A computational workflow for the automated generation of models of genetic designs

Göksel Mısırlı,[†,‖] Tramy Nguyen,[‡,‖] James Alastair McLaughlin,[¶] Prashant Vaidyanathan,[§] Timothy S. Jones,[§] Douglas Densmore,[§] Chris Myers,[*,‡] and Anil Wipat[*,¶]

†*School of Computing and Mathematics, Keele University, Staffordshire, UK*

‡*Department of Electrical and Computer Engineering, University of Utah, Salt Lake City, UT*

¶*ICOS, School of Computing, Newcastle University, Newcastle upon Tyne, UK*

§*Department of Electrical and Computer Engineering Boston University, Boston, MA*

‖*Contributed equally to this work*

E-mail: myers@ece.utah.edu; anil.wipat@ncl.ac.uk

Abstract

Computational models are essential to engineer predictable biological systems and to scale up this process for complex systems. Computational modelling often requires expert knowledge and data to build models. Clearly, manual creation of models is not scalable for large designs. Despite several automated model construction approaches, computational methodologies to bridge knowledge in design repositories and the process of creating computational models has still not been established. This paper describes a workflow for automatic generation of computational models of genetic circuits from data stored in design repositories using existing standards. This workflow leverages the software tool SBOLDesigner to build structural models that are then enriched

by the Virtual Parts Repository API using Systems Biology Open Language (SBOL) data fetched from the SynBioHub design repository. The iBioSim software tool is then utilized to convert this SBOL description into a computational model encoding using the Systems Biology Markup Language (SBML). Finally, this SBML model can be simulated using a variety of methods. This workflow provides synthetic biologists with easy to use tools to create predictable biological systems, hiding away the complexity of building computational models. This approach can further be incorporated into other computational workflows for design automation.

Keywords

SBOL, SBML, data standards, VPR API, modeling, genetic design automation

A substantial amount of information is being produced about biological parts that can be used to implement complex designs. However, this information is usually only available for human interpretation and is often solely at the DNA sequence level. Computational modeling in silico is often exercised manually in order to predict the behavior of designs that can be implemented *in vivo* or *in vitro*. In these models, functional relationships and design constraints between parts in a design are captured in a formal modeling language for simulation. Although this approach may be sufficient for small designs and part libraries, automation of the model generation process is necessary to evaluate larger combinatorial design spaces.

Model-based design approaches are particularly useful to derive physical systems through computational simulations. Several tools reviewed in[1–3] have been developed to understand and optimize the behavior of genetic designs. Some of these tools also offer graphical user interfaces (GUIs), such as iBioSim.[4] iBioSim allows users to create genetic circuits and specify interactions between biological components manually. The user of the tool provides the necessary knowledge through the GUI for the creation of genetic circuits. iBioSim allows the specification of mathematical models for genetic circuits using reaction-based networks.

These models can then be simulated using a variety of deterministic (ODE) and stochastic (SSA) simulators. However, creating models from scratch every time can be difficult, time consuming, and error prone.

There is also research with an emphasis on utilizing existing models. The Virtual Parts Repository[5] (VPR) has been developed to provide reusable and modular models of biological parts. These models are further annotated with metadata to facilitate model composition. The data in the repository is accessible computationally. In a manual design process using the VPR, a user is expected to provide genetic circuit specifications in terms of constituting biological parts. The repository is also ideal to explore large design spaces of biological systems through heuristic evolutionary computing and artificial intelligence-based automated approaches,[6] in which information about biological components and functional relationships are used to optimize existing, or create new designs, until solutions are found. The use of the VPR for such automated approaches is particularly suitable for tool developers wishing to automate model design.

In order to construct a genetic circuit design, sequence editing tools such as SBOLDesigner[7] provide a GUI and assume no programming experience, simplifying the process of designing biological systems. In SBOLDesigner, basic DNA-based biological components are represented with standard glyphs from the SBOL Visual (SBOLv) standard[8] that can then be layed out sequentially along a DNA strand. Often challenges arise in the design process due to a lack of curated information. SBOLDesigner connects to the online design repository SynBioHub[9] to retrieve linked information for genetic parts. An alternative approach to genetic circuit construction is to use domain specific languages,[10–15] which can be particularly efficient and easier to use in a scripting environment. One important example is the tool Cello,[16] which applied electronic design automation principles to synthetic biology to automate the design of synthetic genetic circuits encoded as Boolean logic formula in the Verilog language.

Although these constructions methods are efficient to construct genetic circuits struc-

turally at the DNA-level, mathematical and simulatable models are necessary to explore and further understand the behavior of these circuits. For example, while Cello demonstrates that engineering principles can be applied to build robust multilevel circuits, the circuit performance is primarily focused on the steady-state function of the circuit. The ability to describe biological networks' behavior over time, represents an important step towards building robust genetic circuits.[17] Models constructed for well-established circuits, like the ones studied in Cello, can be simulated to better understand their time-series behavior. Therefore, it is desirable to reuse existing synthetic biology tooling to construct these models.

One way to re-utilize existing research efforts, and to integrate design and modeling tools, is to develop workflows. Workflows have already been acknowledged in synthetic biology, and applications and infrastructures to facilitate computational workflows have been demonstrated.[6,18–21] Synthetic biologists would greatly benefit from workflows that are not just for tool developers, but also directly for end users. These workflows can simplify the design and modeling of genetic circuits using adopted data standards.

Data standards are particularly important to facilitate design automation, and to pass information between different computational tools when implementing complex workflows. The Synthetic Biology Open Language (SBOL)[19,22,23] has emerged as an international standard to exchange genetic circuit designs. This standard is useful to specify designs in terms of constituent components. Using SBOL, the order and sequences of biological components in a design can be captured, and these designs can be hierarchically reused in more complex designs. Importantly, SBOL supports capturing molecular interactions between these components. This information is invaluable when creating computational models that can be encoded using different languages such as the Systems Biology Markup Language (SBML)[24] and CellML.[25] SBML, in particular, has been adopted by more than 280 software tools. SBML allows embedding external information in the form of annotations. Once these models are annotated, they can be used to derive physical systems that can encode the behavior in

models.[26,27] Models can also be derived directly from SBOL documents. The latter approach has been demonstrated by providing information about molecular interactions manually.[28]

To ease the burden on genetic designers, it is desirable to automate the process of model construction in the workflow. There are already different data repositories emerging for synthetic biology. Data from these repositories can be used to create models. The iGEM Registry of Standard Biological parts hosts a large repository of parts collected from participants of the International Genetically Engineered Machine (iGEM) competition, including free-text characterization data and usage experience. JBEI-ICE[29] also allows information about parts to be uploaded, and has been used to publish designs accompanying papers in the ACS Synthetic Biology journal.[30] SynBioHub[9,31] recently emerged as a repository to upload information about both biological parts and their interactions. This information is machine-accessible in the form of SBOL documents. The SynBioHub database is backed-by a graph database, in the form of a Resource Description Framework (RDF) triplestore, and allows uploading, downloading SBOL documents and querying the underlying data using SPARQL. Data can be programmatically accessed by a higher-level API.

Building upon these promising efforts, this paper presents a data integration based approach, enabled by data standards, to facilitate the automated creation of computational models from simple definitions of genetic circuits. These definitions may include information that is necessary for DNA synthesis to allow models to be implemented directly as DNA in the laboratory. Computational models are then constructed by extracting knowledge about these DNA components and other interacting entities such as proteins, small molecules, complexes and so on. The developments presented here bridge the gap between designing circuits and gaining insights into their behavior.

Results and Discussion

This paper illustrates a workflow for the automated generation of models of genetic designs. Information about the biological components necessary for their modeling is extracted from remote data sources. This information can be derived through manual curation processes or through data mining approaches, and can be deposited in repositories for computational access. An example workflow is proposed to illustrate these standardized automated procedures shown in Figure 1. This paper then presents how data mining approaches can be used to create information that is necessary to generate models. This approach is illustrated by creating two datasets from publicly available information. In order to automate the execution of model generation workflows, both the syntax to represent data and the semantics to define the meaning of data should be standardized. To facilitate this process, this paper describes a semantics-based approach in order to capture information about biological interactions in a standard manner using the SBOL syntax.

Genetic Circuit Construction

The process to construct genetic circuits in our workflow is independent of the design tools used, and designs can be created either manually or computationally, as depicted in Figure 2. The example workflow begins by using a design tool that supports the SBOL data standard and can connect to the SynBioHub repositories to query information about biological parts. Users can then use these parts that they have fetched from SynBioHub to construct their own genetic designs within their software tool. Both the SBOLDesigner[7] and Cello[16] tools support this genetic circuit construction methodology. Furthermore, both of these tools also support the uploading of their final genetic designs to a SynBioHub repository to allow them to be shared publicly and reused.

