Technical Note

ShortBOL: A language for scripting designs for engineered biological systems using SyntheticBiology Open Language (SBOL)

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

ACS Synth. Biol., **Just Accepted Manuscript** • DOI: 10.1021/acssynbio.9b00470 • Publication Date (Web): 04 Mar 2020

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 Mısırlı,[‡] Bryan Bartley,[¶] Jacob Beal,[¶] Angel Goñi-Moreno,[*,†] and Anil Wipat[*,†]

†*School of Computing, Urban Sciences Building, Science Square, Newcastle upon Tyne, UK*

‡*School of Computing and Mathematics, Keele University*

¶*Raytheon BBN Technologies, Cambridge, USA*

§*These authors contributed equally to the project*

E-mail: angel.goni-moreno@ncl.ac.uk; anil.wipat@ncl.ac.uk

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 non-trivial designs. Here, we describe ShortBOL, a light-weight 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, bio-design, Synthetic Biology Open Language (SBOL), synthetic biology, RDF

Synthetic Biology Open Language (SBOL) version 2 has emerged as a data standard for synthetic biology.[1] SBOL facilitates computational design, exchange, and reproducibility of biological systems and is defined as a data model with an RDF/XML serialization. While well-suited for precise machine communication, SBOL RDF/XML is too verbose and complex for humans to manually edit designs, particularly for those involving many components and features.

Software tools and libraries have been developed to manipulate SBOL. For example, libSBOLj[2] and pySBOL[3] can be linked to other software, enabling them to read, write, and manipulate SBOL data. While these libraries support tool developers and others with strong programming skills, using them presents an extremely challenging learning curve for most synthetic biologists. Computer-aided Design (CAD) and visualization tools have also been developed to visualize designs and make the designs easier for humans to communicate.[4–6] These visual design tools, however, are often limited in the features of the representation that they can access and visual editing is often a slow and rather manual process. Thus, there is a need for a light-weight SBOL scripting language that bridges the gap between manual editing, visual design, and direct use of libraries.

Here, we describe such a language, ShortBOL (v.1.0), a human readable/writable shorthand for describing biological designs in SBOL. This language is developed for those who are familiar with the SBOL data model but wish to rapidly sketch synthetic biology designs using a simple, text-based scripting language instead of writing code that utilizes the SBOL libraries. Using this language, SBOL data can be generated easily and quickly from simple textual descriptions. The utility of such domain-specific languages has long been recognised by the synthetic biology community, and languages such as the Genotype Specification

Language[7] and Eugene[8] have previously been developed, in particular to enable automated assembly and the exploration of the synthetic biological design space. ShortBOL shares many design aims and characteristics with these languages. However, being an abstraction of SBOL data, ShortBOL inherits the richness of the SBOL data model and the ability to encapsulate design information of unique importance to synthetic biological constructs. Moreover, the ability to describe arbitrary RDF data in ShortBOL provides a flexibility and extensibility that will be important in producing ever greater abstraction, modularity, and concision.

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 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.

```
ComponentDefinition(t)
(
  Identified(ComponentDefinition)
  type = t
)

DnaComponent()
(
  ComponentDefinition(DNA)
)

Promoter()
(
  DnaComponent()
  role = promoter
)
```

Figure 1: An example of a ShortBOL template for a promoter. Here, a promoter is defined from a DnaComponent which is, in turn, defined using a ComponentDefinition. Users can define templates to create specialized representations of design patterns used in their SBOL designs.

Custom templates can be used to provide simple aliases, application-specific syntax, access to common terminologies, and can even be used to model complex parameterized multi-component 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/`). 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.

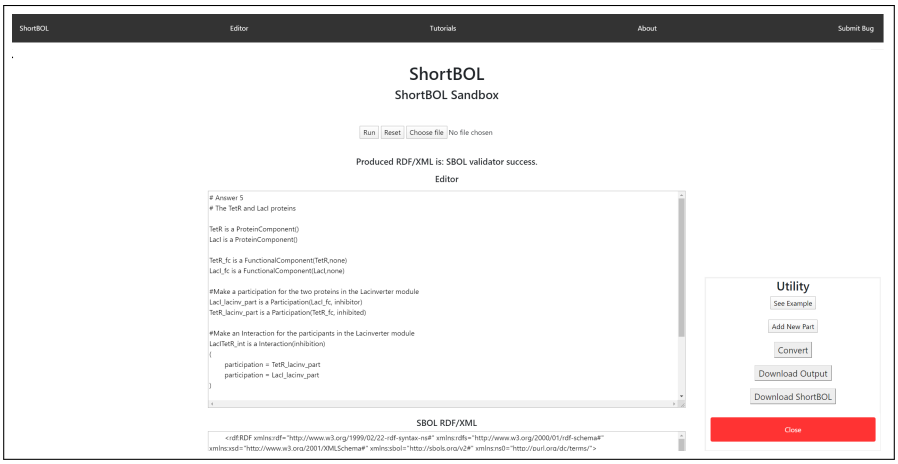

Figure 2: Screenshot of the ShortBOL Web application showing the built-in editor and output window.

