ACS SyntheticBiology

Research Article

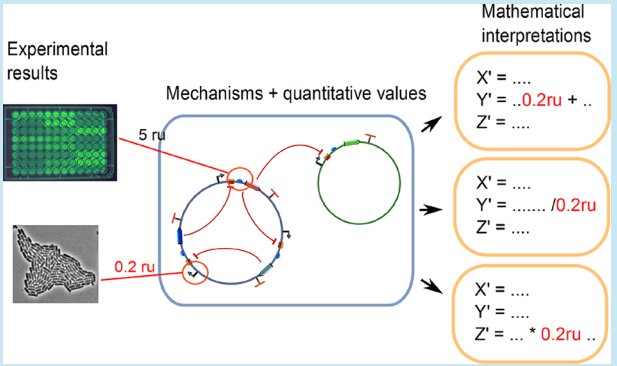
pubs.acs.org/synthbio

Hierarchical Modeling for Synthetic Biology

Deepak Chandran* and Herbert M. Sauro

Department of Bioengineering, University of Washington, Box 355061, William H. Foege Building, Room N210E, Seattle, Washington 98195-5061, United States

ABSTRACT: One of the characteristics of synthetic biology is that it often combines mathematical modeling with experimental work. The link between modeling and experiments is carried out by human researchers who have a conceptual understanding of the underlying biological system. At present, there is no method for representing a conceptual description that can be used to connect mathematical models and experimental data, especially sequence annotations, pertaining to the same underlying biological system. One reason for this limitation is that there *can* exist different mathematical models of the same biological system. In such cases, the same annotation in a DNA sequence would map differently to different models of the same system. In order to enable software support for synthetic biology, a software framework is needed such that it is able to capture a conceptual description of a biological system, including quantitative values, without confining itself to one mathematical model. The novel use of hierarchical modeling inside TinkerCell (www.tinkercell.com) provides one potential software solution for representing a "conceptual diagram" of a biological system. The conceptual diagram does not assume any underlying model. Rather, the diagram is mapped automatically to one of several models. The diagram can then contain information relevant for both modeling and experimental work. Computer-aided design (CAD) can be very useful to synthetic biology. CAD allows engineers to spend more effort at the design stage and less at the construction stage by automatically performing many tasks that are currently performed by human researchers. The ability to automatically link models and experimental results will be one step in the development of practical CAD systems for synthetic biology.



KEYWORDS: *CAD, synthetic biology, hierarchical modeling, ontology, TinkerCell, conceptual model*

Synthetic biology is a highly interdisciplinary field that builds on methods from various established areas of research. If one would observe the laboratory practices in synthetic biology, one might not find any real differences from laboratory practices in molecular biology or microbiology. Similarly, if one looks at mathematical analysis methods in synthetic biology, one may find almost identical methods used in systems biology and related fields. The combination of methods from different disciplines channeled toward a common goal, engineering biological systems, is arguably a characteristic feature of synthetic biology. However, the cost of such interdisciplinary research can be lack of integration. A researcher who is an expert in molecular biology techniques may have difficulty understanding all the details of research performed by an expert in control theory (and vice versa). Such lack of integration within an interdisciplinary field can delay progress. Ideally, software tools should assist researchers in bridging such gaps in knowledge. For example, software tools that can show how parameters in a mathematical model are related to sequence features on a DNA may help researchers in one field better comprehend the research results from another field.

Part of the reason why integration is difficult in software is because of the variety of ways by which a biological system can be represented. When a biological system is defined for the purpose of dynamics systems analysis, researchers may opt to use ordinary differential equations (ODEs) to define the system. When the same system is being defined for the purpose of metabolic engineering, the system might be represented as a stoichiometric matrix.[1] For the purpose of genetic engineering, a system might be a represented using a well annotated DNA sequence. In all of these cases, the underlying biology is the same, but the representation is different. In much of synthetic biology research, these different representations are used together. For example, in order to explain the differentiation mechanism of *B. subtilis*, it was necessary to combine the use of modeling with corresponding changes to the DNA sequence.[2] Use of logic gates in synthetic biology combines the use of digital logic and DNA sequence annotation.[3] In order to use the ribosomal binding site (RBS) strength calculator[4] for designing a system, one must have the sequence of the RBS as well as a quantitative model that utilizes the RBS strength value to predict the dynamics of the system. All of these examples demonstrate the need for integrating sequence-level information with models.

Special Issue: Bio-Design Automation

Received: April 16, 2012
Published: July 23, 2012

NEED FOR INTEGRATING SEQUENCE INFORMATION WITH MATHEMATICAL MODELS

As synthetic biology advances, mathematical models and experimental work would need to be tightly integrated in order for research to progress efficiently. For example, recent work[5] relating to the evolutionary stability of synthetic constructs suggested that metabolic load and repeats in the DNA sequence might be two key factors that determine the time taken for natural selection to cause synthetic systems to malfunction. It might be possible to predict the longevity of a synthetic system using the result of such studies. However, in this particular example, a predictive algorithm would need to use the DNA sequence as well a mathematical model. The model would be used to estimate the metabolic load, and the sequence would be used to identify repeated sequences. Using two separate files as input into the algorithm, i.e., one for the model and one for the DNA sequence, might be prone to mistakes because any changes in the model must correspond to a change in the DNA sequence. Moreover, having two separate input files also limits the value of the algorithm because the algorithm will not be able to automatically adjust the DNA sequence and/or the model in order to optimize the longevity of the synthetic system. Automatic adjustments are not possible because a human is required to map DNA sequence features to model components.

SHORTCOMINGS OF EXISTING METHODS

Integrating sequence information with mathematical models is not as simple as adding additional annotations to existing file formats such as the Systems Biology Markup Language (SBML[7]) or GenBank.[8] The SBML is a file format for representing chemical reactions. It is possible to use the SBML representation to generate stoichiometric matrices, ODEs, or stochastic models. Similarly, the GenBank file format allows software tools to add arbitrary annotations, which can be used to link a SBML file to a GenBank file. The complication arises from the fact that the mapping between sequence and models is not one-to-one. As an example, let us use a system where a protein that is inhibiting its own transcription, i.e., negative autoregulation (Figure 1). The

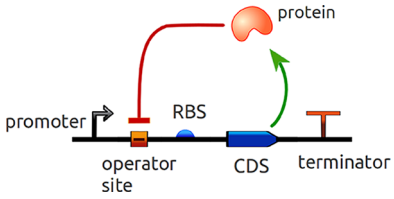


Figure 1. Conceptual diagram of negative autoregulation. The labels show the component *class*. For example, the CDS produces a protein. This diagram can be represented using multiple mathematical models because the biological process that produces the protein from the CDS can be modeled in different ways. The symbols for the promoter, operator site, RBS, and CDS used in this diagram are from the Synthetic Biology Open Language (SBOL).[6]

mathematical model for such a system would probably be the same regardless of the exact protein that is used to construct the real system. For example, one can use the LacI or the TetR transcription factor to build the physical system, but the model in both cases would be the same. Of course, depending on the choice of transcription factor, the parameters in the model might vary, but the basic structure of the model will not change. The DNA sequence encoding the system using LacI will be entirely different from the one using TetR, because the promoter region as well as the CDS sequences will be different. This demonstrates the existence of one-to-many mapping between the model and the DNA sequence. Similarly, there is a one-to-many mapping in the reverse direction as well. Suppose we are interested in building a model for a system where LacI is inhibiting its own transcription. We can opt to model this process using a single ODE, where the production rate of LacI is a Hill function and the degradation rate of LacI is a linear function of the LacI concentration. An alternative way to model the same system might consist of two ODEs: one for the LacI protein and one for the mRNA from which the protein is translated. In yet another version of the model, we may opt to use mass-action kinetics instead of Hill equations. Even for such a simple system, the different ways of modeling can yield entirely different simulated outcomes.[9] Even though the models are different, the underlying DNA sequence as well as many of the other components will remain unchanged. This example demonstrates that there are many ways to model the biological system where LacI is inhibiting its own transcription. An SBML file, arguably the most comprehensive way of capturing a mathematical model of a biochemical system, can still represent only a single mathematical model. Therefore, its formats fail to capture the multiple biological mechanisms of negative autoregulation that might be possible. Similarly, a GenBank file can capture only a specific implementation of negative autoregulation. However, the DNA sequence itself does not capture the *mechanism* of negative autoregulation. Herein lies the shortcoming of existing methods: existing methods are good for capturing specific aspects of a biological system, but none of them are sufficient for capturing the overall concepts themselves, which is what is required for uniting the specific aspects of the system.

Modular modeling methods, such as the ones provided by Antimony[10] and ProMoT[11] software tools, can partially satisfy the issues addressed above. For example, the Antimony model definition language can define a module named "negative_autoregulation", where a module is a model that can be reused inside a larger model. The limitation of existing modular software tools in synthetic biology is that the modules do not have a *type*; in other words, there is no way to determine whether two modules represent the same, or similar, biological mechanisms. The name might provide some clue if the module is named appropriately by the author, but a computer program cannot use the name to identify the underlying mechanism that the module might represent.

