ACS SyntheticBiology

pubs.acs.org/synthbio

Technical Note

LOICA: Integrating Models with Data for Genetic Network Design Automation

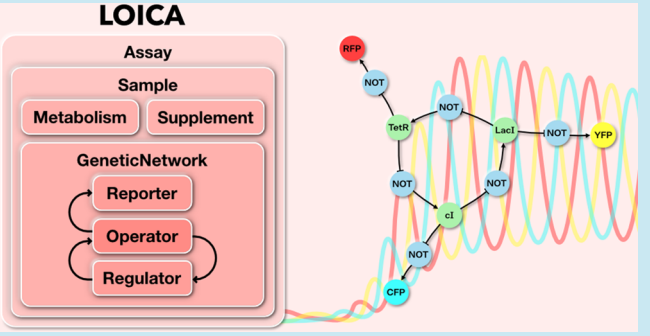
Gonzalo Vidal, Carlos Vidal-Céspedes, and Timothy J. Rudge*

Cite This: *ACS Synth. Biol.* 2022, 11, 1984–1990

Read Online

ACCESS | Metrics & More | Article Recommendations

ABSTRACT: Genetic design automation tools are necessary to expand the scale and complexity of possible synthetic genetic networks. These tools are enabled by abstraction of a hierarchy of standardized components and devices. Abstracted elements must be parametrized from data derived from relevant experiments, and these experiments must be related to the part composition of the abstract components. Here we present Logical Operators for Integrated Cell Algorithms (LOICA), a Python package for designing, modeling, and characterizing genetic networks based on a simple object-oriented design abstraction. LOICA uses classes to represent different biological and experimental components, which generate models through their interactions. These models can be parametrized by direct connection to data contained in Flapjack so that abstracted components of designs can characterize themselves. Models can be simulated using continuous or stochastic methods and the data published and managed using Flapjack. LOICA also outputs SBOL3 descriptions and generates graph representations of genetic network designs.



KEYWORDS: *genetic network, genetic design automation, modeling, characterization, dynamical systems, design abstraction*

INTRODUCTION

Synthetic biology is an interdisciplinary field that mixes life sciences and engineering. From this perspective, living systems are objects to engineer, and a rational way to design them is by modifying their genetic code. This can be done by introducing synthetic DNA that encodes a synthetic regulatory network, also known as a genetic network or genetic circuit. The design–build–test–learn (DBTL) cycle is central to engineering disciplines, and each phase requires appropriate tools, standards, and workflows, which are still in development. Synthetic Biology Open Language (SBOL) is an open standard for the representation of *in silico* biological designs that covers the DBTL cycle and has attracted a community of developers that have produced an ecosystem of software tools.[1−4]

Modeling is key to the DBTL cycle and is essential to the design and learn stages since a model states a well-defined hypothesis about the system operation. Abstraction enables the construction and analysis of models based on components, devices, and systems that can be used to compose genetic networks and derive their DNA sequences. It is the basis for genetic design automation (GDA), which can accelerate and automate the genetic network design process by compiling models into DNA sequences. In order for GDA to proceed in a rational way, the abstract elements of genetic networks must be accessible to characterization, allowing parametrization of models of their operation and interactions.

Functional abstraction of DNA sequences as parts such as transcriptional promoters, ribosome binding sites (RBSs), coding sequences (CDSs), terminators, and other elements has enabled the assembly of relatively small genetic networks.[5−7] However, for large-scale genetic network design, higher-level abstractions are required, as provided by the logic formalism.[8] In this approach, network compositions are abstracted into genetic logic gates that transition between discrete low and high steady-state gene expression levels according to input signals, either external or internal to the network.[9] These genetic logic networks can be designed automatically using Cello,[8] in an analogous way to electronic circuits, on the basis of the required discrete logical truth table, but this specification requires knowledge of the domain-specific programming language Verilog.[10]

Despite the discrete logical design formalism, these genetic networks are dynamical systems and can have autonomous, continuous, non-steady-state dynamics, displaying complex and rich behaviors from bistability to oscillations and even

