ShortBOL: A Language for Scripting Designs for Engineered Biological Systems Using Synthetic Biology Open Language (SBOL)

Matthew Crowther,|| Lewis Grozinger,|| Matthew Pocock,|| Christopher P. D. Taylor,|| James A. McLaughlin, Göksel Misirli, Bryan Bartley, Jacob Beal, Angel Goñi-Moreno,* and Anil Wipat*

**Cite This:** https://dx.doi.org/10.1021/acssynbio.9b00470

**ABSTRACT:** The Synthetic Biology Open Language (SBOL) is an emerging synthetic biology data exchange standard, designed primarily for unambiguous and efficient machine-to-machine communication. However, manual editing of SBOL is generally difficult for nontrivial designs. Here, we describe ShortBOL, a lightweight SBOL scripting language that bridges the gap between manual editing, visual design tools, and direct programming. ShortBOL is a shorthand textual language developed to enable users to create SBOL designs quickly and easily, without requiring strong programming skills or visual design tools.

**KEYWORDS:** programming language, biodesign, Synthetic Biology Open Language (SBOL), synthetic biology, RDF

**RESULTS**

ShortBOL v1.0 is designed to be easy to use for synthetic biologists who may not have much software development training but understand the fundamentals of the SBOL data model. Those with software development training can also find ShortBOL useful as a rapid method of producing SBOL more simply than by writing code that uses the SBOL libraries. The language is text-based, but has a simplified syntax that abstracts some of the more complex features of SBOL. Moreover, by following the tutorial, users who are new to the SBOL data model will be able to quickly and easily produce SBOL designs.

**Received:** November 22, 2019  
**Published:** March 4, 2020

**ShortBOL**

**SBOL RDF/XML**
model can gain exposure to the terminology and approach without having to work with the SBOL code libraries.

ShortBOL is currently built around a minimal selection of language constructs. A typical shorthand document is a list of imports, variable assignments, and template statements to be expanded. A standard template library is provided with ShortBOL, which allows different aspects of genetic designs to be generated using the SBOL data model in response to keywords in the ShortBOL language (Figure 1).

The standard library templates themselves are also written in shorthand, in the same way that a user might create their own template libraries to capture abstractions common within their designs or the synthetic biology domain. These new templates may extend any number of existing templates, or be built from scratch. Furthermore, if libraries are shared, they can then be imported, used, and extended by others.

Custom templates can be used to provide simple aliases, application-specific syntax, access to common terminologies, and can even be used to model complex parametrized multicomponent designs. Variable assignments, on the other hand, 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.

**ShortBOL Usage.** ShortBOL can be used from both the command line and from a custom Web application (http://shortbol.org/) (Figure 2). The ShortBOL repository on GitHub includes documentation on how to compile ShortBOL text files to SBOL XML files using the supplied Python software at the command line. The web application allows ShortBOL documents to be written in the web-based editor and automatically compiled to an SBOL RDF/XML file, which the user can then download. A tutorial describing how to use ShortBOL is also provided, which also introduces features of the SBOL data model. When ShortBOL code is executed via the command line or web application, the output is validated for compliance with the SBOL specification, ensuring ShortBOL output will interoperate with other SBOL tooling.

**Implementation.** SBOL entities are created within the shorthand by using the (is a) operator to expand a template (Figure 3A). For example, lacI_cds is a CDS introduces a new identifier lacI_cds whose properties will be set according to the pattern described by the CDS template. In this particular case,
the CDS template further expands to a SBOL:ComponentDefinition template, which sets the type property to the DnaRegion BioPAX term and role property to the CDS (SO:000316) Sequence Ontology term, as recommended in the SBOL best practices for encoding a CDS using SBOL (Figure 3B). Templates can also be parametrized 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. In combination with the recursive expansion of templates, this can allow collections of specialized, domain-specific templates to be composed from generic ones. Template expansions can also contain a block of ShortBOL expressions. These are used to declare additional properties and their values. For example, the template application lacI_cds is a CDS may be followed by a bracketed block containing the property assignment description = "The lacI CDS".

