ACS Synthetic Biology
Technical Note
Cite This: ACS Synth. Biol. XXXX, XXX, XXX−XXX
pubs.acs.org/synthbio

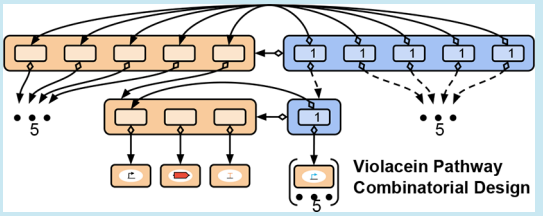
Specifying Combinatorial Designs with the Synthetic Biology Open Language (SBOL)

Nicholas Roehner,*[,†] Bryan Bartley,[†] Jacob Beal,[†] James McLaughlin,[‡] Matthew Pocock,[§] Michael Zhang,[∥] Zach Zundel,[∥] and Chris J. Myers[∥]

[†]Raytheon BBN Technologies, Cambridge, Massachusetts 02138, United States
[‡]Newcastle University, Newcastle upon Tyne NE1 7RU, UK
[§]Turing Ate My Hamster, Ltd., Tyne and Wear, NE27 0RT, UK
[∥]University of Utah, Salt Lake City, Utah 84112, United States

Supporting Information

ABSTRACT: As improvements in DNA synthesis technology and assembly methods make combinatorial assembly of genetic constructs increasingly accessible, methods for representing genetic constructs likewise need to improve to handle the exponential growth of combinatorial design space. To this end, we present a community accepted extension of the SBOL data standard that allows for the efficient and flexible encoding of combinatorial designs. This extension includes data structures for representing genetic designs with "variable" components that can be implemented by choosing one of many linked designs for existing genetic parts or constructs. We demonstrate the representational power of the SBOL combinatorial design extension through case studies on metabolic pathway design and genetic circuit design, and we report the expansion of the SBOLDesigner software tool to support users in creating and modifying combinatorial designs in SBOL.



KEYWORDS: *combinatorial design, combinatorial libraries, biodesign automation, standards, SBOL*

Synthetic biologists are generating increasingly rich and complex libraries of genetic construct variants using many different techniques for the combinatorial assembly of genetic parts.[1−5] Such combinatorial libraries can play an important role in genetic design by allowing designers to explore the impact of part choice, order, and orientation on construct behavior. In order to support the design of such libraries, an expanding collection of tools and formalisms have been developed to enable the specification, permutation, and sampling of combinatorial genetic design spaces.[6−9] In turn, these tools and formalisms have given rise to the need for a standard representation of combinatorial genetic designs in order to enable sharing of such designs between tools and laboratories as well as simplify human and machine reasoning over them.

As a basis for this representation, we have chosen the Synthetic Biology Open Language (SBOL), an existing community standard for representing both structural and functional aspects of genetic designs.[10,11] SBOL has support for hierarchical design, modular composition, and partial specification, making it a natural fit for representing combinatorial design templates and variables. Accordingly, we have developed an extension of SBOL to represent combinatorial designs, and we have incorporated this extension into the SBOL 2.2 specification[12] and SBOL software libraries (www.sbolstandard.org/libsbol).[13−15]

RESULTS

Here we briefly summarize the data model for this extension and discuss its application in two example use cases—a library of pathway variants to optimize enzyme expression,[16] and a library of genetic circuit variants to optimize logic gate function[2,17]—as well as in an updated version of the SBOLDesigner software tool.[18]

Representing Combinatorial Design. Building on the core data model of SBOL, the representation of combinatorial design is a relatively lightweight extension. Namely, the representation of a combinatorial design (a CombinatorialDerivation) involves the specification of (1) a design template and any constraints on its structure (a ComponentDefinition and its sub-Components), (2) the variable portions of the template and their cardinality (sub-Components specified by the VariableComponents of the CombinatorialDerivation), and (3) the variants or values that these variables can assume (possible ComponentDefinitions for each sub-Component as specified by the VariableComponents of the CombinatorialDerivation). SBOL does not require any particular algorithm or data structure to be used in enumerating designs from a combinatorial specification, but provides rules and best