Received: December 1, 2021
Published: May 4, 2022

https://doi.org/10.1021/acssynbio.1c00603
ACS Synth. Biol. 2022, 11, 1984–1990

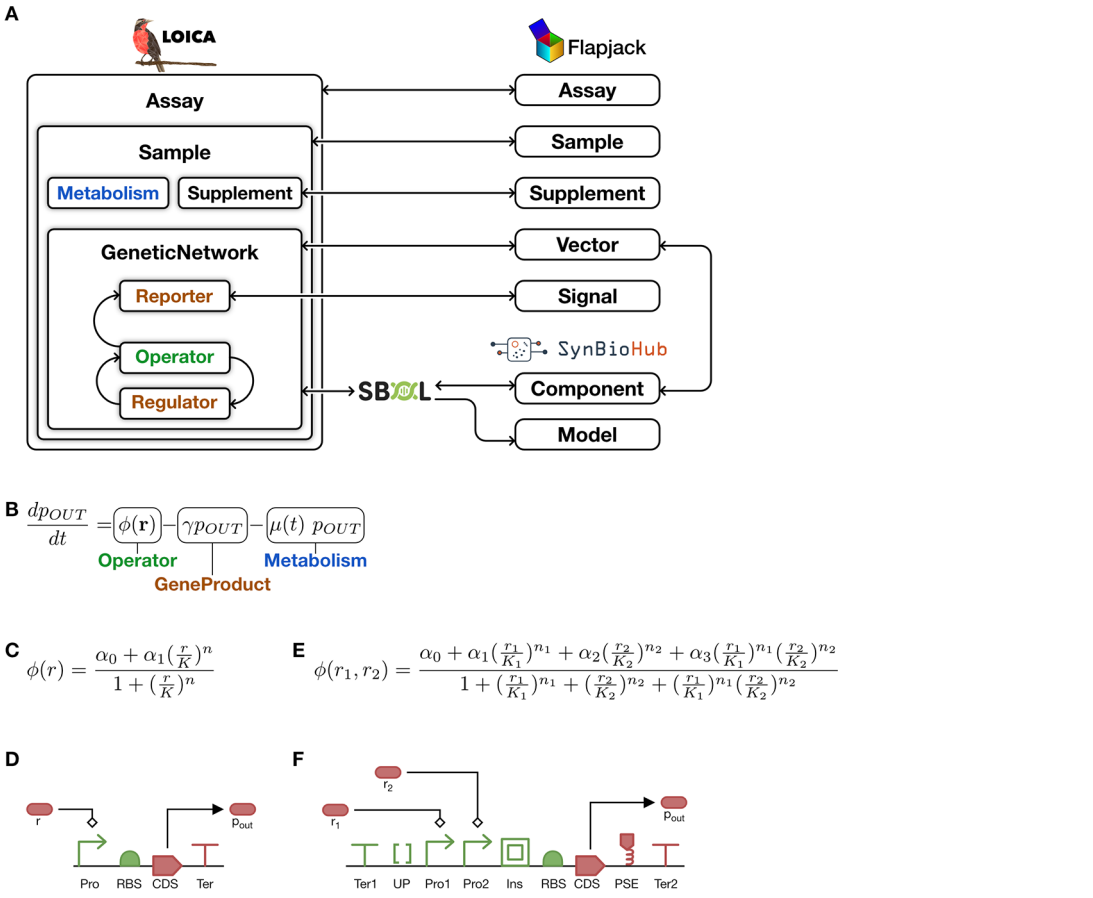


Figure 1. Model generation in LOICA. (A) Diagram of an *Assay* encapsulating a *Sample* that in turn encapsulates *Metabolism*, *Supplement*, and *GeneticNetwork*. In the latter, the *Operator* and *Regulator* interact to generate a model. On the right side the different interactions with the Flapjack and SBOL models are shown. (B) General mathematical model of gene expression of the *GeneProduct* p_{OUT} (*Regulator* or *Reporter*). In the *Operator*, ϕ is a transfer function that maps the concentration of the the input r into the p_{OUT} synthesis rate. In the *GeneProduct*, γ is the degradation rate of p_{OUT}. In *Metabolism*, $\mu(t)$ is the instantaneous growth rate that dilutes p_{OUT}. (C) Transfer function of a one-input *Operator*, where α_0 and α_1 are the nonregulated and regulated synthesis rates, respectively, r is the input concentration, K is the switching concentration, and n is the cooperativity degree. (D) SBOL diagram of a simple transcriptional unit that can instantiate a one-input *Operator* connected to an output *GeneProduct*. (E) Transfer function of a two-input *Operator*, where α_0, α_1, α_2, and α_3 are the nonregulated, promoter 1 regulation, promoter 2 regulation, and joint regulation synthesis rates, respectively, r_1 is the input concentration of *Regulator* 1, r_2 is the input concentration of *Regulator* 2, K_1 is the switching concentration of promoter 1, K_2 is the switching concentration of promoter 2, and n_1 and n_2 are the cooperativity degrees of the *Regulators* with respect to the promoters. (E) SBOL diagram of a complex transcriptional unit that can instantiate a two-input *Operator* connected to an output *GeneProduct*. SBOL diagrams were made using SBOLCanvas.[2] The SynBioHub logo was adapted with permission from the developers[16] and shared under a BSD 2-Clause License. Copyright 2018 SynBioHub Developers. The Flapjack logo was adapted with permission from the developers[13] and shared under an MIT license. Copyright 2022 RudgeLab.

chaos.[4−6] Furthermore, typical operating conditions for engineered networks such as colonies, bioreactors, and microbiomes are time-varying, which can lead to complex behaviors from even simple genetic networks.[11]

To design genetic networks, we therefore require kinetic gene expression data generated at the test phase. These data must be integrated with models to enable characterization of abstracted parts, devices, and systems as well as metadata, including the DNA part composition and sequence, to enable automated design. Thus, there is a need for software design tools that integrate abstract network designs, dynamical models, kinetic gene expression data, DNA part composition, and sequence *via* common exchange standards in a user-friendly and accessible fashion.