Initially, a DNA-level design constructed using one of these tools is annotated with DNA-level parts that have been fetched from the available datasets stored in various SynBioHub

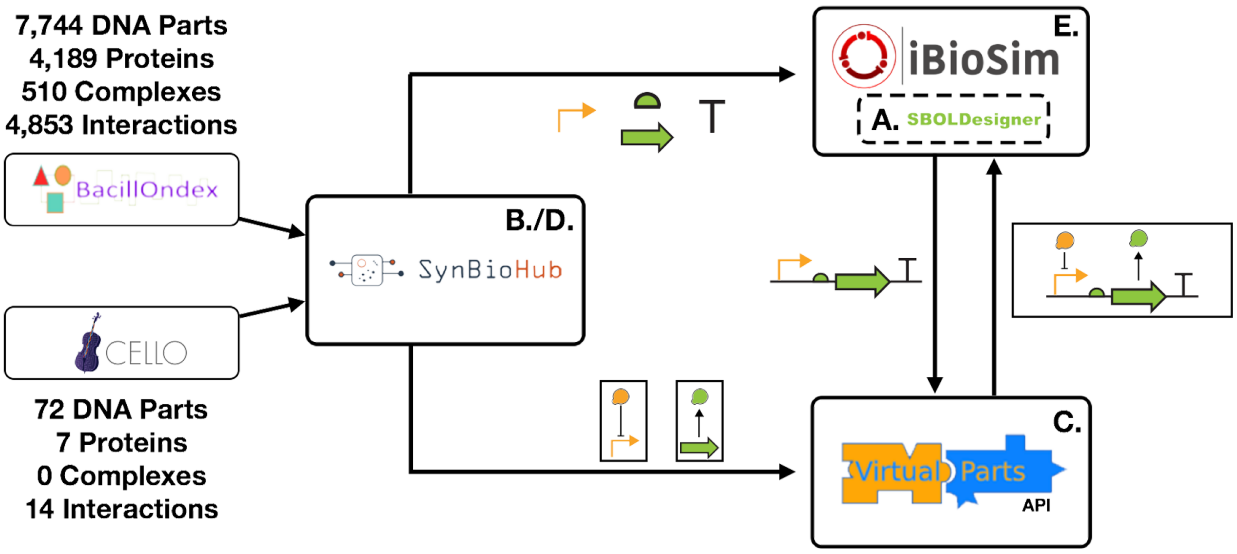

Figure 1: This figure illustrates how model generation is used within our workflow. **A.** The workflow starts off by a user creating a generic design of a transcriptional unit using the SBOLDesigner tool supported within iBioSim as a plugin. **B.** This DNA-level design is annotated with DNA parts fetched from datasets, such as the BacillOndex or Cello datasets, available in SynBioHub repositories. **C.** When the DNA-level design has been completely annotated, the design is sent to the VPR API to perform the enriching process. **D.** This process adds to the DNA-level design any proteins and complexes that affect the behavior of the circuit. This process also adds interactions to connect these proteins and complexes to the DNA-level design. **E.** iBioSim takes the design with the newly added component and converts the design into an SBML model. Genetic circuits expressed as SBML models allow further simulation and other in silico analysis and verification.

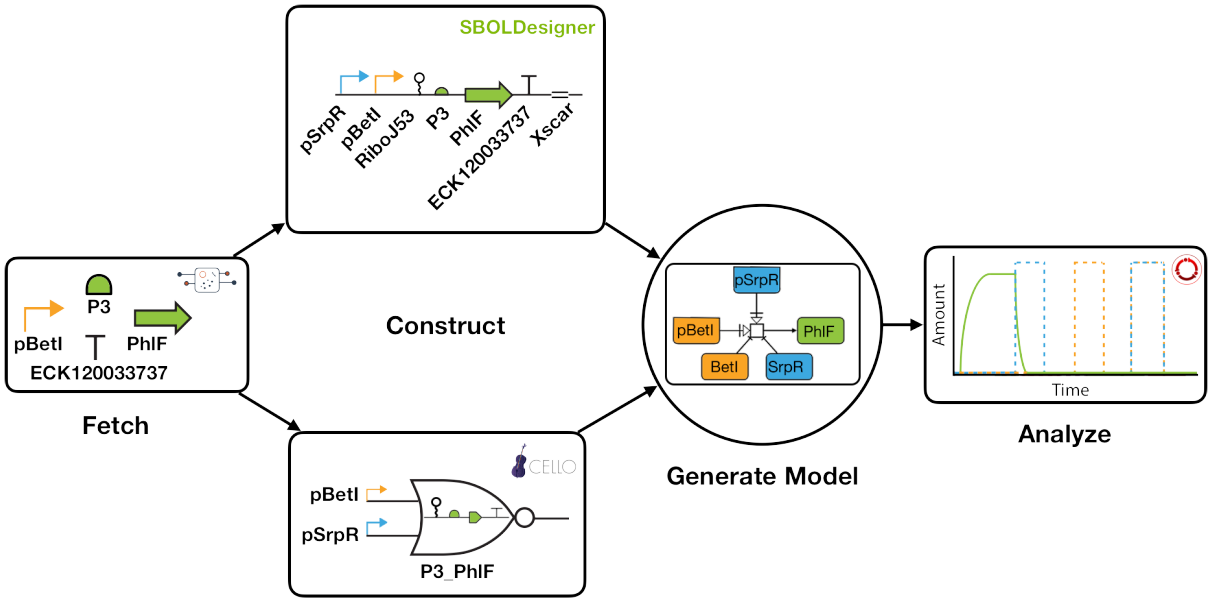

Figure 2: This figure illustrates how a CAD tool can be used to support a workflow for VPR model generation. As mentioned previously, this workflow has been incorporated in SBOLDesigner and Cello. Designs constructed within one of these tools are capable of fetching parts from a SynBioHub repository instance. Annotating a tool's design with parts taken from SynBioHub allows the VPR API to process additional interactions that are needed for modeling. The output design produced from the VPR API is then transformed into the SBML modeling format that allows a user to perform reaction-level simulation and analysis to verify the behavior of their design using a software tool such as iBioSim.

repositories. *Bacillus subtilis* and Cello parts are example datasets of curated DNA parts, proteins, complexes, and their interactions. Desired DNA-level designs can then be selected for submission to the VPR API. In the enriching process, the VPR API looks at the individual DNA-level designs and communicates with the SynBioHub design repositories using standard graph query languages, such as SPARQL. SPARQL allow users to fetch proteins and complexes that interact with the DNA components within the designs. Any components and interactions found are then added to the corresponding DNA level designs. Once the enriching process is complete, the new designs are returned from the VPR API. iBioSim takes these new designs to update its knowledge with the new contents. The interaction information that is added, allows dynamic models to be derived to verify the behavior of the genetic designs using iBioSim's SBOL to SBML converter.[28] Having genetic circuits represented in SBML enables genetic circuit designers to test their designs through in silico simulation and analysis under a variety of environments, parameters, initial values, and stimuli.

Deriving Dynamic Models

SBOL, by design, does not include all of the information needed for dynamic simulations. Therefore, expressing a model in SBML is necessary to verify the behavior of a design. Once the enriching process is complete, the VPR API returns to iBioSim an `SBOLDocument` that has a list of SBOL `ModuleDefinitions`. Each SBOL `ModuleDefinition` represents either a genetic design containing the added protein , complex, and small molecule components and their interactions with the DNA components, or a composition of such `ModuleDefinitions`. This new information is utilized by iBioSim's SBOL to SBML conversion utility[28] to derive a dynamic model in the SBML format. This conversion procedure maps specific SBOL objects to SBML elements and adds extra information necessary to create a dynamic model. More information about the SBOL to SBML conversion process is discussed within the methods section.

Workflow Example

This section demonstrates our proposed workflow using the Rule 30 example from.[16] The Rule 30 example is inspired by an update function for a cellular automaton that is capable of producing complex patterns that display chaotic behavior.[32] The Rule 30 example is one of the genetic circuits that was designed by Cello software in[16] using parts from the Cello library. These parts have been deposited into the Living Computing Project (LCP) SynBioHub repository . As shown in Figure 3A, the data for these parts include not only the DNA parts and their sequences, but also the proteins, complexes, and small molecules that interact with these DNA parts. In addition to the description of these components, the repository also includes all the interactions between them such as inhibition of promoters, encoding of proteins by coding sequences (CDS), complex formation reactions, and degradation reactions.

