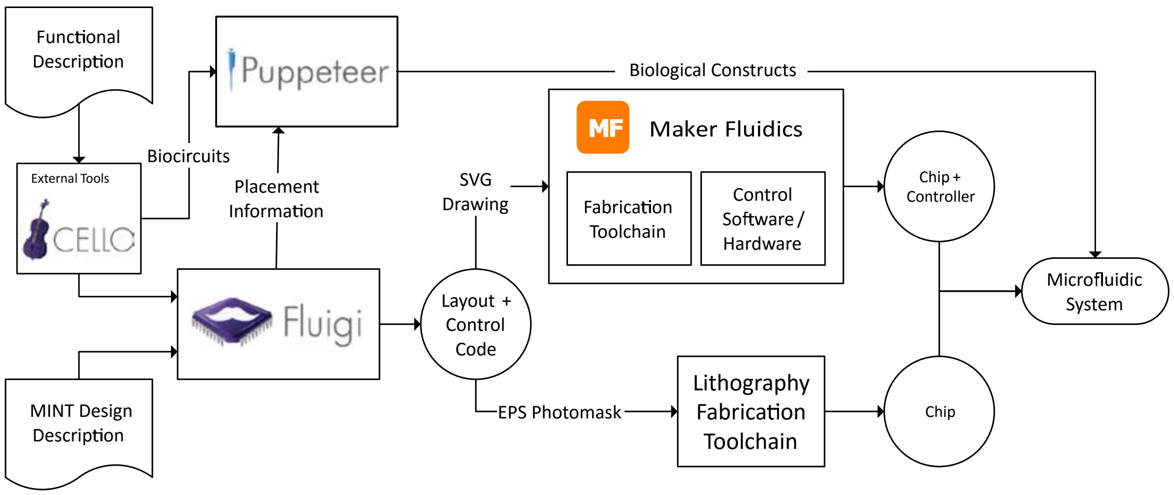
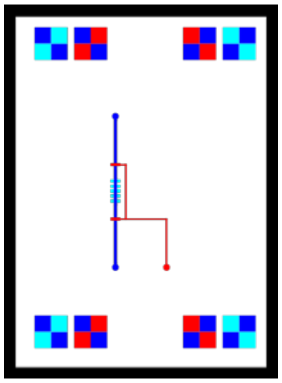


Figure 1: Fluigi Work-flows. Shown is how a MINT Description or a Functional Description of a biological circuit can be synthesized into a microfluidic device. It also shows how the different research projects will interact with Fluigi to automate entire process in the future.

```
DEVICE net_test

LAYER FLOW
PORT p1, p2 r=100;
V LONG CELL TRAP ct1 numChambers=10
    chamberWidth=100 chamberLength=100
    chamberSpacing=50 channelWidth=100;

CHANNEL c1 from p1 3 to ct1 1 w=100;
CHANNEL c2 from ct1 2 to p2 1 w=100;

TERMINAL 1 p1;
TERMINAL 2 p2;

END LAYER

LAYER CONTROL
PORT p3 r=100;
VALVE v1 on c1 w=300 l=100;
VALVE v2 on c2 w=300 l=100;
NET net1 from p3 1 to v1 4, v2 4
    channelWidth=50;
END LAYER
```



```
IMPORT net_test

DEVICE module_test


net_test nt1,nt2,nt3,nt4;

LAYER FLOW

CHANNEL cx1 from nt1 1 to nt2 2 w = 100;
CHANNEL cx2 from nt2 1 to nt3 2 w = 100;
CHANNEL cx3 from nt3 1 to nt4 2 w = 100;


END LAYER
```

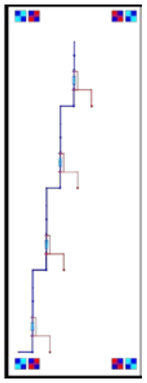


Figure 2: Importing in MINT. The Figure shows MINT code snippets and their corresponding outputs generated by Fluigi on their right. The first code snippet describes the device *net_test* and the second describes *module_test* which imports, instantiates the device *net_test* and makes connections between the instances.

[3] S. K. W. Dertinger, D. T. Chiu, N. L. Jeon, and G. M. Whitesides. Generation of Gradients Having Complex Shapes Using Microfluidic Networks. *Analytical Chemistry*, 73(6):1240–1246, Mar. 2001.

[4] M. S. Ferry, I. A. Razinkov, and J. Hasty. Chapter fourteen - Microfluidics for Synthetic Biology: From Design to Execution. In C. Voigt, editor, *Methods in Enzymology*, volume 497 of *Synthetic Biology, Part A*, pages 295–372. Academic Press, 2011.

[5] F. Hadlock. A shortest path algorithm for grid graphs. *Networks*, 7(4):323–334, 1977.

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

[7] H. Huang and D. Densmore. Fluigi: Microfluidic device synthesis for synthetic biology. *J. Emerg. Technol. Comput. Syst. Special Issue on Synthetic Biology*, 11(3):26:1–26:19, Dec. 2014.

[8] 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.

[9] A. Prindle, P. Samayoa, I. Razinkov, T. Danino, L. S. Tsimring, and J. Hasty. A sensing array of radically coupled genetic /'biopixels/'. *Nature*, 481(7379):39–44, Jan. 2012.

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

[11] T. M. Squires and S. R. Quake. Microfluidics: Fluid physics at the nanoliter scale. *Reviews of Modern Physics*, 77(3):977–1026, Oct. 2005.

[12] T. Thorsen, S. J. Maerkl, and S. R. Quake. Microfluidic Large-Scale Integration. *Science*, 298(5593):580–584, Oct. 2002.

[13] J. P. Urbanski, W. Thies, C. Rhodes, S. Amarasinghe, and T. Thorsen. Digital microfluidics using soft lithography. *Lab on a Chip*, 6(1):96–104, Dec. 2006.

Modular Assembly of an Electronically Integrated Genetic Circuit Library

Luis Ortiz
MCBB Program
36 Cummington St
Boston, MA 02215
+1 (407) 738-9412
Lortiz15@bu.edu

Thomas Costa
Biomedical Engineering
44 Cummington St
Boston, MA 02215
+1 (617) 358-6238
tjcosta@bu.edu

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

ABSTRACT

Genetic circuits have demonstrated their ability to sense environmental stimuli both specifically[1], and across a wide dynamic range[2], and researchers have leveraged these systems to engineer increasingly complex logic devices into biological systems[3,4]. Coupled with advances in modular DNA assembly techniques like Golden Gate, Isothermal Assembly, and MoClo[5], the high-throughput assembly of these devices allows researchers to rationally design, build, and test, a much larger portion of the available design space more rapidly.

However, these genetic circuits are often limited by the availability of fully orthogonal parts[6], and the requirement for separate genetic devices to be spatially constrained in order to relay information. Here we demonstrate the integration of genetic circuits with electronics in an effort to address these limitations. By using a library of modular genetic devices which cause a drop in pH upon the sensing of a chemical stimulus, coupled with Raspberry Pi computers and Twitter as a signal intermediate, we show that these genetic devices can respond to a number of stimuli and have the output of this logic actuate processes in a distinct genetic circuit in a remote location. This relieves the constraint of orthogonality, since genetic devices can be separated into different samples, and allows spatial independence through the use of a computer network to mediate signal propagation between distinct genetic circuits.

Keywords

Synthetic biology; biosensors; MoClo; electronic integration; automated assembly

1. INTRODUCTION

While a considerable effort has been applied to the mining and generation of new orthogonal genetic parts[6], segregation of genetic circuits into distinct, separate populations is an alternative solution which can alleviate the necessity for parts which do not interfere with one another. Unfortunately, the propagation of input and output signals in these genetic devices, most often biomolecules (bacterial quorum sensing molecules, arabinose, etc.), imposes a spatial restriction on these devices as well. Genetic elements encoding logic layers must either be within the same cell or in distinct cells which can move to, or excrete signals for downstream cells to receive, limiting how spatially separated these circuits can be. By comparison, integration with electronic devices, which are not bounded by the same limitations, offers an ideal solution to some of the problems facing performing genetic logic in biological systems.

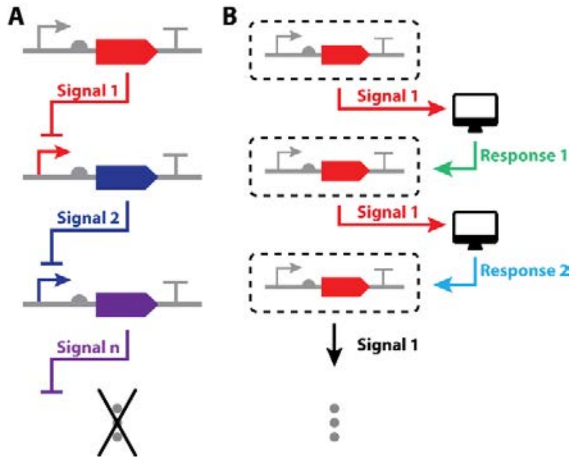


Figure 1: Integrating genetic circuits with electronics alleviates the need for part orthogonality. (A) As the number of inputs increases for a genetic circuit, the number of orthogonal signals must also increase, however this does not scale due to the lack of a large number of orthogonal genetic parts. (B) By comparison, integrating genetic circuits with electronics allows part reuse and removes the constraint of orthogonality, allowing the same biological signal to actuate different responses through electronic intermediates.