Logical Operators for Integrated Cell Algorithms (LOICA) is a tool for the design, modeling, and parametrization of synthetic genetic networks. In contrast to existing genetic network design and modeling tools,[4] rather than composing individual genetic parts, LOICA provides a high-level design abstraction that simplifies the design process by representing networks as combinations of components accessible to parametrization. This parametrization of genetic network models is enabled by direct connection to experimental data *via* Flapjack, which also provides a platform for publishing and sharing simulation results. Furthermore, while LOICA abstracts genetic networks at a higher level, designs can be represented using the latest SBOL3 standard. LOICA is a Python package allowing programmatic design, simulation, parametrization, and analysis of genetic networks. While perhaps not as accessible as a graphical user interface, this approach is more flexible, extensible, and amenable to automation. It can be easily combined with the large ecosystem of biological Python projects[12−15] and uses simple programming concepts that are commonly understood by researchers from a range of disciplines.

https://doi.org/10.1021/acssynbio.1c00603
ACS Synth. Biol. 2022, 11, 1984−1990

A

Create GeneticNetwork
```
osc = GeneticNetwork(vector=fj_vector.id[0])
```

Create and Add Regulators
```
acts = [Regulator(name=f'Act{i+1}', degradation_rate=0.75)
        for i in range(3)]
osc.add_regulators(acts)
rep = Regulator(name='Rep', degradation_rate=0.75)
osc.add_regulator(rep)
```

Create and Add Reporters
```
cfp = Reporter(name='CFP', color='cyan', degradation_rate=0.75,
    signal_id=fj_cfp.id[0])
yfp = Reporter(name='YFP', color='yellow', degradation_rate=0.75,
    signal_id=fj_yfp.id[0])
rfp = Reporter(name='RFP', color='red', degradation_rate=0.75,
    signal_id=fj_rfp.id[0])
osc.add_reporters([cfp,yfp,rfp])
```

Create, Add, and Wire Operators
```
op1 = Hill2([rep,acts[0]], [acts[0],acts[1],cfp],
    alpha=[1,0,100,0], K=[10,10], n=[2,2], name='Op1')
op2 = Hill1(acts[1], [acts[2], yfp],
    alpha=[1,100], K=10, n=2, name='Op2')
op3 = Hill1(acts[2], [rep, rfp],
    alpha=[1,100], K=10, n=2, name='Op3')
osc.add_operators([op1,op2,op3])
```

Draw Graph Representation
```
osc.draw(alpha=1)
```

B

Create Metabolism
```
metab = SimulatedMetabolism(biomass, growth_rate)
```

Create Samples
```
sample = Sample(genetic_network=osc,
                metabolism=metab,
                media=fj_media.id[0],
                strain=fj_strain.id[0])
sample.set_regulator(name='Act2', concentration=5)
```

Create Assay
```
assay = Assay([sample],
              n_measurements=500,
              interval=0.1,
              name=f'Loica Tigges oscillator',
              description='LOICA simulated oscillator',
              biomass_signal_id=fj_biomass_signal.id[0])
assay.run(stochastic=True)
```

Upload Data to Flapjack
```
assay.upload(fj, study.id[0])
```

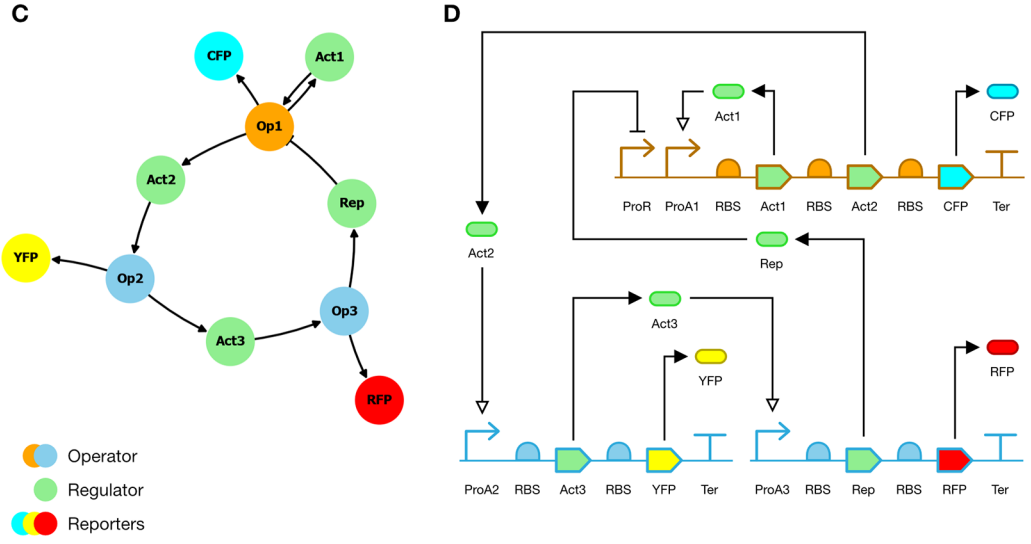


Figure 2. Example of oscillator design, modeling, and analysis in LOICA. (A) Python code that generates an oscillator in LOICA. *GeneticNetwork* construction is the first step, in which the user states all of the objects and their interactions. A graph representation of the model can be drawn in one function call. (B) Next, during *Assay* setup, the user initializes and runs the simulation, and the results can be uploaded to Flapjack. The two-way communication with Flapjack allows data storage and management, enables various analyses to be performed, and allows *Operators* to characterize themselves. (C, D) Comparison of graphical representations for the generated network: (C) LOICA graph output of the oscillator model and respective symbols, which demonstrates how easy it is to visualize generated networks using a higher level of abstraction; (D) SBOL representation of the generated network. It can be seen that as more *Operators* are added, the complexity for visualizing the generated network increases.