Special Issue: IWBDA 2018
Received: February 28, 2019
Published: June 17, 2019

DOI: 10.1021/acssynbio.9b00092
ACS Synth. Biol. XXXX, XXX, XXX−XXX

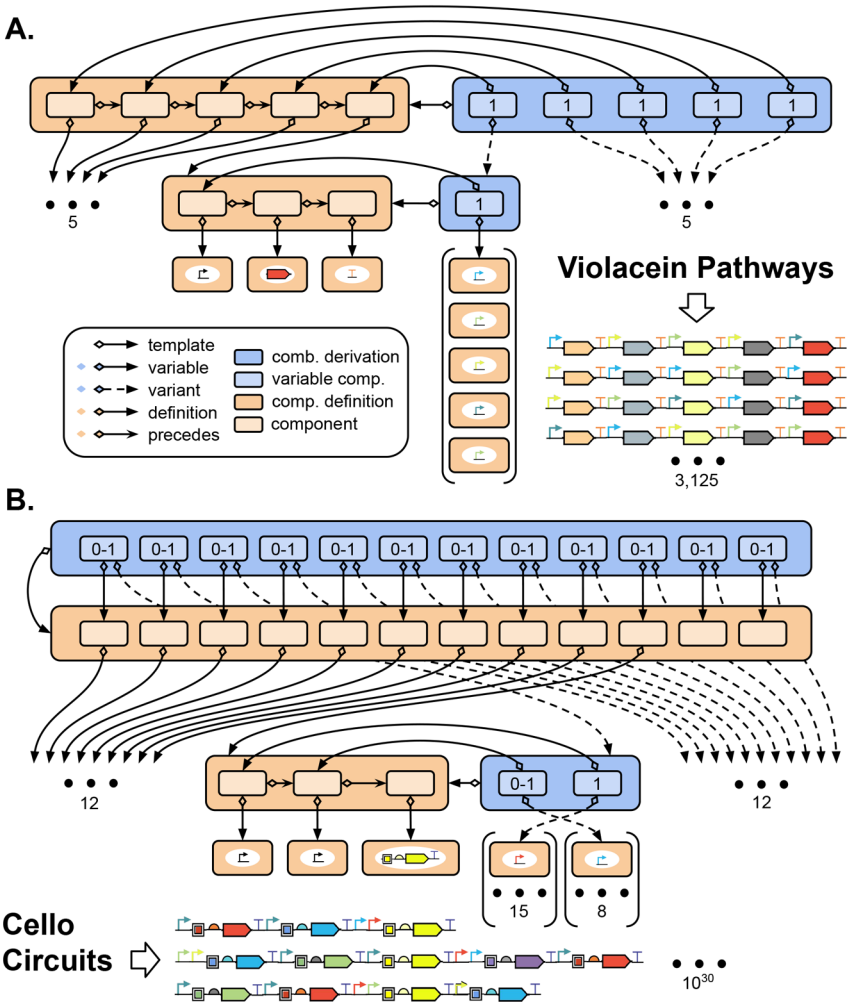


Figure 1. Representation of combinatorial designs for a violacein pathway (top) and Cello logic circuits (bottom) using SBOL. Instances of the new combinatorial design classes are blue, with cardinalities shown as a number on each variable component.

practices for validating whether such designs are correct realizations of their specification.

The new combinatorial data model of SBOL 2.2 consists of two classes: the CombinatorialDerivation class and its associated VariableComponent class (the latter is used to elaborate the properties of the former). The CombinatorialDerivation class is used to link between the template for a library of combinatorial designs and the sets of variables and values that can fill in the template to form specific combinations. The template is defined using a ComponentDefinition: ComponentDefinition is a base class of SBOL that can be used to specify the structure of a biopolymer or other molecule in a modular, hierarchical manner, along with constraints on this structure. For instance, the ComponentDefinition for an abstract transcriptional unit (TU) might contain sub-Components for a promoter, coding sequence (CDS), and terminator (some or all having no specified Sequence), as well as a set of SequenceConstraint objects to assert their relative ordering and orientation. The CombinatorialDerivation class can also be used to broadly recommend how to derive individual designs from the template by setting its strategy property. At present, two generic strategies are defined: exhaustive enumeration of every possible design, or sampling an unspecified subset.

