# Sharing Structure and Function in Biological Design with SBOL 2.0

Nicholas Roehner,[*,†] Jacob Beal,[‡] Kevin Clancy,[¶] Bryan Bartley,[§] Goksel Misirli,[‖] Raik Grünberg,[⊥] Ernst Oberortner,[#] Matthew Pocock,[@] Michael Bissell,[△] Curtis Madsen,[‖] Tramy Nguyen,[▽] Michael Zhang,[▽] Zhen Zhang,[▽] Zach Zundel,[††] Douglas Densmore,[†] John H. Gennari,[‡‡] Anil Wipat,[‖] Herbert M. Sauro,[§] and Chris J. Myers[▽]

*Department of Electrical and Computer Engineering, Boston University, Boston, MA, Raytheon BBN Technologies, Cambridge, MA, Thermo Fisher Scientific, Carlsbad, CA, Department of Bioengineering, University of Washington, Seattle, WA, School of Computing Science, Newcastle University, Newcastle upon Tyne, UK, Institute for Research in Immunology and Cancer, University of Montreal, Montreal, Canada, U.S. Department of Energy Joint Genome Institute, Walnut Creek, CA, Turing Ate My Hamster Ltd, Newcastle upon Tyne, UK, Amyris, Inc., Emeryville, CA, Department of Electrical and Computer Engineering, University of Utah, Salt Lake City, UT, Department of Bioengineering, University of Utah, Salt Lake City, UT, and Department of Biomedical Informatics and Medical Education, University of Washington, Seattle, WA*

E-mail: nicholasroehner@gmail.com

**Abstract**

The *Synthetic Biology Open Language* (SBOL) is a standard that enables collaborative engineering of biological systems across different institutions and tools. SBOL is developed through careful consideration of recent synthetic biology trends, real use cases, and consensus among leading researchers in the field and members of commercial biotechnology enterprises. We demonstrate and discuss how a set of SBOL-enabled software tools can form an integrated, cross-organizational workflow to recapitulate the design of one of the largest published genetic circuits to date, a 4-input AND sensor. This design encompasses the structural components of the system, such as its DNA, RNA, small molecules, and proteins, as well as the interactions between these components that determine the system's behavior/function. The demonstrated workflow and resulting circuit design illustrate the utility of SBOL 2.0 in automating the exchange of structural and functional specifications for genetic parts, devices, and the biological systems in which they operate.

# Introduction

Synthetic biology is a maturing scientific discipline that combines science and engineering in order to design, build, and test novel biological systems aimed at a broad range of applications (*1–3*). This includes the design and construction of entirely new synthetic biological components and devices (*4–6*), as well as the redesign of biological systems found in na-

*To whom correspondence should be addressed
†Boston University
‡Raytheon BBN Technologies
¶Thermo Fisher Scientific
§University of Washington
‖Newcastle University
⊥University of Montreal
#DOE Joint Genome Institute
@Turing Ate My Hamster Ltd
△Amyris, Inc.
▽University of Utah
††University of Utah
‡‡University of Washington

ture (*7–9*). Synthetic biology is often characterized in terms of the following key concepts: predictable off-the-shelf components (such as parts and devices) with standard connections, robust biological chassis (such as yeast, *E. coli*, and mammalian cells) that readily host these components, standardized techniques for assembling components into increasingly sophisticated functional systems (*10*, *11*), and standardized languages for specifying designs in order to facilitate their exchange between investigators (*12–17*).

A vital aspect of using standardized languages to specify designs is the ability to share information on both the intended structure and function of a design (*18*). Sharing this information facilitates useful collaboration between a wide variety of researchers and organizations, satisfying the needs of both individual projects and related projects with common data requirements. Examples of potential collaborators include experts on the design of biological devices and systems, providers of genetic components, and experts on the process of assembling genetic components into functioning devices and systems. Researchers who share a design across different projects need information on both the structure and function of the design's constituent elements so that they can better predict experimental behavior, understand experimental outcomes, and rectify any shortcomings of the design. Furthermore, as designs are communicated via publications or deposited into data repositories by scientists and engineers (for example, see (*19–21*)), their long-term understanding and reuse are dependent upon recording both their structural and functional details and the contextual dependencies (*22*) of these details.

The difficulty of representing both the structural and the functional aspects of a design is particularly acute in the biological sciences, where designs are transformed into living systems and then proceed to interact in often poorly-understood ways with naturally occurring cellular systems. The nature of biological systems is such that a design may have to account for many different types of molecules and their functional interactions, including DNA, RNA, proteins, and small molecules, among others. Previously, biologists have had access to standards for effectively sharing sequence information ("structure") (*23*, *24*) or

systems information ("function") (*25*, *26*). These standards, however, lack the ability to represent both types of information together in a manner that is well-suited to synthetic biology.

The most basic and widely used standard sequence formats, such as FASTA (*23*), describe only raw nucleotide sequences. Another familiar data standard for many biologists is GenBank (*24*), which represents sequences as linearly ordered sequence annotations that describe local sequence features, such as coding sequences (CDS) or promoters. However, GenBank cannot represent the higher-order organization of a design, nor partially complete designs in which the length and arrangement of sub-sequences have not yet been fully determined.

Previous representations of the function of genetic constructs have come primarily from systems biology and have focused on mathematically modeling the interactions between the gene products of constructs and other molecules in the host cell or environment, such as interactions between proteins and metabolites. However, current standards for systems biology, such as the *Systems Biology Markup Language* (SBML) (*25*) and the *Biological Pathway Exchange* (BioPAX) (*26*), do not explicitly represent the interaction of genetic structures with other cellular species very well. For example, they typically cannot readily represent the genetic architecture and sequence design involved in the interaction of a transcription factor and RNA polymerase with a promoter to initiate transcription.

To better communicate designs and realize the vision of predictable engineering of biological systems (*27*), an open consortium of members of the synthetic biology community is developing the *Synthetic Biology Open Language* (SBOL). SBOL version 1.1 (*14*) standardized the exchange of structural specifications of synthetic genetic designs, improving over formats such as GenBank by capturing the hierarchical and heterarchical organization of these designs. This includes the organization of DNA components into composite DNA components that represent transcriptional units, operons, and plasmids, as well as partially complete designs with sequences that are not yet fully determined.

