

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

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



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



Read Online

ACCESS |



Metrics & More



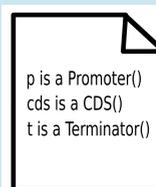
Article Recommendations



Supporting Information

**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*



ShortBOL



```

<<SBOL>
<sbol:ComponentDefinition rdf:about="http://nbold.prefix.org/1">
  <sbol:role rdf:resource="http://nbold.prefix.org/1/0000101">
    <sbol:displayId>cds</sbol:displayId>
    <sbol:version>1</sbol:version>
    <sbol:type rdf:resource="http://www.synbio.org/vocab/synbio.owl#DNA">
      <sbol:persistentIdentity rdf:resource="http://nbold.prefix.org/1">
        <sbol:ComponentDefinition>
          <sbol:role rdf:resource="http://nbold.prefix.org/1">
            <sbol:displayId>cds</sbol:displayId>
            <sbol:version>1</sbol:version>
            <sbol:type rdf:resource="http://www.synbio.org/vocab/synbio.owl#DNA">
              <sbol:persistentIdentity rdf:resource="http://nbold.prefix.org/1">
                <sbol:ComponentDefinition>
                  <sbol:role rdf:resource="http://nbold.prefix.org/1">
                    <sbol:displayId>cds</sbol:displayId>
                    <sbol:version>1</sbol:version>
                    <sbol:persistentIdentity rdf:resource="http://nbold.prefix.org/1">
                      <sbol:ComponentDefinition>
                        <SBOL>

```

SBOL RDF/XML

Synthetic Biology Open Language (SBOL) version 2 has emerged as a data standard for synthetic biology.<sup>1</sup> 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<sup>2</sup> and pySBOL<sup>3</sup> 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.<sup>4–6</sup>

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 lightweight 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 recognized by the synthetic biology community, and languages such as the Genotype Specification Language<sup>7</sup> and Eugene<sup>8</sup> 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 on 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