RESULTS

LOICA provides a high-level genetic design abstraction using a simple and flexible object-oriented programming approach in Python. LOICA integrates models with experimental data *via* two-way communication with Flapjack, a data management and analysis tool for genetic network characterization.[13] LOICA objects can be represented using SBOL3, enabling direct translation to part composition and DNA sequence as well as connection to repositories such as SynBioHub[16] and the ecosystem of SBOL tools.[1−4] All of the code, examples, and documentation are publicly available at https://github.com/RudgeLab/LOICA.

Design Abstraction for Genetic Networks. The basic objects in LOICA are *Operator* and *GeneProduct*, which may be either a *Regulator* or *Reporter* (Figure 1A). A *Regulator* generates a molecular species that regulates gene expression. A *Reporter* generates a molecular species that provides a measurable signal, such as a fluorescent protein. The *Operator*

https://doi.org/10.1021/acssynbio.1c00603
ACS Synth. Biol. 2022, 11, 1984−1990

maps one or more *Regulator* concentrations to one or more *GeneProduct* synthesis rates. An *Operator* can be implemented in DNA as a combination of promoters and their upstream elements and downstream RBSs, and the *GeneProduct* can be a combination of CDSs, possibly a Protein Stability Element (PSE), and terminators. These objects can be represented by SBOL *Components* containing features that describe individual parts and their DNA sequences. The interactions between the *Operators* and the *Regulators* encode models for genetic network temporal dynamics, which can be simulated with ordinary differential equations (ODEs) or the stochastic simulation algorithm (SSA). The system of ODEs is thus:

$$\frac{d\mathbf{p}}{dt} = \mathbf{\Psi}(\mathbf{r}) - \mathbf{\Gamma}\mathbf{p} - \mu(t)\mathbf{p} \tag{1}$$

$$\mathbf{\Psi}(\mathbf{r}) = \sum_k \mathbf{\Phi}_k(\mathbf{r}) \tag{2}$$

where $\mathbf{p} = (p_0, p_1, ..., p_{N-1})^T$ is the vector of *GeneProducts*, which includes different *Regulators* ($\mathbf{r} = (r_0, r_1, ..., r_{M-1})^T$) and *Reporters* ($\mathbf{s} = (s_0, s_1, ..., s_{N-M-1})^T$). The nonlinear operator $\mathbf{\Psi}$ maps *Regulator* concentrations to *GeneProduct* synthesis rates. $\mathbf{\Gamma}$ is a diagonal matrix of *GeneProduct* degradation rates γ_i, and $\mu(t)$ is the instantaneous growth rate of the cells. Equation 1 shows the overall system, where $\mathbf{\Psi}$ encodes the whole network and consists of a sum of individual LOICA *Operators* $\mathbf{\Phi}_k$ (eq 2).

In the stochastic simulation approach, these *Operators* encode the *GeneProduct* production reactions ($* \rightarrow p_i$) with propensities a_i given by the sum of *Operator* synthesis rates:

$$a_i = \sum_j \mathbf{\Phi}_j(\mathbf{r}) \tag{3}$$

where the sum is over all *Operators* that synthesize *GeneProduct* i. The degradation rate γ_i and growth rate $\mu(t)$ determine the propensities b_i of the *GeneProduct* extinction reactions ($p_i \rightarrow *$):

$$b_i = \gamma_i + \mu(t). \tag{4}$$

To make this abstract framework more concrete, Figure 1C−F shows two *Operators* currently implemented in LOICA. In the first, the output expression rate is a simple Hill function of the input *Regulator* concentration (Figure 1C). Depending on the parameters α_0 and α_1, this *Operator* may encode NOT logic ($\alpha_0 > \alpha_1$) or a Buffer ($\alpha_0 < \alpha_1$). The *Operator* is the set of genetic parts that regulate gene expression (Figure 1D). At its core is a promoter containing repressor or activator binding sites, such that the input *Regulator* either increases or decreases transcription and thus the gene expression rate. The second *Operator* is a two-input function (Figure 1E) that models two promoters in tandem (Figure 1F). Depending on the parameters α_0, α_1, α_2, and α_3, the *Operator* may encode a range of logic, including the NOR operation for $\alpha_0 > \alpha_1, \alpha_2, \alpha_3$. *Operator* instantiations may include terminators to isolate from adjacent transcription, an UP element to insulate from upstream DNA context, or an RBS and an insulator to ensure independence of the promoter and RBS function (Figure 1F).