The recently released SBOL 2.0 (*13*) extends the structural representation of SBOL to non-DNA components and combines this structural, sequence feature-based representation with the core aspects of functional, systems biology-based representations, in particular molecular interactions and biological modules with inputs and outputs. This joint structural-functional representation enables synthetic biologists to share the lower- and higher-order structure and function of any biological system and identify key relevant features or potential problems in its design.

SBOL 2.0 was accepted by the synthetic biology community after several proposals spanning a period of four years since the release of SBOL 1.0. Some of these proposals (*28*, *29*) were documented and published, eventually contributing to the official SBOL 2.0 specification (*13*). SBOL 2.0 is already used in several software tools and it is expected that the number of tools adopting the standard will continue to grow. Software libraries that facilitate the use of SBOL by these tools are already freely available in different programming languages and are listed on the SBOL website[1].

This paper examines how SBOL 2.0 can enable an integrated workflow involving a variety of tools and organizations. Following a brief review of the key classes of information represented in SBOL 2.0, this paper analyzes two use cases. The first use case is a microRNA detector inspired by RNA logic circuits that were recently used to detect cells with a cancerous phenotype (*30*). The detector circuit described here is a simplified version of the original, consisting of a microRNA detector that turns off a reporter gene in the presence of a cognate microRNA. Although relatively straightforward, no prior standard format could be used to specify this design, highlighting the importance of the SBOL data model and its ability to connect biological structure and function, as well as its versatility in representing data from synthetic biology, systems biology, and biotechnology. The second use case recapitulates the design of a 4-input AND sensor (*31*), one of the largest synthetic systems constructed to date. This example illustrates how the SBOL data model can support emerging trends

[1] `http://www.sbolstandard.org`

towards increasingly sophisticated and complex biological engineering, demonstrating how multiple organizations using separate software tools can execute an integrated workflow that includes both sequence design (CAD) and functional simulation (CAE).

## Overview of SBOL 2.0

Here, we briefly introduce the core concepts of SBOL 2.0 in order to better explain the use cases that follow. In general, SBOL 2.0 not only enables the representation of structural features of genetic designs and their composition, but also makes it possible to document the functional roles of designs and the interactions between their constituent DNA, RNA, protein, and small molecule components, among others. The primary classes of information captured by version 2.0 of the SBOL data model and their relationships to each other are outlined in Figure 1.

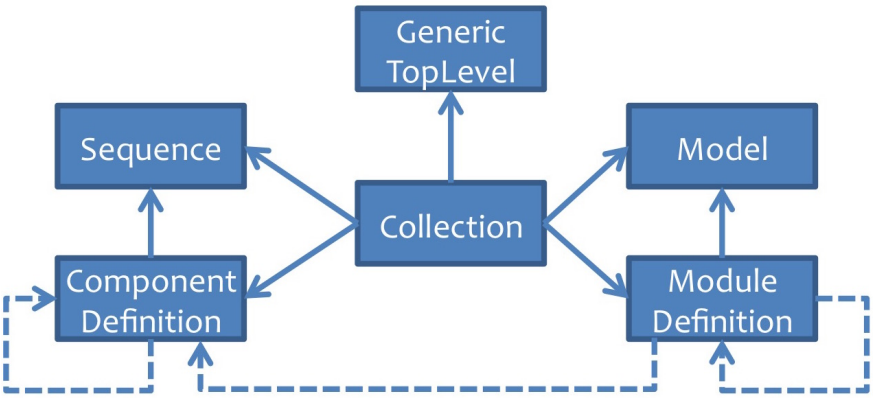


Figure 1: Primary classes of information represented by version 2.0 of the SBOL standard and their relationships. Solid lines represent direct relationships between these classes, whereas dashed lines indicate indirect relationships with the help of intermediate classes that are not shown here. The **ComponentDefinition** and **ModuleDefinition** classes are the basic units for the hierarchical composition of genetic structure and function, respectively. Primary structures, such as nucleotide sequences, are represented using the **Sequence** class. The **Model** class links a **ModuleDefinition** with quantitative or qualitative models in other standard formats, such as SBML. These primary classes can be grouped using the **Collection** class. Application-specific data that are outside the scope of the SBOL data model can be captured using the **GenericTopLevel** class.

SBOL 2.0 includes two main classes that match the structural/functional distinction

above:

- The **ComponentDefinition** class describes the structural aspects of a genetic design, such as its nucleic acid sequences, proteins, and metabolites. Each **ComponentDefinition** can be associated with a **Sequence** in order to specify its primary structure. In addition, **ComponentDefinition**s use terms from ontologies (controlled vocabularies) to document their type (for example, DNA, RNA, or protein) and biological role(s) (for example, promoter, mRNA, or transcription factor). Lastly, as the structure of a design scales in complexity, **ComponentDefinition**s can be reused and organized into a hierarchy of subcomponents. The exact locations of subcomponents or regions of interest can be specified using sequence annotations, while partial orderings of subcomponents can be specified using sequence constraints.

- The **ModuleDefinition** class effectively groups components that work together to perform an intended function and describes the component **Interactions** required to implement this function, such as binding reactions or repression and activation relationships. Each **ModuleDefinition** can also use the **Model** class to document and link to external models written in standards other than SBOL, such as SBML (*25*) and CellML (*32*). Finally, as the function of a design scales in complexity, **ModuleDefinition**s can be reused and organized into a hierarchy of submodules that documents the connections between the input and output components of each submodule.

A microRNA (miRNA) detector is an example of a device with a diversity of biological structures and functions that can be specified using SBOL 2.0. Figure 2 is a combination of a genetic circuit diagram and a pathway/network diagram that illustrates how SBOL can be used in this way, integrating knowledge from both synthetic biology and systems biology. In particular, Figure 2 shows how the miRNA detector can be specified abstractly as a **Module Definition** with a miRNA input and an X-gal output, and in greater detail as a set of DNA, RNA, and protein components and their **Interaction**s, including transcription, translation,

catalysis, and degradation.