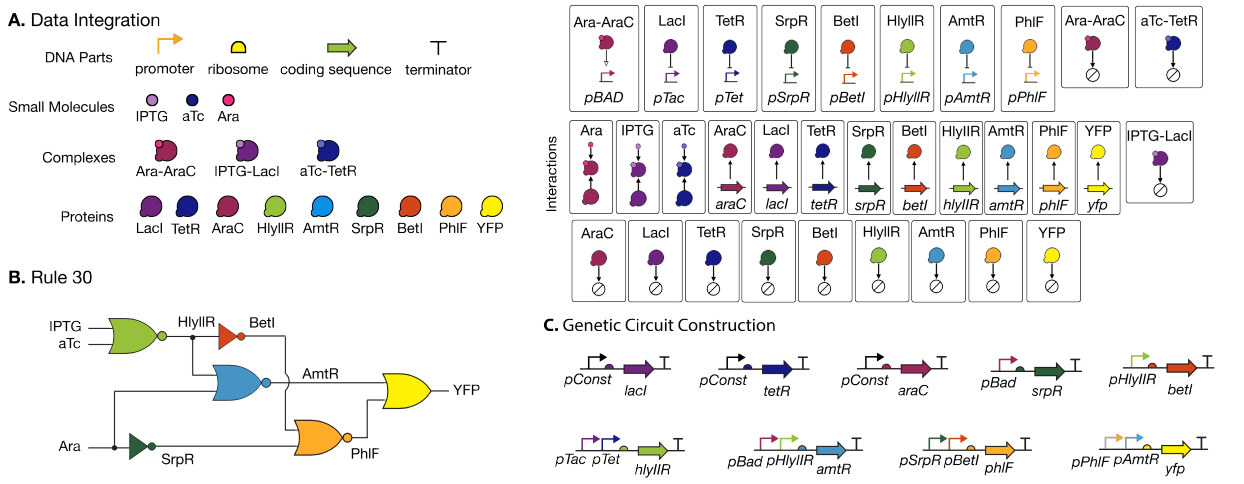

Figure 3: **A.** The Cello components and their interactions utilized in this example. **B.** An electronic schematic representation of the Rule 30 example. **C.** Nine DNA-level designs constructed with Cello parts that can be constructed using either SBOLDesigner or Cello. These nine DNA-level designs represent parts created for the three input sensors for the LacI, TetR, and AraC proteins, the two NOT gates for production of SrpR and BetI proteins, the three NOR gates for production of HlyllR, AmtR, and PhlF proteins, and a YFP reporter for the production of YFP.

As shown in 3B, the Rule 30 example is composed of NOT and NOR gates linked to an OR gate. The OR gate represents the output reporter that drives the expression of YFP.

The inputs to the Rule 30 example are driven by input sensors for three small molecules, IPTG, aTC, and Ara. The genetic circuit that can be constructed for this design using either Cello or SBOLDesigner is composed of nine DNA-level circuits as shown in Figure 3C. The first three DNA-level circuits are the input sensors that produce constitutively the LacI, TetR, and AraC proteins, respectively. The next five circuits are used to construct the Rule 30 example are composed of two NOT gates and three NOR gates. An OR gate is attached to the end of the Rule 30 example to represent the output sensor. Each of these DNA-level circuit designs is expressed in SBOL as `ComponentDefinition`s, with `Component`s that refer to definitions of DNA parts found in the LCP SynBioHub repository.

The DNA-level circuit designs are now sent to the VPR API as a single SBOL document, so that the design can be enriched with additional SBOL data. The result of this process is shown in Figure 4. The VPR API creates SBOL `ModuleDefinition`s for each DNA-level design. It also adds to the DNA-level design any proteins that interact with the corresponding DNA-level design. In this example, nine SBOL `ModuleDefinition`s are created with SBOL `Component`s and SBOL `Interaction`s. These interactions include one stimulation interaction, nine genetic production interactions, and ten inhibition interactions. A tenth SBOL `ModuleDefinition` is created by the VPR API enriching process to instantiate the other nine `Module`s to construct the full Rule 30 example. In this top-level module, the biochemical interactions are added, including three complex-formation reactions and twelve degradation reactions.

The final step of the model generation workflow takes the enriched SBOL description into the iBioSim software and converts it into an SBML model shown in Figure 5. This process converts the ten SBOL `ModuleDefinition`s into ten SBML `ModelDefinition`s. Each SBOL component is converted into an SBML `species`. The interactions are converted into a SBML `reaction`s. SBO terms are assigned to the `species` and `reactions` to preserve the information about the types of these elements. Finally, the SBOL `MapsTo` objects are converted to SBML `port`s and SBML `replacement`s to connect the SBML models to build

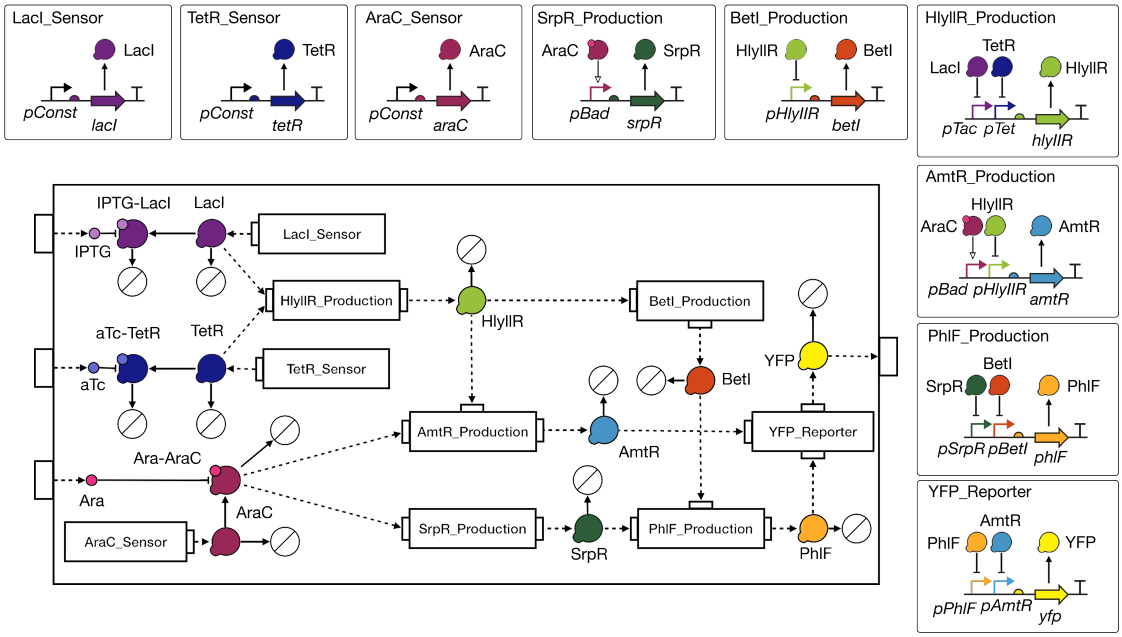

Figure 4: The enriching process takes the nine transcriptional units built from Figure 3C and adds in the proteins and interactions that were curated from the LCP SynBioHub instance for the Cello dataset. Nine of the transcriptional unit designs created in Figure 3C are stored within `ModuleDefinition`s. For each transcriptional unit that are found in a `ModuleDefinition`, interactions have been created. Three of the input sensors have productions interactions. Two of the NOT gates have a total of one activation interaction, one inhibition interaction, and two production interactions. The three NOR gates have one activation interaction, five inhibition interactions, and three production interactions. The last OR gate for the YFP reporter has one inhibition interaction and one production interaction. The VPR API also adds to the output designs a top level `ModuleDefinition` to connect the nine transcriptional units together and form an entire Rule 30 example circuit. This top level design also contains twelve degradations interactions and three complex interactions that are formed between the input signals represented as small molecules binding to three input sensors.

the full model of the Rule 30 example.

Once the model generation workflow has concluded, the resulting model can then be tested in silico. To do this, however, a testing environment, similar to one shown in Figure 6, must be constructed in SBML. This testing environment is used to apply different stimulus to the design and to observe how the generated model responds to the different input signals. An ODE simulation for our Rule 30 example is depicted in Figure 6, which demonstrates the circuit performing as originally specified above. Namely, YFP is produced when either IPTG or aTc is provided but not Ara, or only Ara is provided.

Data Integration

The workflow just described requires data to be integrated from a wide variety of sources. While a substantial amount of biological information has been produced, this data are often available in different formats and the meaning of the data varies between different databases. To make the most of this data in synthetic biology, it is important that these heterogeneous datasets are integrated so that they can be used easily both by humans and software tools. SyBiOntKB[33] is an integrated dataset for synthetic biology applications and has initially been populated with an integrated *Bacillus subtilis* dataset.[34] SyBiOntKB is represented in RDF and the semantics of entities are defined using the SyBiOnt ontology.

As part of this work, SyBiOntKB has been mined for genetic parts. The knowledge base contains information about thousands of biological concepts and relationships between them, regarding gene regulatory networks, metabolic pathways, proteins, RNA molecules and annotations. Gene products are represented based on different biological roles they may have, such as acting as a protein, RNA, enzyme, transcription factor (TF), etc. DNA-based entities such as promoters, CDSs, terminators and so on are annotated with Sequence Ontology (SO)[35] terms. In addition, information about genetic parts from the Cello framework[16,36] was used to create another SBOL dataset. This Cello library contains biological parts that are used to construct Boolean logic gates.

