# Synthetic Biology

pubs.acs.org/synthbio

Technical Note

# Synthetic Biology Curation Tools (SYNBICT)

Nicholas Roehner,\* Jeanet Mante, Chris J. Myers, and Jacob Beal

Cite This: ACS Synth. Biol. 2021, 10, 3200–3204



ACCESS

#### Metrics & More

E Article Recommendations

**ABSTRACT:** Much progress has been made in developing tools to generate component-based design representations of biological systems from standard libraries of parts. Most biological designs, however, are still specified at the sequence level. Consequently, there exists a need for a tool that can be used to automatically infer component-based design representations from sequences, particularly in cases when those sequences have minimal levels of annotation. Such a tool would assist computational synthetic biologists in bridging the gap between the outputs of sequence editors and the inputs to more sophisticated design tools, and it would facilitate their development of automated workflows for design curation and quality control. Accordingly, we introduce *Synthetic Biology Curation Tools* (SYNBICT), a Python tool suite for



automation-assisted annotation, curation, and functional inference for genetic designs. We have validated SYNBICT by applying it to genetic designs in the DARPA *Synergistic Discovery & Design* (SD2) program and the *International Genetically Engineered Machines* (iGEM) 2018 distribution. Most notably, SYNBICT is more automated and parallelizable than manual design editors, and it can be applied to interpret existing designs instead of only generating new ones.

**KEYWORDS:** sequence annotation, design specification, network inference, SBOL, SYNBICT

hile much progress has been made in developing tools to generate component-based design representations for biological systems from standard libraries of parts, these tools all assume that their output representations are new designs to be manually specified by a  $human^{1-3}$  or automatically generated from a functional specification.<sup>4,5</sup> Most biological designs, however, are still specified at the sequence level, either for historical reasons or because they are created using sequence editors and other low-level design approaches. Consequently, there exists a need for a tool that can be used to automatically infer component-based design representations from sequences in a retrospective manner, particularly in cases when those sequences have minimal levels of annotation. Such a tool would assist computational synthetic biologists in bridging the gap between the outputs of sequence editors and the inputs to more sophisticated design tools, and it would facilitate their development of automated workflows for design curation and quality control.

While existing bioinformatics tools for sequence annotation such as SnapGene (available at http://snapgene.com) can be used to generate annotation-based design representations, many of these tools lack support for representing the multiple distinct classes of design information relevant to synthetic biology designs (for example, representing a genetic circuit both in terms of its structural organization into transcriptional units and its functional organization as a regulatory network). Consequently, it is much more difficult to unambiguously reason about the higher-order structure and function of designs represented with these tools.

For functional inference, there do exist bioinformatics tools and resources<sup>6</sup> for inferring biochemical networks based on the genes and molecular species present. These tools, however, tend to focus on gene–gene and protein–protein interactions instead of the DNA–protein interactions that are common to synthetic biology designs. In addition, while a computational synthetic biology workflow has been developed to infer networks based on component hierarchies,<sup>7</sup> it does not automate generation of these hierarchies, nor does it automate abstraction of the inferred networks.

Accordingly, in order to support inference of rich component-based design representations from genetic sequences, we introduce *Synthetic Biology Curation Tools* (SYNBICT), a tool suite for automation-assisted annotation, curation, and functional inference for genetic designs.

**Received:** May 19, 2021 **Published:** November 10, 2021



pubs.acs.org/synthbio

Technical Note



**Figure 1.** Workflow diagram for SYNBICT. SYNBICT supports multiple workflows that can begin with either selecting target DNA components for sequences\_to\_features or features\_to\_circuits, or by selecting target network modules for circuits\_to\_truth\_tables. Optional steps are able to be bypassed by following an overlapping arrow.



**Figure 2.** Examples of SYNBICT outputs, including (A) an annotated component hierarchy from sequences\_to\_features, (B) a network module from features\_to\_circuits, and (C) a truth table from circuits\_to\_truth\_tables. All glyphs are defined by the SBOL Visual standard.<sup>11</sup> In (A), SYNBICT adds subcomponent annotations to the target construct component based on exact sequence matches and removes an existing generic annotation for YFP. In (B), SYNBICT creates a network module that contains the annotated construct component and adds regulatory submodules based on whether they contain DNA components with the same ID as a subcomponent of the annotated construct. In this case, submodules are added for AraC, pBAD, PhIF, and YFP. Then SYNBICT infers which non-DNA components are inputs and outputs on the basis of the absence of incoming and outgoing regulatory arcs, respectively. In this case, SYNBICT infers that L-Arabinose is an input and YFP\_protein is an output. In addition, SYNBICT infers transcription interactions between promoter subcomponents and CDS subcomponents on the basis of their locations in the annotated construct. In this case, SYNBICT infers that pPhIF stimulates YFP and pBADmin stimulates PhIF. These interactions do not appear in submodules because they do not exist outside of the context of the annotated construct. Also note that in this diagram all instances of the same glyph are assumed to map to a single component in the design. In (C), SYNBICT generates each row of the truth table by setting an input component's value to zero and then propagating that value to other components in the network based on the logical interpretations of the regulatory interactions between these components. In this case, the value of the output component YFP\_protein is inferred to be the negation of the input component L-Arabinose and the other intermediate components.