A NEW APPROACH

We propose a new approach for representing the concepts of a synthetic biology system. This approach uses four key features: a custom ontology, hierarchical modeling, local parameters, and parent/child relationships. These four features, when used together, can be used to map sequence information to parameters in models and vice versa. Additionally, the approach can be used to map the same conceptual model to multiple mathematical models. Here, a mathematical model refers to a set of reactions with specific reaction rate equations, which can be best represented by an SBML file. The same method can also be used to map experimental results to parameters or variables in models, although this article will focus on just the integration of DNA sequence information and mathematical models.

"CONCEPTUAL MODEL" VERSUS "MODEL"

In the context of this article, the term "model" refers to any description of a system that is amenable to mathematical analysis.

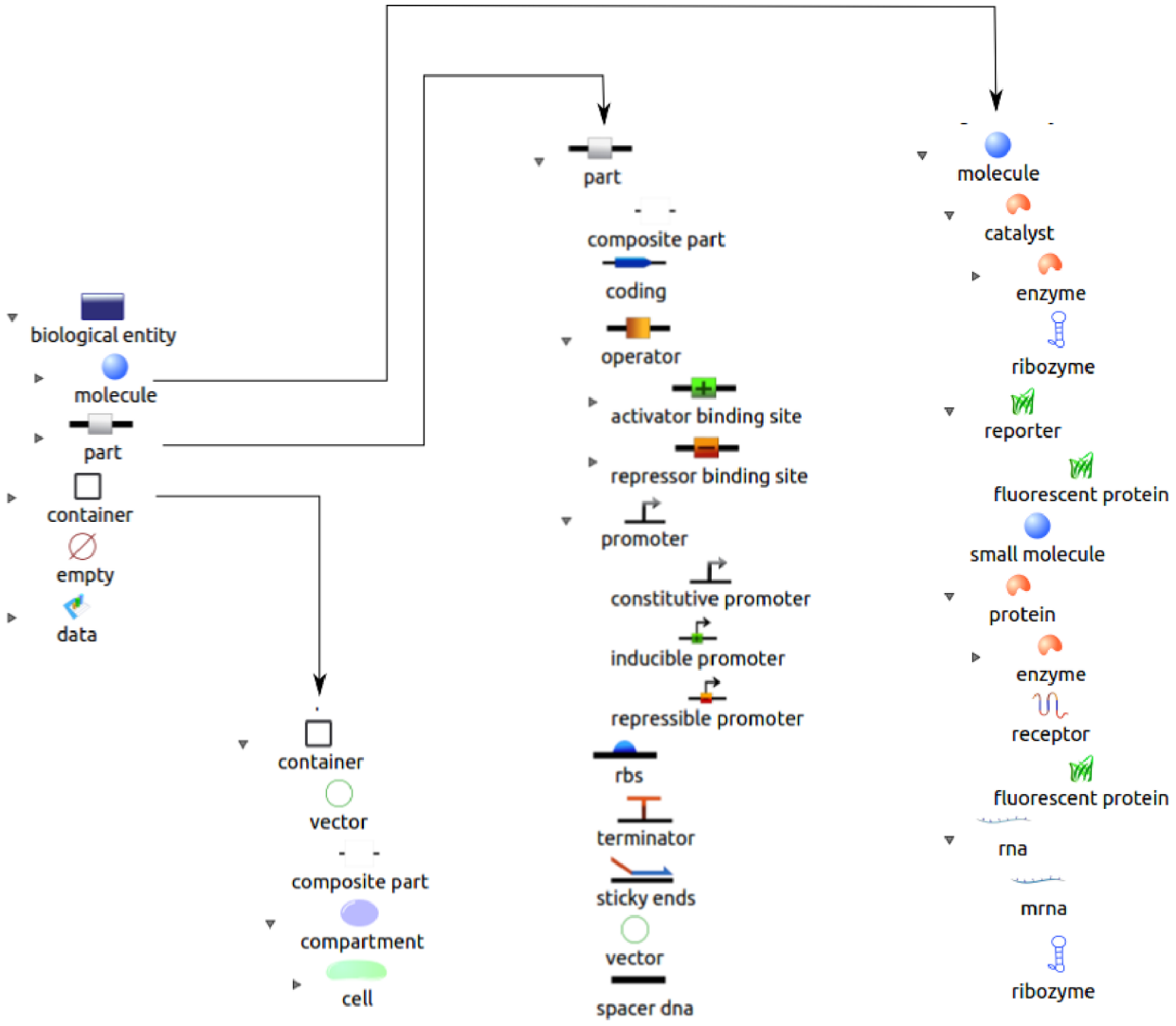


Figure 2. Classes categorized as the biological entity and their inheritance relationships. The ontology is divided into three major classes, of which the Biological Entity class describes any object that can act as a participant in a biological process. This figure shows almost all of the Biological Entity families. The three main branches within the biological entity class are "container", "part", and "molecule", each of which are further expanded in the figure (shown by the black lines). In the figure, indentations indicate inheritance relationships. For example, at the bottom right, both "mRNA" and "ribozyme" are subclasses of the class "RNA". Some classes (e.g., "fluorescent protein") are subclasses of multiple parent classes (e.g., "protein" and "reporter").

An SBML file is an example of a "model", by this definition. It is important to note that an SBML file can be simulated using stochastic simulation algorithms, deterministic simulation algorithms, or sometimes even Boolean simulation algorithms. However, the mechanisms represented by each of those algorithms is the same.

In the context of this article, the term "conceptual model" refers to a description that does not specify the assumptions and detailed mechanisms that are necessary to create a complete model, such as an SBML file. The purpose of the conceptual model is to act as the link between different types of analyses, databases, and even experimental data. It is important to note that the conceptual model *does* contain parameters. These parameters reflect real measurable parameters rather than mathematical model parameters. These parameters can be used in different ways in different mathematical models.

Analysis of Conceptual Models at Present. One can argue that it is possible to build minimal mathematical models that reflect only the known interactions of a conceptual model. For example, it is possible to describe gene regulation using a single equation that reflects how the transcription factor(s) influence transcription of a target gene. Although a simple mathematical model might be sufficient to describe a large number of cases, the problem in describing a conceptual model with a single mathematical model is that we entangle the assumptions of the mathematical model with the conceptual model. Suppose we realize that it is necessary to include the mRNA in the mathematical model in order to capture the observed experimental data, then we cannot simply change the mathematical model without changing the conceptual model. Decoupling the conceptual model from the mathematical model (in software) allows one to update the mathematical model without changing the basic conceptual diagram of the system under study.

OVERVIEW

The ontology provides the necessary terminology and structure for labeling processes such as transcription regulation or enzyme catalysis.