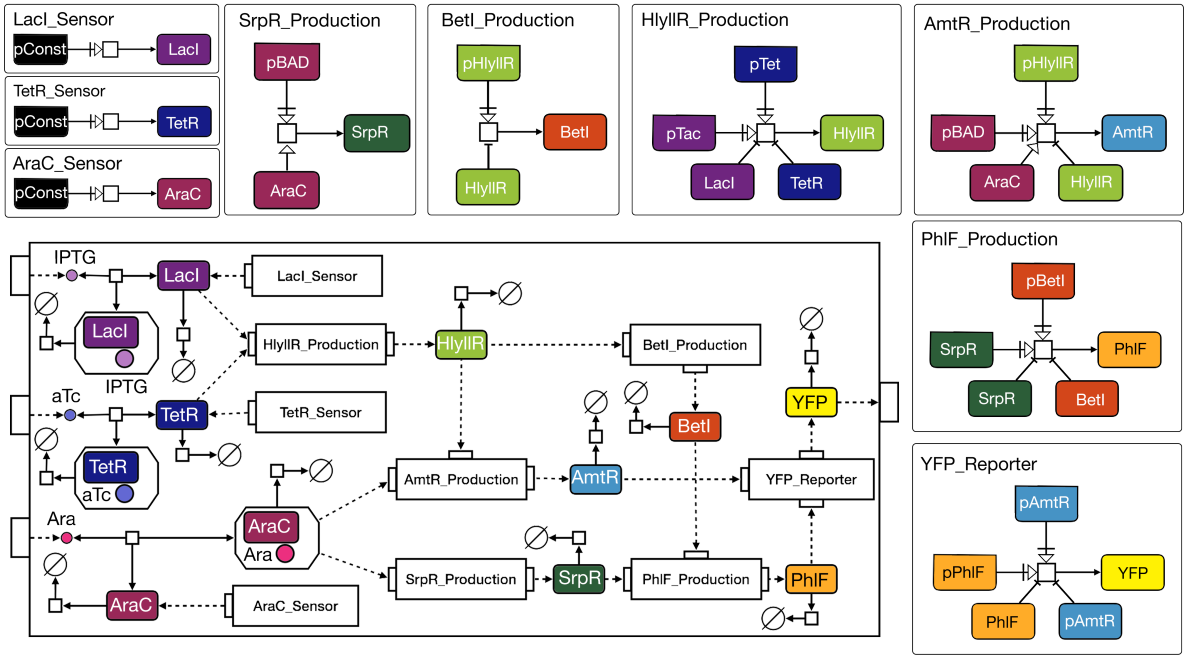

Figure 5: The updated DNA-level designs from Figure 4 are converted into ten SBML models. Each of these models contain interactions that have been enriched by the VPR API thus allowing reaction based models to be created. The small white boxes represent reactions. Arrows that point to the reaction can act as a modifier. Arrows pointing away from the reaction acts as a product. In this Rule 30 example, the three input sensors for LacI, TetR, and AraC have a total of three interactions representing modifiers for the pConst promoter and three interactions for production of LacI, TetR, and AraC proteins. The two NOT gates that represent the production of SrpR and BetI proteins have two modifiers, one activation, one inhibition, and two production interactions. The three NOR gates that represent the production of HlyllR, AmtR, and PhlF proteins are enriched with six modifiers, two activation interactions, five inhibition interactions, and three production interactions. The single NOR gate that represents the production of YFP proteins has two modifiers, two inhibition interactions, and one production interaction. The top-level SBML model is the same model that was converted from the SBOL top level design generated from the VPR API. In this model, two new interactions are included. As shown in the top level model, there are a total of twelve degradation interactions and three complex interactions that are created.

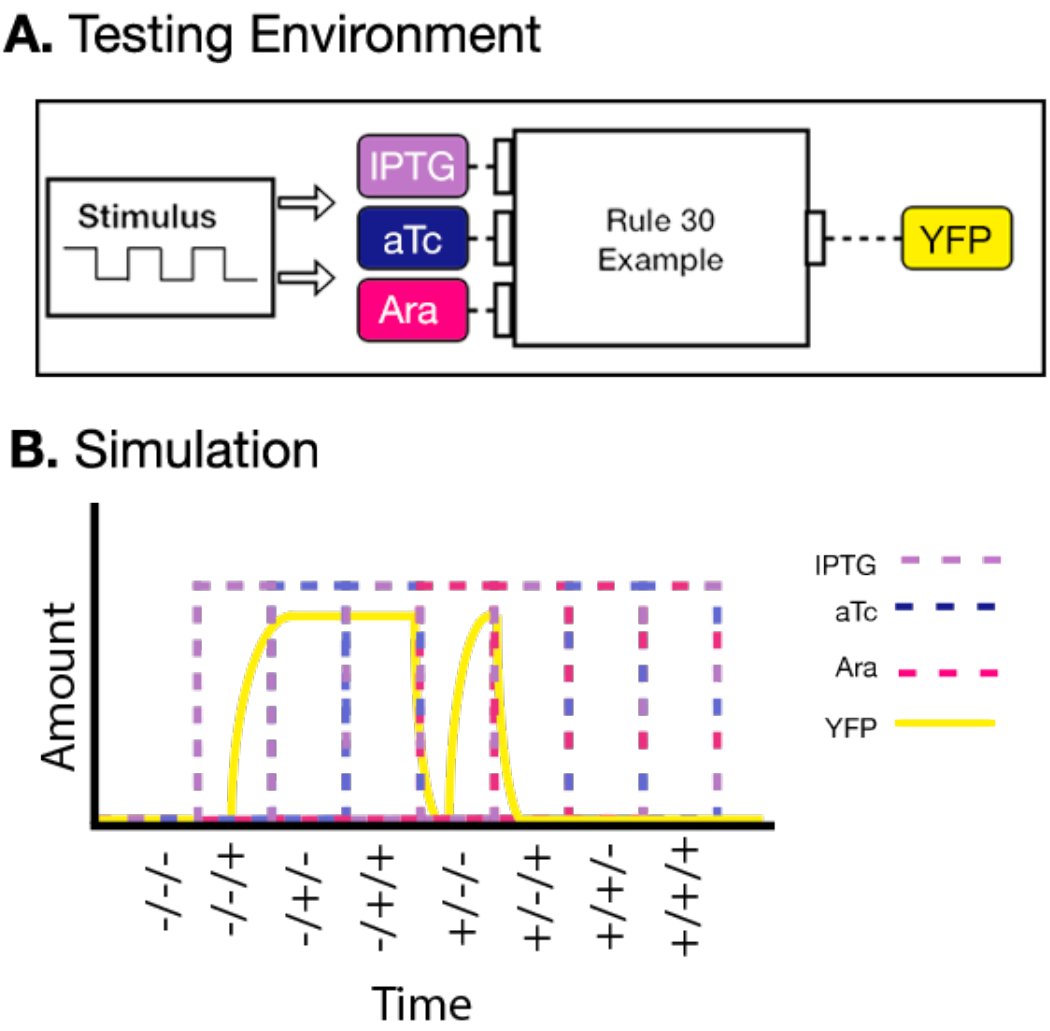

Figure 6: A testing environment to verify the behavior for the Rule 30 example. **A.** In this environment, an instantiated model of the Rule 30 example is created from Figure 5 top-level model. The inputs in the testing environment are connected to the example model so that alternating stimuli are applied by changing the amounts of the three small molecules, IPTG, aTc, and Ara. The alternating states can be observed as the changes in the amounts of YFP proteins produced. **B.** A simulation representing the Rule 30 example producing YFP when different inputs are applied. Here we observe that Rule 30 example only produce a HIGH output state when either IPTG or aTc are HIGH and Ara is LOW, or only Ara is HIGH. This simulation environment produces the precise behavior that we expect to have based on the results reported for Cello[16]

In this work, SBOL is used to represent the relationships between gene products and DNA-based entities. SBOL definitions of genetic parts and interactions are created according to these rules and stored in a single SBOL document:

- A transcriptional stimulation interaction is created between a gene product (such as a protein or TF) and a promoter, if the gene product binds to a positively regulated operator that is part of a promoter.

- A transcriptional inhibition interaction is created between a gene product and a promoter, if the gene product binds to negatively regulated operator that is part of a promoter.

- A phosphorylation interaction is created if a gene product is phosphorylated by another, or the gene product is activated by a protein, which is annotated with the Gene Ontology's[37] `response regulator` molecular function term.

- A genetic component definition is created for concepts representing promoter, CDS, RBS, operator, terminator, spacer sequence parts with known nucleotide sequences.

- A genetic component definition is created for proteins that have the `encoded by` relationship to CDS concepts.

- Part-subpart relationships are created using SBOL's hierarchical representation of genetic parts. For example if a promoter contains an operator part, the operator is represented as a subcomponent of the promoter in the SBOL representation, with start and end locations.

- A reusable module definition is created for a genetic construct, which is defined together with interactions such as transcriptional activations or inhibitions. Genetic components and interactions are converted into the SBOL format as defined above.

