ShortBOL: A shorthand for SBOL

Matthew Pocock*
Turing Ate My Hamster Ltd
turingatemyhamster@gmail.com

Chris Taylor
School of Computing Science, Newcastle University
c.p.d.taylor@newcastle.ac.uk

Göksel Mısırlı
ICOS, School of Computing Science, Newcastle University
goksel.misirli@ncl.ac.uk

James Alastair McLaughlin
ICOS, School of Computing Science, Newcastle University
j.a.mclaughlin@newcastle.ac.uk

Anil Wipat*
ICOS, School of Computing Science, Newcastle University
anil.wipat@ncl.ac.uk

1. INTRODUCTION

Synthetic Biology Open Language (SBOL) version 2 [1] is one of the emerging synthetic biology (synbio) data standards. It facilitates the computational design and exchange of novel, reproducible and composable designed biological systems. SBOL is defined as a data model and RDF/XML serialization. This data model is generic and extensible, but also very explicit, sometimes at the cost of being verbose. A single biological design concept many be captured through multiple SBOL entities, and this level of normalization can make it difficult for a person to edit SBOL data manually. While well-suited for precise machine communication, SBOL RDF/XML is too verbose and complex for humans to directly edit non-trivial designs.

Software tools and libraries are emerging to manipulate SBOL. For example, libSBOLj [5] can be linked to other software, enabling them to read, write and manipulate SBOL data. While libSBOLj supports tools developers, it does not directly help synthetic biologists work with SBOL. CAD and visualisation tools have been developed to visualise designs and make the designs easier for humans to communicate [4, 3, 2]. However, visual design is an essentially manual process. We have identified a need for a light-weight SBOL scripting language that bridges the gap between manual, visusal design, and software development.

Here, we present ShortBOL, a human readable/writable shorthand language for SBOL. This language is developed for synthetic biologists who wish to rapidly sketch synthetic biology designs using a simple, text based scripting language. The terminology used is designed to be much closer to the concepts synthetic biologists have when describing designs. Using this language, design information can be captured more easily and quickly, without limiting the functionality that SBOL already provides.

2. THE SHORTBOL LANGUAGE

ShortBOL is designed to be easy to use for synthetic biologists who may not have much software development training. The language is text based, has a simple syntax that uses white spacing to demarcate blocks, as in popular programming languages such as Python rather than punctuation, as in RDF/Turtle or JSON. A standard template library is provided, covering the SBOL data model and its use for capturing genetic designs. Power users can create new templates to capture abstractions common within their designs or their synbio domain. If these libraries are made available on the Web, they can then be imported and used by others.

The ShortBOL shorthand is built around a minimal selection of language constructs. A typical shorthand document is a list of imports, property assignments and template applications.

Template libraries are pulled into a ShortBOL document using import statements. These libraries are themselves written in shorthand, and declare new templates. Custom templates can be used to provide simple aliases, application-specific syntax, access to common terminologies, and can even be used to model complex parameterised multi-component designs.

Assignment statements 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.

SBOL entities are created within the shorthand by using the colon (`:`) operator to apply a template (Figure 1A). For example, `lacI_cds : CDS` introduces a new identifier `lacI_cds` that will hold the value of expanding the `CDS` template. In this case, the `CDS` template expands to a `SBOL:ComponentDefinition` template with the type set to the `DnaRegion` BioPAX term and role set to the CDS (`SO000316`) Sequence Ontology term, as recommended in the SBOL best practices for encoding a CDS using SBOL (Figure 1B). Some templates are parameterised by one or more arguments. For example, the `DNASequence` template expects a single argument, containing a DNA string. When the `DNASequence` template is expanded, the `elements` property of the resulting `SBOL:Sequence` to that value 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.