dx.doi.org/10.1021/sb300033q | *ACS Synth. Biol.* 2012, 1, 353–364

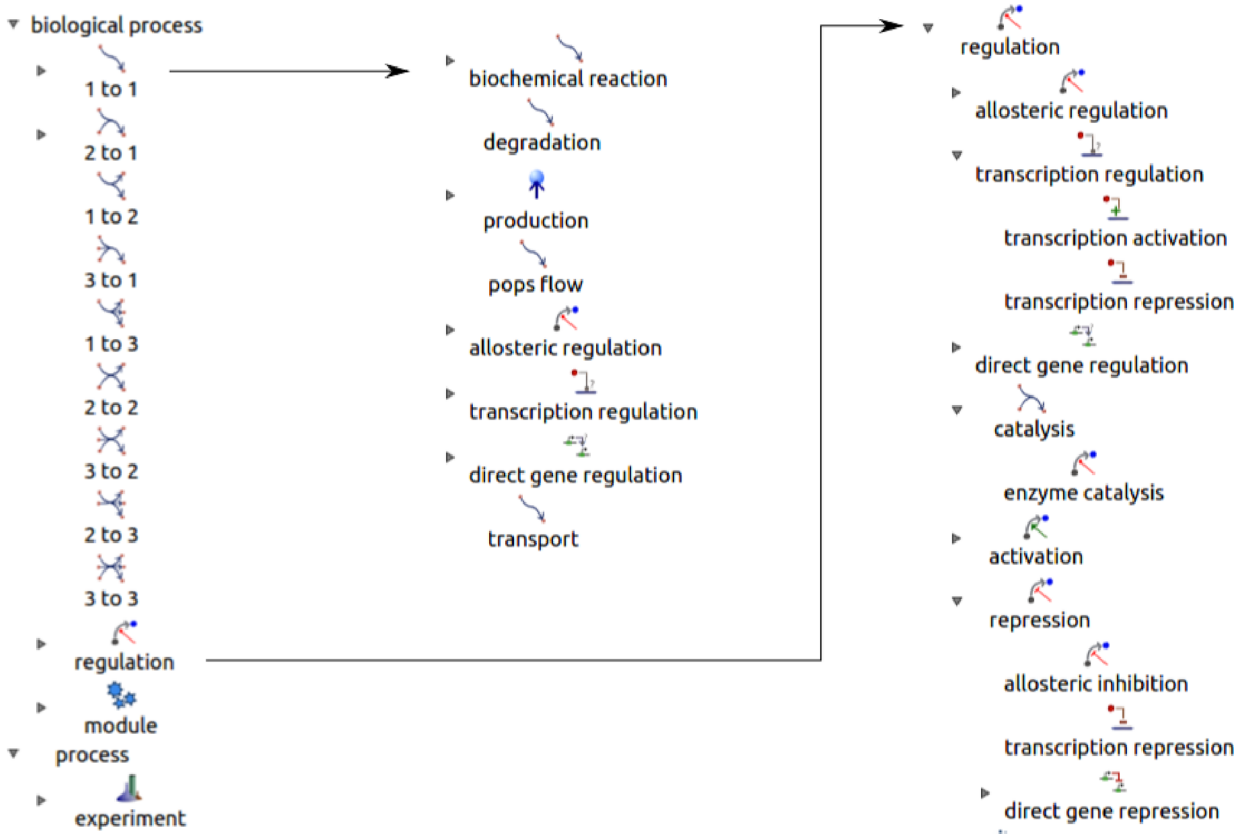


Figure 3. Classes categorized as the Biological Processes and their inheritance relationships. The ontology is divided into three major classes, of which the Biological Process class describes regulations and regulations involving biological entities as participants. Two main classes, "1 to 1" and "regulation", that are subclasses of the Biological Process class are further expanded in the figure. The "1 to 1" class represents processes where only two molecules are involved, and the "regulation" class represents processes such as catalysis, repression, or upregulation. In the figure, indentations indicate inheritance relationships. For example, at the top right, both "transcription activation" and "transcription repression" are subclasses of the class "transcription regulation".

Hierarchical modeling is used to map each process to one or more specific models. Local parameters such as promoter strength or protein degradation rates are used to map parameters in one model to matching parameters in another model. Parent/child relationships are used to group components of a system, such as molecules that belong inside a cell, parts that belong inside a composite part, or parts that belong on the same plasmid. The details of each individual feature will be described in detail in the next four sections.

■ ONTOLOGY

An ontology, for the purpose of the proposed method, is a structured vocabulary that allows computer programs to give meaning to terms by relating the terms to one another. There are a handful of existing ontologies that are used to describe different aspects of biological systems. Gene Ontology (GO[12]), Systems Biology Ontology (SBO[13]), BioPAX,[14] and Sequence Ontology (SO[15]) are four examples of ontologies (or structured vocabularies) that are used to describe gene functions, kinetics, biochemical pathways, and DNA sequence features, respectively. The method proposed in this work uses a custom ontology, but the custom ontology, in the future, should be an extension or combination of these existing ontologies. This section will describe the basic features of the custom ontology and how existing ontologies can be reused in future work to replace this custom ontology.

The custom ontology used to achieve the goals of this article consists of three main classes: biological entities, biological processes, and participants. The class of biological entities refers to molecules such as proteins, RNA, or metabolites as well as DNA "parts" such as protein coding sequences, RBSs, promoters, and operator sites. The class of biological processes consists of reactions and regulations that involve two or more biological entities. For example, transcriptional regulation is a biological process involving a transcription factor and an operator site. Similarly, enzyme catalysis is a biological process involving an enzyme and two metabolites. The participants class describes the *role* of a biological entity inside a biological process. For example, in enzyme catalysis, the roles include *catalyst*, *substrate*, and *product*. The substrate and product roles are associated with the two metabolites, and the catalyst role is associated with the enzyme. The biological entities class and the biological process class may contain *attributes*, or values that provide additional information about the entity or process. The DNA sequence is an example of an attribute that is present in all DNA parts. Similarly, the dissociation constant is another attribute that will be present in biological processes involving binding, such as allosteric regulation and transcriptional regulation.

There are subclasses within the three major classes in the ontology. The three main subclasses of biological entities are *molecules*, *DNA parts*, and *compartments*. Subclasses of *molecules* include *proteins*, *RNA*, and *small molecules*. Some classes can belong to two or more parent classes. For example, the class

dx.doi.org/10.1021/sb300033q | *ACS Synth. Biol.* 2012, 1, 353–364

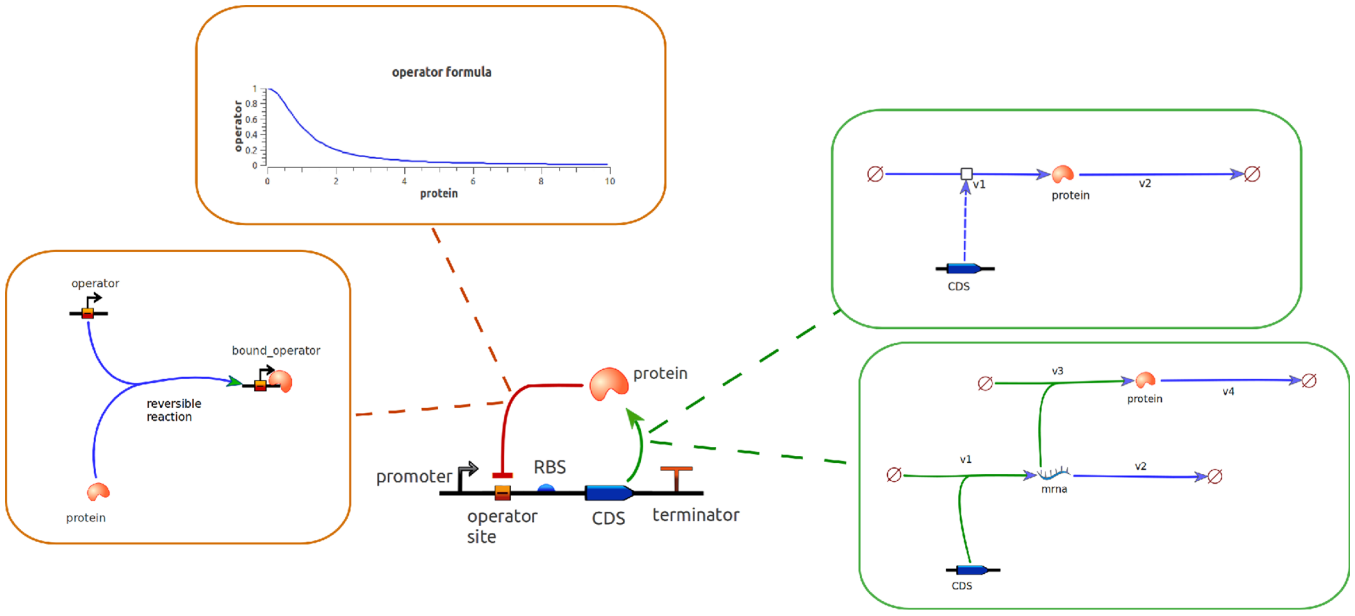


Figure 4. A conceptual diagram of negative autoregulation and different ways of generating a complete mathematical model. In this diagram, there are two main processes: the CDS producing the protein and the protein regulating the promoter via the operator site. There are different ways to mathematically model each of these processes. In other words, each process can be mapped to one of many mathematical models. Using hierarchical modeling and ontologies, such mapping from a conceptual model to more specific models is possible.

named *fluorescent proteins* is a subclass of *proteins* and *reporter molecules*. Subclasses under the class of *DNA parts* include *promoters*, *RBS*, *coding region* (synonym: *CDS*), *operator*, and *transcriptional terminator* (synonym: *terminator*). The class names can have synonyms. For example, *parts* and *DNA parts* are synonyms for the same class. Figure 2 shows the complete set of classes in the biological entities category.

The class of biological processes is divided into two major categories: reactions and regulations. The category of reactions are represented by several classes: "1 to 1", which represents processes involving two molecules, "1 to 2" and "2 to 1", both of which represent processes involving three molecules, and so on (see Figure 3). Regulations consist of enzyme catalysis, allosteric regulation, and transcriptional regulation. Two major subclasses of regulation are *repression* and *activation*. Some classes may have more than one parent class. For example, *transcriptional repression* is a subclass of *transcriptional regulation* and *repression*, and *allosteric activation* is a subclass of *allosteric regulation* and *activation*. Biological processes can also contain parameters. For example, the process named *transcriptional regulation* would contain a parameter called *Kd*, the dissociation constant for the transcription factor. Similarly, *enzyme catalysis* consist of two parameters: *Km*, the Michaelis–Menten constant, and *Kcat*, the catalytic constant. Note that these parameters do not imply a mathematical model. They can be used in the context of mathematical modeling, but the reason that the parameters are defined within the ontology is so that parameters from one mathematical model can be mapped to parameters in another mathematical model for the same underlying system. Moreover, some of these parameters can be measured experimentally, and therefore, the ontology can be used to connect experimentally measured values to the appropriate parameters in mathematical models. Figure 3 shows the complete set of classes under the biological process category.