2. GENETIC CIRCUIT CONSTRUCTION & ELECTRONIC INTEGRATION

Using Modular Cloning[7] (MoClo), we were able to rapidly build a library of genetic circuits where the expression of β-galactosidase is controlled by various inducible promoters. These circuits operate as AND logic gates where β-galactosidase expression drives the breakdown of lactose in the culture medium, lowering pH only when both the correct inducer and lactose are present. We chose pH as a measureable output for these circuits due to the ease of measurement and integration with inexpensive electronics.

The pH of cultures are continuously monitored using a standard glass electrode pH sensor connected to a Raspberry Pi, which sends a tweet to the @TweeColi Twitter page upon sensing the pH drop below a predesignated threshold. The tweet serves as an output from the first genetic circuit, and verifies that both signals were sensed (lactose & inducer molecule to activate genetic circuit). In a distinct location a second Raspberry Pi listens for this tweet, and once the tweet is registered, a script actuates a syringe pump which adds homoserine lactone (HSL) to a bacterial culture. This second bacterial culture carries a HSL-responsive genetic circuit which produces the fluorescent protein YFP upon addition of HSL. This is a specific example of a more extensible paradigm where biology is actuated externally, performs sensing, responds, and the electronic measurement of that response actuates new biology downstream.

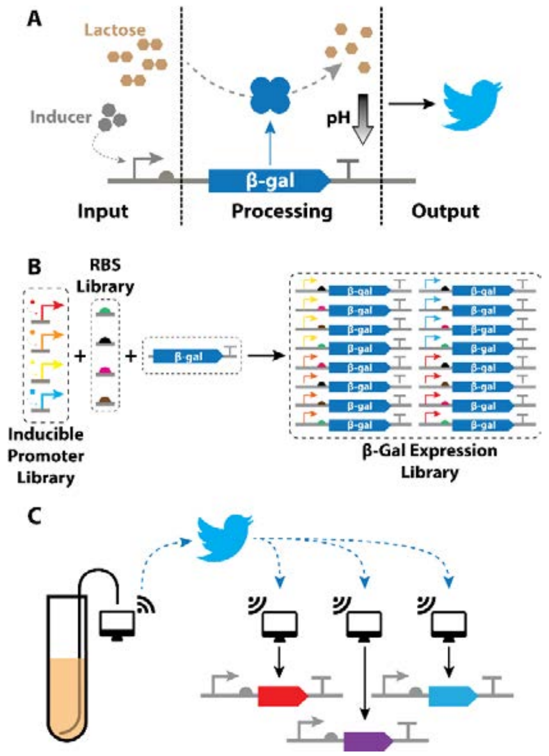


Figure 2: Electronic integration of genetic circuits. (A) An inducible promoter drives the expression of beta-galactosidase which metabolizes lactose, causing a drop in pH which is measured electronically. (B) A modular library of beta-galactosidase expressing genetic circuits was rapidly generated via MoClo, using a library of inducible promoters and ribosomal binding sites (RBS). (C) pH of cultures are monitored continuously, sending out a tweet only when the pH drops below a specified value. This sensing is then distributed to remote systems via Twitter to actuate changes in new genetic circuits.

3. CHALLENGES

There are a number of challenges associated with the proposed research. This system, while amenable to changes in the types of outputs generated by genetic circuits (fluorescence/luminescence vs. pH), must still produce outputs that are easily detectable with electronics. Genetic circuits must also be tuned to produce outputs within the detectable range of the electronics used to sense them. Furthermore, genetic circuits, which generally operate on the order of minutes to hours, are much slower than electronic devices which can respond near instantaneously. Lastly, the application of these integrated genetic circuits is limited to the sensing methodology of the electronics. For instance, pH modulation and sensing is much easier to execute *in-vitro* as opposed to *in-vivo*.

That being said, the integration of genetic circuits with electronics combines the strength of biological sensing with the flexibility and connectivity of computer systems, allowing for genetic part reuse and spatially agnostic connectivity between genetic circuits. Leveraging the power of computers also extends beyond direct integration with genetic circuits. Genetic part assignments can be automated via software, based on performance data, and used to tune genetic device outputs to match the measureable dynamic range of the electronics used to measure them. Integration with electronics, especially computers, also allows for real-time data capture and analysis which can serve as the basis for automated and dynamic control of biological systems.

4. ACKNOWLEDGMENTS

We would like to thank Sonya Iverson and Alan Pacheco for preliminary work on pH modulating genetic circuits. We also thank Ryan Silva for technical expertise in establishing the Raspberry Pi and Twitter integration for these experiments.

5. REFERENCES

[1] Wang, B., Barahona, M. & Buck, M. A modular cell-based biosensor using engineered genetic logic circuits to detect and integrate multiple environmental signals. *Biosens Bioelectron.* **40** (1), 368-376, doi:10.1016/j.bios.2012.08.011, (2013).

[2] Rogers, J. K. *et al.* Synthetic biosensors for precise gene control and real-time monitoring of metabolites. *Nucleic Acids Res.* **43** (15), 7648-7660, doi:10.1093/nar/gkv616, (2015).

[3] Tamsir, A., Tabor, J. J. & Voigt, C. A. Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. *Nature.* **469** (7329), 212-215, doi:10.1038/nature09565, (2011).

[4] Moon, T. S., Lou, C., Tamsir, A., Stanton, B. C. & Voigt, C. A. Genetic programs constructed from layered logic gates in single cells. *Nature.* **491** (7423), 249-253, doi:10.1038/nature11516, (2012).

[5] Casini, A., Storch, M., Baldwin, G. S. & Ellis, T. Bricks and blueprints: methods and standards for DNA assembly. *Nat Rev Mol Cell Biol.* **16** (9), 568-576, doi:10.1038/nrm4014, (2015).

[6] Stanton, B. C. *et al.* Genomic mining of prokaryotic repressors for orthogonal logic gates. *Nat Chem Biol.* **10** (2), 99-105, doi:10.1038/nchembio.1411, (2014).

[7] Weber, E., Engler, C., Gruetzner, R., Werner, S. & Marillonnet, S. A modular cloning system for standardized assembly of multigene constructs. *PLoS One.* **6** (2), e16765, doi:10.1371/journal.pone.0016765, (2011).

Modular composition of synthetic biology designs using rule-based models

Göksel Mısırlı[*]
ICOS, School of Computing Science, Newcastle University
goksel.misirli@ncl.ac.uk

William Waites[*]
School of Informatics, University of Edinburgh
wwaites@tardis.ed.ac.uk

Matteo Cavaliere
School of Informatics, University of Edinburgh
mcavali2@staffmail.ed.ac.uk

Paolo Zuliani
ICOS, School of Computing Science, Newcastle University
paolo.zuliani@ncl.ac.uk

Vincent Danos[†]
School of Informatics, University of Edinburgh
vdanos@inf.ed.ac.uk

Anil Wipat[†]
ICOS, School of Computing Science, Newcastle University
anil.wipat@ncl.ac.uk

1. INTRODUCTION

Synthetic biology aims to extend classical genetic engineering by applying principles such as modularity, standardization and abstraction to design complex biological systems and even entire genomes. This process is challenging: biological systems have very large design spaces. As the complexity of engineered systems increases, computational approaches become ever more important to identify biologically feasible solutions. Computational modelling and simulation is crucial to design and verify genetic circuits in a efficient manner. The availability of modular and reusable models is desirable to automate this process. Such models can be composed to create larger models to specify the behaviour of genetic circuits *in vivo* or *in vitro*.

We previously developed an approach termed Standard Virtual Parts (SVPs) to represent models of DNA-based biological parts such as promoters, coding sequences (CDSs) and ribosome binding sites (RBSs) [3]. These models are reusable and annotated with machine-readable information to assist in automated composition. Models are currently implemented using a reaction-based formalism where much of the information required for composition is tightly coupled. Rule-based formalisms on the other hand offer inherent modularity and allow decoupling part models. This approach was previously demonstrated in the Kappa Bio-Brick Framework (KBBF) [4], which helps in the creation of rule-based models of genetic circuits. This framework provides rules that describe the transcription and translation of DNA parts and a modeller provides rules for the interactions of gene products such as transcription factors (TFs). A series of parameters is also provided by the modeller.

Here, we present our work based on SVPs and KBBF aimed at automating the design of genetic circuits using rule-based models (RBMs). Extending KBBF, we have defined abstract templates that can be used to instantiate rules associated with *cis* and *trans* interactions. We then sketch how to annotate rule-based models, using an annotation framework [2] previously developed, in order to automate composition for RBMs and produce executable composite rule-based models. We also outline protocols to incorporate model repositories into the design process. The overall described workflow bridges together well studied topics and suggests, in this way, a feasible protocol (currently, under development by the authors) for the modular design and composition of synthetic circuits.

2. RULE-BASED MODEL TEMPLATES

Templates were created by extending those from KBBF based on a general (and abstract) notion of sliding, docking (binding) and fall off [4]. Instantiating such a template yields specific rules modelling transcriptional and translational processes. For instance, templates can be then used to instantiate rules for:

- Binding: RNA polymerases (RNAPs) to promoter, ribosomes to RBSs and TFs to DNA.
- Sliding: RNAPs moving on DNA (transcription) and ribosomes moving on RNA (translation).
- Fall off: RNAPs or TFs from DNA, or ribosomes from RNA.