Logical *Operators* can thus be instantiated as genetic devices that are regulated by input *Regulators* and output *GeneProduct* synthesis rates. As well as the one-input and two-input *Operators* described above, LOICA currently includes signal *Receivers* and constitutive *Sources*. LOICA currently cannot represent networks with nodes with more than two inputs, but all *Operators* can drive multiple output *GeneProducts*. However, it should be noted that LOICA can be used to define an *Operator* as any operation that maps an input *Regulator* concentration to an output synthesis rate, which may correspond to different genetic implementations than those described here. Thus, by expanding the range of *Operator* classes, in the future LOICA could be extended to represent a larger range of genetic networks.

Model Generation and Simulation. Oscillators offer a useful dynamical system case study because they produce continuous sustained oscillations that cannot be captured by ON/OFF logic.[5,7,17−19] We consider a genetic network based on the topology of the mammalian oscillator developed by Tigges *et al.*,[18] consisting of positive and negative feedback loops.

The design is made programmatically (Figure 2A, B) and includes three *Operators*, a two-input Hill function *Operator* (orange node in Figure 2C), which is both negatively and positively regulated, and two one-input Hill function *Operators* (blue nodes in Figure 2C), which are both activated by their respective *Regulators*. Each *Operator* also outputs a fluorescent protein *Reporter* (RFP, YFP, or CFP). The *Reporters* are linked to the Flapjack *Signal* model and together with the *Operators* and *Regulators* are incorporated into a *GeneticNetwork*, linked to the Flapjack *Vector*, which with the *Metabolism* drives the dynamics of the *Sample* (Figure 1A). The *Sample* belongs to an *Assay*, and both are connected to their corresponding Flapjack counterparts (Figure 1A). The code to create the *GeneticNetwork* model is shown in Figure 2A, and the code to define a context for modeling the genetic network is shown in Figure 2B. This approach is used to generate synthetic data using either ODEs (Figure 3A) or the SSA[20] (Figure 3B)—contained in a LOICA *Assay*—from models that can be uploaded to Flapjack. It is then easy to access Flapjack's genetic network characterization tools, data management, and data visualization through its Python package (pyFlapjack) or web interface (http://flapjack.rudge-lab.org).

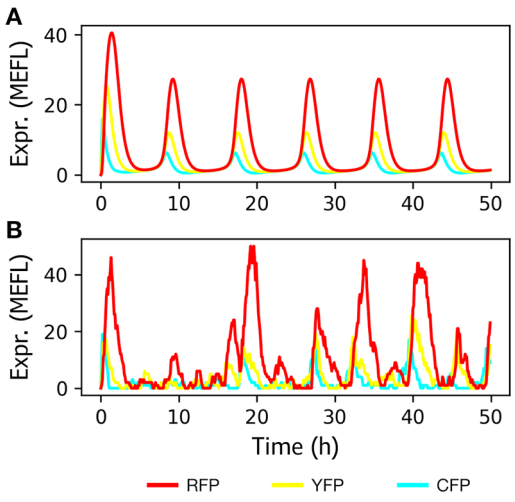


Figure 3. Plots of simulated data using different models. (A) Plot of simulated data from the previous network using the ODE model. (B) Plot of simulated data from the same network using the SSA model.

Encoding Designs in SBOL3. *GeneticNetwork* has a method to generate an SBOL document representation of itself. *GeneticNetwork* maps to a Component that contains

https://doi.org/10.1021/acssynbio.1c00603
ACS Synth. Biol. 2022, 11, 1984−1990

representations of *Operators* and *GeneProducts* and their interactions. The combination of an *Operator* and its *GeneProducts* is represented by SubComponents of a transcriptional unit Component of type DNA with role Engineered Region. Each *GeneProduct* has a genetic production Interaction that generates a molecular species (protein or RNA). If the molecular species is a *Regulator*, then it has an inhibition or stimulation Interaction with one or more *Operators*. Whether it is an inhibition or stimulation depends on the parameters of the *Operator*. A Model is added to the SBOL document and includes the source, language, and framework. To enable the synthesis or assembly of the design, the *Operator* and *GeneProduct* Components should include a Sequence. Constraints are added to ensure correct part order.

LOICA can also generate graph representations of *GeneticNetworks*, which can be used for further analysis of their structure and for visual inspection (see Figure 2C). In comparison with the SBOL Visual representation of the same *GeneticNetwork* (Figure 2D), the LOICA representation abstracts implementation details in favor of providing a simplified design overview. Load and save functions also provide a simple way to store and exchange high-level designs.