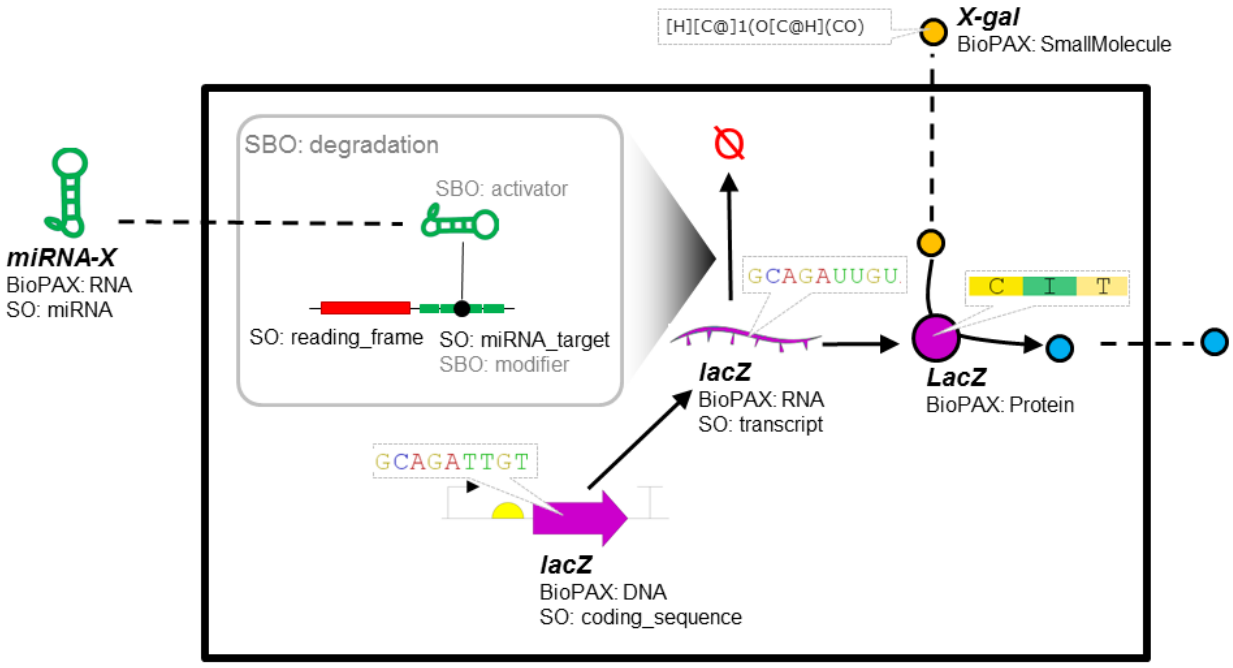


Figure 2: Visualization of using the SBOL 2.0 data model to capture the design of a miRNA detector. In this design, a miRNA input shuts off a lacZ reporter by activating the degradation of its mRNA. **ComponentDefinition**s are represented using colored glyphs labeled with SO and BioPAX terms and associated with structural data (primary sequences). Each DNA **ComponentDefinition** is represented using a glyph from the SBOL Visual standard (*33*, *34*). The degradation of the lacZ transcript is an **Interaction** activated by a miRNA participant and is represented by a gray box labeled with SBO terms. This **Interaction** is activated by a miRNA input. Other **Interactions** are represented by arrows, including transcription, translation, and enzyme catalysis (which converts the substrate X-gal into a blue product). The bounding black box represents the overall **ModuleDefinition**, with inputs and outputs indicated by dashed lines.

Also included in Figure 2 are the ontology terms that decorate each SBOL entity and describe their types and roles. Specifically, terms taken from the *BioPAX* ontology and *Sequence Ontology* (SO) (*35*) indicate the type and role of each **ComponentDefinition**, respectively, while terms taken from the *System Biology Ontology* (SBO) (*36*) document the different types of **Interaction**s and the roles that components play in them. The SBOL 2.0 standard unifies many different ontologies into a coherent model for biological design, combining perspectives from synthetic and systems biology and demonstrating interoperabil-

ity between many different standards. Interoperability is crucial to a strong and versatile standard that fulfills many different use cases from related fields. This spirit empowers the SBOL standard to serve as a platform for CAD, automation, and closer collaboration in the synthetic biology community. A more detailed description of the SBOL 2.0 data model can be found in the Supporting Information and the official SBOL 2.0 specification (*13*).

# Results

A standard can enable and aid in the automated orchestration of multiple software tools at collaborating organizations. In addition, the ability of a standard to associate structural, manufacturing specifications with a parameterized design space has proven extremely valuable to many other engineering disciplines. Such advances in automated design have revolutionized printed circuit board design, for example. The SBOL 2.0 standard enables similar advances in bioengineering.

In Figure 3 we illustrate the workflow capabilities of SBOL 2.0 with respect to state-of-the-art synthetic biology design. The workflow follows a formalized engineering approach that couples computer-aided engineering (CAE) (the dynamic simulation of systems) with computer-aided design (CAD) (the physical specification of a design for manufacturing). This extends our previous work using SBOL 1.1 to couple CAD with computer-aided manufacturing (CAM) (the assembly and synthesis of DNA), which is an integral task in synthetic biology.

In particular, Figure 3 shows one iteration in which a team of synthetic biologists cooperates to design the structure and function of a genetic circuit from the bottom up. Given an abstract functional specification for a genetic logic circuit, the team obtains the sequences of the requisite DNA components from public repositories and then composes the DNA components into genetic constructs. These intermediate constructs are then passed to circuit CAE tools to compose a genetic circuit design and link it to a mathematical model describing the

system's behavior. Ultimately, the team stores the circuit designs and its associated system models in appropriate repositories in order to support the reuse of the designs in similar experiments and more complex designs.