Although a single template can be used to instantiate multiple rules, their rates need not be uniform and can depend on the DNA parts present in the instance. For example, lower rates of a fall off instance can be used to model leakiness of stop codons or terminators.

Templates include the possibility to instantiate rules for modelling gene fusion. In fact, when stop codons are not included, ribosomes can keep sliding (Fig. 1) and produce protein chains (for simplicity, we assume that each protein in a chain retains its interactions). We also explicitly include the possibility to model the degradation of protein and of mRNA chains (extending KBBF) with the addition of templates for the binding of protease to proteins and the degradation of same. In this way, one can explicitly model induced degradation by increasing the binding rate of protease to chaperon proteins in a chain.

Docking templates are defined for arbitrary promoter architectures where activator or repressor sites (binding sequences) can be placed anywhere in a genetic circuit. These

[*]These authors contributed equally to this work
[†]To whom correspondance should be addressed

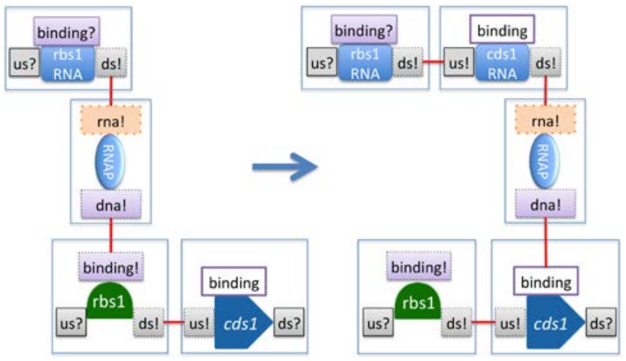


Figure 1: A transcriptional sliding rule. The RNAP starts sliding through DNA (left hand side). As a result, the mRNA chain is extended by transcribing the CDS (right hand side). Outer boxes include agents (rbs1, cds1, rbs1RNA, cds1RNA, RNAP) and their corresponding sites (us, ds, binding, rna, dna), and lines represent agent connections.

templates yield rules that take into account complex regulatory effects of TFs (e.g., Fig. 2).

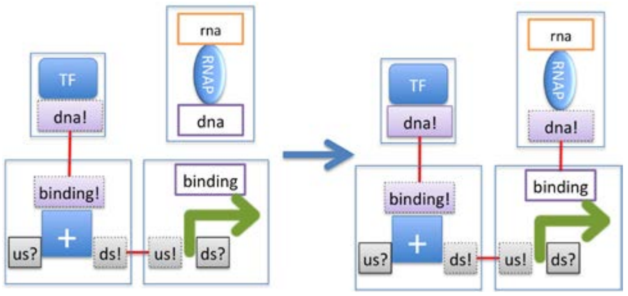


Figure 2: A rule showing the binding of a RNAP to a promoter, when a TF (which may be part of a larger chain) is bound to an upstream binding sequence.

3. MODEL COMPOSITION

Using these templates, we defined rule-based models of basic biological parts[1]. These models can be associated with quantitative parameters to create particular parts models, which can then be merged into executable models. This process can be automated using the information in the models, either available in the specific rule-based syntax or in machine-accessible annotations.

The standard templates use the `DNA` and `P` agent definitions for all DNA and protein entities respectively. A single molecule is distinguished from others using a corresponding identifier for the internal state of a `part` site. These agent definitions increase the reusability and modularity of rules. Models composed of these rules can then be joined together for given genetic circuits.

The specification for a genetic circuit can be given using the Synthetic Biology Open Language (SBOL) [1] terms. A `ComponentDefinition` entity with `subcomponents` can be used to specify a particular genetic circuit, or an abstract definition to represent large solution spaces. Models of parts ideally come from model repositories such as the Virtual Parts Repository. To specify custom rules that may not be in databases — such as for a host — explicit annotated rule-based models can be used. We devised the following protocol in order to access models from different model repositories:

- Mapping agents. The `bqbiol:is` term from the BioModels.net qualifiers is used to specify agents representing the same biological entity.
  ```
  #^ :PlacI bqbiol:is db:BBaR0010.
  ```
- Identifying the model of a part. The user specifies an abstraction, or set of templates, a list of parts and a list of non-part agent identifiers. For each part the client sends this information to the database which responds with and annotated Kappa file containing the corresponding rules.
- Communicating with repositories. The simplest option is just to use the same annotation mechanism and specify a source within the circuit for each part:
  ```
  #^ :MyCircuit rbc:sources (
  #^      rbc:part ex:prom; rbc:source <...>
  #^ ), ...
  ```
- Merging mapped agents. The Kappa files are then rewritten by a *union-find* operation driven by the `bqbiol:is` annotations followed by renaming and then simply concatenated and (naively) deduplicated.

4. CONCLUSIONS

We have briefly presented our work in progress based on SVPs and KBBF to automate the design of biological systems using RBMs. The proposed workflow is based on the definition of modular templates that can be used to instantiate rules for basic biological parts. Such rules can be annotated leading to a feasible protocol to automate their composition for the scalable modelling of synthetic systems.

5. ACKNOWLEDGMENTS

The Engineering and Physical Sciences Research Council grant EP/J02175X/1 (to A.W., V.D., G.M. and M.C.) and the European Union's Seventh Framework Programme for research, technological development and demonstration grant 320823 (to W.W.).

Additional Authors Ricardo Honorato-Zimmer (School of Informatics, University of Edinburgh)

References

[1] B. Bartley, J. Beal, K. Clancy, G. Misirli, N. Roehner, E. Oberortner, M. Pocock, M. Bissell, C. Madsen, T. Nguyen, Z. Zhang, J. H. Gennari, C. Myers, A. Wipat, and H. Sauro. Synthetic Biology Open Language (SBOL) Version 2.0.0. *J Integr Bioinform*, 12(2):272, 2015.

[2] G. Misirli, M. Cavaliere, W. Waites, M. Pocock, C. Madsen, O. Gilfellon, R. Honorato-Zimmer, P. Zuliani, V. Danos, and A. Wipat. Annotation of rule-based models with formal semantics to enable creation, analysis, reuse and visualisation. *Bioinformatics*, 2015.

[3] G. Misirli, J. Hallinan, and A. Wipat. Composable modular models for synthetic biology. *ACM J Emerg Technol Comput Syst*, 11(3):22:1–22:19, Dec. 2014.

[4] J. Wilson-Kanamori, V. Danos, T. Thomson, and R. Honorato-Zimmer. *Computational Methods in Synthetic Biology*, chapter Kappa Rule-Based Modeling in Synthetic Biology. 2015.

[1]Available at http://github.com/rbm/composition

Owl v2.0 : A Web-Application Workspace for Synthetic Biology

Yury V. Ivanov, Prashant Vaidyanathan, Evan Appleton, Zach Chapasko, Arash Khoshparvar, Douglas Densmore
Department of Electrical and Computer Engineering
Boston University
Boston, MA 02215, USA
yvi1@bu.edu; prash@bu.edu; eapple@bu.edu; ztc@bu.edu; arashk@bu.edu; dougd@bu.edu

Keywords

Owl; Statistical data analysis; datasheets; Bioinformatics; CIDAR; CAD; SBOL; visualization; graphs.

1. INTRODUCTION

Synthetic biology workflows have four major categories of problem-solving: specification, design, construction, and testing. These functions utilize forward engineering approaches, combined with knowledge of Molecular Biology and Bioinformatics, to engineer predefined genetic circuits[4]. Underlining components for these core functions are data. While data storage was addressed with a development of Clotho[6] and data exchange with the Synthetic Biology Open Language (SBOL)[3], data analysis, however, has not been addressed yet.

Owl v1.0.0. uses LaTeX typesetting system to produce PDF datasheets[2]. We are developing Owl v2.0 that aims to introduce data analytics services (statistics, bioinformatics, and machine learning), as well as data visualization services, and provide a user-friendly interface that can be rendered on any mobile device or modern internet browser.

2. IMPLEMENTATION

2.1 Owl Stack

The new Owl v2.0 ecosystem consists of a more comprehensive client-side and server-side architecture (Fig. 1).

2.1.1 Client-side

The client-side (or user web-interface) of Owl is written in HTML, CSS, and JavaScript and thus provides a cross-platform electronic workspace.

2.1.2 Server-side

The server-side uses the MEN-stack framework which consists of **M**ongoDB (noSQL database), **E**xpressJS and **N**ode.js (a server-side JavaScript environment). To perform real-time queries, Owl uses Elasticsearch[1].

2.2 Integration

2.2.1 Interface

At the highest level, Owl provides users with an electronic workspace consisting of a dynamic web-interface (**Step 1**,

[1]https://www.elastic.co/

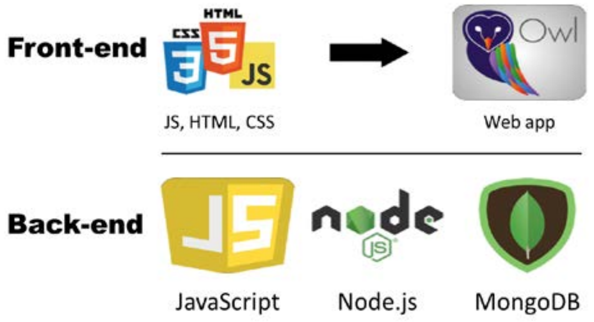


Figure 1: The front end and back end of Owl are build using the most popular web-development tools and languages.