Provenance about this data mining process is also documented as part of the same SBOL document. As an RDF/XML based language, SBOL facilitates embedding useful information that may not be directly captured by the data model. Here, the PROV-O ontology is used to create a data mining `activity`, and SBOL definitions of genetic parts are linked to this activity.

The resulting SBOL document, containing information about *B. subtilis* gene regulatory networks, has been uploaded to a publicly shared SynBioHub instance (`https://synbiohub.org`). SynBioHub is used to group definitions of genetic parts and interactions as the `Bacillus subtilis collection`. Related publications are associated with the collection. The data are available both manually through the Web interface, and computationally.

Data from the Cello project have been deposited into the LCP SynBioHub repository (`https://synbiohub.programmingbiology.org`) to allow the synthetic biology community easy access to the parts, interactions, and additional functional information to characterize the Boolean logic gates designed in the Cello framework. In this interconversion process, Cello's `parts` and `gate_parts` collections including information about genetic parts, and the `gate_parts` collection containing information about transcriptional genetic constructs are mapped to SBOL entities as defined below. The other collections are added into the SynBioHub repository as JSON file attachments.

Enriching SBOL Designs

This section defines a common semantics to unify the representation of biological interactions using the SBOL syntax. This defined semantics is used in the workflow by the VPR API to examine the DNA-level information provided by Cello and SBOLDesigner and enrich it with biological details using data found in the SynBioHub repositories. Typically, in a manual design, this information is provided by a domain expert who knows the order of biological components and intricate constraints about how these components work together

to achieve a desired function. In our workflow, biological complexity is hidden from the user as much as possible. Initially, users decide which components to include using metadata in SynBioHub, and users may not know all the interactions of these selected components. Especially for large designs, it becomes more difficult to analyze every component and add relevant molecular interactions to analyze the behavior of a resulting circuit manually. In this project, simple definitions of biological designs are expanded to complex networks including as much information as possible in order to create predictive models.

This work extends the VPR[5] API to create these networks. As shown in Figure 7, simple networks that only contain information about DNA-level components are enriched with additional nodes that represent other biological components such proteins, protein complexes, small molecules, etc. Information regarding interactions between these biological components are also incorporated in the form of edges. Examples include genetic production of proteins, transcriptional activation or inhibition of promoters, formation of protein complexes, post-translational modification of proteins, etc.

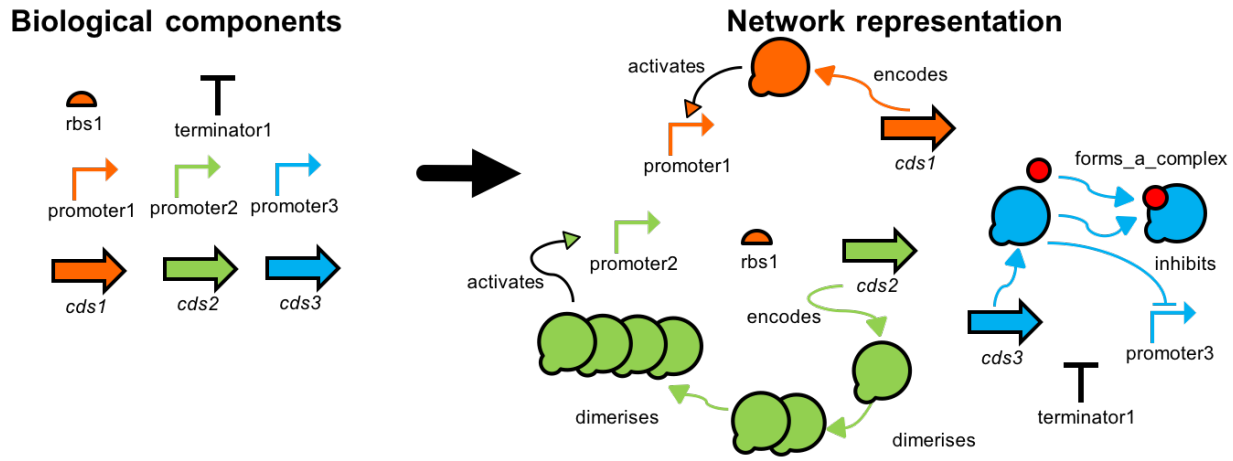

Figure 7: An example depicting the derivation of a network-based representation of the biological design components. Information about DNA-based biological design components is used to create an enriched network that includes further details about other biological components that may affect the behavior of genetic circuits.

SBOL is both used to represent genetic circuits and these complex networks. The VPR API accepts SBOL documents that only contain information about the type and order of

biological components, and returns back the same information together with the network representation of the biological details. In SBOL, genetic circuits are described in terms of how they are composed of individual DNA-based components. Interaction entities then provide a generic syntax to represent molecular interactions. Each interaction may have participants with defined roles, and these participants can either be individual design components or genetic circuit designs themselves. This flexibility requires an agreed semantics with regards to the representation of these interactions for machine interoperability.

In order to provide machine interoperability in computational workflows, the VPR API uses the Systems Biology Ontology (SBO)[38] terms when providing types of interactions, and roles of participants in each interaction[23] (Figure 8). These terms make the resulting SBOL documents further machine tractable and facilitate deriving models. Figure 8 depicts the representation for (A) stimulation, (B) inhibition, (C) genetic production, and (D) non-covalent binding (or complex formation).

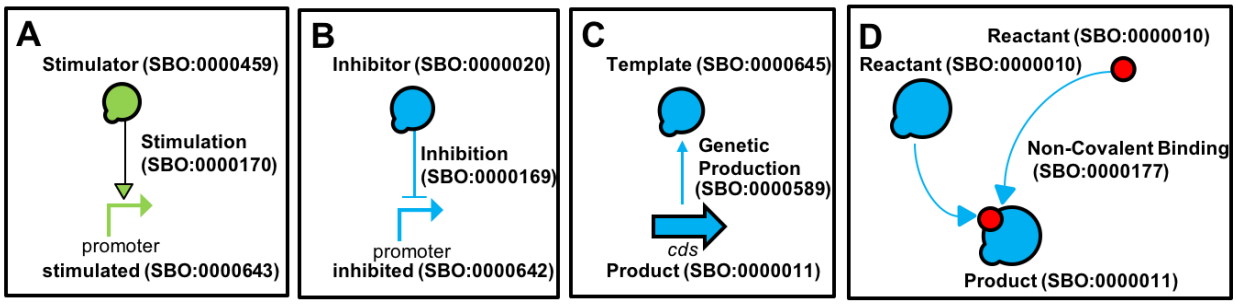

Figure 8: Example formal representations of molecular interactions using SBOL and established ontologies. **A**. Transcription activation of a promoter by a TF. **B**. Transcription repression of a promoter by a TF. **C**. Genetic production (transcription and translation) of a protein. **D**. Formation of complexes by non-covalent binding. The final complex has the product role and other participants have the reactant role. Dimerization and formation of tetramers also follow this rule.

As shown in Figure 9, the VPR API uses the gene centric abstraction when representing molecular interactions in order to support iBioSim's model creation process. In this approach, genetic circuit designs are represented in terms of transcriptional units which contain a set of biological components such as promoters, RBSs, CDSs and terminators. Molecular interactions are delegated to these transcriptional units to simplify the repre-

sentation of modeling of complex applications. These transcriptional units replace single DNA-based components in interactions. For example, genetic production of a protein is represented using the protein and the transcriptional unit that contains the CDS component. Similarly, transcriptional activation of a promoter is represented between the TF and the transcriptional unit that contains the promoter component. Transcriptional units can be concatenated or may include composite devices. The latter approach is particularly useful to incorporate Cello designs which may be formed of promoters and composite RBS-CDS-terminator designs.

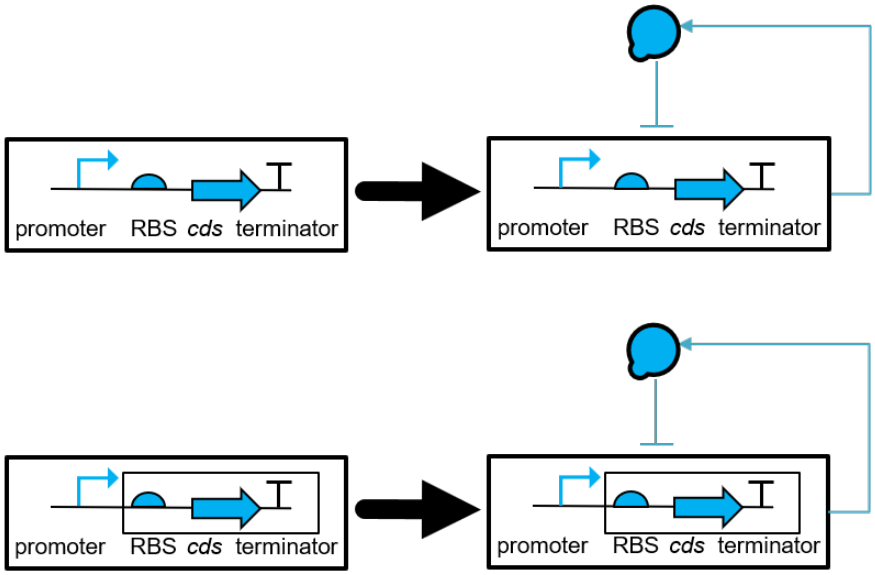

Figure 9: Transcriptional unit-based abstraction for representing interactions. In the figure, a simple negative-autoregulatory circuit is shown. Interactions of DNA-based components with other components are captured using the transcriptional units in which these components are included. As shown in the second part of the figure, transcriptional units may also include composite parts.