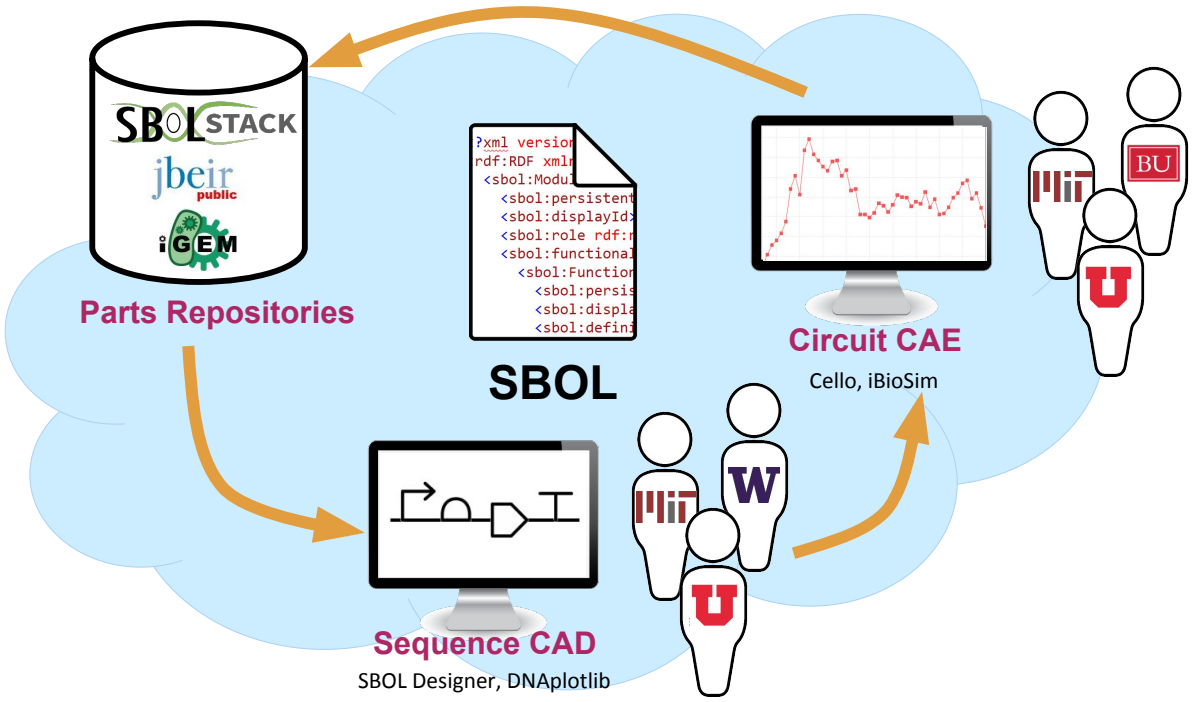


Figure 3: An SBOL-enabled workflow for computer-aided design (CAD) and engineering (CAE) of genetic circuits. Examples of relevant SBOL-compliant tools, repositories, and organizations that develop the tools are shown next to each step in the workflow. The individual tools are described in greater detail later when discussing the application of the workflow to the design of a 4-input AND sensor.

## Design of a 4-input AND Sensor

As an example of what can be accomplished using the SBOL-enabled workflow described in the previous section, we now consider applying this workflow to the design of a 4-input AND sensor that was originally designed and tested by Moon *et al.* (*31*). This example demonstrates:

1. Storage and distribution of SBOL data and linked data from other standards via public or shareable repositories.

2. The ability of SBOL 2.0 to connect structural design tools (for example, sequence editors and construct design tools) and functional design tools (for example, circuit design tools and simulators).

3. The backwards compatibility of SBOL 2.0, which allows for continued integration of SBOL 1.1 tools and files.

4. The interchange of SBOL 2.0 with more quantitative function-centric standards, such as SBML (*25*) and CellML (*32*), as well as with structure-centric standards, such as FASTA (*23*) and GenBank (*24*) (via their conversion to SBOL 1.1).

As shown in Figure 4, the first step in the applied workflow for designing the 4-input AND sensor is to obtain the necessary DNA components from publicly accessible repositories. All named promoters and CDSs were obtained from the JBEI-ICE repository (*19*) as SBOL 1.1 files. Unfortunately, the original publication did not provide identifiers for the constitutive promoter, ribosome binding sites (RBSs), and terminators in this circuit, and its authors were unable to locate the sequences for these components when contacted. This experience motivates the importance of using standards such as SBOL to capture design information at the time of publication. As a workaround, we obtained DNA sequences for a constitutive promoter, ribosome binding site, and terminator from the iGEM registry[2] (*37*).

During the second step of the applied workflow, these DNA components are composed into the constructs making up the 4-input AND sensor using the University of Washington's SBOL Designer tool[3] and a version of MIT's DNAplotlib[4] (modified by the University of Washington). In particular, the modified DNAplotlib was used to compose the constructs for the *araC* and *ipgC* genes, while SBOL Designer was used to compose all other constructs. The resulting composite DNA components are then exported as SBOL 1.1 files and sent to the University of Utah for the third step of the applied workflow, circuit design.

[2] `http://parts.igem.org/`
[3] `http://clarkparsia.github.io/sbol`
[4] `https://github.com/SynBioDex/dnaplotlib`