The class of participants has far fewer subclasses when compared to the class of biological entities or biological processes. Participants include roles that biological entities take inside a biological process. Some of these roles include *regulator target*, *catalyst*, *reactant*, and *product*. The participant classes also follow a hierarchical structure. For example, *activator* and *repressor* are two classes that are subclasses of *regulator* class, and *substrate* is a subclass of the generic *reactant* class.

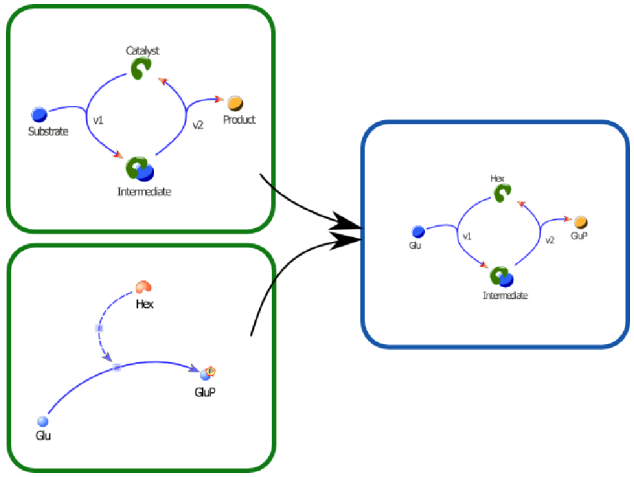


Figure 5. Automatic mapping of a simple conceptual model to a specific model. The conceptual model (bottom left) simply indicates that an enzyme nameds Hex is converting a molecule called Glu to GluP. The conceptual model is annotated using the ontology (not shown in diagram). A software tool is able to take a specific model of enzyme catalysis (top left) and automatically map components and parameters in the conceptual model to variables in the specific model.

Relationship to Existing Ontologies. There are a few ontologies that are used in the fields of systems biology and bioinformatics, with Gene Ontology (GO[12]), Systems Biology Ontology (SBO[13]), and BioPAX[14] being three prominent ones. Of these three, the custom ontology described in this article is most similar to the BioPAX standard, which is used to represent biochemical pathways. BioPAX defines biological processes in

dx.doi.org/10.1021/sb300033q | *ACS Synth. Biol.* 2012, 1, 353–364

terms of participants, hence the similarity. However, BioPAX does not define parameters for biological processes. Further, the biological entities in BioPAX do not include all of the entities that are required in synthetic biology, such as the class of DNA parts. The other two ontologies, GO and SBO, do not define the participants that comprise a biological process, and for that reason, they cannot be used to achieve our goals. Note that SBO does define parameters, and therefore, there is some overlap between SBO and the custom ontology that we have proposed.

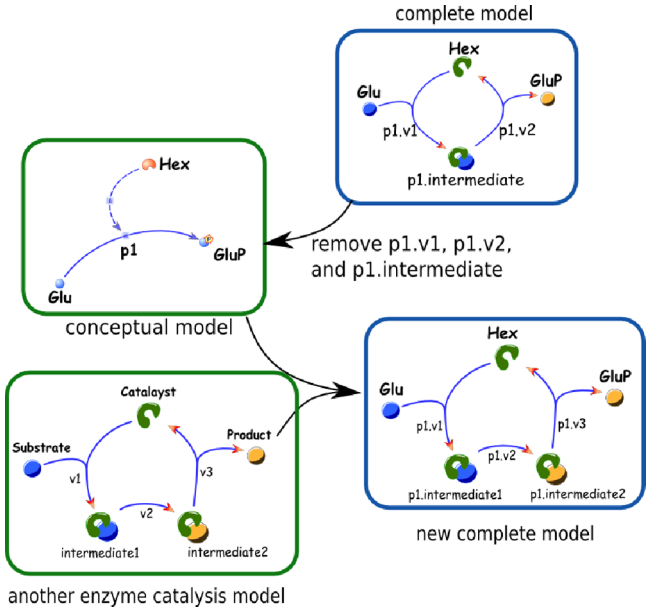


Figure 6. Automatic mapping of a simple conceptual model to multiple specific models. The conceptual model indicates that an enzyme names Hex is converting a molecule called Glu to GluP. With the support of the ontology the components in the conceptual model can be mapped to variables in the specific model. The specific model can be defined in any modeling format, e.g., SBML. The criteria is that the variables and parameters in the specific model must match the participant roles in the ontology. Note that the additional variables, such as the intermediate complexes, are child items of the process called J1, hiding them from the conceptual diagram.

An Example Where Existing Ontologies Fail. Consider a relative simple example where transcription factor *X* acts as a repressor for promoter region *P*, and the kinetics of that repression is captured by the dissociation constant *Kd*. This is a conceptual model because it is neither a physical construct nor a mathematical model. It can be mapped to specific physical constructs, e.g., the LacI protein being used in the place of *X* and the pLac promoter for *P*. Similarly, it can be mapped to a specific set of reactions for generating a mathematical model. The GO can identify this process as a "negative regulation of RNA polymerase II transcriptional preinitiation complex assembly". However, GO alone would not allow a software tool to make the association between *X* and LacI because GO would only provide a description for the whole process and not its individual participants. A software tool that uses GO to label this specific process would not be able to determine the fact that LacI cannot be mapped to *P*. BioPAX can identify the repression as a process called "TemplateReactionRegulation" with *X* being the controller and *P* being the controlled participant, and thus, BioPAX can provide sufficient information for mapping *X* to LacI and *P* to *pLac*. However, the kinetic description, *Kd*, cannot be captured using the BioPAX formalism (and GO as well). SBO can be used to capture the *Kd* value and its biological meaning, but like GO, SBO does not contain any information that allows a software tool to map *X* to LacI. The inability to perform such mappings makes these ontologies inadequate for hierarchical modeling where abstract designs need to be mapped to specific models or specific physical constructs.

LOCAL PARAMETERS

Mathematical models (e.g., in the form of ODEs) are generally defined using global parameters, or parameters that belong with the model. In the SBML format, local parameters behave differently from global parameters because local parameters cannot be accessed globally. In the proposed framework, all parameters are locally defined but globally accessible. In other words,

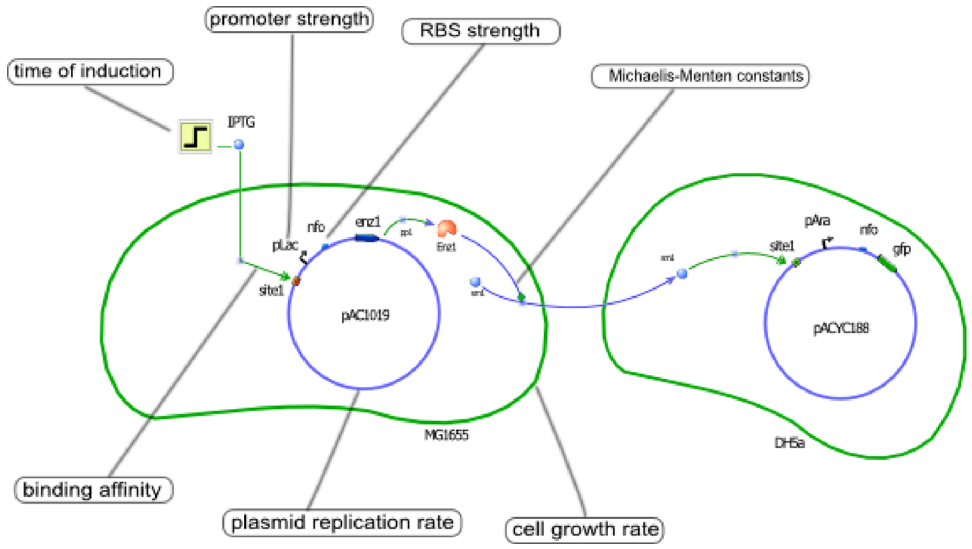


Figure 7. Conceptual diagram created using TinkerCell. TinkerCell implements hierarchical modeling, supported by an ontology. This feature allows TinkerCell to map conceptual models such as this quorum sensing model to multiple mathematical models. Each complete mathematical model can be exported as a SBML file or MATLAB code or simulated within TinkerCell, using the COPASI systems biology package.[16]