The other class, VariableComponent, is used to specify how the template of a CombinatorialDerivation is filled in to create fully implemented designs. Each instance of the VariableComponent class specifies a set of available ComponentDefinition variants that can define a sub-Component variable from the template ComponentDefinition. These variant ComponentDefinitions can be aggregated individually or as part of an SBOL Collection, or they can also be derived in accordance with another CombinatorialDerivation (a variantDerivation), thereby enabling the specification of a hierarchical combinatorial design. The operator property of the VariableComponent then specifies how many Component objects are expected to be derived from the sub-Component variable with one of four cardinalities: one, zero-or-one, zero-or-more, or one-or-more. A more detailed description of the CombinatorialDerivation and VariableComponent classes can be found in the SBOL 2.2 technical specification.[12]

Theorem 1. *The expressive power of a CombinatorialDerivation is equivalent to a regular language.*

Proof. Assume an alphabet that includes all possible ComponentDefinitions and that A and B are CombinatorialDerivations.

A CombinatorialDerivation with an empty template ComponentDefinition results in the empty language, $\{\epsilon\}$.

DOI: 10.1021/acssynbio.9b00092
ACS Synth. Biol. XXXX, XXX, XXX−XXX

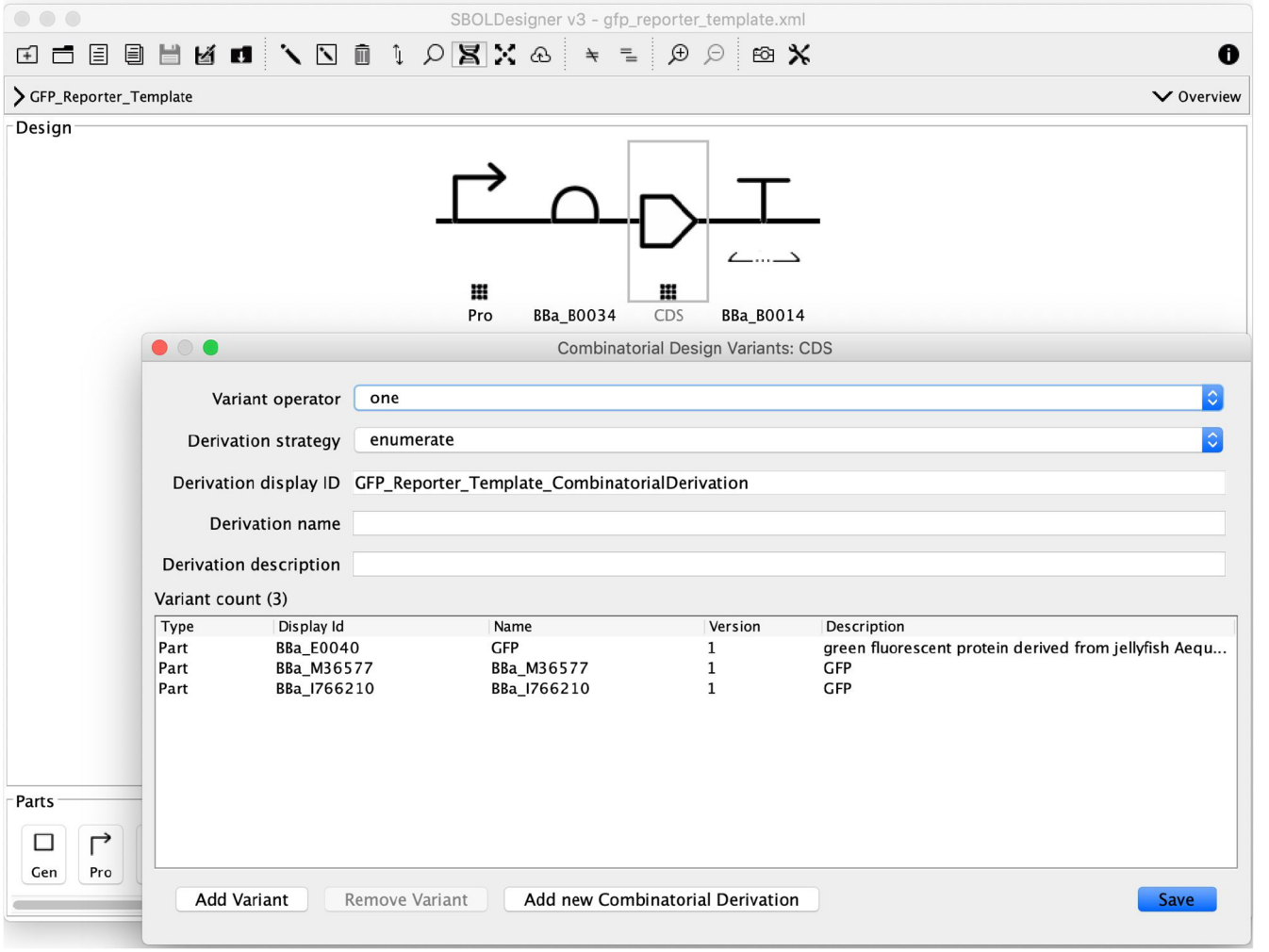


Figure 2. SBOLDesigner, a sequence-based computer-aided design tool, and its application to a combinatorial design encoded in SBOL. Each part labeled with a grid symbol corresponds to a variable component that can be implemented by choosing one of multiple variant parts or constructs, or potentially omitted if the variant operator includes "zero".

A CombinatorialDerivation with a template ComponentDefinition that includes a single sub-Component, which has a VariableComponent with operator "one", a variable that refers to this Component, and a variant that refers to a single ComponentDefinition a results in the singleton language $\{a\}$.