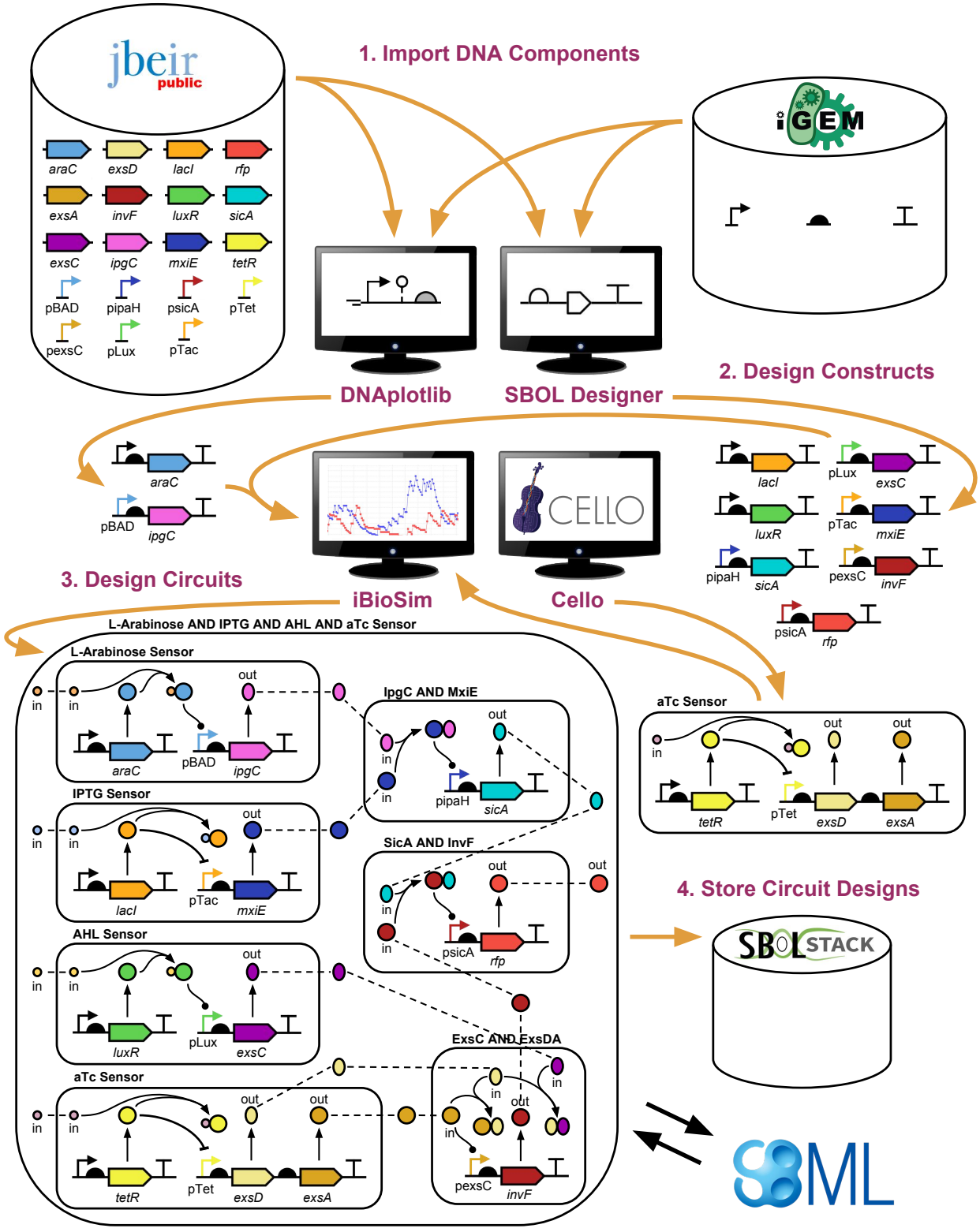


Figure 4: SBOL-enabled workflow for genetic circuit design applied to a 4-input AND sensor.

During the circuit design step, all constructs designed in previous steps are imported into the University of Utah's iBioSim tool (*38*). Note that during import, all SBOL 1.1 files are converted to SBOL 2.0 using native functionality available in the Java library for SBOL 2.0, libSBOLj (*39*). This conversion is necessary to enable the addition of new classes of data only found in SBOL 2.0, including non-DNA components, modules with designated inputs and outputs, and interactions between components. The entire 4-input sensor genetic circuit design was composed using iBioSim, with the exception of the aTc sensor, which was designed using MIT/Boston University's Cello software[5] (*40*, *41*) and exported to iBioSim as SBOL 2.0. Simultaneously, iBioSim is used to construct a quantitative model of the 4-input AND sensor's behavior written in SBML (*25*). This SBML model is then simulated to check its functional correctness for different parameter sets. In order to link the SBOL circuit design to its SBML model, an instance of the SBOL **Model** class is included in the SBOL circuit design. To link in the opposite direction, the SBML model is annotated with SBOL using the scheme described in (*42*).

During the final step of the applied workflow, the SBOL files encoding the 4-input AND sensor circuit design are deposited into Newcastle University's SBOL Stack repository (*21*). Consequently, the qualitative structure and function of the genetic circuit design are documented in a publicly accessible resource, where they can be reused in other designs and linked by referring publications. Both the SBOL and SBML files for the 4-input AND sensor are available in the Supporting Information of this manuscript.

# Discussion

Through collaborative recapitulation of the design of a 4-input AND sensor, one of the largest genetic circuits to date (*31*), we have demonstrated how SBOL 2.0 can enable workflows for rapidly creating and sharing more diverse, complete descriptions of structure and function in biological design. Notably, once the necessary interfaces between people and software were

[5] `https://github.com/CIDARLAB/cello`

established to support the workflow described in this manuscript, the actual design process was able to be executed quickly, taking less than a day.

The ability to establish efficient workflows such as this represents significant progress toward one of the core goals of synthetic biology software development: to give users the ability to design a system, demonstrate its functionality through simulation, and associate this functional design with the structural elements needed to assemble it. As in other areas of engineering, it is expected that flexible workflows comprising many tools are likely to be much more effective and sustainable than monolithic tools. As such, a key need to meet this goal has been the development of a suitable data exchange format that would permit the community to exchange both functional and structural information in a defined, extensible, and shareable format. The SBOL 2.0 standard is extremely versatile, permitting synthetic biologists to create, share, and publish functional and structural descriptions of designs ranging from simple DNA components to genes, networks, pathways, and whole cells. The standard also enables scientists to link these descriptions to quantitative behavioral models written in other standards, such as CellML (*43*), SBML (*44*), and BNGL (*45*).

Even a versatile standard like SBOL 2.0, however, does not completely solve the problem of information exchange. As previously discussed, a key strength of SBOL 2.0 is that it permits users to represent biological designs in nearly any hierarchical, modular form that they might require, provided that the appropriate ontology terms for specifying the biological structures and functions that make up the design are available. Thus, groups of users can easily reuse and extend their designs as their projects progress. This freedom, however, can also interfere with the ability to exchange information. For example, if two groups choose different ontologies to represent information about the function of components in their system, then it may not be clear how to translate between those representations. To mitigate this concern, we have incorporated recommendations into the standard on which ontology terms to use and how to use them, recommendations which were instrumental for realizing and executing the workflow reported in this paper.

As workflows continue to be developed and integrated across more use cases and domains, such recommendations and best practices will need to be further elaborated. We anticipate that decisions of this type will best be informed by the actual experiences of various working groups as the standard becomes more widely used. The lessons learned will inform the future development of the SBOL standard, especially in codifying different patterns of SBOL usage for different subfields of synthetic biology, such as the engineering of biosynthetic pathways, genetic circuits, and protein-protein interaction networks. The continued development of new tools that use SBOL and their adoption by the wider scientific community will thus also be critical to improving the standard.