dx.doi.org/10.1021/sb300033q | *ACS Synth. Biol.* 2012, 1, 353–364

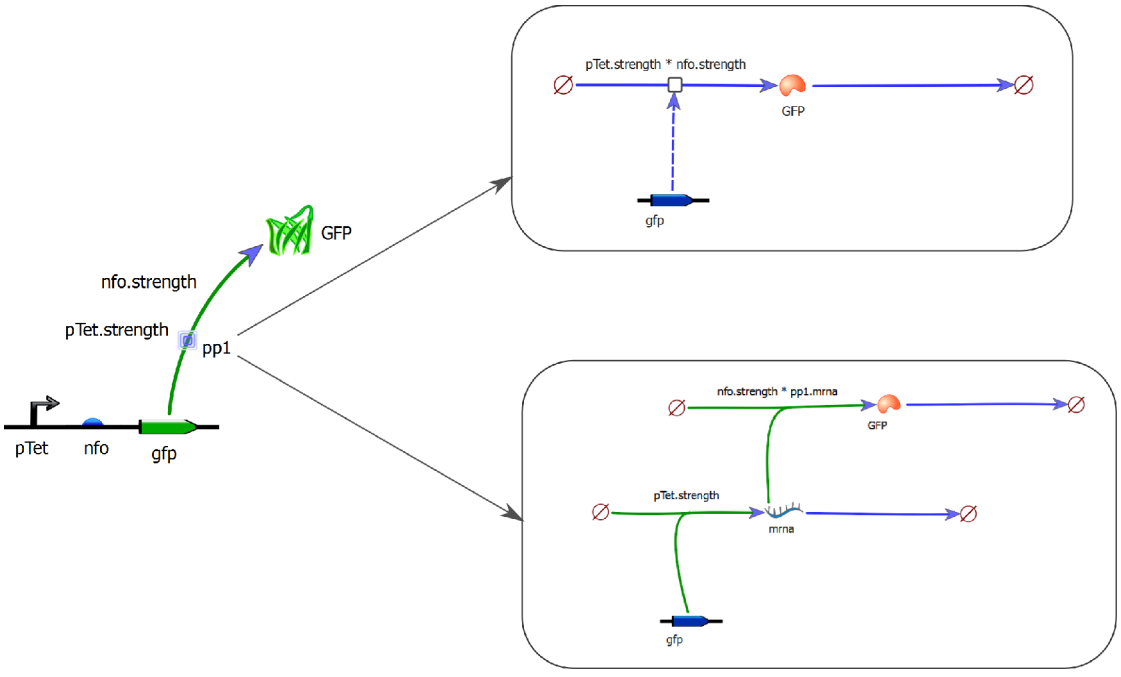


Figure 8. A higher-level conceptual diagram of protein production is mapped to two potential mathematical models. This is a simple example, but the general idea is applicable to larger and more complicated cases. The parameters in the conceptual model are the promoter strength and RBS strength. One of the mathematical model does not contain the mRNA intermediate, and therefore, for that particular model, the rate of production of the protein is determined by the product of the RBS strength and promoter strength (note that mRNA degradation is assumed to be 1/min in this case). Of course, the TinkerCell user can edit any of these default equations or can create an entirely new mathematical model for the protein production process.

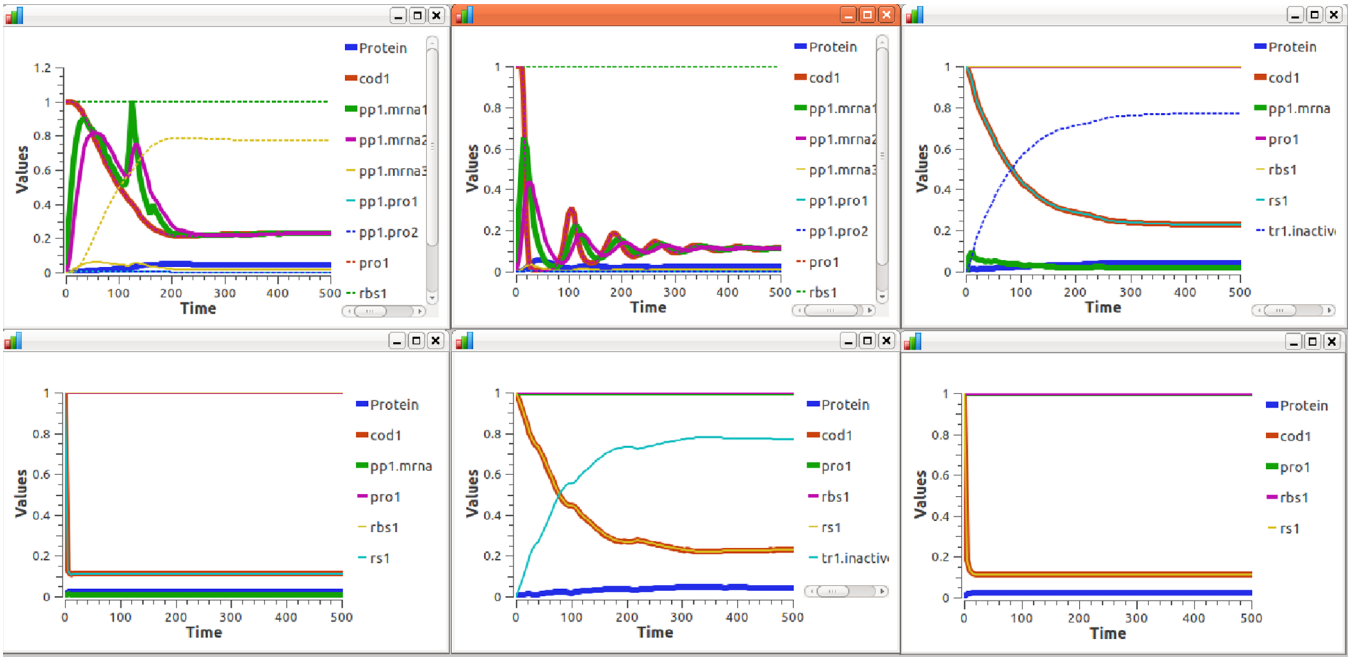


Figure 9. Different mathematical representations of the same conceptual diagram can result in *qualitative* differences as well as quantitative difference in the analysis. The set of graphs shown here are the outputs from a TinkerCell add-on that automatically generates different possible mathematical models from the same conceptual diagrams and performs the necessary parameter mappings. These graphs represent different possible behaviors for the negative autoregulation system shown in Figure 4 depending on the details of the mathematical model.

local parameters behave similar to normal parameters except for the fact that they contain a prefix that describes their origin. For example, the strength of Promoter part *P1* will be named *P1strength* (*P1_strength* is also appropriate). Similarly, the dissociation constant associated with a biological process named *reg1* will be named *reg1Kd* (or *reg1_Kd*). In this sense, all parameters are associated with some model component. While this created restrictions on parameter names in a model, there is a clear advantage from the point of view of databases. Suppose a software program is trying to load a promoter part from a database and wants to update the values of the corresponding mathematical model. If the parameters are named without any

dx.doi.org/10.1021/sb300033q | *ACS Synth. Biol.* 2012, 1, 353–364

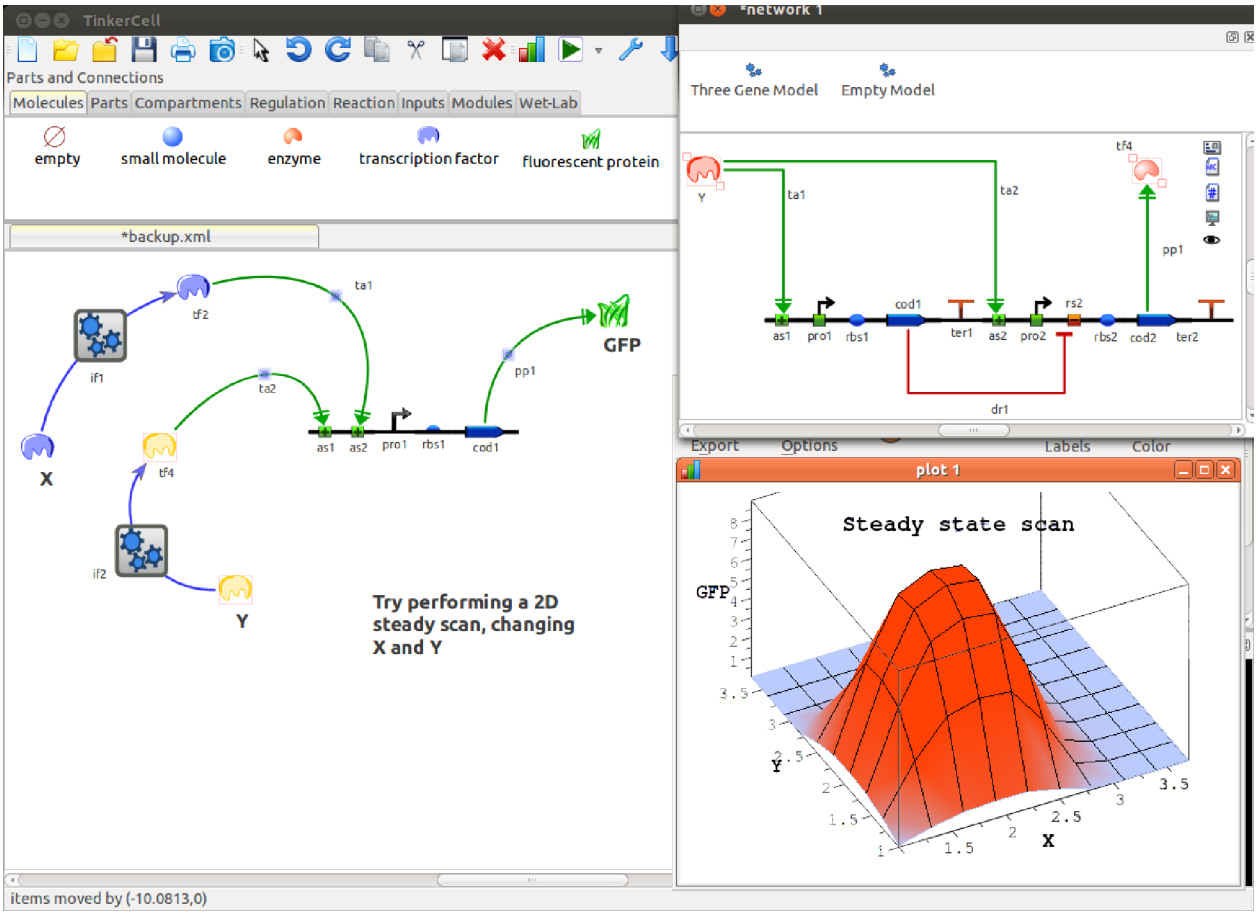


Figure 10. Modular feedforward networks used inside a larger network. A feedforward network can be considered a "module" because of its specific input-output response. Assuming that this behavior is preserved when one connects the module to another genetic circuit, it is possible to design larger systems using such modules. This TinkerCell screenshot shows how a user can design complex networks using modules. The boxes in this screenshot contain feedforward networks within them. If needed, the user can double-click on the icons to see and edit the individual modules.

