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

Nicholas Roehner[1], Bryan Bartley[1], Jacob Beal[1], James McLaughlin [2], Matthew Pocock[3], Michael Zhang[4], Zach Zundel[4], Chris Myers[4], Anil Wipat[2]
[1]Raytheon BBN Technologies, [2]Newcastle University, [3]Turing Ate My Hamster, Ltd., [4]University of Utah
nicholas.roehner@raytheon.com

## 1 INTRODUCTION

During the last decade, new technologies have been developed for the combinatorial assembly of genetic parts [8, 9], enabling synthetic biologists to more readily generate libraries of genetic construct variants. These types of 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, new tools and formalisms have been developed to enable the specification, permutation, and sampling of combinatorial genetic design spaces [1, 2]. In turn, these 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 and to 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 [4, 7]. 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 [3] and SBOL software libraries (www.sbolstandard.org/libsbol). 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 [5], and a library of genetic circuit variants to optimize logic gate function [6, 9].

## 2 REPRESENTING COMBINATORIAL DESIGN

Building on the core data model of SBOL, the representation of combinatorial design is a relatively lightweight extension. Namely, its representational semantics involve the specification of a design template and any constraints on its structure, the variable portions of the template and their cardinality, and the variants or values that these variables can assume. 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 practices to validate whether these designs are a correct realization of their specification.

There are two classes in the new SBOL 2.2 combinatorial data model: the CombinatorialDerivation class and the VariableComponent class (Figure 1). The CombinatorialDerivation class is used to specify a template for a library of combinatorial designs and to link that template to a collection of variables and values that will fill in the template to form specific combinations. The template is defined

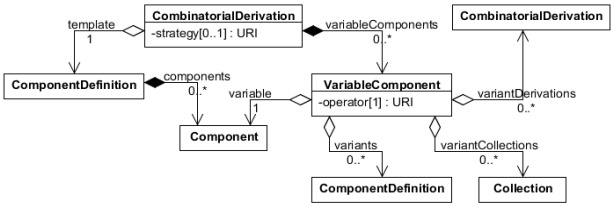


**Figure 1: Combinatorial designs can be specified in SBOL 2.2 using two new classes: CombinatorialDerivation and VariableComponent.**

using a ComponentDefinition: ComponentDefinition is a base class of SBOL used to specify the structure of a biopolymer in a modular, hierarchical manner, along with constraints on this structure. For instance, the ComponentDefinition for an abstract transcriptional unit (TU) would likely contain sub-Component objects for a promoter, coding sequence (CDS), and terminator without sequences and a set of SequenceConstraint objects to assert their relative ordering and orientations. The CombinatorialDerivation class can also be used to broadly recommend how many individual designs to derive from the template by setting its strategy property. At present, two strategy values are defined: either exhaustive enumeration of every possible design or sampling an unspecified subset.

The other class, VariableComponent, is used to specify the way in which a CombinatorialDerivation template is filled in to create fully instantiated designs. Each instance of the VariableComponent class specifies a set of available ComponentDefinition variants that can define a Component from the template. These variants can be aggregated individually or as part of an SBOL Collection, or can be derived in accordance with another CombinatorialDerivation, enabling the specification of a hierarchical combinatorial design. The operator property then specifies how many Component objects are expected to be derived from the template Component (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 [3].

## 3 EXAMPLE USE CASES

*Use Case: Pathway Design.* Figure 2 demonstrates how SBOL can be used to encode the combinatorial design of a library of 3,125 violacein pathway variants originally designed by the Dueber lab [5]. The SBOL representation consists of a two-level hierarchy of ComponentDefinition and CombinatorialDerivation objects. The root ComponentDefinition is a template that specifies the complete