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 parameterized 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

A)

```
@prefix igem = <http://parts.igem.org/>
@prefix igem

lacI_cds is a CDS()
(
   description = "The lacI CDS"
   name = "lacI"
   sequence = lacI_seq
)
lacI_seq is a DNASequence("atggtgaatgt")
```

B) ↓ *Template Expansion*

```
lacI_seq is a Sequence()
(
   encoding = iupacDNA
   displayId = "lacI_seq"
   elements = "atggtgaatgt"
)
lacI_cds is a ComponentDefinition()
(
   role = cds
   type = dna
   displayId = "lacI_cds"
   description = "The lacI CDS"
   name = "lacI"
   sequence = lacI_seq
 )
```

C) ↓ *Rendering to SBOL RDF/XML*

```
<sbol:ComponentDefinition rdf:about="http://parts.igem.org/lacI_cds/1">
  <sbol:persistentIdentity rdf:resource="http://parts.igem.org/lacI_cds"/>
  <sbol:version>1</sbol:version>
  <sbol:displayId>lacI_cds</sbol:displayId>
  <sbol:sequence rdf:resource="http://parts.igem.org/lacI_seq/1"/>
  <dcterms:title>lacI</dcterms:title>
  <dcterms:description>The  lacI  CDS</dcterms:description>
  <sbol:role rdf:resource="http://identifiers.org/so/SO:0000316"/>
  <sbol:type rdf:resource="http://www.biopax.org/release/biopax-level3.owl#Dna"/>
</sbol:ComponentDefinition>
<sbol:Sequence rdf:about="http://parts.igem.org/lacI_seq/1">
  <sbol:elements>atggtgaatgt</sbol:elements>
  <sbol:displayId>lacI_seq</sbol:displayId>
  <sbol:encoding rdf:resource="http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html"/>
  <sbol:persistentIdentity rdf:resource="http://parts.igem.org/lacI_seq"/>
  <sbol:version>1</sbol:version>
</sbol:Sequence>
```

Figure 3: 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.

`lacI_cds is a CDS` may be followed by an 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 complex computational tools or the need for a mediating GUI. 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 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. Moreover, we envisage community-driven development of template libraries to intuitively design biological systems according to the needs of different labs.

Author Contributions

M.P., C.T., G.M. and J.A.M. designed the initial version of the ShortBOL language. M.P. and C.T. implemented the Scala proof-of-concept implementation. L.G. and M.C. developed the Python implementation described in this paper. M.P, G.M., A.G.-M., J.B., B.B and A.W wrote the paper and tested the system. M.P., L.G, M.C., and A.W. wrote the documentation. A.W., M.P., and A.G.-M. supervised the project.

Acknowledgement

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 34000024085 (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.

Supporting Information Available

The following files are made available

- `shortbol-1.0_ACS_synthetic_biology_build.zip` : The complete source code and examples from the ShortBOL project presented in this paper, and can also be obtained from `https://github.com/intbio-ncl/shortbol`.

Conflict of Interest

The authors declare no conflicts of interest.

References

(1) Madsen, C. et al. Synthetic Biology Open Language (SBOL) Version 2.3. *J. Integr. Bioinform.* **2019**, *16*, 25.

(2) Zhang, Z.; Nguyen, T.; Roehner, N.; Misirli, G.; Pocock, M.; Oberortner, E.; Samineni, M.; Zundel, Z.; Beal, J.; Clancy, K.; Wipat, A.; Myers, C. J. libSBOLj 2.0: A Java Library to Support SBOL 2.0. *IEEE Life Sci. Lett.* **2015**, *1*, 34–37.

(3) 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.* **2018**, 1515–1518.

(4) McLaughlin, J. A.; Pocock, M.; Misirli, G.; Madsen, C.; Wipat, A. VisBOL: Web-based tools for synthetic biology design visualization. *ACS Synth. Biol.* **2016**, *5*, 874–876.

(5) McLaughlin, J. A.; Misirli, G.; Pocock, M.; Wipat, A. An Environment for Augmented Biodesign Using Integrated Data Resources. Proceedings of 8th International Workshop on Bio-Design Automation. Newcastle University, 2016.

(6) Zhang, M.; McLaughlin, J. A.; Wipat, A.; Myers, C. J. SBOLDesigner 2: An Intuitive Tool for Structural Genetic Design. *ACS Synth. Biol.* **2017**, *6*, 1150–1160.

(7) Wilson, E. H.; Sagawa, S.; Weis, J. W.; Schubert, M. G.; Bissell, M.; Hawthorne, B.; Reeves, C. D.; Dean, J.; Platt, D. Genotype Specification Language. *ACS Synth. Biol.* **2016**, *5*, 471–478.

(8) Bilitchenko, L.; Liu, A.; Cheung, S.; Weeding, E.; Xia, B.; Leguia, M.; Anderson, J. C.; Densmore, D. Eugene – A Domain Specific Language for Specifying and Constraining Synthetic Biological Parts, Devices, and Systems. *PLoS One* **2011**, *6*, e18882.

Graphical TOC Entry