During the development of the demonstrated workflow, we have found that many software tools and repositories still support the export of data conforming to the SBOL 1.1 specification. The primary reason for this is that the SBOL 2.0 specification is fairly new. Therefore, many of the DNA sequences exchanged in the workflow are captured in SBOL 1.1 files that are later automatically converted to SBOL 2.0 files using libSBOLj (*39*), for example. The support of automated conversion between different versions of SBOL ensures backward compatibility of tools and files and eases the process of upgrading to the most current version of the standard.

By encouraging the adoption and use of version 2.0 of the SBOL standard, we seek to improve the reproducibility of results in the field by supporting the development of tools that aid in the engineering of biological systems. We encourage biologists, developers, and managers interested in using our tools to visit the SBOL website, `sbolstandard.org`. Practitioners who wish to actively participate in the development of this standard should contact the SBOL editors at `editors@sbolstandard.org`.

Finally, the SBOL Developers Group will continue its open community process of developing extensions to the SBOL standard beyond version 2.0. For example, there is a growing consensus within the community concerning the need to specify the context-dependence of biological designs. That is, there is a clear need to express the fact that designs will often

work in one host organism, but work differently, or not at all, in a second host. To address this issue, a working group has been formed to discuss how to express this information within SBOL 2.0 and what modifications are necessary to support the specification of host context. Similar working groups are likely to be formed in other areas, such as the specification of assembly protocols, sequencing results, and combinatorial library design. These working groups will be driven by the evolving priorities and needs of the community, either to develop extensions to the SBOL standard or to develop best practices for interfacing SBOL with other standards that represent and communicate data relevant to biological design.

# Methods

## DNAplotlib

The genetic constructs for the arabinose sensor module were composed programmatically using Python scripts and DNAplotlib, an open-source Python package under development at MIT. DNAplotlib makes it easy to create custom figures for data visualization and publication and was used to generate the SBOL Visual (*33*, *34*) glyphs in the figures of this manuscript. DNAplotlib has undergone further extension at the University of Washington to support reading and writing of SBOL version 1.1 and rendering of SBOL data.

The promoters and coding sequences of the sensor were imported into DNAplotlib from the JBEI-ICE repository as SBOL files. The ribosome binding sites and terminators were imported separately as SBOL files from the iGEM parts registry. These components were then composed into a single construct and exported as an SBOL version 1.1 file.

## SBOL Designer

SBOL Designer is a user-friendly tool for creating and modifying genetic constructs that make up genetic circuits and other biological systems. Individual DNA components are displayed as SBOL Visual (*33*, *34*) glyphs in a straightforward graphical user interface. DNA

components can be imported from repositories or created directly within SBOL Designer before being composed hierarchically with or without complete sequences. After a design is finished, SBOL Designer can export it as an SBOL file.

To design many of the genetic constructs for the 4-input AND sensor, sequences for individual promoters, RBSs, etc., were obtained from the iGEM registry and used to create elementary DNA components in SBOL Designer. Other DNA components were imported as SBOL files from the JBEI-ICE repository. These components were then composed using SBOL Designer's point-and-click interface to form composite DNA components for each gene and exported as SBOL documents to iBioSim. Note that SBOL Designer is only capable of creating and exporting data adhering to version 1.1 of the SBOL data model. In order to add information on higher-order function, this data must be converted to SBOL 2.0 using libSBOLj (*39*), as done in this workflow with iBioSim. A newer version of SBOL Designer[6] is currently being developed at the University of Utah to support SBOL 2.0 and integrate with the SBOL Stack, a repository in this workflow.

## Cello

Cello (*40*, *41*) is a tool developed at Boston University and MIT for automatically converting combinational logic specifications written in Verilog (*46*) to complete DNA sequences encoding transcriptional logic circuits. Cello takes as input a Verilog file and a User Constraint File (UCF) containing data on the structure and performance of a library of genetic NOT and NOR gates, input promoters, and output CDSs. Cello can export data in a variety of formats, including Eugene (*15*) and SBOL 2.0.

As part of the workflow described in this manuscript, Cello was used to design the aTc sensor module from a Verilog file describing an inverter and a custom UCF extending the default Cello UCF with a constitutive input promoter and the *exsDA* output CDSs (see Figure 4). Typically, Cello is used to design transcriptional logic circuits that use

[6] `https://github.com/SynBioDex/SBOLDesigner`

inducible promoters as inputs, leaving out the portions of the circuit that express the small molecule-sensitive transcription factors that regulate these promoters. For the purposes of this manuscript, a constitutive input promoter was added to the UCF in order to recapitulate the design of the 4-input AND sensor as presented in (*31*). Also note that Cello can be used to design larger circuits than the aTc sensor module shown here.

## iBioSim

The last tool in the workflow is `iBioSim` (*38*). iBioSim is a *Genetic Design Automation (GDA)* tool developed at the University of Utah that is capable of modeling, visualizing, and simulating genetic circuits. Genetic circuits are primarily represented in iBioSim using SBML (*25*). While SBOL is used to represent the qualitative structure and function of these circuits, SBML is used to mathematically define their quantitative behavior. Since both representations are useful for different purposes, converters between the two standards have been developed and implemented in iBioSim (*47*, *48*).

In the genetic circuit design workflow discussed in this manuscript, the 4-input AND sensor was modeled in iBioSim (*38*) as SBML annotated with SBOL (*42*) and SBO (*36*) terms to indicate which chemical species and reactions represent genetic components and their interactions. This annotation scheme facilitates the conversion between SBML and SBOL. In the case of the 4-input AND sensor, its SBML model was manually annotated with DNA components and constructs imported from SBOL Designer and DNAplotlib as SBOL 1.1 files and converted to SBOL 2.0 using libSBOLj (*39*). Note that the SBML for the aTc sensor did not need to be manually annotated in this way, since it was imported from Cello as SBOL 2.0 and converted to SBOL-annotated SBML using the SBOL-to-SBML converter described in (*48*). Once composed, the overall SBOL-annotated SBML model for the whole 4-input AND sensor was converted to a SBOL 2.0 file using the SBML-to-SBOL converter described in (*47*) and stored in the SBOL Stack repository (*21*).