Fig. 2). This web interface can be rendered on any mobile device and modern internet browser.

2.2.2 Data Analytics and Visualization

The new version of Owl is extended to provide data analytics and visualization packages (**Step 2**). These include: (i) LaTeX typesetting system for PDF datasheet generation, (ii) JavaScript packages to create dynamic and publishable-quality static graphs, (iii) statistical packages, (iv) use of shell scripts, and (v) bioinformatics packages.

2.2.3 Data Storage

The Owl v2.0 ecosystem connects to a MongoDB database via Clotho (using a websocket API available in Java and JS), which allows data persistence as well as conversion to formats that are SBOL-v2.0 compliant (**Step 3**).

2.2.4 API

External applications can use Owl v2.0 features using a RESTful API. These APIs will be available in Java, Python and JavaScript(**Step 4**) and can be used by tools like Phoenix: an automated design-build-test-learn tool [1], Cello [4], Double Dutch [5], and DNAplotlib[2].

2.3 Use-Case Scenario

[2]https://github.com/VoigtLab/dnaplotlib

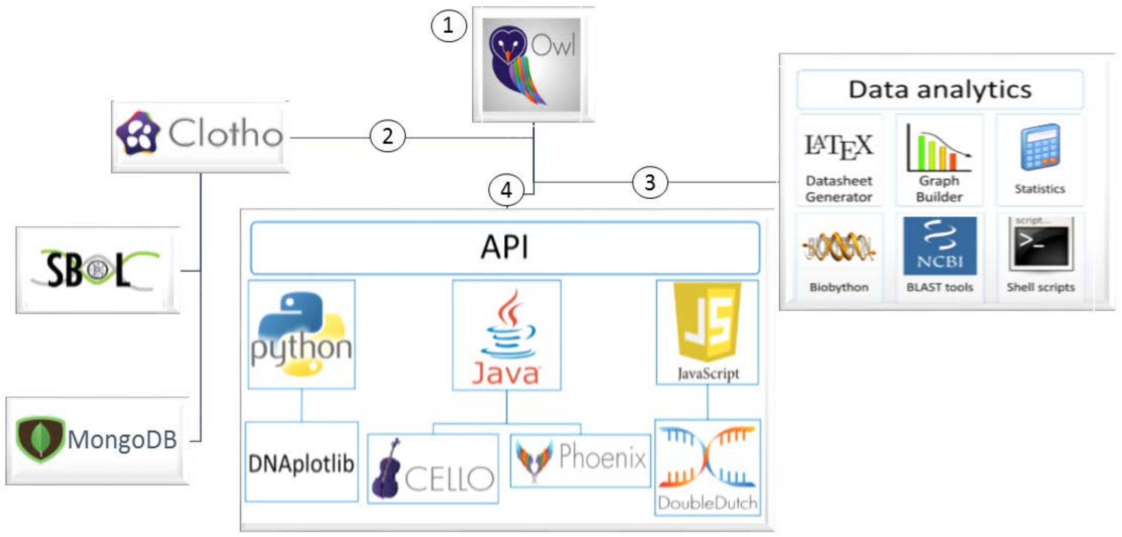


Figure 2: Owl v2.0 ecosystem contains a web application (1) and a persistence database manager Clotho (2). Data analytics packages are performed on a server-side (3). Owl v2.0 is connected to other software via RESTful APIs (4).

2.3.1 *Authentication*

In order to use Owl, a user first has to create a profile (**Step 1**, Fig. 2); this profile can be used with any Clotho-compatible tool. Owl supports OAuth 2.0. secure authentication protocol to login via popular social networks, like Facebook and Google+.

2.3.2 *Create a Project*

An authenticated user can create projects in Owl. For example, the user can search a repository of synthetic parts (*e.g.*, iGEM), save them for that project in an SBOL 2.0 compliant format (**Step 2**), and perform optional Basic Local Alignment Searches (BLAST, **Step 3**) on those parts.

2.3.3 *Design, Build, and Test*

Design-build-test tools are connected to Owl through RESTful APIs (**Step 4**). Through Owl, a user can send project-associated parts, and other files (if necessary) to perform various tasks provided by those tools. For example, Phoenix designs, builds, and tests, while Double Dutch optimizes gene expression. Some projects may require execution of shell-scripts, bioinformatic scripts, or statistical analysis modules, all of which are executed at the server side of Owl (**Step 3**) using command line execution.

2.3.4 *Visualization*

Assembled designs are visualized in Owl using DNAplotlib (**Step 4**) or Pigeon (not shown). Graphs are generated by the server-side of Owl using JavaScript libraries for static and dynamic graphs. Owl can also generate PDF datasheets (**Step 3**), documenting timestamps, parts used, project ID, title, graphs, protocols, and more using pdfLatex.

3. CONTRIBUTIONS

Creation of Owl v2.0 was motivated by the need to efficiently manage the vast amount of data gathered in Synthetic Biology. We are developing an electronic workspace that allows a user to access design-build-test tools, analyze, visualize, save, and transfer the data in a standardized way. Since Owl is connected to Clotho and provides APIs for programs written in Python, Java, and JavaScript, a user can access data from any tool that is connected to Owl v2.0 's ecosystem.

4. REFERENCES

[1] E. Appleton, E. Oberortner, P. Vaidyanathan, Z. Chapasko, Y. Pacheco, Alan andAgarwal, N. Roehner, T. Haddock, and D. Densmore. Phoenix: An automated design-build. poster presented at the International Workshop on Bio-Design Automation (IWBDA), August 2015.

[2] 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.

[3] B. Bartley, J. Beal, K. Clancy, G. Misirli, N. Roehner, E. Oberortner, M. Pocock, M. Bissell, C. Madsen, T. Nguyen, et al. Synthetic biology open language (sbol) version 2.0. 0. *0 J. Int. Bioinformatics*, 12:272, 2015.

[4] A. A. 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):aac7341, 2016.

[5] N. Roehner and D. Densmore. Double dutch: A tool for designing libraries of variant metabolic pathways. poster presented at the SynBERC 2014 Spring Retreat, March 2015.

[6] B. Xia, S. Bhatia, B. Bubenheim, M. Dadgar, D. Densmore, and J. C. Anderson. Developer's and user's guide to Clotho v2.0 A software platform for the creation of synthetic biological systems. *Meth. Enzymol.*, 498:97–135, 2011.

Phagebook Alpha
A Software Environment for Social Synthetic Biology

Johan Ospina[1,*], Inna Turshudzhyan[1,*], Allison Durkan[1,*], Kristel Tan[2,*], Anna Goncharova[2,*], Prashant Vaidyanathan[1], Nicholas Roehner[1] and Douglas Densmore[1]

[1]Department of Electrical & Computer Engineering, Boston University, Boston, MA
[2]Department of Computer Science, Boston University, Boston, MA
[*]These authors contributed equally to this work
{johanos,innatur,azulad7,ktan,agonchar,prash,nroehner,dougd}@bu.edu

1. INTRODUCTION

Synthetic biology as a field has grown considerably in recent years. With the increasing use of Biodesign automation(BDA) to augment the process of engineering reliable genetic systems, various research groups have introduced and developed applications that focus on specific sub-problems within the main phases of BDA: Specify, Design, Build, Test, Verify, and Learn. To achieve forward engineering of living systems to create novel solutions for many of society's grand challenges in human health, materials, energy, and environmental remediation, a highly cross-disciplinary team with varying skill sets and areas of expertise is required. It is very important to allow members of these cross-disciplinary teams to communicate across projects, tools and software environments while maintaining a sense of community. To facilitate collaboration on a large scale, we have developed a social networking platform: Phagebook[1], which uses the Clotho 3.0 ecosystem [1] to connect users and synthetic biology applications to one another. Phagebook lets users create a profile, connect with other researchers, and and publications, and post progress updates for projects, and manage lab inventory and ordering.

Clotho 3.0 was created as a database management system to provide an ecosystem to store data in a standardized format. The default data model in Clotho 3.0 provides a community-designed model, incorporates user feedback and adopts community standards such as SBOL. Clotho 3.0 is more flexible since programmers can use a Clotho instance as a service called from their programs, or write apps designed to run within Clotho. Clotho also provides a convenient library (in Java and JavaScript) for working directly with Clotho default model objects, without the need to manually manage each command. Such an ecosystem would be ideal to store data for biological parts, features, sequences or any data that can be stored in a JavaScript Object Notation (JSON) format.

Phagebook utilizes built-in Clotho features to store and manage data. Additionally Phagebook exchanges information with various Clotho applications. This integration allows users to easily incorporate Phagebook into existing aspects of their research as highlighted in Fig 1.

[1]Phagebook is currently in Alpha testing and can be accessed at https://phagebook.org

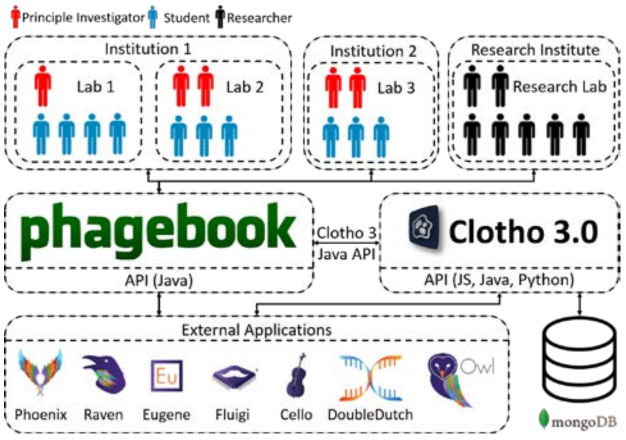


Figure 1: Phagebook ecosystem. Users can have various roles and can use Phagebook to connect to one another. Phagebook uses the Clotho ecosystem to connect users to data stored in Clotho Apps like Cello and Phoenix.