In its current form, SYNBICT is targeted toward computational synthetic biologists with some experience using the Python programming language, but in the future it could be made more accessible via the development of graphical or webbased interfaces for use by synthetic biologists in general. The remainder of this manuscript is organized as follows: we present the architecture of SYNBICT, discuss how it has been applied, and finally discuss implications and future work.

# RESULTS

SYNBICT is implemented as three Python applications that curate designs at increasing levels of abstraction. Its efficacy has been validated through application to genetic designs in the DARPA Synergistic Discovery & Design (SD2) program and the International Genetically Engineered Machines (iGEM) 2018 distribution.

**Implementation of SYNBICT.** The architecture of SYNBICT is illustrated in Figure 1. First, the sequences\_to\_-features module handles sequence annotation and curation, then features\_to\_circuits infers networks of interactions, and finally circuits\_to\_truth\_tables abstracts the behavior of regulatory networks as logic functions. SYNBICT reads and writes design representations encoded in the *Synthetic Biology Open Language* (SBOL) 2 standard<sup>8</sup> using SBOL's native Python software library, pySBOL2.<sup>9</sup> SBOL is a community standard for modular, hierarchical representation of biological designs that can be used to represent individual DNA components as well as the overall biochemical systems that encompass them.

sequences\_to\_features. For sequence annotation, SYN-BICT uses the Aho-Corasick string matching algorithm as implemented by the flashtext<sup>10</sup> Python package. Given an SBOL 2 file containing a library of DNA components, SYNBICT simultaneously matches their sequences to those of target components in an SBOL 2, FASTA, or GenBank file. Whenever a library component's sequence is a complete, exact match to a contiguous portion of a target component s sequence, SYNBICT annotates the target component with the location of the match and includes the library component as a subcomponent of the target component, thus creating an SBOL 2 component hierarchy as its output (see Figure 2A). If there are no exact matches, then no component hierarchy is created.

Since library and target component sequences may be similar but not match exactly, sequences to features can also extend a library of DNA components prior to sequence annotation by aligning their sequences to portions of the target component's sequence that have already been annotated with another tool (such as an open reading frame finder). If the alignment score for a library component's sequence is above a user-defined threshold, and if the name of the library component is contained by the name of the annotation (or vice versa), then the matching portion of the target component's sequence is used to derive a new DNA component that is effectively a sequence variant of the library component. This is a fairly conservative approach to extending component libraries given the requirement that names match as well as sequences, but it can be necessary in cases when identical sequences have different functional roles (such as a guide RNA CDS and its binding site).

Besides annotating sequences, sequences\_to\_features can also curate sequence annotations by identifying those that overlap and are potentially redundant. Depending on which options are set, these annotations can be automatically merged according to certain criteria, or they can be returned to the user to decide which to keep and which ones to delete. Sequence annotations can also be automatically deleted if their properties include a particular functional role or if they do not specify the location of a subcomponent.

features\_to\_circuits. SYNBICT's approach to network composition is the following: given an SBOL 2 file containing one or more target components and an SBOL 2 file containing a library of network modules that include DNA subcomponents, SYNBICT checks for each network module whether all of its DNA subcomponents match at least one subcomponent of the target component(s) by ID. If so, then SYNBICT makes the network module a submodule of a new root network module. After composing the root network module, SYNBICT maps between subcomponents (DNA and otherwise) under the assumption that multiple copies of the same subcomponent function identically. Finally, SYNBICT infers the existence of transcription interactions by comparing the locations and orientations of promoter subcomponents to those of CDS subcomponents. Whenever SYNBICT finds a promoter that precedes a CDS, is in the same orientation, and is close enough according to a user-defined threshold, it adds a transcription interaction between them to the root network module (see Figure 2B). While Figure 2 only includes submodules that contain stimulation, inhibition, and genetic production interactions, SYNBICT can generally take as input modules that contain any type of interaction (e.g., complex formation), provided that they also contain at least one DNA component.