Operator Model Parametrization. A description of how to use LOICA to generate and analyze simulated data from models has been provided. However, another workflow scenario goes from data to model parametrization. We demonstrate this process for a two-input *Operator* using simulated data (see the example notebook https://github.com/RudgeLab/LOICA/blob/master/notebooks/Hill2.ipynb)

In order to characterize the two-input *Operator*, three auxiliary genetic networks are required. Each genetic network needs a *Reporter* as a measurable output. In this example, we used LOICA to generate simulated kinetic time-course data. The outputs must be quantified with respect to model parameters, and therefore, in an experimental setup the measurements should be properly calibrated.[21] Two genetic networks composed of an input *Supplement* (*e.g.*, an acyl-homoserine lactone), a Receiver *Operator*, and a *Reporter* must be characterized. The Receiver *Operator* transforms the input *Supplement* concentrations into output expression rates of the *Reporter*, modeled as a Hill function. Each of these genetic networks was simulated over a range of input concentrations, and the data were uploaded to Flapjack. This allowed parametrization of the Receiver *Operators* by fitting of their dynamic models to the simulated data.

Next, the two Receiver *Operators* were composed into a genetic network to drive the *Regulator* inputs of the two-input *Operator*, which in turn drives the output *Reporter*. This genetic network was then simulated over a range of concentrations of input signals, and the data were uploaded to Flapjack. The two-input *Operator* model, combined with the parameters of the two Receivers, was then fitted to the simulated data.

The *Operator* class provides a single function that performs this parametrization process, connecting to Flapjack *via* identifiers of the appropriate genetic networks, automatically combining all of the available data.

DISCUSSION

The DBTL cycle is fundamental in synthetic biology, and thus, various tools have been developed to optimize the different stages. Within this cycle, modeling and characterization are essential for the design and learn stages, for which we have designed LOICA, a tool that connects these processes in an automated fashion. LOICA integrates the design, modeling, and characterization of genetic networks and their components into Python workspaces, providing a powerful and easy-to-understand high-level design abstraction for GDA that is implemented using simple object-oriented programming principles. Importantly, this programming interface does not require specialist or domain-specific knowledge but instead leverages common programming skills, making it accessible but also providing customization capabilities for advanced users.

LOICA genetic network designs are composed of objects that correspond to DNA sequences and are capable of characterizing themselves *via* links to specific experimental data in Flapjack.[13] In this way, designs and their DNA instantiations are associated with online repositories of experimental data, which enable upload and sharing by multiple users or laboratories. This allows division of labor and reuse of experimental data that could be integrated into the workflow of distributed biofoundries. *GeneticNetworks* can automatically build SBOL3 representations of their structure that encapsulate SBOL3 Components defining LOICA objects and incorporating their part composition and DNA sequence. As well as enabling GDA, this provides an easy way to construct SBOL3 documents promoting the use of the standard and providing the capacity to export SBOL3 files, which then can be loaded to different SBOL-based tools. In this way, LOICA not only links models to part composition and DNA sequence, allowing automated assembly[22] or synthesis, but also connects designs to DNA provenance and other metadata contained in repositories such as SynBioHub.[16]

Therefore, LOICA provides simple, easy-to-use, high-level design abstraction for modeling and characterization of genetic networks and their components, which connects to existing synthetic biology standards, tools, and repositories of experimental data to enable GDA. As with any abstraction, simplification comes at the cost of some limitations, which will be addressed in future revisions. The LOICA *Operator*−*Regulator* abstraction assumes a one-step *Regulator* synthesis model. If mRNA dynamics is important to network function, this is clearly not appropriate. Furthermore, since *Regulators* interact only with *Operators*, protein−protein or RNA−RNA interactions are not possible. These limitations may be overcome by implementing more complex models of *Operators* that track the dynamics of internal events such as mRNA production and input *Regulator*−*Regulator* interactions. Also, all *GeneProducts* synthesized by an *Operator* are currently assumed to be expressed at the same rate, which may be overcome by specifying more complex *Operator* models.

Another issue is calibration of experimental measurement data with respect to model parameters and the use of different units. Following best practice, we propose the use of Molecules of Equivalent Fluorescein (MEFL) as units for outputs using GFP or its derivatives.[21] However, since many datasets are not calibrated, future versions of LOICA will incorporate explicit specification of units, allowing for conversion of experimental measurements to values directly comparable to model simulations.

LOICA explicitly has the *Metabolism* as a model component but currently includes limited interaction between this and the *GeneticNetwork*. It has previously been shown that gene expression is modulated by resource competition because of metabolic limitations and in turn has an effect on metabolism, including growth rates.[23−26] These interactions are not

https://doi.org/10.1021/acssynbio.1c00603
ACS Synth. Biol. 2022, 11, 1984−1990

currently included in LOICA models, but the necessary interactions between classes are present, meaning that given suitable models, the interactions between *Metabolism* and *GeneticNetwork* and their components could be encoded in a straightforward manner. Furthermore, the characterization method implemented in LOICA assumes that genetic parts are not affected by their compositional context. Various methods have been developed to reduce such effects to a minimum,[27,28] but they cannot be discounted completely. It may be appropriate to develop a constraint-based specification of such interactions between specific parts, similar to the approach of Cello.[8]

Future work also includes parametrization of stochastic models,[29] which will extend the existing characterization of continuous models. A major improvement will be the implementation of spatiotemporal dynamics of gene expression in multicellular populations, including a connection to CellModeller[30] for individual-based modeling. SBOL integration will continue to be improved to leverage more features and to allow model consistency checking based on known interactions between *Operators* and *Regulators*, such that for example a known repressor cannot be encoded in a model as an activator. Ultimately, we aim to complete and automate the DBTL cycle through an open-source workflow that incorporates LOICA, Flapjack,[13] SynBioHub,[16] and tools powered by SBOL.[1]

■ ASSOCIATED CONTENT

Special Issue Paper

Invited contribution from the 13th International Workshop on Bio-Design Automation.