Assume C is a CombinatorialDerivation that has a template ComponentDefinition with two sub-Components in series, and two VariableComponents with operator "one" that each refer to one of the Components as a variable. If the VariableComponent that refers to the first Component has a variantDerivation that refers to A and the VariableComponent that refers to the second Component has a variantDerivation that refers to B, then the result is the concatenation of A and B (i.e., $A{\cdot}B$).

Assume C is a CombinatorialDerivation that has a template ComponentDefinition with one sub-Component, and one VariableComponent with operator "one" that refers to the Component as a variable. If the VariableComponent has A and B as variantDerivations, then the result is the union of A and B (i.e., $A \cup B$).

Assume C is a CombinatorialDerivation that has a template ComponentDefinition with one sub-Component, and one VariableComponent with operator "zero-or-more" that refers to the Component as a variable. If the VariableComponent has A as a variantDerivation, then the result is the Kleene star of A (i.e., A^*).

Use Case: Pathway Design. Figure 1A demonstrates how SBOL can be used to encode the combinatorial design of a library of 3125 violacein pathway variants originally designed by the Dueber lab.[16] The SBOL representation consists of a two-level hierarchy of ComponentDefinition and CombinatorialDerivation objects, with root objects at the top of this hierarchy and leaf objects at the bottom. The root ComponentDefinition is a template that specifies the complete ordering of five partially abstract TUs, each defined by a ComponentDefinition containing an abstract promoter followed by one of the enzyme CDSs from the violacein pathway and the terminator tADH1, all with inline orientation. The root CombinatorialDerivation then specifies that each of the five TUs in the template should be filled in with one of five possible TUs with different promoters as specified by a leaf CombinatorialDerivation. Each of these leaf CombinatorialDerivations refers to the same set of five promoter variants but refers to a different template ComponentDefinition with a different enzyme CDS.