Like sequences\_to\_features, features\_to\_circuits can also extend a library of network modules based on the similarity of their DNA subcomponents to annotated portions of a target component's sequence. If one or more of a network module's DNA subcomponents appear to be variants as described in the previous section, then SYNBICT derives a new network module with the same type of interaction(s) between the variant DNA and protein subcomponents (and nonvariant subcomponents, if any).

circuits\_to\_truth\_tables. SYNBICT uses several simplifying assumptions to abstract a network module as a truth table of logic functions and values for its non-DNA subcomponents. First, SYNBICT assumes that the network module does not contain cycles since it currently only supports inference of truth tables for combinational logic circuits. Second, SYNBICT only considers stimulation and inhibition interactions in the network module that are described using the Systems Biology Ontology (SBO)<sup>12</sup> terms recommended by the SBOL standard. Third, SYNBICT only considers each subcomponent in the network module that is not regulated by more than two other subcomponents (either directly or via regulation of any transcriptional units producing the subcomponent) in order to simplify the assignment of a logic function to compute its value. A complete description of how SYNBICT abstracts different interaction and transcriptional unit motifs can be found in its documentation (see Data Availability).

**Application to Design Collections.** SYNBICT has been tested by application to genetic designs from the iGEM registry and the DARPA Synergistic Discovery and Design (SD2) program.

For the iGEM registry, we applied SYNBICT to the 2018 iGEM distribution. For this application, we used a component library derived from the RegulonDB database<sup>13</sup> (8824

# **ACS Synthetic Biology**

promoter components, 218 CDS components) to annotate the sequences of 1410 target constructs greater than 700 bp in length, which took 15 min. Of these constructs, SYNBICT produced annotations for 266 of them, identifying a total of 361 promoters and 20 CDS components, or an average of 1.4 annotations per construct. Of the 381 annotations made, SYNBICT identified 285 (275 promoter components and 10 CDS components) as potentially novel features that were not previously annotated in the target constructs. SYNBICT also inferred 8 transcriptional interactions between promoter and CDS components in the target constructs. In this case, however, no circuit designs were able to be generated, likely indicating an insufficient overlap between the functional interactions contained in RegulonDB and the functional interactions included in the 2018 iGEM distribution.

For the DARPA SD2 program, SYNBICT was applied to sequences for genetic circuits designed by several laboratories. In particular, we used SYNBICT and component libraries for CRISPR-dCas9<sup>14</sup> and TetR-homologue<sup>4</sup> genetic circuits to generate designs for 92 out of 101 strains with available DNA sequences (31 *E. coli* strains and 61 yeast strains). Of the remaining nine strains, six were base strains that were not engineered with a genetic circuit (i.e., SYNBICT correctly generated a null output), and only three were not adequately covered by the input component libraries. Figure 2B shows an example of one of the TetR-homologue genetic circuit designs generated during SD2.

# DISCUSSION

With respect to prior approaches to generating biological design representations, SYNBICT's key strengths are (1) it is more automated and parallelizable than manual design editors, and (2) it can be applied to interpret existing genetic designs, as opposed to other automated tools that focus instead on generating new circuit or pathway designs. SYNBICT also represents designs using an open community standard (SBOL) that can represent both the structural and functional aspects of a design, which prior representations from bioinformatics and systems biology cannot do.

All three of SYNBICT's applications have opportunities for improvement. In the case of sequences to features, its library extension capability currently requires the existence of annotations made with other tools, but in the future this capability could be made less dependent on existing knowledge, potentially at the cost of introducing some false variants and increasing runtime. Additionally, this module's annotation curation capability currently focuses primarily on overlapping annotations and selectively deleting or merging them, but it could do more to analyze the details of these annotations and highlight potential errors or inconsistencies. In general, SYNBICT could be augmented to take user constraints on the structure and/or function of a design as input and verify whether these constraints are satisfied. For features to circuits, its inference of network modules and interactions is currently limited to a root network module and transcription interactions between promoter and CDS components, but this capability could be expanded to inference of intermediate network modules based on interactions between components and their proximity on the same DNA construct or other considerations. For circuits\_to\_truth\_tables, its capability for network abstraction could be made to apply to networks that include feedback and a broader class of regulatory motifs. This

capability could also be applied to other types of high-level function, such as metabolic synthesis.

Finally, future tooling for automatic generation of biological design representations could become even more flexible by reducing or eliminating the requirement for a library of components as input. This, however, will likely require more accurate models for predicting the function of a biological sequence from its structure alone, which will in turn require some combination of improvements in our mechanistic understanding of biology and machine learning to extend hypotheses into areas that are less well understood.

**Data Availability.** Source code and documentation for SYNBICT are available on GitHub at https://github.com/SD2E/SYNBICT under the Apache License, Version 2.0.

# ASSOCIATED CONTENT

#### **Special Issue Paper**

Invited contribution from the 12th International Workshop on Bio-Design Automation.