## SBOL Stack

The SBOL Stack (*21*) is a *Resource Description Framework* (RDF) (*49*) database developed by Newcastle University that has been specifically designed for storing and sharing SBOL data. Data in the Stack is stored in triplestore repositories where it can be accessed using SPARQL queries (*50*), a provided RESTful API, or a provided Web interface. The SBOL Stack takes advantage of the fact that SBOL data is RDF and stores that data as RDF triples instead of storing actual files. By storing the data as triples, the Stack is able to perform automatic data integration. Thus, data attached to resources that use the same URI are combined to help form a more complete picture of the resource.

In the workflow presented in this paper, the SBOL files describing the 4-input AND sensor designed and constructed by the other tools were deposited into Newcastle's publicly facing instance of the SBOL Stack. This RDF data is now available for users to query and download using one of the aforementioned methods of accessing the data in the Stack. Additionally, if more data is uploaded to the Stack concerning any of the components used in the 4-input AND sensor, then this data is automatically integrated with the data already present, allowing future users to download an SBOL document containing more complete information on each component.

## Acknowledgement

This material is based upon work supported by National Science Foundation under grants No. DBI-1355909 and DBI-1356401, and the Engineering and Physical Sciences Research Council under grant EP/J02175X/1. 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 these funding agencies. We thank Bryan Der (MIT), Oge Nnadi (JBEI), and James McLaughlin (Newcastle University) for their support in demonstrating the workflow presented in this manuscript.

## Supporting Information Available

The supporting information consists of a zip archive that contains a supplementary PDF document and two iBioSim project directories. The supplementary document describes version 2.0 of the SBOL data model, serialization, and libraries in greater detail, while the project directories contain SBOL and SBML files describing the miRNA sensor and 4-input sensor. In addition, the project directory for the 4-input AND sensor contains time-series simulation data generated by iBioSim from the SBML files for the design.

This material is available free of charge via the Internet at `http://pubs.acs.org/`.

## References


1. Benenson, Y. (2012) Biomolecular computing systems: principles, progress and potential. *Nature Reviews of Genetics 13*, 455–468.

2. Woolston, B., Edgar, S., and Stephanopoulos, G. (2013) Metabolic Engineering: past and future. *Annual Review of Chemical and Biomolecular Engineering 4*, 259–288.

3. Markson, J., and Elowitz, M. (2014) Synthetic Biology of Multicellular Systems: New Platforms and Applications for Animal Cells and Organisms. *ACS synthetic biology 3*, 875–876.

4. Rhodius, V., Segall-Shapiro, T., Sharon, B., Ghodasara, A., Orlova, E., Tabakh, H., Burkhardt, D. H., Clancy, K., Peterson, T., Gross, C., and Voigt, C. (2013) Design of orthogonal genetic switches based on a crosstalk map of $\sigma$s, anti-$\sigma$s, and promoters. *Molecular systems biology 9*, 702.

5. Stanton, B., Nielsen, A., Tamsir, A., Clancy, K., Peterson, T., and Voigt, C. (2014) Genomic mining of prokaryotic repressors for orthogonal logic gates. *Nature chemical biology 10*, 99–105.

6. Y-J.Chen,, Liu, P., Nielsen, A., Brophy, J., Clancy, K., Peterson, T., and Voigt, C. (2013) Characterization of 582 natural and synthetic terminators and quantification of their design constraints. *Nature methods 10*, 659–664.

7. Temme, K., Zhao, D., and Voigt, C. (2012) Refactoring the nitrogen fixation gene cluster from Klebsiella oxytoca. *Proceedings of the National Academy of Sciences 109*, 7085–7090.

8. Cummings, M., Breitling, R., and Takano, E. (2014) Steps towards the synthetic biology of polyketide biosynthesis. *FEMS microbiology letters 351*, 116–125.

9. Kim, E., Moore, B. S., and Yoon, Y. (2015) Reinvigorating natural product combinatorial biosynthesis with synthetic biology. *Nature chemical biology 11*, 649–659.

10. Casini, A., Storch, M., Baldwin, G. S., and Ellis, T. (2015) Bricks and blueprints: methods and standards for DNA assembly. *Nature Reviews Molecular Cell Biology 16*, 568–576.

11. Appleton, E., Tao, J., Haddock, T., and Densmore, D. (2014) Interactive assembly algorithms for molecular cloning. *Nature Methods 11*, 657–662.

12. Wilson, E. H., Sagawa, S., Weis, J. W., Schubert, M. G., Bissell, M., Hawethorne, B., Reeves, C. D., Dean, J., and Platt, D. (2016) Genotype specification language. *ACS Synth. Biol.* DOI: 10.1021/acssynbio.5b00194.

13. Bartley, B., Beal, J., Kevin, C., Goksel, M., Roehner, N., Oberortner, E., Pocock, M., Bissell, M., Madsen, C., Nguyen, T., Zhang, Z., Gennari, J. H., Myers, C., Wipat, A., and Sauro, H. (2015) Synthetic Biology Open Language (SBOL) Version 2.0.0. *Journal of Integrative Bioinformatics 12*, 272.

14. Galdzicki, M. et al. (2014) SBOL: A community standard for communicating designs in synthetic biology. *Nature Biotechnology 32*.

15. Oberortner, E., Bhatia, S., Lindgren, E., and Densmore, D. (2014) A Rule-Based Design Specification Language for Synthetic Biology. *J. Emerg. Technol. Comput. Syst. 11*, 25:1–25:19.

16. Bilitchenko, L., Liu, A., Cheung, S., Weeding, E., Xia, B., Leguia, M., Anderson, J. C., and Densmore, D. (2011) Eugene - a domain specific language for specifying and constraining synthetic biological parts, devices, and systems. *PLoS One 46*, e18882.

17. Beal, J., Lu, T., and Weiss, R. (2011) Automatic compilation from high-level biologically-oriented programming language to genetic regulatory networks. *PLoS ONE 6*, e22490.

18. Stattery, W. (1971) An Index of US Voluntary Engineering Standards. *National Standards Organizations in the US 329*.