convention (e.g., *k1*, *k2*, *k3*), it is practically impossible to determine which parameter should be updated. In contrast, with the proposed method, it is relatively simple to find all parameters associated with a specific promoter: suppose the updated promoter was named *P1*, then the software program simply needs to update all parameters that have the prefix *P1.* or *P1_*.

Local Parameters Are a Way of Organizing a Model. The notion of a local parameter, as provided in the previous paragraph, is intended to be quite simple. From a computer science point of view, the concept is comparable to the difference between object-oriented programming and functional programming. In object-oriented programming, a function (or method) is called through the class that contains that function, e.g., "V.func()", whereas in functional programming languages, the same program can be written without any classes, and the function would simply be called by a programmer-defined name, e.g., "funcV() or Vfunc()". While any object-oriented code can be converted to functional code, the additional layer of organization that object-oriented methodology provides makes it appealing and is one reason why it has become the more popular means of programming. The idea of local parameters is analogous.

Parameter Names Are Provided by the Software Tool. It is important to note that a user would not write the complete parameter names, e.g., "P1k1". Rather, the user would only define the individual names, "P1" and "k1". The software tool that is generating the models would automatically generate the complete parameter names. This automation is necessary in order to avoid inconsistencies in the names.

HIERARCHICAL MODELING

Hierarchical modeling is used to represent a conceptual, or high-level, description of a model. A gene negatively regulating itself is an example of a conceptual model (Figure 4). This description is a conceptual model because the negative regulation can occur via different mechanisms, and each mechanism would require a different model. For example, the interaction between the transcription factor and its target operator site can be described using two separate reactions representing the binding and unbinding of the transcription factor, or the same process can be described using a single Hill equation. Both approaches are useful, depending on the purpose of the mathematical model. However, the DNA sequence for implementing a specific negative autoregulation would remain unchanged in either case because the DNA sequence does not contain the mechanistic details.

Automatic Mapping to Multiple Models. Hierarchical modeling is used in conjunction with the ontology to provide mapping from complete models to complete mathematical models. The basic idea behind performing this mapping is to map individual biological processes to predefined models, where these predefined models can be SBML files, ODE equations, or any

dx.doi.org/10.1021/sb300033q | *ACS Synth. Biol.* 2012, 1, 353–364

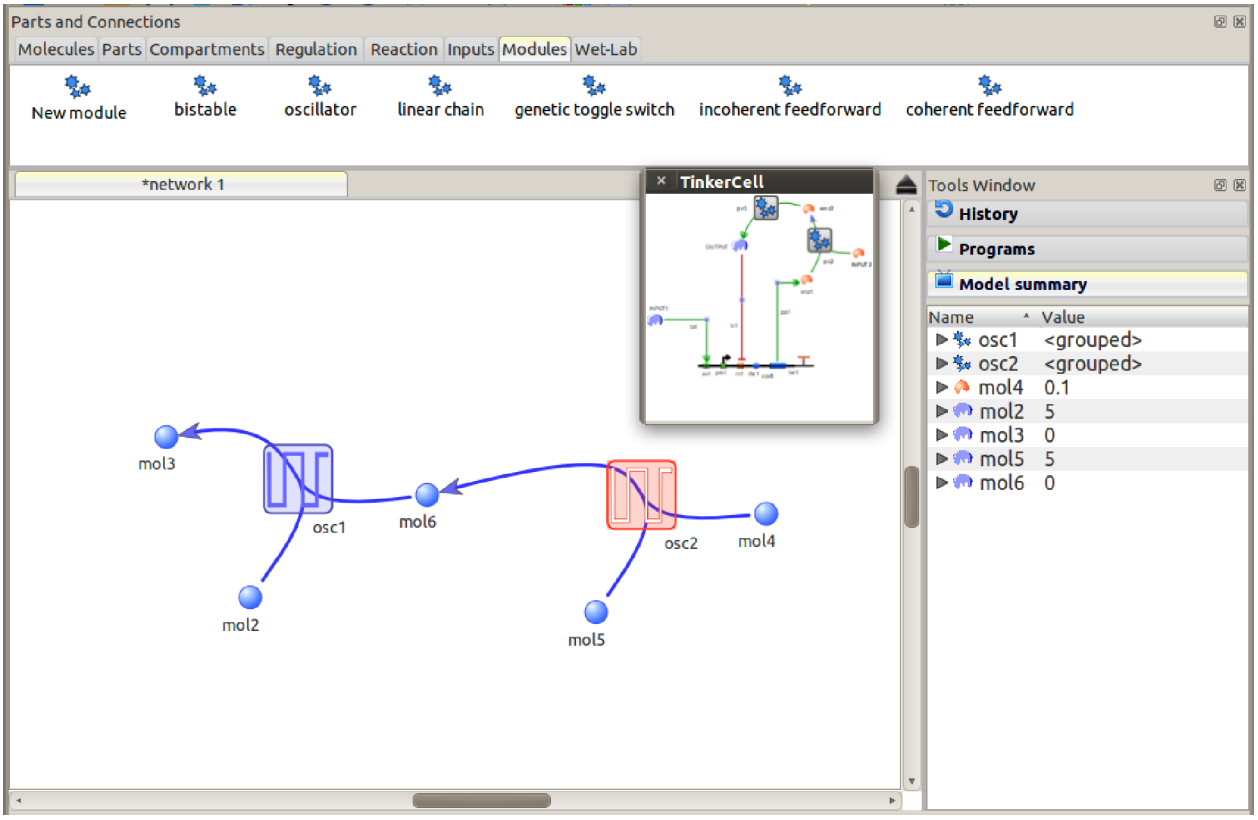


Figure 11. TinkerCell screenshot showing a modular network composed of two coupled oscillators. Each square block represents an oscillator with three participants. Suppose an engineer is interested in connecting one oscillator to another and is not interested in the specific details of each oscillator. This diagram captures the high-level description of a system that the engineer is interested. Each oscillator can be mapped to one or more specific mathematical models or constructs representing specific oscillator designs. The small window in the figure shows the user how one of the oscillators is implemented, which the user can replace without affecting the conceptual model.

other form of complete model. Figure 5 shows how this is accomplished for one biological process. In this figure, the conceptual model is just one process, enzyme catalysis. The participants include one enzyme, named Hex, and two metabolites, named Glu and GluP. The conceptual model must be supported by an ontology in order to generate the complete model automatically. The ontology would describe the term "enzyme catalysis" as a process involving a *catalyst*, which belongs to the class *enzyme*; a *substrate*, belonging to class *molecule*; and a *product*, belonging to class *molecule*. Additionally, this process also contains two parameters: *Km* and *Kcat*. Any predefined model that uses the three variables catalyst, substrate, and product and the two parameters *Km* and *Kcat* is a valid candidate model for automatic mapping. Figure 6 shows how one conceptual model can be mapped to multiple candidate models. Note that the individual models have additional variables and parameters. For example, the intermediate enzyme−substrate and enzyme−product complexes exist in the complete model but not the conceptual model. Further, the different models would have additional parameters for the additional reactions that are not visible in the conceptual model. All of these additional details are treated as *child* components and parameters, which is discussed in the next section.

Automatic Mapping of Parameters. In order to map parameters between models, the individual models, such as the ones shown in Figures 5 and 6, must use the parameter names defined in the ontology. In the example shown in Figure 6, the ontology defines two parameters for the Enzyme Catalysis process: *Km* and *Kcat*. The submodels shown in Figure 6 can choose to use these parameter names in different ways, depending on the assumptions of each model. However, it is assumed that the person creating these submodels understand the biological meaning of the two parameters and will use the parameters in a way that reflects their biological significance. A submodel may choose to ignore one or more of the parameters.