2. FEATURES

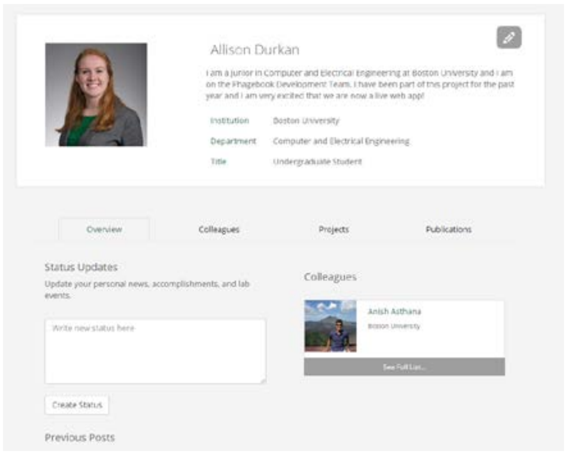
Phagebook is mainly divided into 3 main features: Personal, Project Management, and Lab inventory management/Ordering portal.

2.1 Personal

Phagebook uses Clotho 3.0's authentication and authorization features to create a new user. Users have to enter their email ids while registering for a new account. Phagebook then sends an activation link which can activate a user and let them log in to Phagebook. All user information and credentials are stored and secured by Clotho 3.0 .

2.1.1 *Personal information*

Phagebook users can enter relevant personal information like: first and last name, organization (institution, research institution or industry) affiliation, department, position and role (principle investigator(PI), graduate student, post-doctoral researcher, researcher, etc). This is displayed in the user's default home page (Fig 2). Users can also enter status messages and can tag objects in Clotho 3.0 (provided they have read or write access to that object) in their status. For instance, a user can post a status saying that they just created a new Biological Part and can provide a link to the Part's metadata through Clotho.



Allison Durkan

I am a junior in Computer and Electrical Engineering at Boston University and I am on the Phagebook Development Team. I have been part of this project for the past year and I am very excited that we are now a live web app!

Institution	Boston University
Department	Computer and Electrical Engineering
Title	Undergraduate Student

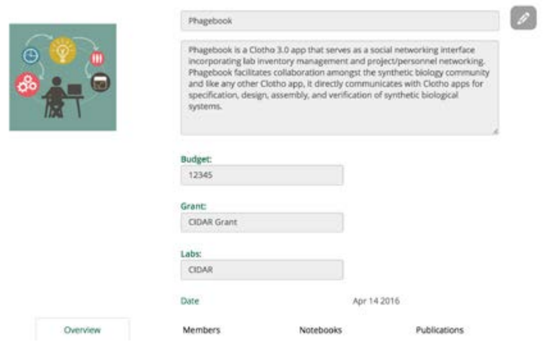
Figure 2: Profile Page. The profile page lets users search and add colleagues, add new statuses and publications. Data security is managed by Clotho 3.0

2.1.2 *Adding colleagues*

Phagebook uses the query feature in Clotho 3.0 to search for other Phagebook users based on entries in the database that belong to the *Person* schema as well as the first or last name specified in the query. Once a request to add a colleague has been submitted, the colleague can go to a portal to manage requests and choose to accept or decline the request from the user.

2.1.3 *Adding publications*

Phagebook uses Pubmed's ESummary API and the Crossref API to get a user's publication from a Pubmed id or a Digital object identifier(DOI) code respectively. Users can also add custom publications to their list of publications These publications are stored in Clotho 3.0 and added to the user's list of publication. This enables multiple users who are co-authors on the same publication to be linked to the same publication.



Phagebook

Phagebook is a Clotho 3.0 app that serves as a social networking interface incorporating lab inventory management and project/personnel networking. Phagebook facilitates collaboration amongst the synthetic biology community and like any other Clotho app, it directly communicates with Clotho apps for specification, design, assembly, and verification of synthetic biological systems.

Budget: 12345

Grant: CIDAR Grant

Labs: CIDAR

Date Apr 14 2016

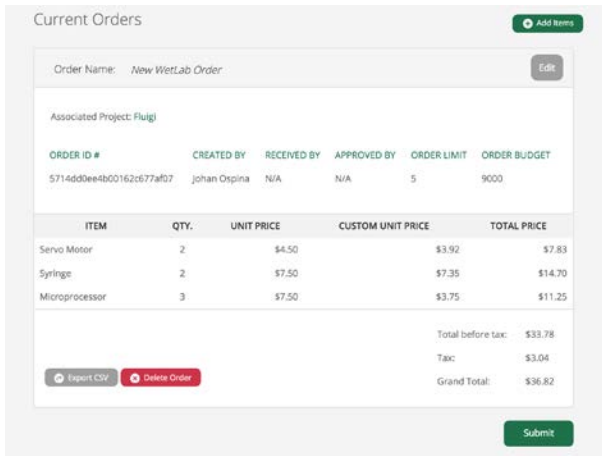
Figure 3: Project Page. PIs can create projects and add collaborators for each project. Project members can add statuses and publications for a project.

2.2 Labs and Project management

Users registered as PIs can create labs, research groups and projects (Fig. 3) associated with each research group. Members of a project can invite colleagues from the same lab or collaborators from other labs, research institutions or the industry to be a part of the project. Members of a project can post status updates to a project (and can choose to send that update out as an email to the entire group or just post it in the Project's statuses). Each member is also granted a lab notebook for each project they are a part of.

2.3 Ordering portal

The ordering portal lets users add Vendors and Products commonly ordered by a lab, to Clotho. Users can then create orders which are tied to a specific project. Users can also specify the maximum number of items that can be placed in each order as well as the maximum budget for each order. The ordering portal lets users search and select products, set the approximate tax to be applied for the entire order as well as set any special vendor discounts that may apply for a particular order. Once an order is submitted for approval, it is sent to the PIs in that project, for approval. Once any PI approves the order (which appears in the manage requests portal), the amount in the order is deducted from the total budget for that order. Users can view all their current (Fig. 4) and past orders (Submitted orders that have been approved, orders that were not approved) and can create customizable order forms that can be downloaded as comma separated values (CSV) files.



Current Orders

Order Name: *New WetLab Order*

Associated Project: Fluigi

ORDER ID #	CREATED BY	RECEIVED BY	APPROVED BY	ORDER LIMIT	ORDER BUDGET
5714dd0ee4b00162c677af07	Johan Ospina	N/A	N/A	5	9000

ITEM	QTY.	UNIT PRICE	CUSTOM UNIT PRICE	TOTAL PRICE
Servo Motor	2	$4.50	$3.92	$7.83
Syringe	2	$7.50	$7.35	$14.70
Microprocessor	3	$7.50	$3.75	$11.25

Total before tax:	$33.78
Tax:	$3.04
Grand Total:	$36.82

Figure 4: Ordering Portal. Users can add/remove products to an order. Products can have custom prices. Orders can specify a budget cap and a limit on the number of products in a single order

3. ACKNOWLEDGEMENTS

The authors would like to thank Kathleen Lewis and Kara Le Fort for their contributions to the project.

4. REFERENCES

[1] S. Paige et al. Clotho 3.0: An improved common framework for synthetic biology computing. poster presented at Synthetic Biology Boston, June 2014.

A powerful, open source, extensible, cloud tool to drive biological design and construction

Edinburgh Genome Foundry
Daniel Rutherford Building G.24
School of Biological Sciences
University of Edinburgh
Edinburgh EH9 3BF

Autodesk Bio/Nano Research
Autodesk Bio/Nano Research
Pier 9, The Embarcadero
San Francisco, CA 94111

Eli Groban
Autodesk Bio/Nano Research
Pier 9, The Embarcadero
San Francisco, CA 94111
1-415-971-7517
Eli.groban@autodesk.com

ABSTRACT

The Bio/Nano Research Group at Autodesk, in collaboration with the Edinburgh Genome Foundry, is releasing an extensible, open source, cloud tool to drive biological design and complex DNA construction. The current software toolbox for genetic design offers solutions that are either relatively good but expensive, or low cost but quite limited. A different approach is necessary as more scientists want to design and build higher numbers of increasingly complex constructs. This paper provides an overview of our high level biological design and construction tool.

CCS Concepts

• Applied computing~Genetics • Information systems~Web applications • Information systems~Open source software • Information systems~RESTful web services • Computing methodologies~Machine learning

Keywords

Biological design, open source, cloud based computer aided design, extensibility

1. INTRODUCTION

The current software toolbox for genetic design offers solutions that are either relatively good but expensive, or low cost but relatively limited [1]. A different approach is necessary as more scientists want to design and build higher numbers of increasingly complex constructs.