**Received:** November 22, 2019

**Published:** March 4, 2020

```

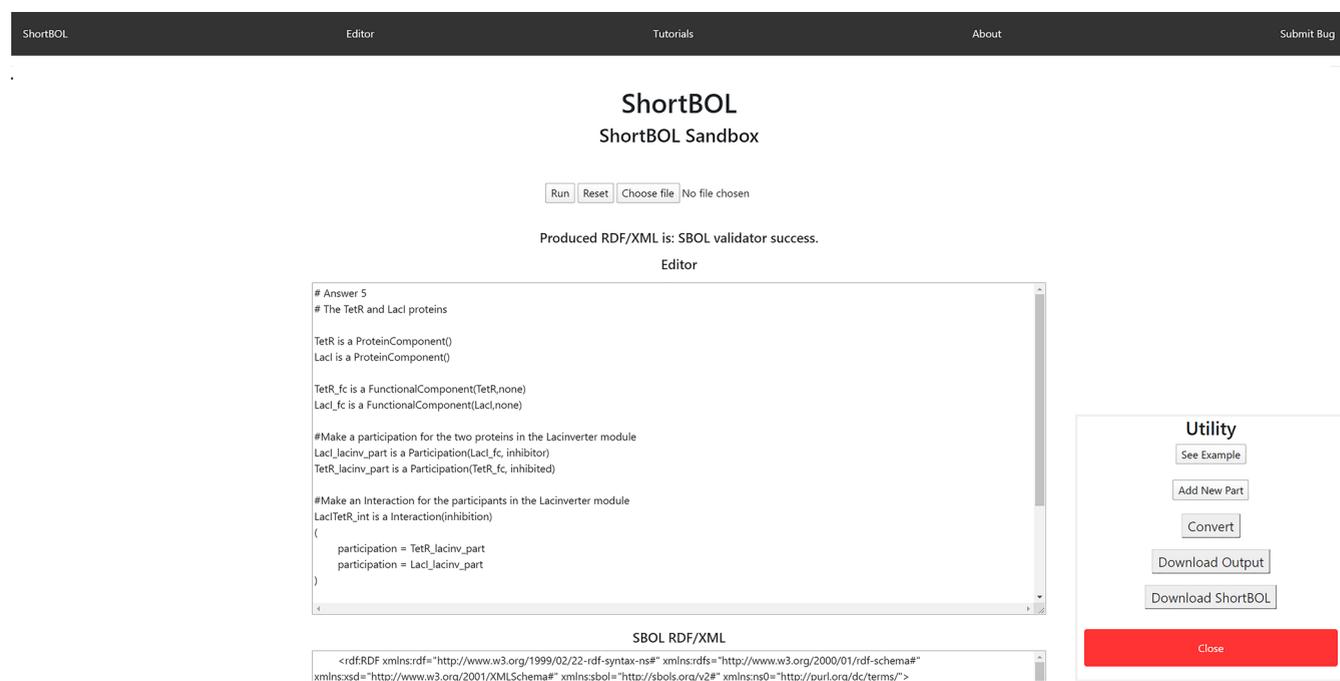
ComponentDefinition(t)
(
  Identified(ComponentDefinition)
  type = t
)

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.



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

70 model can gain exposure to the terminology and approach  
71 without having to work with the SBOL code libraries.

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

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

86 Custom templates can be used to provide simple aliases,  
87 application-specific syntax, access to common terminologies,  
88 and can even be used to model complex parametrized  
89 multicomponent designs. Variable assignments, on the other  
90 hand, associate a value with an identifier, using the equals (=)  
91 operator. For example, repressor = tetR associates the value tetR

with the identifier repressor. This can be used to set up aliases to  
92 provide more natural local names for remotely defined terms  
93 and design components. 94

**ShortBOL Usage.** ShortBOL can be used from both the  
95 command line and from a custom Web application (<http://shortbol.org/>) (Figure 2). The ShortBOL repository on GitHub  
96 includes documentation on how to compile ShortBOL text files  
97 to SBOL XML files using the supplied Python software at the  
98 command line. The web application allows ShortBOL docu-  
99 ments to be written in the web-based editor and automatically  
100 compiled to an SBOL RDF/XML file, which the user can then  
101 download. A tutorial describing how to use ShortBOL is also  
102 provided, which also introduces features of the SBOL data  
103 model. When ShortBOL code is executed *via* the command line  
104 or web application, the output is validated for compliance with  
105 the SBOL specification, ensuring ShortBOL output will  
106 interoperate with other SBOL tooling. 107

**Implementation.** SBOL entities are created within the  
108 shorthand by using the (is a) operator to expand a template  
109 (Figure 3A). For example, lacI\_cds is a CDS introduces a new  
110 identifier lacI\_cds whose properties will be set according to the  
111 pattern described by the CDS template. In this particular case,  
112

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("atgggtgaatgt")