Interpretation. The statements contained in shorthand documents are interpreted sequentially, and from each template expansion statement a RDF graph is generated. The union of these graphs is then serialized as RDF/XML to produce a valid SBOL document. The steps involved in interpreting a shorthand statement depend on the type of that statement:

- **Import statements:** 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 in all subsequent statements.
- **Template declaration:** Templates are associated with their identifier, and made available for future expansion.
- **Template expansion:** If the name of a template application matches a registered template, expand that template and set all the nested properties.

### DISCUSSION

ShortBOL v1.0 fulfills the need for an SBOL shorthand. This version is designed to be true to the SBOL data model, allowing synthetic biologists to read and write SBOL, and for the rapid creation and exchange of synthetic biology designs without...
ShortBOL comes with a formal syntax and semantics, and so is also suitable for machine exchange. ShortBOL is not intended to replace SBOL, however, which can represent additional complex design information, including material that is not textual or that has user-defined semantics. Moreover, SBOL is based on RDF and can benefit from existing Semantic Web tooling. Instead, 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.

Following the syntax and approach of the SBOL model has the advantage of making the ShortBOL syntax familiar to developers but can be daunting to biologists not familiar with SBOL terms and approaches. There is a further need for future development of ShortBOL to abstract away the more complex features of the SBOL data model and use a syntax that is more commonplace in the synthetic biology community. The current version of ShortBOL is centered around SBOL version 2.0, which allows synthetic biology designs to be encoded. However, subsequent SBOL versions also include features such as capturing the lineage of designs, combinatorial assembly, encoding parameters and measures, and recording experimental data. Modifications and extensions to the standard library included with ShortBOL will be required in order to support these features of the data model.

Development of a new version that includes the newer features above, together with a fully online editor and expansion pipeline is ongoing, supporting while-you-type integration with other SBOL tooling, including VisBOL. We hope that the open nature of ShortBOL template libraries will support rapid development of SBOL extensions and domain-specific design terminologies. Moreover, we envisage community-driven development of template libraries to intuitively design biological systems according to the needs of different laboratories.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acssynbio.9b00470.

The complete source code and examples from the ShortBOL project presented in this paper (can also be obtained from https://github.com/intbio-ncl/shortbol) (ZIP)

AUTHOR INFORMATION

Corresponding Authors

Angel Goni-Moreno — School of Computing, Newcastle University, Newcastle upon Tyne NE4 5ST, U.K.; orcid.org/0000-0002-2097-2507; Email: angel.goni-moreno@ncl.ac.uk

Anil Wipat — School of Computing, Newcastle University, Newcastle upon Tyne NE4 5ST, U.K.; orcid.org/0000-0001-7310-4191; Email: anil.wipat@ncl.ac.uk

Authors

Matthew Crowther — School of Computing, Newcastle University, Newcastle upon Tyne NE4 5TG, U.K.

Lewis Grozinger — School of Computing, Newcastle University, Newcastle upon Tyne NE4 5TG, U.K.; orcid.org/0000-0002-9024-701X

Matthew Pocock — School of Computing, Newcastle University, Newcastle upon Tyne NE4 5TG, U.K.

Christopher P. D. Taylor — School of Computing, Newcastle University, Newcastle upon Tyne NE4 5TG, U.K.

James A. McLaughlin — School of Computing, Newcastle University, Newcastle upon Tyne NE4 5TG, U.K.

Goësli Misrili — School of Computing and Mathematics, Keele University, Keele, Newcastle ST5 5BG, U.K.

Bryan Bartley — Raytheon BBN Technologies, Cambridge, Massachusetts 02138, United States; orcid.org/0000-0002-1597-4022

Jacob Beal — Raytheon BBN Technologies, Cambridge, Massachusetts 02138, United States

Complete contact information is available at: https://pubs.acs.org/10.1021/acssynbio.9b00470

Author Contributions


Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

The authors of this work are supported by The Engineering and Physical Sciences Research Council grants EP/J02175X/1, EP/R003629/1, EP/N031962/1 (J.A.M. and A.W.), and EP/R019002/1 (A.G.-M.), EPSRC studentship 3400024085 (M.C.), and the European CSA 820699 (A.G.-M. and A.W.). J.B. and B.B. are supported in part by the Air Force Research Laboratory (AFRL) and DARPA under contract FA875017CO184. J.A.M. is supported by FUJIFILM Diosynth Biotechnologies and M.C. by Doulix Ltd. M.C. is supported by the EPS. 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 the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

REFERENCES