Genetic circuit designs are enriched using the SBOL syntax, agreed semantics as described above, and the gene centric abstraction. The following rules are applied for this process:

- A single SBOL `ModuleDefinition` entity is created for a given transcriptional unit design, which may be formed of promoters, RBSs, CDSs, and terminators. This entity is used to encapsulate molecular interactions between biological components.

- Biological molecules interacting with any of the DNA-based components are added to the `ModuleDefinition`.
- Interactions of proteins (excluding protein-DNA interactions) produced by the transcriptional unit are also included in the `ModuleDefinition`.
- If a biological molecule is not produced by any entities in the enriched SBOL design, then the corresponding SBOL `FunctionalComponent` is marked as an input. Otherwise, the corresponding `FunctionalComponent` is marked as both input and output. These inputs and outputs are used when creating hierarchical designs.

Discussion

The standard enabled design workflow presented here is important to abstract details of genetic circuit design processes. Particularly, this workflow simplifies the complexity of designing computational models of complex biological systems. These models are of value in the creation of predictable biological systems but can be challenging to create for every possible biological solution. Using this workflow, users can initially design simple DNA constructs and, in return, they are presented with information about how these designs may behave. Functional information about the relationships between DNA and elements, such as proteins that can be produced by these DNA constructs, together with mathematical models that can be simulated to gain detailed insights into the intended biological systems are then presented. These models can further be expanded to include other proteins, small molecules, and so on that interact with the designed system. As demonstrated here, simplifying the design process using a tools such as Cello and SBOLDesigner has significant benefits. Designs can relatively easily be created by users who can design circuits using DNA parts and still benefit from computational simulations.

Moreover, this approach presented here facilitates automation and exploring large design spaces of biological systems. As information for biological components becomes more

available from online repositories, automation will help speed up the process of building and modeling a genetic design *in silico*. Building and modeling genetic designs *in silico* minimizes the time and errors that could occur when implementing these designs *in vivo* or *in vitro* and thus increasing the rate of having a successful working circuit that meets the goal of the design specification.

So far, a variety of genetic design software tools and biological data repositories have been introduced, such as SBOLDesigner, Cello, iBioSim, and SynBioHub. However, existing software tools often deal with all aspects of specifying genetic circuits and creating models to understand the behavior of these circuits. The automated workflow approach presented here utilizes existing resources where possible, for example, by computationally constructing genetic circuits from biological components, for which curated data are available in online design repositories. This proposed workflow provides insights on how genetic software tools can incorporate this automated procedure into their tool.

The standards SBOL and SBML are critical since they serve as domain specific languages that seamlessly connect all the tools within this workflow. The workflow described in this paper uses SBOL to capture the initial user designs, which are then enriched using annotations and biological constraints about biological design components. Resulting data contains enough information to create computational models in the form of SBML. Like all data conversions, it is important to enforce the integrity of the data by ensuring that information remains consistent when data are translated between differing formats. Loss of data limits the ability to reproduce designs and models. Currently, SBOL objects and SBML elements are directly mapped with a one-to-one correspondence.

Mapping between an SBML modeling entity and a design component in SBOL is often facilitated by ontological terms. In this work, SBO terms are particularly used to map functional relationships in the form of biochemical reactions. Currently, we support a subset of reactions that are commonly used in genetic circuits. These include genetic production, transcriptional regulation, degradation, binding of molecules, etc. We envisage that VPR

and other software components can add new information and other software components will still continue working with the set of data that they understand. As the VPR API obtains access to more curated interaction, protein, and small molecule data, we will need to update the type of SBO terms used in the conversion to represent the growing amount of information that is added to the VPR API enriching process. Ideally, if the VPR API expands the types of interactions that are added during this process, then the SBOL to SBML conversion should reflect this change. Currently, in simulations we use nominal rate parameters. Our goal is to update the software components in the workflow to provide annotations in SBOL files and to reflect these values in SBML models. For instance, the Cello dataset has characterization data that are not yet supported in this workflow. Such information is useful to evaluate the quality of the generated genetic designs. We are looking into how characterization data can be incorporated into this workflow and how metrics can be applied to score the quality of constructed designs.

Genetic circuit design automation, and the creation of associated models, rely upon the availability of data. Finding data that is often available in different syntax and semantics, integrating this data and presenting in standard formats is certainly challenging. SynBioHub offers a platform to build upon for data integration and access. Currently, the VPR API can pull data from a SynBioHub instance using the SBOL syntax and standard RDF tooling. We have already populated the publicly available SynBioHub instances with an information rich *B. subtilis* and Cello datasets. We plan to extend this approach to other organisms and make the modeling process easier in general.

Here, we addressed some of the challenges related to the predictability of synthetic genetic circuits. As more information becomes available through characterization, the presented workflow can further be extended. This workflow supports the separation of data in different layers. For example, information about orthogonal parts can be defined using SBOL, which supports capturing provenance information, and hence allows linking orthogonal parts. Computational modeling of these parts can then be implemented using this

workflow.

This paper presents the use of standards such as SBOL and SBML and extends the available tooling to work in computational workflows to simplify the design of genetic circuits for end users. As we demonstrated, tools can take advantage of our approach and framework by delegating the construction of detailed mathematical models, automated approaches to search for solutions in large biological design spaces can also benefit. Through standardization and the availability of data and APIs, we envisage that other tooling can also be incorporated to this workflow for genetic design automation. Indeed, since standards are used throughout this process, any DNA-level design tool, modeling tool, and data repository that supports these standards can be used in the place of the tools presented here.

Methods

Standards

SBOL and SBML are two data standards that have been adopted by the synthetic biology community. To facilitate the integration of the resources and approaches presented here in other workflows, these standards are utilized in this work. SBOL is an example of a knowledge standard that is used to represent the information about a genetic design. A genetic design encoded in the SBOL data model is composed of `ComponentDefinitions`, `ModuleDefinitions`, `Sequences`, and other elements. `ComponentDefinitions` are used to represent DNA, small molecules, and proteins, and link to `Sequence` elements that describe how they are constructed. `ModuleDefinitions` are used to group the components together that make up the parts of a genetic design. `ModuleDefinitions` also include a description of the interactions between these components, such as inhibition, stimulation, genetic production, etc. An SBOL `ModuleDefinition` is depicted in Figure 10(A) using SBOLv,[8] a visual standard for genetic designs represented in SBOL.

SBML is an example of a modeling standard that can be used to describe the behavior of

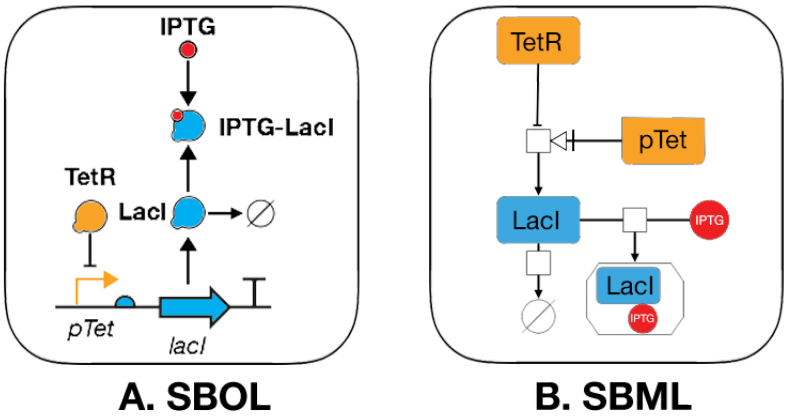

Figure 10: This figure shows an example of a LacI Inverter using SBOL Visual glyphs to show SBOL Data objects and SBGN glyphs to show SBML elements. DNA parts (i.e. pTet and lacI), proteins (i.e. TetR and LacI), small molecules (IPTG), and complexes (i.e. IPTG-LacI) are represented as ComponentDefinitions in SBOL and Species in SBML. SBOL Interactions and SBML Reactions are represented as arrows and lines with a straight end point. The arrows are used to denote inhibition and genetic production. Lines with a straight end point are used to denote inhibition. The rectangular bounding box represented in both SBOL visual and SBGN glyphs represent an SBOL ModuleDefinition and an SBML Model.

genetic designs. A genetic design modeled in SBML is composed of core elements: parameters, species, reactions, and compartments, among other elements. SBML parameters are used to describe named constants and variables. Genetic components, such as DNA, proteins, and small molecules, are represented using SBML species. SBML reactions describe how species transform from one form to another. For example, reactions allow the expression of protein degradation, complex formation, or regulated genetic production, and many other processes. SBML compartments are used to group SBML species and SBML reactions. SBML has further elements that can be used in simulation, such as function definitions, unit definitions, initial assignments, rules, constraints, and events. SBML also contains package extensions. One of the packages used by this work is the hierarchical composition package (comp),[39] which is used to compose models of genetic designs composed of multiple transcriptional units. An SBML model is depicted in Figure 10(B) using the Systems Biology Graphical Notation (SBGN),[40] a visual standard for biological models.