```

B) ↓ *Template Expansion*

```

lacI_seq is a Sequence()
(
  encoding = iupacDNA
  displayId = "lacI_seq"
  elements = "atgggtgaatgt"
)
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>atgggtgaatgt</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.

114 the CDS template further expands to a SBOL:ComponentDe-  
 115 finition template, which sets the type property to the DnaRegion  
 116 BioPAX term and role property to the CDS (SO:000316)  
 117 Sequence Ontology term, as recommended in the SBOL best  
 118 practices for encoding a CDS using SBOL (Figure 3B).  
 119 Templates can also be parametrized by one or more arguments.  
 120 For example, the DNASequence template expects a single  
 121 argument, containing a DNA string. When the template is  
 122 expanded, the elements property of the resulting SBOL:Se-  
 123 quence is set to be equal to the supplied argument. This  
 124 mechanism allows common design and composition patterns to  
 125 be captured relatively easily within templates, without requiring  
 126 a full programming language. In combination with the recursive  
 127 expansion of templates, this can allow collections of specialized,  
 128 domain-specific templates to be composed from generic ones.  
 129 Template expansions can also contain a block of ShortBOL  
 130 expressions. These are used to declare additional properties and  
 131 their values. For example, the template application lacI\_cds is a  
 132 CDS may be followed by a bracketed block containing the  
 133 property assignment description = "The lacI CDS".  
 134 **Interpretation.** The statements contained in shorthand  
 135 documents are interpreted sequentially, and from each template  
 136 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

158 complex computational tools or the need for a mediating GUI.  
 159 ShortBOL comes with a formal syntax and semantics, and so is  
 160 also suitable for machine exchange. ShortBOL is not intended to  
 161 replace SBOL, however, which can represent additional complex  
 162 design information, including material that is not textual or that  
 163 has user-defined semantics. Moreover, SBOL is based on RDF  
 164 and can benefit from existing Semantic Web tooling. Instead, the  
 165 ShortBOL syntax simplifies the creation of SBOL documents. As  
 166 a textual language with a defined syntax, it has the advantage of  
 167 describing design information unambiguously for machines,  
 168 compared to visual languages, which are for human  
 169 consumption.

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

185 Development of a new version that includes the newer  
 186 features above, together with a fully online editor and expansion  
 187 pipeline is ongoing, supporting while-you-type integration with  
 188 other SBOL tooling, including VisBOL.<sup>4</sup> We hope that the open  
 189 nature of ShortBOL template libraries will support rapid  
 190 development of SBOL extensions and domain-specific design  
 191 terminologies. Moreover, we envisage community-driven  
 192 development of template libraries to intuitively design biological  
 193 systems according to the needs of different laboratories.

## 194 ■ ASSOCIATED CONTENT

### 195 ■ Supporting Information

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

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

## 202 ■ AUTHOR INFORMATION

### 203 Corresponding Authors

204 **Angel Goñi-Moreno** – School of Computing, Newcastle  
 205 University, Newcastle upon Tyne NE4 5TG, U.K.; [orcid.org/0000-0002-2097-2507](https://orcid.org/0000-0002-2097-2507); Email: [angel.goni-moreno@ncl.ac.uk](mailto:angel.goni-moreno@ncl.ac.uk)  
 206  
 207 **Anil Wipat** – School of Computing, Newcastle University,  
 208 Newcastle upon Tyne NE4 5TG, U.K.; [orcid.org/0000-0001-7310-4191](https://orcid.org/0000-0001-7310-4191); Email: [anil.wipat@ncl.ac.uk](mailto:anil.wipat@ncl.ac.uk)  
 209

### 210 Authors

211 **Matthew Crowther** – School of Computing, Newcastle University,  
 212 Newcastle upon Tyne NE4 5TG, U.K.  
 213 **Lewis Grozinger** – School of Computing, Newcastle University,  
 214 Newcastle upon Tyne NE4 5TG, U.K.; [orcid.org/0000-0002-9024-701X](https://orcid.org/0000-0002-9024-701X)  