■ AUTHOR INFORMATION

Corresponding Author

Timothy J. Rudge − *Interdisciplinary Computing and Complex BioSystems (ICOS) Research Group, School of Computing, Newcastle University, Newcastle upon Tyne NE1 7RU, U.K.;* orcid.org/0000-0001-9446-9958; Email: tim.rudge@newcastle.ac.uk

Authors

Gonzalo Vidal − *Institute for Biological and Medical Engineering, Schools of Engineering, Biology, and Medicine, Pontificia Universidad Católica de Chile, Santiago 7820244, Chile; Interdisciplinary Computing and Complex BioSystems (ICOS) Research Group, School of Computing, Newcastle University, Newcastle upon Tyne NE1 7RU, U.K.;* orcid.org/0000-0003-3543-520X

Carlos Vidal-Céspedes − *Institute for Biological and Medical Engineering, Schools of Engineering, Biology, and Medicine, Pontificia Universidad Católica de Chile, Santiago 7820244, Chile;* orcid.org/0000-0003-3867-0395

Complete contact information is available at: https://pubs.acs.org/10.1021/acssynbio.1c00603

Author Contributions

Conceptualization, data curation, investigation, methodology, project administration, resources, supervision, validation: G.V. and T.J.R.; software, formal analysis, visualization, writing—original draft, writing—review editing: G.V., C.V., T.J.R; funding acquisition: T.J.R.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

G.V. was supported by a scholarship from the Institute for Biological and Medical Engineering, Pontificia Universidad Católica de Chile, a scholarship from the School of Computing, Newcastle University, and ANID Fondecyt Regular 11140601. G.V., C.V., and T.J.R. were supported by ANID PIA Anillo ACT192015 and ANID Fondecyt Regular 1211598. The authors thank Chris Myers, Jacob Beal, Tom Mitchell, Guillermo Yañez, and Gabriel Miranda for their helpful comments and discussions.

■ REFERENCES

(1) McLaughlin, J. A.; Beal, J.; Mısırlı, G.; Grünberg, R.; Bartley, B. A.; Scott-Brown, J.; Vaidyanathan, P.; Fontanarrosa, P.; Oberortner, E.; Wipat, A.; et al. The Synthetic Biology Open Language (SBOL) version 3: simplified data exchange for bioengineering. *Front. Bioeng. Biotechnol.* **2020**, *8*, 1009.

(2) Terry, L.; Earl, J.; Thayer, S.; Bridge, S.; Myers, C. J. SBOLCanvas: A Visual Editor for Genetic Designs. *ACS Synth. Biol.* **2021**, *10*, 1792−1796.

(3) Hatch, B.; Meng, L.; Mante, J.; McLaughlin, J. A.; Scott-Brown, J.; Myers, C. J. VisBOL2—Improving Web-Based Visualization for Synthetic Biology Designs. *ACS Synth. Biol.* **2021**, *10*, 2111−2115.

(4) Watanabe, L.; Nguyen, T.; Zhang, M.; Zundel, Z.; Zhang, Z.; Madsen, C.; Roehner, N.; Myers, C. iBioSim 3: a tool for model-based genetic circuit design. *ACS Synth. Biol.* **2019**, *8*, 1560−1563.

(5) Elowitz, M. B.; Leibler, S. A synthetic oscillatory network of transcriptional regulators. *Nature* **2000**, *403*, 335−338.

(6) Gardner, T. S.; Cantor, C. R.; Collins, J. J. Construction of a genetic toggle switch in Escherichia coli. *Nature* **2000**, *403*, 339−342.

(7) Danino, T.; Mondragón-Palomino, O.; Tsimring, L.; Hasty, J. A synchronized quorum of genetic clocks. *Nature* **2010**, *463*, 326−330.

(8) Nielsen, A. A.; Der, B. S.; Shin, J.; Vaidyanathan, P.; Paralanov, V.; Strychalski, E. A.; Ross, D.; Densmore, D.; Voigt, C. A. Genetic circuit design automation. *Science* **2016**, *352*, No. aac7341.

(9) Shin, J.; Zhang, S.; Der, B. S.; Nielsen, A. A.; Voigt, C. A. Programming Escherichia coli to function as a digital display. *Mol. Syst. Biol.* **2020**, *16*, No. e9401.

(10) Thomas, D. E.; Moorby, P. R. *The Verilog® Hardware Description Language*; Springer, 1990.

(11) Vidal, G.; Vidal-Céspedes, C. I.; Rudge, T. J. Accurate reconstruction of dynamic gene expression and growth rate profiles from noisy measurements. *bioRxiv* **2021**, DOI: 10.1101/2021.03.16.435606.