# AUTHOR INFORMATION

# **Corresponding Author**

Nicholas Roehner – Raytheon BBN Technologies, Cambridge, Massachusetts 02138, United States; • orcid.org/0000-0003-4957-1552; Email: nicholas.roehner@raytheon.com

### Authors

- Jeanet Mante Department of Biomedical Engineering, University of Colorado Boulder, Boulder, Colorado 80309, United States
- Chris J. Myers Department of Electrical, Computer, and Energy Engineering, University of Colorado Boulder, Boulder, Colorado 80309, United States; Orcid.org/0000-0002-8762-8444
- Jacob Beal Raytheon BBN Technologies, Cambridge, Massachusetts 02138, United States

Complete contact information is available at: https://pubs.acs.org/10.1021/acssynbio.1c00220

#### **Author Contributions**

NR and JB designed SYNBICT and NR implemented its three modules as Python applications. NR applied SYNBICT with advice from JB to genetic designs obtained from the DARPA SD2 program and iGEM 2018 distribution. CJM and JM contributed to the design of the features\_to\_circuits module and JM tested the sequences\_to\_features module. NR wrote the manuscript, and all authors revised the manuscript.

#### Notes

The authors declare no competing financial interest.

#### ACKNOWLEDGMENTS

NR and JB are supported by DARPA Award HR0011-15-C-0084. CM and JM are supported by National Science Foundation under Grant No. 1939892. JM is supported by a Dean's Graduate Assistantship at the University of Colorado Boulder. Finally, CM is supported by DARPA FA8750-17-C-0229. 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 U.S. International Traffic in Arms Regulation or U.S. Export Administration Regulations.

#### REFERENCES

(1) 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.

(2) Czar, M. J.; Cai, Y.; Peccoud, J. Writing DNA with GenoCAD. *Nucleic Acids Res.* **2009**, *37*, W40–7.

(3) Chandran, D.; Bergmann, F.; Sauro, H. TinkerCell: modular CAD tool for synthetic biology. *J. Biol. Eng.* **2009**, *3*, 19.

(4) Nielsen, A. A. K.; Der, B. S.; Shin, J.; Vaidyanathan, P.; Paralanov, V.; Strychalski, E. A.; Ross, D.; Densmore, D.; Voigt, C. A. Genetic Circuit Design Automation. *Science* **2016**, 352, aac7341.

(5) Beal, J.; Lu, T.; Weiss, R. Automatic Compilation from High-Level Biologically Oriented Programming Language to Genetic Regulatory Networks. *PLoS One* **2011**, *6*, No. e22490.

(6) Escorcia-Rodriguez, J. M.; Tauch, A.; Freyre-Gonzalez, J. A. Abasy Atlas v2.2: The Most Comprehensive and Up-To-Date Inventory of Meta-Curated, Historical, Bacterial Regulatory Networks, Their Completeness and System-Level Characterization. *Comput. Struct. Biotechnol. J.* **2020**, *18*, 1228–1237.

(7) Misirli, G.; Nguyen, T.; McLaughlin, J. A.; Vaidyanathan, P.; Jones, T. S.; Densmore, D.; Myers, C.; Wipat, A. A Computational Workflow for the Automated Generation of Models of Genetic Designs. *ACS Synth. Biol.* **2019**, *8*, 1548–1559.

(8) Roehner, N.; Beal, J.; et al. Sharing Structure and Function in Biological Design with SBOL 2.0. ACS Synth. Biol. 2016, 5, 498-506.
(9) Mitchell, T.; Bartley, B.; Toll, B. pySBOL2; 2020. https://github.

com/SynBioDex/pySBOL2/releases/tag/v1.2.

(10) Singh, V. Replace or Retrieve Keywords In Documents at Scale. *arXiv*, October 31, 2017, 1711.00046.

(11) Beal, J.; Nguyen, T.; et al. Communicating structure and function in synthetic biology diagrams. ACS Synth. Biol. 2019, 8, 1818–1825.

(12) Courtot, M.; Juty, N.; et al. Controlled Vocabularies and Semantics in Systems Biology. *Mol. Syst. Biol.* 2011, 7, 543.

(13) Santos-Zavaleta, A.; Salgado, H.; et al. RegulonDB v 10.5: Tackling Challenges to Unify Classic and High Throughput Knowledge of Gene Regulation in E. coli K-12. *Nucleic Acids Res.* **2019**, 47, D212–D220.

(14) Gander, M. W.; Vrana, J. D.; Voje, W. E.; Carothers, J. M.; Klavins, E. Digital Logic Circuits in Yeast with CRISPR-dCas9 NOR Gates. *Nat. Commun.* **2017**, DOI: 10.1038/ncomms15459.