PARENT/CHILD RELATIONSHIPS

The parent/child relationship between components of a conceptual diagram is essential for describing physical aspects of biological system. Examples include molecules inside a cell, where the cell is the parent component that contains the molecules, or DNA parts on a plasmid, where the plasmid is the parent component. For the purpose of synthetic biology, composite parts, or parts composed of multiple parts, is another case where parent/child relationship is needed to describe the physical structure of the DNA appropriately.

The parent/child relationship is also necessary for supporting certain aspects of hierarchical modeling. Hierarchical modeling, as it is used in the proposed framework, allows connecting a general model to one or more specific models. Returning to the example as the one shown in Figure 6, suppose that we choose to model the enzyme catalysis using one intermediate complex. When using this approach, an additional molecular species, i.e., the enzyme−substrate intermediate complex, would need to be included in the model that does not exist in the original conceptual model. This additional component will be a *child*, or subcomponent, of the enzyme catalysis process (named J1 in Figures 5 and 6). The reason for making this complex a child

dx.doi.org/10.1021/sb300033q | *ACS Synth. Biol.* 2012, 1, 353−364

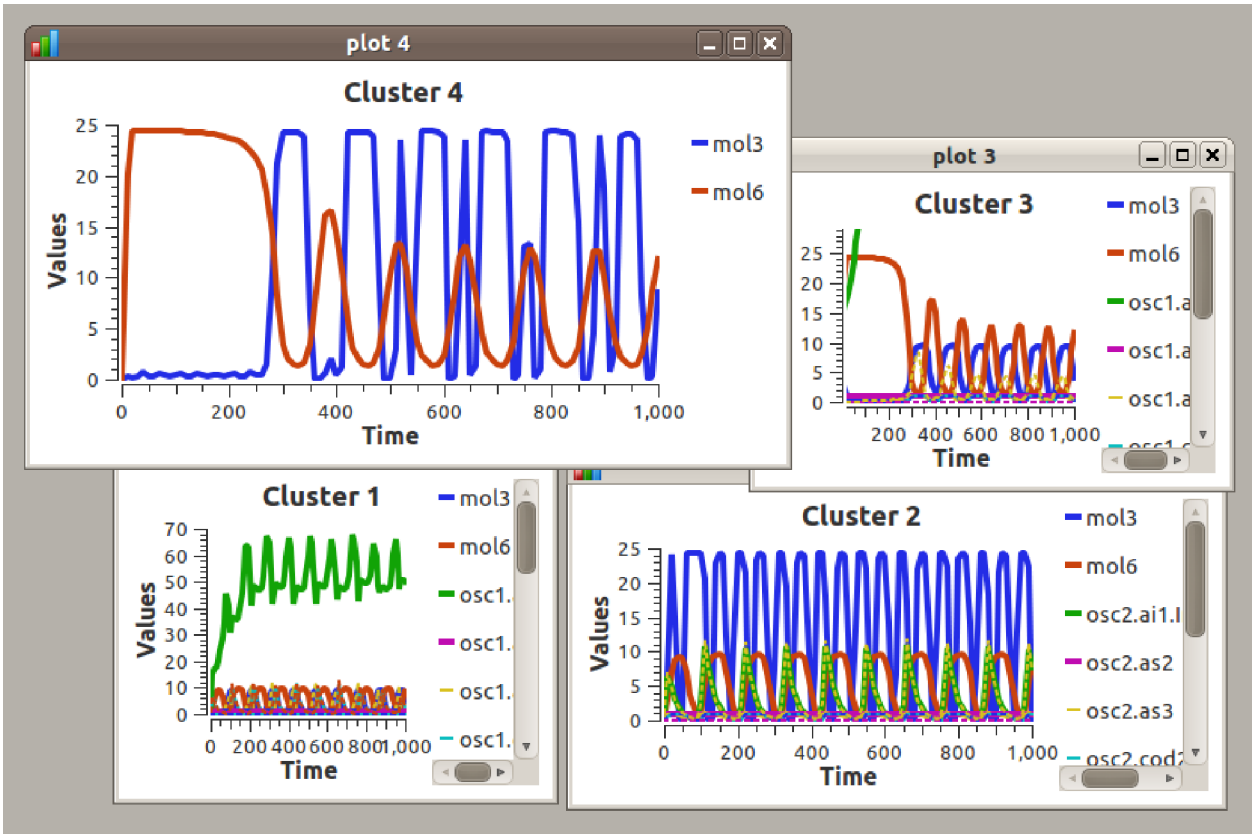


Figure 12. Simulated output from modular network composed of two coupled oscillators shown in Figure 11. The different mathematical models are generated automatically by mapping each oscillator process to two predefined oscillator models (thus a total of four possible combinations). Interestingly, one of the combinations (cluster 4) results in one oscillator controlling the frequency of the other (for the default parameter set).

component is because the complex only comes into the model if we choose a specific type of modeling approach, and therefore, this complex cannot be part of the generalize model shown in 1, which is supposed to map to multiple modeling approaches. The parent/child relationship is an integral feature for supporting hierarchical modeling because it is essential for representing intermediate complexes that exist in one model but not the other.

IMPLEMENTATION

The software tool TinkerCell[17,18] is a demonstration of an integrated framework that combines sequence features with mathematical models via the use of ontologies, hierarchical modeling, local parameters, and parent/child relationships. Figure 7 shows an example of a conceptual model created using TinkerCell. Note that this figure combines aspects of a physical construct, such as the plasmids, which may not be relevant for a mathematical modeling. A parameter, such as promoter strength, can be traced back to a DNA sequence on the plasmids, and therefore, when a user changes the plasmid sequence, it is possible for TinkerCell to relate the change in sequence back to specific parameters in a mathematical model. Each arc in the figure represents a biological process that can be mapped to one or more mathematical models. The complete model can then be simulated within TinkerCell itself, using the COPASI package,[16] or exported to SBML or MATLAB code.

Quorum Sensing Example. Figure 7 is a typical example of a conceptual diagram that can be represented using the framework described in this manuscript. The figure was generated using the TinkerCell software tool; the additional labels on the figure—binding affinity, plasmid replication rate, cell growth rate, Michaelis–Menten constants, RBS strength, promoter strength, and time of induction—are not part of the original TinkerCell screenshot. These additional labels show some of the parameters associated with the conceptual model itself. For example, the plasmid replication rate is a parameter that describes some physical aspect of the plasmid (i.e., the replication origin). This parameter might be used in different ways in different mathematical interpretations of the conceptual model. These parameters are the *local* parameters because they are associated with some component. The parameters describing the dynamics of individual processes, such as the enzyme catalysis, are also local parameters because they belong with that particular process. The parent/child relationship is somewhat obvious in this example: all of the DNA parts on the plasmid are children of the plasmid. Similarly, the plasmid itself is a child of the cell that it resides in. The reactions and regulations within a cell are also children of that cell. This particular model includes some additional detail, such as the step function (labeled as "time of induction" in Figure 7). The step function describes how the IPTG molecule is administered into the system. In the next section, the mapping of local parameters to multiple mathematical models will be explained.

Mapping to Two Mathematical Models of Protein Production. The production of a protein from a gene requires an mRNA intermediate. However, when modeling the process mathematically, it is not always necessary to include this intermediate.[19] TinkerCell allows for this flexibility. As shown in Figure 8, the parameters in the conceptual model are automatically mapped to the lower, more detailed, models. In this example, the parameters include promoter strength and RBS