(12) Bartley, B. A.; Choi, K.; Samineni, M.; Zundel, Z.; Nguyen, T.; Myers, C. J.; Sauro, H. M. pySBOL: a python package for genetic design automation and standardization. *ACS Synth. Biol.* **2019**, *8*, 1515−1518.

(13) Yáñez Feliú, G.; Earle Gómez, B.; Codoceo Berrocal, V.; Muñoz Silva, M.; Nuñez, I. N.; Matute, T. F.; Arce Medina, A.; Vidal, G.; Vidal Céspedes, C.; Dahlin, J.; et al. Flapjack: Data management and analysis for genetic circuit characterization. *ACS Synth. Biol.* **2021**, *10*, 183−191.

(14) Yeoh, J. W.; Swainston, N.; Vegh, P.; Zulkower, V.; Carbonell, P.; Holowko, M. B.; Peddinti, G.; Poh, C. L. SynBiopython: an open-source software library for Synthetic Biology. *Synth. Biol.* **2021**, *6*, No. ysab001.

(15) Chapman, B.; Chang, J. Biopython: Python tools for computational biology. *ACM Sigbio Newsl.* **2000**, *20*, 15−19.

(16) McLaughlin, J. A.; Myers, C. J.; Zundel, Z.; Mısırlı, G.; Zhang, M.; Ofiteru, I. D.; Goni-Moreno, A.; Wipat, A. SynBioHub: a standards-enabled design repository for synthetic biology. *ACS Synth. Biol.* **2018**, *7*, 682−688.

(17) Potvin-Trottier, L.; Lord, N. D.; Vinnicombe, G.; Paulsson, J. Synchronous long-term oscillations in a synthetic gene circuit. *Nature* **2016**, *538*, 514−517.
(18) Tigges, M.; Marquez-Lago, T. T.; Stelling, J.; Fussenegger, M. A tunable synthetic mammalian oscillator. *Nature* **2009**, *457*, 309−312.
(19) Stricker, J.; Cookson, S.; Bennett, M. R.; Mather, W. H.; Tsimring, L. S.; Hasty, J. A fast, robust and tunable synthetic gene oscillator. *Nature* **2008**, *456*, 516−519.
(20) Gillespie, D. T. Stochastic simulation of chemical kinetics. *Annu. Rev. Phys. Chem.* **2007**, *58*, 35−55.
(21) Beal, J.; Haddock-Angelli, T.; Baldwin, G.; Gershater, M.; Dwijayanti, A.; Storch, M.; De Mora, K.; Lizarazo, M.; Rettberg, R. Quantification of bacterial fluorescence using independent calibrants. *PLoS One* **2018**, *13*, No. e0199432.
(22) Storch, M.; Haines, M. C.; Baldwin, G. S. DNA-BOT: a low-cost, automated DNA assembly platform for synthetic biology. *Synth. Biol.* **2020**, *5*, ysaa010.
(23) Weiße, A. Y.; Oyarzún, D. A.; Danos, V.; Swain, P. S. Mechanistic links between cellular trade-offs, gene expression, and growth. *Proc. Natl. Acad. Sci. U. S. A.* **2015**, *112*, E1038−E1047.
(24) Gorochowski, T. E.; Avcilar-Kucukgoze, I.; Bovenberg, R. A.; Roubos, J. A.; Ignatova, Z. A minimal model of ribosome allocation dynamics captures trade-offs in expression between endogenous and synthetic genes. *ACS Synth. Biol.* **2016**, *5*, 710−720.
(25) Qian, Y.; Huang, H.-H.; Jiménez, J. I.; Del Vecchio, D. Resource competition shapes the response of genetic circuits. *ACS Synth. Biol.* **2017**, *6*, 1263−1272.
(26) Ceroni, F.; Boo, A.; Furini, S.; Gorochowski, T. E.; Borkowski, O.; Ladak, Y. N.; Awan, A. R.; Gilbert, C.; Stan, G.-B.; Ellis, T. Burden-driven feedback control of gene expression. *Nat. Methods* **2018**, *15*, 387−393.
(27) Lou, C.; Stanton, B.; Chen, Y.-J.; Munsky, B.; Voigt, C. A. Ribozyme-based insulator parts buffer synthetic circuits from genetic context. *Nat. Biotechnol.* **2012**, *30*, 1137−1142.
(28) Carr, S. B.; Beal, J.; Densmore, D. M. Reducing DNA context dependence in bacterial promoters. *PLoS One* **2017**, *12*, No. e0176013.
(29) Revell, J.; Zuliani, P. Stochastic rate parameter inference using the cross-entropy method. *Lect. Notes Comput. Sci.* **2018**, *11095*, 146−164.
(30) Rudge, T. J.; Steiner, P. J.; Phillips, A.; Haseloff, J. Computational modeling of synthetic microbial biofilms. *ACS Synth. Biol.* **2012**, *1*, 345−352.

https://doi.org/10.1021/acssynbio.1c00603
ACS Synth. Biol. 2022, 11, 1984−1990