2. TOWARDS ABSTRACT DESIGN

2.1 Abstraction of parts from sequence

Designing at the base pair level is quite useful for small, simple genetic programs, but quickly becomes untenable as the length and complexity of these programs expands [2]. In order to

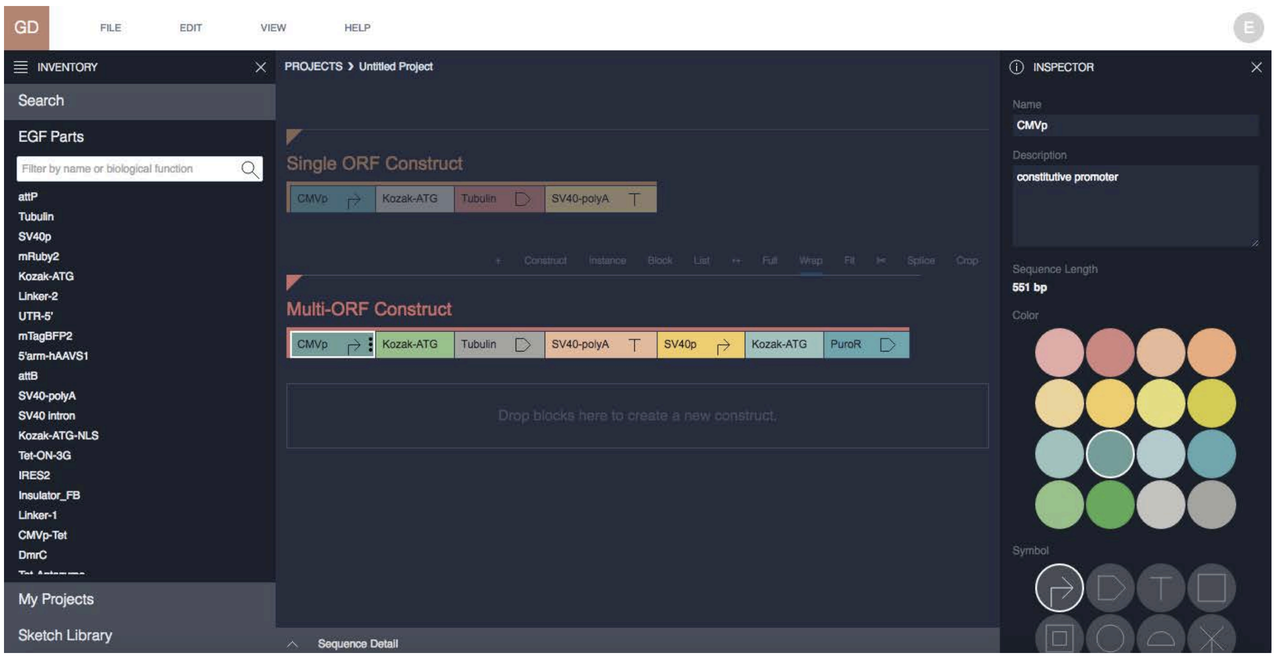


Figure 1. Abstraction of Parts from Sequence. DNA Parts for two different designs shown in a graphical representation

empower scientists to program organisms with higher efficiency and increasing complexity, our application works at higher levels of abstraction than current software packages (Figures 1 and 2). This tool focuses less on the A/C/T/G letters in DNA code and more on high level design.

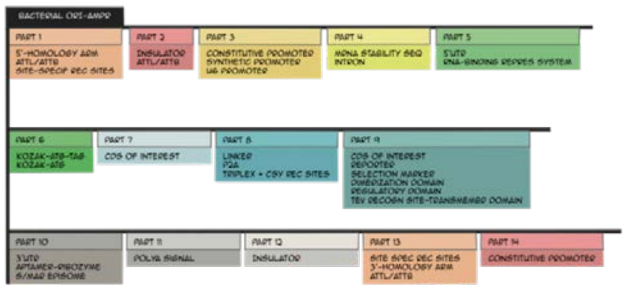


Figure 2. Part layout towards a design intent.

2.2 Abstraction of design from parts

Autodesk is a leader in the space of goal directed design [3]. In this emerging design paradigm, a user does not dictate the design of a structure, but instead communicates its purpose, or design constraints (Figure 3). The design software then generates alternatives and determines the fit against the objective. The user is then presented with a set of designs and their scores.

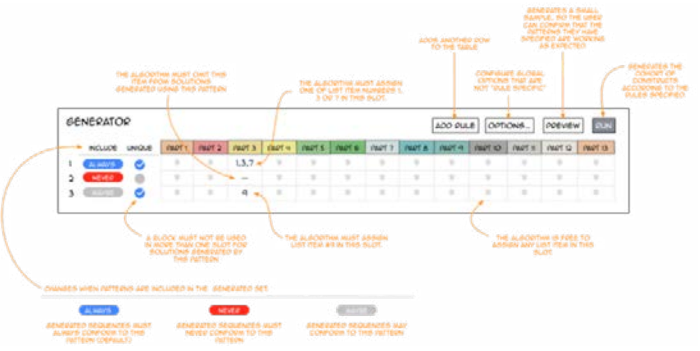

GENERATOR
ADD RULE
OPTIONS...
PREVIEW
RUN
INCLUDE
UNIQUE
PART 1
PART 2
PART 3
PART 4
PART 5
PART 6
PART 7
PART 8
PART 9
PART 10
PART 11
PART 12
PART 13
ALWAYS
NEVER
1,3,7
4
THE ALGORITHM MUST OMIT THIS ITEM FROM SOLUTIONS GENERATED USING THIS PATTERN
THE ALGORITHM MUST ASSIGN ONE OF LIST ITEM NUMBERS 1, 3 OR 7 IN THIS SLOT
ADDS ANOTHER ROW TO THE TABLE
CONFIGURE GLOBAL OPTIONS THAT ARE NOT "RULE SPECIFIC"
GENERATES A SMALL SAMPLE, SO THE USER CAN CONFIRM THAT THE PATTERNS THEY HAVE SPECIFIED ARE WORKING AS EXPECTED
GENERATES THE COHORT OF CONSTRUCTS ACCORDING TO THE RULES SPECIFIED
A BLOCK MUST NOT BE USED IN MORE THAN ONE SLOT FOR SOLUTIONS GENERATED BY THIS PATTERN
THE ALGORITHM MUST ASSIGN LIST ITEM #4 IN THIS SLOT
THE ALGORITHM IS FREE TO ASSIGN ANY LIST ITEM IN THIS SLOT
CHANGES WHEN PATTERNS ARE INCLUDED IN THE GENERATED SET
ALWAYS
NEVER
GENERATED SEQUENCES MUST ALWAYS CONFORM TO THIS PATTERN (DEFAULT)
GENERATED SEQUENCES MUST NEVER CONFORM TO THIS PATTERN
GENERATED SEQUENCES MAY CONFORM TO THIS PATTERN

Figure 3. GUI for rule definition to feed into design specifications.

As we continue to standardize and characterize biological parts, we move closer to being able to apply the same generative design algorithms to the biological space.

3. A link with robust manufacturing

3.1 The gap between academic and industrial DNA fabrication technologies

On the manufacturing side, high throughput, long DNA fabrication technologies exist but are locked behind industry doors. For example, an industrial DNA fabrication unit can produce 2,000 constructs at 20,000 BP/construct with a turnaround time of 3-4 weeks. Academic researchers and small biotech start-ups, however, cannot take advantage of these capabilities.

This leads to a situation where academic researchers and small biotech start-ups are at a disadvantage as they do not have the CAD software driving CAM based foundries required to rapidly engineer their biology.

3.2 Integration with the Edinburgh Genome Foundry

In order to empower scientists to program organisms with higher efficiency and increasing complexity, our application will integrate with DNA Foundries seamlessly to produce those designs. This will enable all scientists, regardless of rank or privilege, access to academic DNA foundries as they come online. For example, the tool will support designing in the application and then placing an order with the Edinburgh Genome Foundry. Unlike previous tools, which simply send a DNA sequence to a fabrication facility, this tool plans to send a bill of materials that encompasses more than just a DNA sequence. Moreover, we hope to connect to the specific foundry API to provide real time feedback as to whether a design is buildable and track the status of the construction process.

3.3 Open source, free, cloud based and extensible

Unlike other high power tools in this space, our software will be open source and freely available to the community. It is also cloud based and built with a plug-in architecture to allow for customization and increasing levels of complexity. We encourage members of the Synthetic Biology community to write plug-in features to keep this as a high powered tool that stays in sync with state of the art knowledge.

In this presentation, we will unveil the tool to the biodesign community, showcase some of its capabilities, discuss future directions, and illustrate how to interact with the tool from a biology and a software development perspective.

4. ACKNOWLEDGMENTS

Our thanks to Autodesk Research for funding this work and the folks at the University of Edinburgh for collaborating on the software and the DNA fabrication process.

5. REFERENCES

[1] Kearse, M. et al. 2012. Geneious Basic: an integrated and extendable desktop software platform for the organization and analysis of sequence data. *Bioinformatics*. 15, 5 (Apr 2012), 1647-1649.