Template applications can be directly followed by an indented block of ShortBOL expressions. These are used to declare additional properties and their values. For example, the template application `lacI_cds : CDS` may be followed by an indented block containing the assignment `description = "The lacI CDS"`. This sets the `description` prop-

*To whom correspondance should be addressed

erty of `lacI_cds` to `"The lacI CDS"`.

Figure 1: Rendering SBOL documents using ShortBOL. A genetic circuit representation in ShortSBOL 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.

A)

```
import shorthand:sbol2

lacI_cds : CDS
   description = "The lacI CDS"
   name = "lacI"
   sequence = lacI_seq

lacI_seq : DNASequence("atggtgaatgt")
```

↓ *Template Expansion*

B)

```
lacI_seq : Sequence
   encoding = <SBOL:IUPACDNA>
   displayId = "lacI_seq"
   elements = "atggtgaatgt"

lacI_cds : ComponentDefinition
   role = <SBOL:CDS>
   type = <SBOL:DNA>
   displayId = "lacI_cds"
   description = "The lacI CDS"
   name = "lacI"
   sequence = lacI_seq
```

↓ *Rendering to SBOL RDF/XML*

C)

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

2.1 Expansion

Shorthand documents go through an expansion pipeline, being re-written until they become RDF/XML documents. The steps are as follows:

- *Import processing:* 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.
- *Template registration:* Templates are associated with their identifier, and made available for future application.
- *Variable substitution:* Identifiers are looked up in the current assignment dictionary, and if found, replaced with the corresponding value.
- *Template expansion:* If the name of a template application matches a registered template, expand that template and set all the nested properties.
- *Hoisting:* If a property's value is set to a complex object where a reference was expected (e.g. the sequence property of a ComponentDefinition holds as value a Sequence instance), hoist the value up to the top level and replace the value with a reference.
- *Assigning identifiers:* Mint identifiers (URIs) for any instances that are anonymous in shorthand but need identifiers in an RDF rendering.
- *Transliteration to RDF/XML:* Generate XML/RDF from the processed shorthand script. This is a direct transliteration of the URIs associated with identifiers, property names and remaining template applications to RDF resources, predicates and classes.

3. CONCLUSIONS

ShortBOL fulfills the need for an SBOL shorthand. It is designed to be easy for biologists to read and write, allowing the rapid creation and exchanging of synbio designs without heavyweight computational tools or the need for a mediating GUI. ShortBOL comes with a formal syntax and semantics, so is also suitable for machine exchange. Development of 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.

4. ACKNOWLEDGMENTS

The Engineering and Physical Sciences Research Council grant EP/J02175X/1 (to A.W. and G.M.). JM is supported by FUJIFILM DioSynth biotechnologies.

5. REFERENCES

[1] B. Bartley, J. Beal, K. Clancy, G. Misirli, N. Roehner, E. Oberortner, M. Pocock, M. Bissell, C. Madsen, T. Nguyen, Z. Zhang, J. H. Gennari, C. Myers, A. Wipat, and H. Sauro. Synthetic Biology Open Language (SBOL) Version 2.0.0. *J Integr Bioinform*, 12(2):272, 2015.

[2] L. Clark & Parsia. SBOL design tool powered by semantic technologies. http://clarkparsia.github.io/sbol/.

[3] J. A. McLaughlin, G. Misirli, M. Pocock, and A. Wipat. An environment for augmented biodesign using integrated data resources, submitted. In *8th International Workshop on Bio-Design Automation*, 2016.

[4] J. A. McLaughlin, M. Pocock, G. Misirli, C. Madsen, and A. Wipat. Visbol: Web-based tools for synthetic biology design visualization. *ACS Synth Biol*, 2016.

[5] Z. Zhang, T. Nguyen, N. Roehner, G. Misirli, M. Pocock, E. Oberortner, M. Samineni, Z. Zundel, J. Beal, K. Clancy, A. Wipat, and C. J. Myers. libsbolj 2.0: A java library to support sbol 2.0. *IEEE Life Sci Lett*, 1(4):34–37, Dec 2016.