Data Curation

The information about functional relationships about biological components was mined from SyBiOntKB.[33,34] This knowledgebase is encoded using RDF/XML which is ideal to represent information as graphs in which nodes represent biological entities and edges represent how these entities relate to each other. In order to extract information, we use the SPARQL[41] querying languages. A SPARQL query itself is a graph and is used to find data using pattern matching. Using this approach, we created several SPARQL queries and identified biological components that participate in biological reactions. Definitions of these components and reactions are then encoded using SBOL. `ComponentDefinition` and `ModuleDefinition` entities are used respectively to represent information about biological components and their interactions. This mapping is implemented using libSBOLj,[42] an open-source Java library that has been developed to read and construct SBOL documents. The application to mine for information is implemented in Java, using Maven.

Data Repositories

Structural and functional information about biological parts is documented in SBOL and stored in SynBioHub repositories.[9] SynBioHub is an RDF database intended for the dissemination of genetic circuit designs and their parts. As well as serving as a database, SynBioHub provides interoperability between different registries enabled by the canonicalization of part data in SBOL. SynBioHub is used as the backend for the VPR, allowing the VPR to access DNA and protein parts as well as their interactions via the SynBioHub SPARQL interface, which can in turn be used as a reference for model generation. SynBioHub also allows SBML models to be attached to biological parts as attachments, which can be used to associate a part with a pre-generated model.

Data Conversion

Since different types of information can be accessed using different data standards, there exist data conversion methods to convert between them. SBOL to SBML[28] is the modeling conversion used within this workflow to verify the behavior of genetic designs. A simplified mapping of the SBOL to SBML conversion is shown in Figure 11. In the SBOL to SBML conversion, SBOL `ComponentDefinitions` that represent the biological parts are translated to SBML `Species`. SBOL `Interactions` are converted to SBML `Reactions` and `Participants` of an `Interaction` are translated to SBML `Reactants`, `Products`, or `Modifiers`, depending on the role of the participant in the interaction. SBOL `ModuleDefinitions` are converted to SBML `ModelDefinitions`. URI annotations are added to SBML elements to reference the associated SBOL components.[43] For example, SBML `Species` are annotated with a reference to the SBOL `ComponentDefinition` that describes that particular biological part. In addition, SBO terms are used to fine-tune the conversion to more precise translations. The custom annotations and SBO terms allow for round-trip conversion.

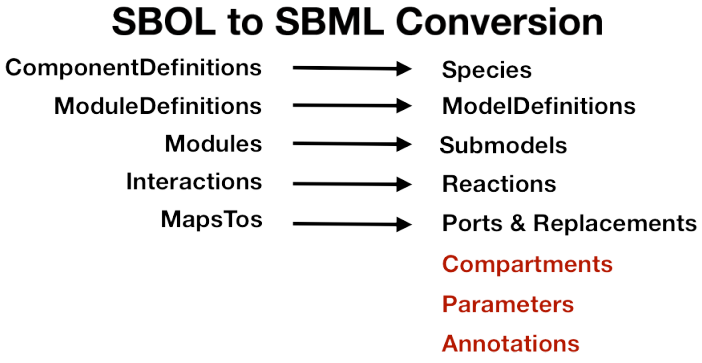

Figure 11: A simplified mapping to show how SBOL objects are converted to SBML elements. SBOL objects for `ComponentDefinitions`, `ModuleDefinitions`, `Modules`, `Interactions`, and `MapsTos`, are directly mapped to SBML elements for `Species`, `ModelDefinitions`, `Submodels`, `Reactions`, and `Ports` & `Replacements`, respectively. The conversion process also adds `Compartments`, `Parameters`, and `Annotations`.

The main purpose when converting from qualitative information to quantitative information, such as the SBOL to SBML conversion utilized in this workflow, is to enable the

simulation of these generated models. To achieve this, the conversion adds default kinetic laws with default parameter values. These defaults can later be modified within the iBioSim model editor GUI. The SBML model constructed in this way can be simulated using a variety of methods, including ordinary differential equations (ODEs) and stochastic simulation methods.

Cello Tool for Construction of Genetic Boolean Logic Gates

The Cello software enables the construction of combinational Boolean logic genetic gates, which are then translated to physical designs in DNA using technology mapping. The Boolean logic genetic gates that were designed for the Cello project have been constructed and validated experimentally. Information on the available biological parts used to construct these genetic circuits, and the genetic constraints that are considered while designing the parts are stored within Cello's User Constraint file (UCF) files. This information is categorized in separate collections within the UCF file. For example, the `parts` collection lists all the available parts in the Cello library: promoters, ribozymes, ribsome binding sites, terminators and scars. The `gate_parts` collection indicates how the parts in the Cello library are composed to create gates. The `gates` collection indicates the regulatory behavior which is characterized as a response function. This collection also indicates the promoter that is regulated by the coding sequence present in any specific gate.

In the data conversion process, Cello's `parts` and `gate_parts` collections are mapped to SBOL entities as defined earlier in the transformation rules. All parts stored within the parts collection of the UCF are converted to individual SBOL `ComponentDefinition`s. The information regarding the regulatory interaction between a gate and a promoter in the `gate_parts` collection is used to construct SBOL `ComponentDefinition`s for the proteins produced by the coding sequences which interact with specific promoters. These interactions are then represented as SBOL `ModuleDefinition`s. An SBOL `Interaction` and its SBO terms are also added to the SBOL `ModuleDefinition` to indicate if a specific protein acts as

an activator or inhibitor for any promoter `ModuleDefinition`s are also created to represent the production of proteins by coding sequences in each gate. A production interaction is then added for the protein produced by each gate. Using this data conversion process, Cello's UCF is converted to SBOL.

In the model generation step, the circuit design produced by Cello is first converted to a list of transcriptional units. Each transcriptional unit is then converted to SBOL `ComponentDefinition`s using the collection created in the previous step. This list of SBOL `ComponentDefinitions` is then enriched with `ModuleDefinition`s which describe the various interactions including the production of protein and the regulation of promoters by proteins. Using the framework mentioned in this paper, this SBOL representation of the circuit design is then converted to SBML using the VPR API.

Author Contribution

G.M. and A.W. carried out the data integration and mining related tasks regarding the SyBiOntKB dataset, and a subset was then represented using SBOL. C.M., P.V., T.J., and D.D. carried out the work regarding the representation of the Cello dataset using SBOL. G.M., A.W., T.N., and C.M. developed the approach to syntactically and semantically represent molecular interactions using SBOL. G.M. and A.W. developed the VPR API, while T.N. and C.M. added support for the VPR API in iBioSim and P.V., T.J, and D.D. added support for the VPR API in Cello. J.A.M. developed the interface to the SynBioHub repository used by the VPR API. P.V., T.J, and D.D. provided the 51 circuits built in Cello to test on the demonstrated workflow. T.N. and C.M. verified the behavior of the 51 circuits generated from the VPR API by converting the enriched designs to SBML models and creating testing environments for each model to create simulation results. All authors contributed to the manuscript.

Conflict of Interest

None declared.

Acknowledgement

The authors would like to thank Chris Voigt of MIT for the information about the Cello parts and circuit designs. The authors of this work are supported by the Engineering and Physical Sciences Research Council (EPSRC) grant EP/J02175X/1 (J.M., A.W.) and the National Science Foundation under Grant No. 1446607 (P.V. and D.D.), 1522074 (T.J., C.M., and D.D.), CCF-1218095 (T.N. and C.M.), and DBI-1356041 (T.N. and C.M.). G. M., in part, supported by the EPSRC 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 the funding agencies.

Supporting Information Available

Our workflow was exercised by constructing 52 genetic circuits taken from Cello.[16] Each of these circuits have been uploaded from iBioSim onto the Living Computing Project SynBioHub instance, under the `Cello_VPRGeneration_Paper` collection. These circuits are available at: `https://synbiohub.programmingbiology.org/public/Cello_VPRGeneration_Paper/Cello_VPRGeneration_Paper_collection/1`. All of the circuits include the SBOL representation for the design, the SBML models, and the simulation results.

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

References

(1) MacDonald, J. T.; Barnes, C.; Kitney, R. I.; Freemont, P. S.; Stan, G.-B. V. Computational design approaches and tools for synthetic biology. Integr. Biol. **2011**, 3, 97–108.