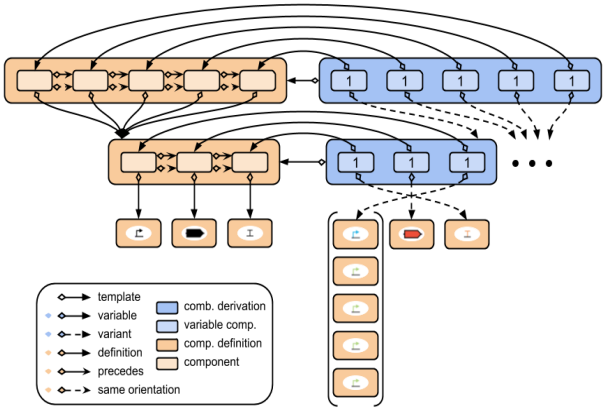


**Figure 2: Representation of violacein pathway combinatorial design using SBOL.**

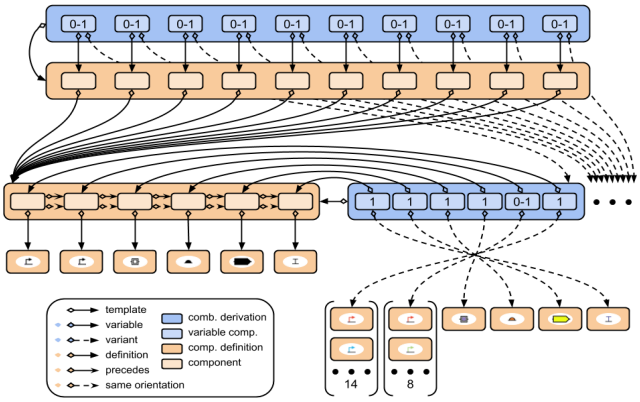


**Figure 3: Representation of Cello circuit combinatorial design using SBOL.**

ordering of five generic TUs, each defined by the same ComponentDefinition containing a promoter followed by a CDS and a terminator, all with the same 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 leaf CombinatorialDerivation refers to the same set of five promoter variants but refers to a different enzyme CDS in the violacein pathway.

*Use Case: Genetic Circuit Design.* Figure 3 demonstrates how SBOL can be used to encode the combinatorial 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 its root CombinatorialDerivation does not specify the relative order or orientation of any of its ten 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 ten, and these TUs can have any ordering or orientation. In addition, each leaf CombinatorialDerivation has a single zero-or-one VariableComponent corresponding to the first 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).

## 4 DISCUSSION AND CONTRIBUTIONS

Currently, SBOL's 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, allowing unambiguous reasoning about complete designs, as well as their relationship to information sources, experimental products, and other designs. We thus anticipate that SBOL representation of combinatorial design will support improved tooling and workflows, facilitating better reuse and attribution of designs, faster engineering of circuits and components, and novel applications across many domains of synthetic biology.

## ACKNOWLEDGMENTS

This work was supported by the DARPA Living Foundries award HR0011-15-C-0084. 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] S. P. Bhatia, M. J. Smanski, C. A. Voigt, and D. M. Densmore. Genetic design via combinatorial constraint specification. *ACS Synth. Biol.*, 6(11):2130–2135, 2017.
[2] L. Bilitchenko et al. Eugene–a domain specific language for specifying and constraining synthetic biological parts, devices, and systems. *PLoS One*, 6(4):e18882, 2011.
[3] R. Cox et al. Synthetic Biology Open Language (SBOL) version 2.2.0. *Journal of Integrative Bioinformatics*, 15(1), 2018.
[4] M. Galdzicki et al. The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nat. Biotechnol.*, 32(6):545–550, 2014.
[5] M. E. Lee, A. Aswani, A. S. Han, C. J. Tomlin, and J. E. Dueber. Expression-level optimization of a multi-enzyme pathway in the absence of a high-throughput assay. *Nucleic Acids Res.*, 41(22):10668–10678, 2013.
[6] A. A. K. Nielsen. Genetic circuit design automation. *Science*, 352(6281):aac7341, 2016.
[7] N. Roehner et al. Sharing structure and function in biological design with SBOL 2.0. *ACS Synth. Biol.*, 5(6):498–506, 2016.
[8] E. Weber, C. Engler, R. Gruetzner, S. Werner, and S. Marillonnet. A modular cloning system for standardized assembly of multigene constructs. *PLoS One*, 6(2):e16765, 2011.
[9] L. B. A. Woodruff et al. Registry in a tube: multiplexed pools of retrievable parts for genetic design space exploration. *Nucleic Acids Res.*, 45(3):1553–1565, 2017.