Use Case: Genetic Circuit Design. Figure 1B demonstrates how SBOL can be used to encode the combinatorial

DOI: 10.1021/acssynbio.9b00092
ACS Synth. Biol. XXXX, XXX, XXX−XXX

design of all 10^{30} genetic circuit variants that can be constructed from the Cello gate NOR/NOT gate library. The key differences between this combinatorial design and that of the violacein pathway are that the root CombinatorialDerivation does not specify the relative order or orientation of any of its 12 generic TUs, nor does it require that each of these TUs be filled in (because each VariableComponent has a zero-or-one operator). Consequently, the circuit derived from this combinatorial design can contain any number of TUs up to 12, and these TUs can have any ordering. In addition, each leaf CombinatorialDerivation has a single zero-or-one VariableComponent corresponding to the second promoter in the template TU ComponentDefinition, thus capturing the fact that each derived TU can have NOT or NOR logic (one promoter or two promoters).

Use Case: SBOLDesigner. SBOLDesigner is a sequence-based computer-aided design tool that supports creating and modifying combinatorial designs represented using SBOL.[18] We have expanded SBOLDesigner to support designs that make use of the new combinatorial design extension. Figure 2 shows an example of the new interface being used to specify a combinatorial design of a green fluorescent protein reporter circuit (shown here on SBOLDesigner's main canvas). In this design, the promoter and CDS are described using combinatorial variants, which can be edited using a variant editor dialogue, shown here being applied to set the choices for implementing the green fluorescent protein CDS. In this case, each combination of promoter and CDS will be enumerated with the same ribosome binding site and a hierarchically defined double terminator.

DISCUSSION

Currently, the SBOL representation of combinatorial design is equivalent in expressive power to a regular language. Though not demonstrated by these use cases, SBOL can be used to represent design patterns in which a particular component or motif is repeated an indefinite number of times. For example, this could be used to represent the design of a promoter with a variable number of operator sites. Should the need arise to represent palindromic design patterns, such as with a context-free language, SBOL can be extended with additional types of constraints to assert that the same number of components must be derived from different parts of the template.

Many key cases of combinatorial library design can be represented using SBOL with the new combinatorial design extension, ranging from existing industrial applications in optimizing biosynthetic pathways to current research in controlling biological systems. This improves over prior representations by integrating combinatorial design with hierarchical, ontology-supported representation, thereby enabling unambiguous reasoning over multiple levels of design abstraction and their relationship to other classes of information (such as experimental products) and other sources of metadata outside of the SBOL standard. We thus anticipate that SBOL representation of combinatorial design will support improved tooling and workflows, better reuse and attribution of designs, faster engineering of circuits and components, and novel applications across many subdomains of synthetic biology.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acssynbio.9b00092.

ZIP archive containing two SBOL files (.xml file extension) for the combinatorial designs from the violacein pathways and Cello logic circuits use cases (ZIP)

ZIP archive containing two SBOL files (.xml file extension) for the combinatorial design and enumerated designs from the GFP reporter use case (ZIP)

AUTHOR INFORMATION

Corresponding Author

*E-mail: nicholas.roehner@raytheon.com.

ORCID

Nicholas Roehner: 0000-0003-4957-1552
Bryan Bartley: 0000-0002-1597-4022
Michael Zhang: 0000-0002-1084-6734
Zach Zundel: 0000-0003-1355-6721
Chris J. Myers: 0000-0002-8762-8444

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

The authors of this work are supported by the National Science Foundation under Grant No., CCF-1748200 (C.M.), DBI-1356041 (M.Z. and C.M.), #1522074 (N.R.), and DARPA award HR0011-15-C-0084 (N.R.), and FA8750-17-C-0229 (Z.Z. and C.M.). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies. This document does not contain technology or technical data controlled under either U.S. International Traffic in Arms Regulation or U.S. Export Administration Regulations.

REFERENCES

(1) Weber, E., Engler, C., Gruetzner, R., Werner, S., and Marillonnet, S. (2011) A modular cloning system for standardized assembly of multigene constructs. *PLoS One 6*, No. e16765.

(2) Woodruff, L. B., Gorochowski, T. E., Roehner, N., Mikkelsen, T. S., Densmore, D., Gordon, D. B., Nicol, R., and Voigt, C. A. (2016) Registry in a tube: multiplexed pools of retrievable parts for genetic design space exploration. *Nucleic Acids Res. 45*, 1553−1565.