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

[3] https://autodeskresearch.com/projects/dreamcatcher

A Statistical Approach Reveals Designs for the Most Robust Stochastic Gene Oscillators

[Extended Abstract]

Mae L Woods
Department of Cell and Developmental Biology
University College London
mae.woods@ucl.ac.uk

Miriam Leon
Department of Cell and Developmental Biology
University College London
miriam.leon.12@ucl.ac.uk

Ruben Perez-Carrasco
Department of Mathematics
University College London
r.carrasco@ucl.ac.uk

Chris P Barnes
Department of Cell and Developmental Biology
University College London
christopher.barnes@ucl.ac.uk

ABSTRACT

The engineering of transcriptional networks presents many challenges due to the inherent uncertainty in the system structure, changing cellular context, and stochasticity in the governing dynamics. One approach to address these problems is to design and build systems that can function across a range of conditions; that is they are robust to uncertainty in their constituent components. Here we examine the parametric robustness landscape of transcriptional oscillators, which underlie many important processes such as circadian rhythms and the cell cycle, plus also serve as a model for the engineering of complex and emergent phenomena. The central questions that we address are: Can we build genetic oscillators that are more robust than those already constructed? Can we make genetic oscillators arbitrarily robust? These questions are technically challenging due to the large model and parameter spaces that must be efficiently explored. Here we use a measure of robustness that coincides with the Bayesian model evidence, combined with an efficient Monte Carlo method to traverse model space and concentrate on regions of high robustness, which enables the accurate evaluation of the relative robustness of gene network models governed by stochastic dynamics. We provide a number of new design principles for the construction of robust oscillators.

CCS Concepts

•Applied computing → Life and medical sciences;

WOODSTOCK '97 El Paso, Texas USA
© 2016 ACM. ISBN 123-4567-24-567/08/06…$15.00
DOI: 10.475/123_4

Keywords

oscillations; robustness; circuit topology; stochastic processes; sequential Monte Carlo

1. INTRODUCTION

A major challenge facing the progress of synthetic biology is the design and implementation of systems that function in the face of fluctuating cellular environments. While it is widely accepted within the field that the task of constructing and rewiring pathways is tractable, predicting *in silico* how an implemented system will behave *in vivo* under different cellular conditions remains a huge challenge [1]. Robust systems perform their function over a wide range of parameters and external influences. If we could design and build synthetic systems that are robust, then not only would the systems have a higher probability of functioning, but we would also enhance their predictability. Robustness in the context of biological systems has been intensively studied for almost two decades [5].

Biological oscillators have been studied extensively as they form the core of many crucial biological processes such as circadian rhythms and the cell cycle. Oscillating systems also serve as a model for the understanding and engineering of complex and emergent phenomena. Various synthetic systems have been implemented both *in vivo* and *in vitro*. There has been much theoretical study of biological oscillators (for reviews see refs [4, 7, 6]). Despite this body of work, a comprehensive study of the robustness of transcriptional oscillators has not been performed because of the technical challenges it poses. To develop more predictable design and modelling frameworks that can calculate realistic estimates of system properties - including robustness - requires approaches that can handle a large number of parameters, parametric uncertainty and stochastic dynamics. This can be achieved using sequential Monte Carlo methods [2]. Here we extend the Monte Carlo framework to include model space exploration. The novelty in our approach is that the algorithm spends time in models and parameters in direct proportion to their robustness, and thus focuses in on interesting regions of joint model-parameter space. This

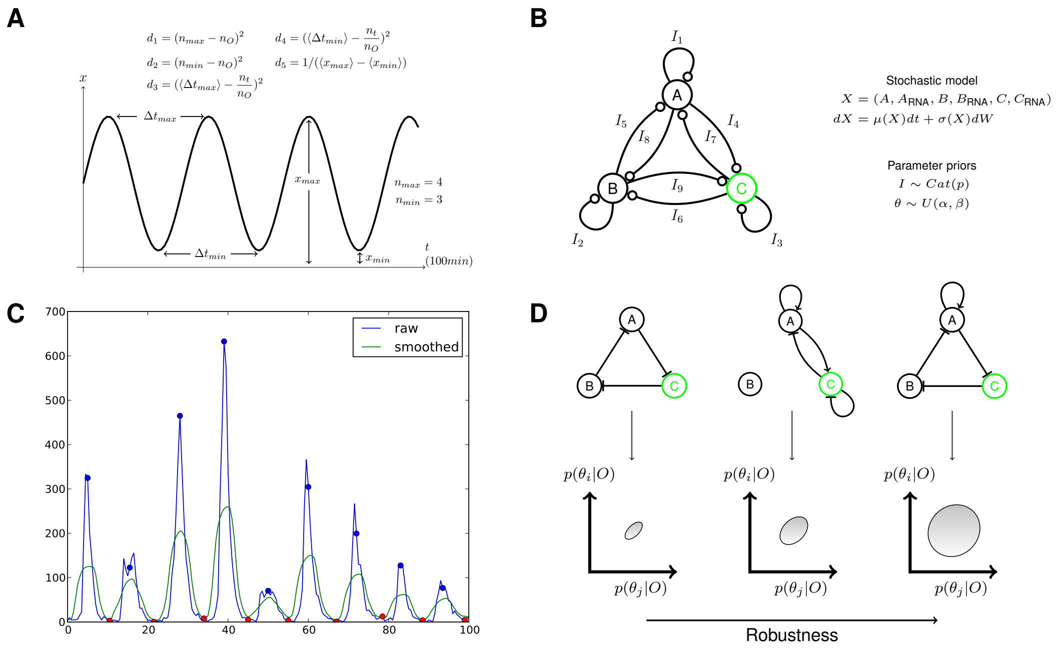

A
$d_1 = (n_{max} - n_O)^2$
$d_2 = (n_{min} - n_O)^2$
$d_3 = (\langle \Delta t_{max} \rangle - \frac{n_t}{n_O})^2$
$d_4 = (\langle \Delta t_{min} \rangle - \frac{n_t}{n_O})^2$
$d_5 = 1/(\langle x_{max} \rangle - \langle x_{min} \rangle)$
x
Δt_{max}
Δt_{min}
x_{max}
x_{min}
$n_{max} = 4$
$n_{min} = 3$
t $(100min)$
B
Stochastic model
$X = (A, A_{RNA}, B, B_{RNA}, C, C_{RNA})$
$dX = \mu(X)dt + \sigma(X)dW$
Parameter priors
$I \sim Cat(p)$
$\theta \sim U(\alpha, \beta)$
C
raw
smoothed
D
$p(\theta_i|O)$
$p(\theta_j|O)$
Robustness

Figure 1: Outline of the method. (A) The objective behaviour is specified through a set of summary statistics and distances on the summaries. (B) Model space is defined, plus a mapping of a graph to a stochastic model. (C) Signal processing is used to extract features from simulations. (D) The output of the algorithm is a set of models that satisfy the objective and maximise robustness.

avoidance of enumeration of all possibilities allows us to address more interesting questions and to assess robustness in a quantitative manner.

2. CONCLUSIONS

We apply our novel framework to investigate the robustness of transcriptional oscillators, an outline of which is given in Figure 1. We examine two main questions regarding the robustness of stochastic transcriptional oscillators: Can we build genetic oscillators that are more robust than those already constructed? Can we make genetic oscillators arbitrarily robust? We find that the most robust two gene oscillators that can provide regular oscillations are of a type already constructed [8]. We also examine the ring oscillator - the repressilator being the classic synthetic implementation [3] - and find that different activation reactions, in addition to positive auto-regulation [9], can increase its robustness. We also determine the topologies that give rise to the most robust three gene systems and find that in general they are more robust than the simple two gene and ring oscillators. The frequency, amplitude and robustness of all transcriptional oscillators, independent of topology, depends strongly on the rates of degradation of the species involved. Finally we find that the number of regulatory interactions increases oscillator robustness up to a plateau, beyond which there is no increase in robustness, which has wide implications for the construction of complex synthetic systems.

3. REFERENCES

[1] A. P. Arkin. A wise consistency: engineering biology for conformity, reliability, predictability. *Current opinion in chemical biology*, Nov. 2013.

[2] C. P. Barnes, D. Silk, X. Sheng, and M. P. H. Stumpf. Bayesian design of synthetic biological systems. *Proceedings of the National Academy of Sciences of the United States of America*, 108(37):15190–15195, Sept. 2011.

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

[4] A. Goldbeter. Computational approaches to cellular rhythms. *Nature*, 420(6912):238–245, Nov. 2002.

[5] H. Kitano. Towards a theory of biological robustness. *Molecular Systems Biology*, 3, Sept. 2007.

[6] P. Lenz and L. Søgaard-Andersen. Temporal and spatial oscillations in bacteria. *Nature reviews Microbiology*, 9(8):565–577, Aug. 2011.

[7] B. Novák and J. J. Tyson. Design principles of biochemical oscillators. *Nature Reviews Molecular Cell Biology*, 9(12):981–991, Dec. 2008.

[8] J. Stricker, S. Cookson, M. R. Bennett, W. H. Mather, L. S. Tsimring, and J. Hasty. A fast, robust and tunable synthetic gene oscillator. *Nature*, 456(7221):516–519, Nov. 2008.

[9] T. Y.-C. Tsai, Y. S. Choi, W. Ma, J. R. Pomerening, C. Tang, and J. E. Ferrell. Robust, tunable biological oscillations from interlinked positive and negative feedback loops. *Science*, 321(5885):126–129, July 2008.