 215

**Matthew Pocock** – School of Computing, Newcastle University,  
 Newcastle upon Tyne NE4 5TG, U.K. 216  
 217  
**Christopher P. D. Taylor** – School of Computing, Newcastle  
 University, Newcastle upon Tyne NE4 5TG, U.K. 218  
 219  
**James A. McLaughlin** – School of Computing, Newcastle  
 University, Newcastle upon Tyne NE4 5TG, U.K. 220  
 221  
**Göksel Misirli** – School of Computing and Mathematics, Keele  
 University, Keele, Newcastle ST5 5BG, U.K. 222  
 223  
**Bryan Bartley** – Raytheon BBN Technologies, Cambridge,  
 Massachusetts 02138, United States; [orcid.org/0000-0002-1597-4022](https://orcid.org/0000-0002-1597-4022) 224  
 225  
**Jacob Beal** – Raytheon BBN Technologies, Cambridge,  
 Massachusetts 02138, United States 226  
 227  
 228

Complete contact information is available at: 229  
<https://pubs.acs.org/10.1021/acssynbio.9b00470> 230

## 231 Author Contributions

232 <sup>||</sup>M.C., L.G., M.P., and C.T. contributed equally to the project. 232  
 233 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 234  
 proof-of-concept implementation. L.G. and M.C. developed the 235  
 Python implementation described in this paper. M.P., G.M., 236  
 A.G.-M., J.B., B.B., and A.W. wrote the paper and tested the 237  
 system. M.P., L.G., M.C., and A.W. wrote the documentation. 238  
 A.W., M.P., and A.G.-M. supervised the project. 239

## 240 Notes

The authors declare no competing financial interest. 241

## 242 ■ ACKNOWLEDGMENTS

The authors of this work are supported by The Engineering and 243  
 Physical Sciences Research Council grants EP/J02175X/1, EP/ 244  
 R003629/1, EP/N031962/1 (J.A.M. and A.W.), and EP/ 245  
 R019002/1 (A.G.-M.), EPSRC studentship 34000024085 246  
 (M.C.), and the European CSA 820699 (A.G.-M. and A.W.). 247  
 J.B. and B.B. are supported in part by the Air Force Research 248  
 Laboratory (AFRL) and DARPA under contract 249  
 FA875017CO184. J.A.M. is supported by FUJIFILM DioSynth 250  
 Biotechnologies and M.C. by Doulix Ltd. M.C. is supported by 251  
 the EPS. Any opinions, findings, and conclusions or recom- 252  
 mendations expressed in this material are those of the author(s) 253  
 and do not necessarily reflect the views of the funding agencies. 254  
 This document does not contain technology or technical data 255  
 controlled under either the U.S. International Traffic in Arms 256  
 Regulations or the U.S. Export Administration Regulations. 257

## 258 ■ REFERENCES

- 259 (1) Madsen, C., et al. (2019) Synthetic Biology Open Language  
 (SBOL) Version 2.3. *J. Integr. Bioinform.* 16, 25. 260
- 261 (2) 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. J. (2015) libSBOLj 2.0: A Java Library to Support SBOL  
 2.0. *IEEE Life Sci. Lett.* 1, 34–37. 262  
 263
- 264 (3) Bartley, B. A., Choi, K., Samineni, M., Zundel, Z., Nguyen, T.,  
 Myers, C. J., and Sauro, H. M. (2019) pySBOL: A Python Package for  
 Genetic Design Automation and Standardization. *ACS Synth. Biol.* 8,  
 1515–1518. 265  
 266  
 267
- 268 (4) McLaughlin, J. A., Pocock, M., Misirli, G., Madsen, C., and Wipat,  
 A. (2016) VisBOL: Web-based tools for synthetic biology design  
 visualization. *ACS Synth. Biol.* 5, 874–876. 269  
 270
- 271 (5) McLaughlin, J. A., Misirli, G., Pocock, M., and Wipat, A. (2016)  
 An Environment for Augmented Biodesign Using Integrated Data  
 Resources. *Proceedings of 8th International Workshop on Bio-Design  
 Automation*. Newcastle University. 272  
 273  
 274  
 275

- 276 (6) Zhang, M., McLaughlin, J. A., Wipat, A., and Myers, C. J. (2017)  
277 SBOLDesigner 2: An Intuitive Tool for Structural Genetic Design. *ACS*  
278 *Synth. Biol.* 6, 1150–1160.
- 279 (7) Wilson, E. H., Sagawa, S., Weis, J. W., Schubert, M. G., Bissell, M.,  
280 Hawthorne, B., Reeves, C. D., Dean, J., and Platt, D. (2016) Genotype  
281 Specification Language. *ACS Synth. Biol.* 5, 471–478.
- 282 (8) Bilitchenko, L., Liu, A., Cheung, S., Weeding, E., Xia, B., Leguia,  
283 M., Anderson, J. C., and Densmore, D. (2011) Eugene – A Domain  
284 Specific Language for Specifying and Constraining Synthetic Biological  
285 Parts, Devices, and Systems. *PLoS One* 6, No. e18882.