(3) Ellis, T., Adie, T., and Baldwin, G. S. (2011) DNA assembly for synthetic biology: from parts to pathways and beyond. *Integr. Biol. 3*, 109−118.

(4) Cobb, R. E., Ning, J. C., and Zhao, H. (2014) DNA assembly techniques for next-generation combinatorial biosynthesis of natural products. *J. Ind. Microbiol. Biotechnol. 41*, 469−477.

(5) Engler, C., and Marillonnet, S. (2013) *Synthetic Biology*, pp 141−156, Springer.

(6) Bilitchenko, L., Liu, A., Cheung, S., Weeding, E., Xia, B., Leguia, M., Anderson, J. C., and Densmore, D. (2011) Eugene−a domain specific language for specifying and constraining synthetic biological parts, devices, and systems. *PLoS One 6*, No. e18882.

(7) Yang, K., Stracquadanio, G., Luo, J., Boeke, J. D., and Bader, J. S. (2016) BioPartsBuilder: A synthetic biology tool for combinatorial assembly of biological parts. *Bioinformatics 32*, 937−939.

(8) Roehner, N., Young, E. M., Voigt, C. A., Gordon, D. B., and Densmore, D. (2016) Double Dutch: a tool for designing

DOI: 10.1021/acssynbio.9b00092
ACS Synth. Biol. XXXX, XXX, XXX−XXX

combinatorial libraries of biological systems. *ACS Synth. Biol. 5*, 507−517.
(9) Bhatia, S. P., Smanski, M. J., Voigt, C. A., and Densmore, D. M. (2017) Genetic design via combinatorial constraint specification. *ACS Synth. Biol. 6*, 2130−2135.
(10) Galdzicki, M., et al. (2014) The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nat. Biotechnol. 32*, 545−550. Computational Biology.
(11) Roehner, N., et al. (2016) Sharing Structure and Function in Biological Design with SBOL 2.0. *ACS Synth. Biol. 5*, 498−506.
(12) Cox, R., et al. (2018) Synthetic biology open language (SBOL) version 2.2.0. *J. Integr. Bioinform.*, DOI: 10.1515/jib-2017-0074.
(13) Zhang, Z., Nguyen, T., Roehner, N., Misirli, G., Pocock, M., Oberortner, E., Samineni, M., Zundel, Z., Beal, J., Clancy, K., Wipat, A., and Myers, C. (2015) libSBOLj 2.0: a Java library to support SBOL 2.0. *IEEE Life Sciences Letters 1*, 34−37.
(14) Bartley, B. A., Choi, K., Samineni, M., Zundel, Z., Nguyen, T., Myers, C. J., and Sauro, H. M. (2018) pySBOL: A Python Package for Genetic Design Automation and Standardization. *ACS Synth. Biol.*, DOI: 10.1021/acssynbio.8b00336.
(15) McLaughlin, J. A., Myers, C. J., Zundel, Z., Wilkinson, N., Atallah, C., and Wipat, A. (2019) sboljs: Bringing the Synthetic Biology Open Language to the Web Browser. *ACS Synth. Biol. 8*, 191−193.
(16) Lee, M. E., Aswani, A., Han, A. S., Tomlin, C. J., and Dueber, J. E. (2013) Expression-level optimization of a multi-enzyme pathway in the absence of a high-throughput assay. *Nucleic Acids Res. 41*, 10668−10678.
(17) Nielsen, A. A. K., Der, B. S., Shin, J., Vaidyanathan, P., Paralanov, V., Strychalski, E. A., Ross, D., Densmore, D., and Voigt, C. A. (2016) Genetic circuit design automation. *Science 352*, 352.
(18) Zhang, M., McLaughlin, J. A., Wipat, A., and Myers, C. J. (2017) SBOLDesigner 2: an intuitive tool for structural genetic design. *ACS Synth. Biol. 6*, 1150−1160.

DOI: 10.1021/acssynbio.9b00092
ACS Synth. Biol. XXXX, XXX, XXX−XXX