SynBIS – The Synthetic Biology Information System

Dr Matthieu Bultelle
Imperial College, London
Dept of Bioengineering
SW7 2AZ London
m.bultelle97@imperial.ac.uk

Dr Inaki Sainz de Murieta
Imperial College, London
Dept of Bioengineering
SW7 2AZ London
i.sainzdemurieta@imperial.ac.uk

Professor Richard Kitney
Imperial College, London
Dept of Bioengineering
SW7 2AZ London
r.kitney@imperial.ac.uk

ABSTRACT

The article addresses the development of SynBIS, an information system that empowers the multistage characterisation pipeline of biological parts devised by the Centre for Synthetic Biology and Innovation at Imperial College and supports the automatic generation of datasheets and their dissemination. SynBIS supports open data standards and offers direct programmatic access for all analysis, simulation and CAD software tools.

Keywords

Synthetic Biology; Registry; Standards; Characterisation

1. RATIONALE FOR SYNBIS

Systematic rational design — also known as design-build-test cycle — has become central to the transformation of synthetic biology into an engineering discipline. Another common concept in much of engineering is that systems can be rapidly produced from the combination of standardised components. For this to happen in synthetic biology, there is a strong need to build large catalogues of fully characterised bioparts and make them available in public registries, so they can be reused for design [1].

Ideally, the bioparts should be characterised in a wide range of contexts (plasmids, hosts, medium…) to the highest standards - using validated protocols ensuring a high level of reproducibility. Also, all the data collected and generated during the characterisation process, as well as the tools/methods used (e.g protocols and computer code) should be made available in widely-supported machine readable data formats - as is the case in biomedicine (for instance the CDISC initiative [2] for clinical studies). Finally, the catalogues must be available online, support widespread data standards and offer programmatic access.

It is important to stress that only by offering a comprehensive description of the characterisation of a part, rather than a short summary of it, can repositories be tool and application-agnostic.

Some CAD tools may use the processed data as is, others may use the raw data to calibrate their own models and add extra rules to the parts. The experimental portion of the data can also be used for systems biology projects and by modelling tools.

IWBDA2016, August 16–18, 2016, Newcastle,UK.

Although several well-known repositories exist (the iGEM parts registry [3], or the virtual parts repository [4] to name a few), we are not aware of any repository that meets all those requirements. This obvious gap has led to the creation of SynBIS - an information system developed at the Centre for Synthetic Biology and Innovation (CSynBI) at Imperial College to support its multistage characterisation pipeline of biological parts.

2. CHARACTERISATION PIPELINES

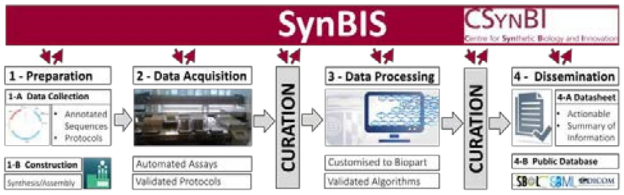


Figure 1. The CSynBI Characterisation Pipeline

SynBIS is the IT spine designed at CSynBI to enable biopart-characterisation efforts on a scale that is difficult-to-impossible for human experimentalists to achieve. First developed with constitutive promoters, it now supports the characterisation of other fundamental bioparts such inducible promoters or RBS.

SynBIS deals with characterisation in term of pipeline. Human intervention is kept to a minimum (curation mainly); automation is used when possible (especially during data acquisition and data analysis). CSynBI pipelines (see Figure 1) comprise three steps.

1. **Data acquisition -** Acquisition has been automated as per [5] – using a set of validated protocols, whose purpose it is to improve reproducibility and increase the throughput. Plate reader and flow cytometry data are typically acquired.
2. **Data Processing -** Experimental data are processed to estimate the relationship between the input and output of the part. Processing is modularised by using libraries of models.
3. **Dissemination -** The information on the biopart is summarised into a datasheet, which is then uploaded to dedicated website (synbis.bg.ic.ac.uk). The datasheet links to all the data related to the characterisation process.

3. SynBIS

3.1 Overview

All SynBIS characterisation information is available for download and use under CC-BY Creative Common license. SynBIS' web interface (synbis.bg.ic.ac.uk) allows users to search the SynBIS catalogue by attributes such as part type, name, and its input-output function (Figure 2-Top), and then access the datasheets of the parts of interest to the users.

Since the influence of the context (chassis, medium, plasmid, reporter, assay protocol and experiments settings) on parts behaviour is poorly understood, SynBIS will host several datasheets (one per context) for the same part.

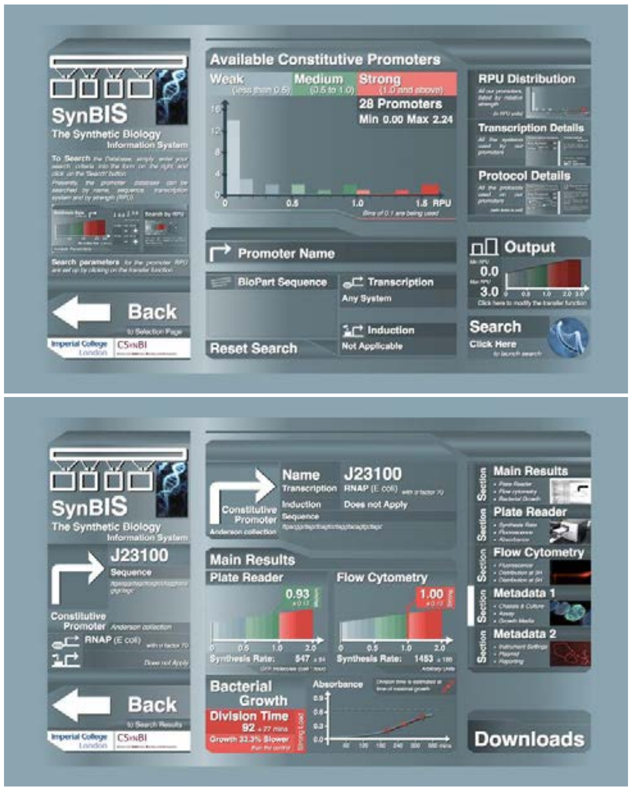


Figure 2. The SynBIS Web Interface (Top: Search Page / Bottom: First Page of the Datasheet of the J23100 Promoter)

3.2 Supporting Popular Data Standards

SynBIS supports established data standards such as SBOL [6] for construct description. To enable the sharing of raw experimental data, SynBIS also supports the first data-acquisition standard in synthetic biology (DICOM-SB [7]). The development of DICOM-SB was an important part of the SynBIS project – such a standard being needed to not only share experimental data, but more widely to support and enable data acquisition at scale.

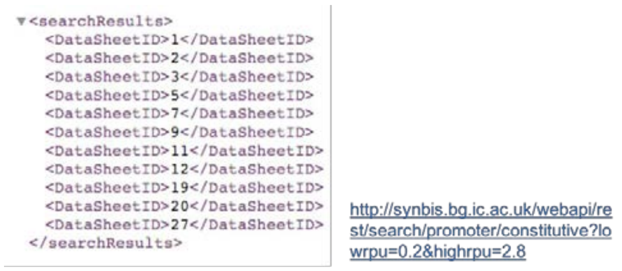
Finally, we have developed a flexible datasheet model – based on the workflow of a canonical characterisation pipeline and serialised in XML – linking a biopart to all the data generated in a characterisation experiment: raw data, analysis results (presenting qualitative and quantitative information describing the part's behaviour) and the metadata required to ensure reproducibility of the experimental and analysis results.

3.3 Programmatic Access

SynBIS implements an API to enable programmatic access from external applications. This way the power users can not only access individual datasheets (and related data) but also run bulk queries (see Figure 3). The API comprises two types of RESTful web services:

Inbound: input of new curated datasheet information into SynBIS (open to SynBIS partners only)
Outbound: retrieval of datasheet information.

- o XML interface: provides the complete description of a datasheet formatted following the SynBIS database structure.
- o SBOL interface: provides the basic information which can be encoded with SBOL-Core.



```
▼<searchResults>
   <DataSheetID>1</DataSheetID>
   <DataSheetID>2</DataSheetID>
   <DataSheetID>3</DataSheetID>
   <DataSheetID>5</DataSheetID>
   <DataSheetID>7</DataSheetID>
   <DataSheetID>9</DataSheetID>
   <DataSheetID>11</DataSheetID>
   <DataSheetID>12</DataSheetID>
   <DataSheetID>19</DataSheetID>
   <DataSheetID>20</DataSheetID>
   <DataSheetID>27</DataSheetID>
 </searchResults>
```

http://synbis.bg.ic.ac.uk/webapi/rest/search/promoter/constitutive?lowrpu=0.2&highrpu=2.8

Figure 3. Search for All Promoters with an RPU between 0.2 and 2.8 with the SynBIS API

4. ACKNOWLEDGMENTS

The development of SynBIS has been supported by the ST-Flow and Flowers consortia.

5. REFERENCES

[1] Kitney, R., and Freemont, P. (2012). Synthetic biology – the state of play. *FEBS Lett.* 586, 2029–2036.

[2] CDISC. http://www.cdisc.org/odm

[3] Registry of Standard Biological Parts. http://parts.igem.org

[4] Virtual Parts Repository. http://sbol.ncl.ac.uk:8081

[5] Hirst, C. D. (2014). Automated BioPart characterisation for synthetic biology. *(PhD thesis, Imperial College, London)*

[6] SBOL – The Synthetic biology Open Language http://sbolstandard.org

[7] Sainz de Murieta, I., Bultelle, M. and Kitney, R.I. (2016). Toward the First Data Acquisition Standard in Synthetic Biology. *ACS synthetic biology*.