19. Ham, T. S., Dmytriv, Z., Plahar, H., Chen, J., Hillson, N. J., and Keasling, J. D. (2012) Design, implementation and practice of JBEI-ICE: an open source biological part registry platform and tools. *Nucleic Acids Research*

20. iGEM Foundation, Registry of Standard Biological Parts. http://parts.igem.org (accessed October 1, 2015).

21. Madsen, C., Misirli, G., Pocock, M., Hallinan, J., and Wipat, A. SBOL Stack: The One-stop-shop for Storing and Publishing Synthetic Biology Designs. 7th International Workshop on Bio-Design Automation. 2015.

22. Cardinale, S., and Arkin, A. P. (2012) Contextualizing context for synthetic biology - identifying causes of failure of synthetic biological systems. *Biotechnology Journal 7*, 856–866.

23. Pearson, W. R., and Lipman, D. J. (1988) Improved tools for biological sequence comparison. *PNAS 85*, 2444–2448.

24. Bilofsky, H. S., and Christian, B. (1988) The GenBank genetic sequence data bank. *Nucleic Acids Research 16*, 1861–1863.

25. Hucka, M. et al. (2003) The Systems Biology Markup Language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics 19*, 524–531.

26. Demir, E. et al. (2010) The BioPAX community standard for pathway data sharing. *Nature Biotechnology 28*, 935–942.

27. Endy, D. (2005) Foundations for engineering biology. *Nature 438*, 449–453.

28. Roehner, N., Oberortner, E., Pocock, M., Beal, J., Clancy, K., Madsen, C., Misirli, G., Wipat, A., Sauro, H., and Myers, C. J. (2014) Proposed data model for the next version of the Synthetic Biology Open Language. *ACS synthetic biology 4*, 57–71.

29. Galdzicki, M. et al. Recent Advances in the Synthetic Biology Open Language. 5th International Workshop on Bio-Design Automation. 2013.

30. Xie, Z., Wroblewska, L., Prochazka, L., Weiss, R., and Benenson, Y. (2011) Multi-input RNAi-based logic circuit for identification of specific cancer cells. *Science 333*, 1307–1311.

31. Moon, T. S., Lou, C., Tamsir, A., Stanton, B. C., and Voigt, C. A. (2012) Genetic programs constructed from layered logic gates in single cells. *Nature 491*, 249–253.

32. Garny, A., Nickerson, D., Cooper, J., dos Santos, R. W., Miller, A., McKeever, S., Nielsen, P., and Hunter, P. (2008) CellML and associated tools and techniques. *Philos. Transact. A: Math Phys Eng Sci 366*, 3017–43.

33. Quinn, J. Y. et al. (2015) SBOL Visual: A Graphical Language for Genetic Designs. *PLoS Biol. 13*, e1002310.

34. Quinn, J., Beal, J., Bhatia, S., Cai, P., Chen, J., Clancy, K., Hillson, N. J., Galdzicki, M., Maheshwari, A., Umesh, P., Pocock, M., Rodriguez, C., Stan, G.-B., and Endy, D. Synthetic Biology Open Language Visual (SBOL Visual), Version 1.0.0. BBF RFC 93, 2013; DOI: 1721.1/78249.

35. Eilbeck, K., Lewis, S. E., Mungall, C. J., Yandell, M., Stein, L., Durbin, R., and Ashburner, M. (2005) The Sequence Ontology: a tool for the unification of genome annotations. *Genome Biology 6*, R44.

36. N. Juty and N. Novere, *Encyclopedia of Systems Biology*; Springer New York, 2013; pp 2063–2063.

37. Vilanova, C., and Porcar, M. (2014) iGEM 2.0–refoundations for engineering biology. *Nature biotechnology 32*, 420–424.

38. Myers, C. J., Barker, N., Jones, K., Kuwahara, H., Madsen, C., and Nguyen, N.-P. D. (2009) iBioSim: a tool for the analysis and design of genetic circuits. *Bioinformatics 25*, 2848–2849.

39. 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. *IEEE Life Science Letters*, In press.

40. Nielsen, A. A. K., Der, B. S., Shin, J., Vaidyanathan, P., Paralanov, V., Strychalski, E. A., Ross, D., Densmore, D., and Voigt, C. A. (2016) Genetic circuit design automation. *Science 352*.

41. Vaidyanathan, P., Der, B., Bhatia, S., Roehner, N., Silva, R., Voigt, C., and Densmore, D. (2015) A Framework for Genetic Logic Synthesis. *Proceedings of the IEEE PP*, 1–12.

42. Roehner, N., and Myers, C. J. (2014) A methodology to annotate Systems Biology Markup Language Models with the Synthetic Biology Open Language. *ACS Synth. Biol. 3*, 57–66.

43. Hedley, W. J., Nelson, M. R., Bellivant, D. P., and Nielsen, P. F. (2001) A short introduction to CellML. *Phil. Trans. R. Soc. Lond. A 359*, 1073–1089.

44. Hucka, M., Bergmann, F. T., Hoops, S., Keating, S. M., Sahle, S., Schaff, J. C., Smith, L. P., and Wilkinson, D. J. The Systems Biology Markup Language (SBML): language specification for Level 3 Version 1 Core. SBML Specification, 2010; http://sbml.org/Documents/Specifications#SBML_Level_3_Version_1_Core.

45. Faeder, J. R., Blinov, M. L., and Hlavacek, W. S. *Systems biology*; Springer, 2009; pp 113–167.

46. (2006) IEEE Standard for Verilog Hardware Description Language. *IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001)* 1–560.

47. Nguyen, T., Roehner, N., Zundel, Z., and Myers, C. J. (2016) A Converter from the Systems Biology Markup Language to the Synthetic Biology Open Language. *ACS Synthetic Biology* DOI: 10.1021/acssynbio.5b00212.

48. Roehner, N., Zhang, Z., Nguyen, T., and Myers, C. J. (2015) Generating Systems Biology Markup Language Models from the Synthetic Biology Open Language. *ACS Synthetic Biology 4*, 873–879.

49. Manola, F., and Miller, E., Eds. *RDF Primer*; W3C Recommendation; World Wide Web Consortium, 2004.

50. Prud'hommeaux, E., and Seaborne, A. SPARQL Query Language for RDF. W3C Recommendation, 2008; `http://www.w3.org/TR/rdf-sparql-query/`.

# Graphical TOC Entry