(2) Appleton, E.; Madsen, C.; Roehner, N.; Densmore, D. Design Automation in Synthetic Biology. Cold Spring Harb Perspect Biol **2017**, 9, a023978.

(3) Purnick, P. E.; Weiss, R. The second wave of synthetic biology: from modules to systems. Nat. Rev. Mol. Cell Biol. **2009**, 10, 410–422.

(4) Madsen, C.; Myers, C.; Patterson, T.; Roehner, N.; Stevens, J.; Winstead, C. Design and test of genetic circuits using iBioSim. IEEE Des Test **2012**, 29, 32–39.

(5) Misirli, G.; Hallinan, J.; Wipat, A. Composable Modular Models for Synthetic Biology. ACM J Emerg Technol Comput Syst **2014**, 11, 22:1–22:19.

(6) Hallinan, J.; Gilfellon, O.; Misirli, G.; Wipat, A. Tuning receiver characteristics in bacterial quorum communication: An evolutionary approach using standard virtual biological parts. IEEE Symp Comput Intell Bioinforma Comput Biol Proc. 2014; pp 1–8.

(7) 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.

(8) Quinn, J. Y. et al. SBOL Visual: A Graphical Language for Genetic Designs. PLoS Biol. **2015**, 13, 1–9.

(9) McLaughlin, J. A.; Myers, C. J.; Zundel, Z.; Misirli, G.; Zhang, M.; Ofiteru, I. D.; Goni-Moreno, A.; Wipat, A. SynBioHub: A Standards-Enabled Design Repository for Synthetic Biology. ACS Synth Biol **2018**, 7, 682–688.

(10) Beal, J.; Phillips, A.; Densmore, D.; Cai, Y. Design and analysis of biomolecular circuits; Springer, 2011; pp 225–252.

(11) Pocock, M.; Taylor, C.; McLaughlin, J.; Misirli, G.; Wipat, A. ShortBOL: A shorthand for SBOL. The 8th International Workshop on Bio-Design Automation. 2016.

(12) Misirli, G.; Lord, P. Tawny-SBOL: Using ontologies to design and constrain genetic circuits. The 8th International Conference on Biomedical Ontology. 2017.

(13) Pedersen, M.; Phillips, A. Towards programming languages for genetic engineering of living cells. J R Soc Interface **2009**, rsif–2008.

(14) Beal, J.; Lu, T.; Weiss, R. Automatic compilation from high-level biologically-oriented programming language to genetic regulatory networks. PloS One **2011**, 6, e22490.

(15) Bilitchenko, L.; Liu, A.; Cheung, S.; Weeding, E.; Xia, B.; Leguia, M.; Anderson, J. C.; Densmore, D. Eugene–a domain specific language for specifying and constraining synthetic biological parts, devices, and systems. PloS One **2011**, 6, e18882.

(16) 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.

(17) Vaidyanathan, P.; Ivison, R.; Bombara, G.; DeLateur, N. A.; Weiss, R.; Densmore, D.; Belta, C. Grid-based temporal logic inference. Decision and Control (CDC), 2017 IEEE 56th Annual Conference on. 2017; pp 5354–5359.

(18) Beal, J.; Weiss, R.; Densmore, D.; Adler, A.; Appleton, E.; Babb, J.; Bhatia, S.; Davidsohn, N.; Haddock, T.; Loyall, J.; Schantz, R.; Vasilev, V.; Yaman, F. An end-to-end workflow for engineering of biological networks from high-level specifications. ACS Synth Biol **2012**, 1, 317–331.

(19) Galdzicki, M. et al. The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. Nat. Biotechnol. **2014**, 32, 545–550.

(20) Myers, C. J.; Beal, J.; Gorochowski, T. E.; Kuwahara, H.; Madsen, C.; McLaughlin, J. A.; Mısırlı, G.; Nguyen, T.; Oberortner, E.; Samineni, M.; Wipat, A.; Zhang, M.; Zundel, Z. A standard-enabled workflow for synthetic biology. Biochem. Soc. Trans. **2017**, 45, 793–803.

(21) Mısırlı, G.; Madsen, C.; de Murieta, I. S.; Bultelle, M.; Flanagan, K.; Pocock, M.; Hallinan, J.; McLaughlin, J. A.; Clark-Casey, J.; Lyne, M.; Micklem, G.; Stan, G.-B.; Kitney, R.; Wipat, A. Constructing synthetic biology workflows in the cloud. Engineering Biology **2017**, 1, 61–65.

(22) Roehner, N. et al. Sharing Structure and Function in Biological Design with SBOL 2.0. ACS Synth Biol **2016**, 5, 498–506.

(23) Beal, J. et al. Synthetic Biology Open Language (SBOL) Version 2.1.0. J Integr Bioinform **2016**, 13, 30–132.

(24) Hucka, M.; Finney, A.; Sauro, H. M.; Bolouri, H.; Doyle, J. C.; Kitano, H.; et al., The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics **2003**, 19, 524–531.

(25) Lloyd, C. M.; Halstead, M. D.; Nielsen, P. F. CellML: its future, present and past. Prog. Biophys. Mol. Biol. **2004**, 85, 433–450.

(26) Misirli, G.; Hallinan, J. S.; Yu, T.; Lawson, J. R.; Wimalaratne, S. M.; Cooling, M. T.; Wipat, A. Model annotation for synthetic biology: automating model to nucleotide sequence conversion. Bioinformatics **2011**, 27, 973–979.

(27) Nguyen, T.; Roehner, N.; Zundel, Z.; Myers, C. J. A Converter from the Systems Biology Markup Language to the Synthetic Biology Open Language. ACS Synth Biol **2016**, 5, 479–486.

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

(29) Ham, T. S.; Dmytriv, Z.; Plahar, H.; Chen, J.; Hillson, N. J.; Keasling, J. D. Design, implementation and practice of JBEI-ICE: an open source biological part registry platform and tools. Nucleic Acids Res. **2012**, 40, e141.

(30) Hillson, N. J.; Plahar, H. A.; Beal, J.; Prithviraj, R. Improving Synthetic Biology Communication: Recommended Practices for Visual Depiction and Digital Submission of Genetic Designs. ACS Synth Biol **2016**, 5, 449–451.

(31) Madsen, C.; McLaughlin, J. A.; Mısırlı, G.; Pocock, M.; Flanagan, K.; Hallinan, J.; Wipat, A. The SBOL Stack: A Platform for Storing, Publishing, and Sharing Synthetic Biology Designs. ACS Synth Biol **2016**, 5, 487–497.

(32) Wolfram, S. A new kind of science; Wolfram media Champaign, 2002; Vol. 5.

(33) Misirli, G.; Hallinan, J.; Pocock, M.; Lord, P.; McLaughlin, J. A.; Sauro, H.; Wipat, A. Data Integration and Mining for Synthetic Biology Design. ACS Synth Biol 5, 1086–1097.

(34) Misirli, G.; Wipat, A.; Mullen, J.; James, K.; Pocock, M.; Smith, W.; Allenby, N.; Hallinan, J. BacillOndex: An Integrated Data Resource for Systems and Synthetic Biology. J Integr Bioinform **2013**, 10, 224.

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

(36) Vaidyanathan, P.; Der, B. S.; Bhatia, S.; Roehner, N.; Silva, R.; Voigt, C. A.; Dens-

more, D. A framework for genetic logic synthesis. Proceedings of the IEEE **2015**, 103, 2196–2207.

(37) Ashburner, M. et al. Gene Ontology: tool for the unification of biology. Nat. Genet. **2000**, 25, 25–29.

(38) Courtot, M. et al. Controlled vocabularies and semantics in systems biology. Mol. Syst. Biol. **2011**, 7, 543.

(39) Smith, L. P.; Hucka, M.; Hoops, S.; Finney, A.; Ginkel, M.; Myers, C. J.; Moraru, I.; Liebermeister, W. SBML Level 3 package: Hierarchical Model Composition, Version 1 Release 3. J Integr Bioinform **2015**, 12, 268.

(40) Novère, N. L. et al. The systems biology graphical notation. Nat. Biotechnol. **2009**, 27, 735–741.

(41) Allemang, D.; Hendler, J. Semantic web for the working ontologist: effective modeling in RDFS and OWL; Elsevier, 2011.

(42) Zhang, Z.; Nguyen, T.; Roehner, N.; Misirli, G.; Pocock, M.; Oberortner, E.; Samineni, M.; Zundel, Z.; Beal, J.; Clancy, K.; Wipat, A.; Myers, C. J. libSBOLj 2.0: A Java Library to Support SBOL 2.0. IEEE Life Sci Lett **2016**, 1, 34–37.

(43) Roehner, N.; Myers, C. J. A methodology to annotate systems biology markup language models with the synthetic biology open language. ACS synthetic biology **2013**, 3, 57–66.

Graphical TOC Entry