dx.doi.org/10.1021/sb300033q | *ACS Synth. Biol.* 2012, 1, 353–364

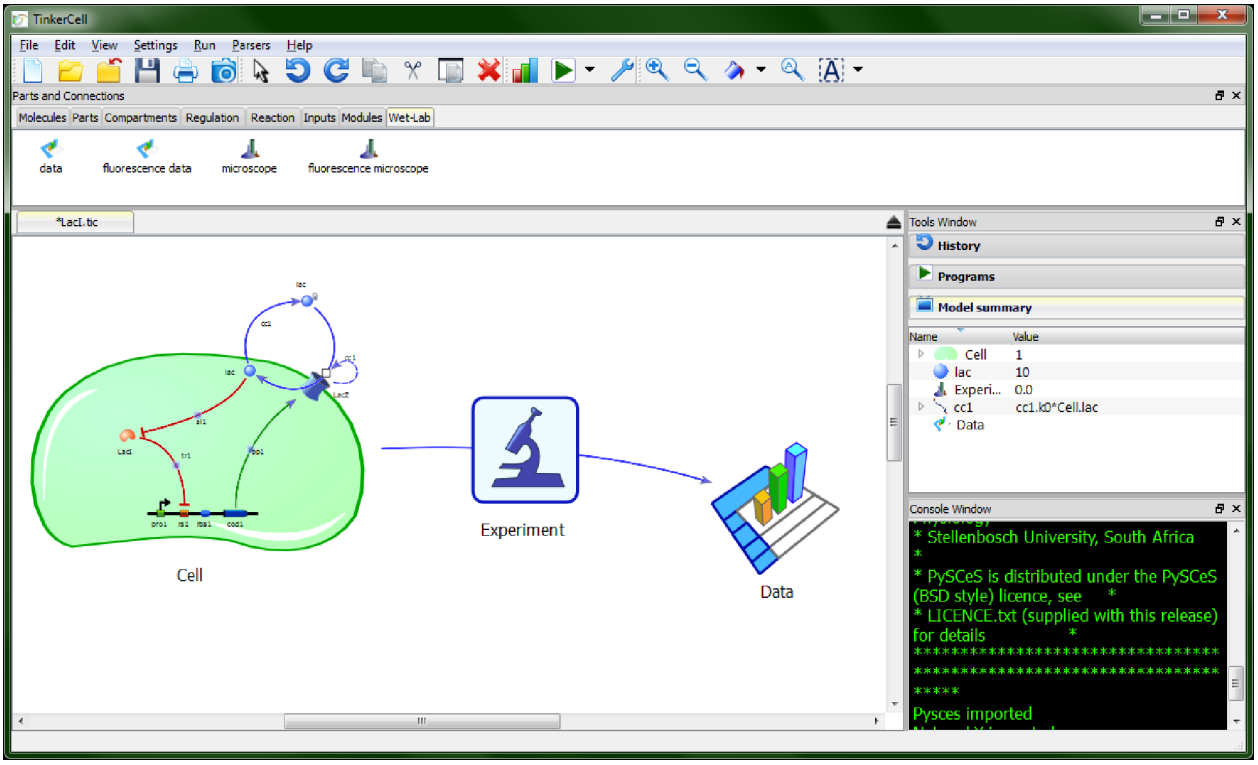


Figure 13. Representing experiments in TinkerCell. This screenshot shows a process where some experiment was performed on an engineered cell in order to obtain some data. This screenshot demonstrates that an "experiment" is simply another type of process, just like all of the other biological processes. Its participants include a cell and a piece of data, the results of the experiment. The experiment itself will have a type, e.g., "fluorescent microscopy movie" or something of that sort. The data would also have a type. The program TinkerCell currently supports the basic framework for creating such diagrams, but no functions are available for performing any sort of analysis on this type of diagram.

strength. In the model without mRNA, the protein production rate is a function of both parameters. This mapping is automatically performed by TinkerCell because the ontology describes "protein production" using two parameters: transcription rate and translation rate. Because the biological meaning of these parameters are understood by the software (via the ontology), it is possible to map the promoter strength parameter to transcription rate and RBS strength to translation rate.

In order to map parameters between models, the individual models, such as the ones shown in Figures 5 and 6, must use the parameters names defined in the ontology. In the example shown in Figure 6, the ontology defines two parameters for the Enzyme Catalysis process: *Km* and *Kcat*. The submodels shown in Figure 6 can choose to use these parameter names in different ways, depending on the assumptions of each model. However, it is assumed that the person creating these submodels understand the biological meaning of the two parameters and will use the parameters in a way that reflects their biological significance. A submodel may choose to ignore one or more of the parameters.

REPRESENTING MODULES

The capability to represent conceptual diagrams automatically enables another feature: modules. Modules, in this context, are subnetworks of a larger network. Additionally, they have some human-defined interpretation. For example, a bistable region of a larger network can be considered a module. Similarly, an oscillatory component of a network can be considered a module. The advantage of using modules to construct models is that it makes a large model more understandable to humans. For example, Figure 10 shows two feedforward networks controlling the production of a fluorescent protein. The feedforward networks are treated as modules, which makes the diagram more understandable visually. If the entire network were presented as a single piece, it might be difficult for a viewer to decipher the overall purpose and structure of the diagram. The ability to create conceptual diagrams automatically enables TinkerCell to support modules such as the ones shown in Figure 10. Figure 11 shows another example where one oscillator is used as an input for another oscillator. This conceptual design is useful when an engineer is interested in connecting two oscillators and is not interested in how each oscillator is implemented. See also Figure 12.

CONCLUSION

Synthetic biology is a highly interdisciplinary field, requiring individuals from different disciplines to come together to solve similar challenges. One role of software is to assist in this integration process. Different disciplines have different ways of representing a biological system, but the fundamental biological concepts in each of those representations are the same. Hence, we realized that a software methodology that can integrate different representations should try to capture the concepts. While there may be different ways to represent biological concepts, we have demonstrated one successfully implemented approach. The approach relies on hierarchical modeling and ontologies. The use of ontologies is needed in order to map between concepts and models. The use of hierarchical modeling needed to map from generic diagrams to details within different regions of that diagram. Additional requirements of the method presented in this article are local parameters and parent/child

dx.doi.org/10.1021/sb300033q | *ACS Synth. Biol.* 2012, 1, 353–364

relationships between components in a diagram. The use of local parameters is needed, once again, for mapping between different mathematical models.

While our demonstration software application, TinkerCell, is primarily a mathematical modeling tool, the method presented in this article can be used to bridge mathematical models with experimental data. The first steps toward integrating experimental data are shown in Figure 13. The figure describes the process of taking a cell with some dynamics and performing some type of experiment in order to obtain data. The experiment itself can be categorized using a detailed ontology, and similarly the type of data can also be categorized. In this manner, it would be possible for a software tool such as TinkerCell to store the necessary details for generating mathematical models as well as connecting variables and parameters in those models to experimental data. Further, it would be possible to connect the same model to multiple experiments (and vice versa) because the conceptual diagram can act as the orchestrator.

AUTHOR INFORMATION

Corresponding Author

*E-mail: deepakc@uw.edu.

Notes

The authors declare no competing financial interest.

Relationship to Prior Publications on TinkerCell. The previous publications on TinkerCell[17,18] did not describe hierarchical modeling because TinkerCell was quite different at the time of those publications. The concepts discussed in this article have not been published before.

REFERENCES

(1) Copeland, W., Bartley, B., Chandran, D., Galdzicki, M., Kim, K., Sleight, S., Maranas, C., and Sauro, H. (2012) Computational tools for metabolic engineering. *Metabolic Engineering 14*, 270−280.

(2) Cagatay, T., Turcotte, M., Elowitz, M. B., Garcia-Ojalvo, J., and Süel, G. M. (2009) Architecture-dependent noise discriminates functionally analogous differentiation circuits. *Cell 139*, 512−22.

(3) Anderson, J. C., Voigt, C. A., and Arkin, A. P. (2007) Environmental signal integration by a modular AND gate. *Mol. Syst. Biol. 3*, 133 ; DOI: 10.1038/msb4100173.

(4) Salis, H. M., Mirsky, E. A., and Voigt, C. A. (2009) Automated design of synthetic ribosome binding sites to control protein expression. *Nat. Biotechnol. 27*, 946−50.

(5) Sleight, S. C., Bartley, B. A., Lieviant, J. A., and Sauro, H. M. (2010) Designing and engineering evolutionary robust genetic circuits. *J. Biol. Eng. 4*, 1−20.

(6) Galdzicki, M.; Wipat, A.; Villalobos, A.; Stan, G. B.; Smith, T.; Sauro, H.; Roehner, N.; Pocock, M.; Plahar, H.; Peccoud, J.; Myers, C.; et al. *Synthetic Biology Open Language (SBOL)*, Version 1.0. 0., 2011.

(7) Hucka, M.; Finney, A.; Sauro, H. M.; Bolouri, H.; Doyle, J. C.; Kitano, H.; Arkin, A. P.; Bornstein, B. J.; Bray, D.; Cornish-Bowden, A.; Others, The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* 2003, *19*, 524−531

(8) Bilofsky, H. S., and Christian, B. (1988) The GenBank genetic sequence data bank. *Nucleic Acids Res. 16*, 1861−1863.

(9) Stricker, J., Cookson, S., Bennett, M. R., Mather, W. H., Tsimring, L. S., and Hasty, J. (2008) A fast, robust and tunable synthetic gene oscillator. *Nature 456*, 516−519.

(10) Smith, L. P., Bergmann, F. T., Chandran, D., and Sauro, H. M. (2009) Antimony: A modular model definition language. *Bioinformatics 25*, 2452−2454.

(11) Mirschel, S., Steinmetz, K., Rempel, M., Ginkel, M., and Gilles, E. D. (2009) PROMOT: modular modeling for systems biology. *Bioinformatics 25*, 687.

(12) Ashburner, M., et al. (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat. Genetics 25*, 25−9.

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

(14) Demir, E., et al. (2010) The BioPAX community standard for pathway data sharing. *Nat. Biotechnol. 28*, 935−42.

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

(16) Mendes, P., Hoops, S., Sahle, S., Gauges, R., Dada, J., and Kummer, U. (2009) Computational modeling of biochemical networks using COPASI. *Methods Mol. Biol. 500*, 17−59.

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

(18) Chandran, D., Bergmann, F. T., and Sauro, H. M. (2010) Computer-aided design of biological circuits using TinkerCell. *Bioeng. Bugs 1*, 274−281.

(19) Chandran, D., Copeland, W. B., Sleight, S. C., and Sauro, H. M. (2008) Mathematical modeling and synthetic biology. *Drug Discovery Today: Dis. Models 5*, 299−309.