Standards, Platforms, and Applications

Stanley Gu * Herbert Sauro†

February 11, 2013

*stanleyg@uw.edu
†hsauro@uw.edu

Contents

1 Summary

With the sequencing of the human genome, it has become apparent that systems biology, the understanding of cellular networks through dynamical analysis is becoming an important part of research for mainstream biologists. One of the indicative trends to emerge in recent years is the development of model interchange standards that permit biologists to easily exchange dynamical models between different software tools. This chapter describes the current and rising standards in systems biology that facilitate knowledge management and physiological model exchange. In addition, software platforms that implement these standards and enables the reuse of software code is discussed. Finally, the range of possible computational applications is described, highlighting the most commonly used and emerging tools in the field.

2 Introduction

Although computational systems biology may seem to be a recent field of endeavor, its origins can be traced as far back as the 1920s and 30s [100]. During this period it was already believed by some that genes were responsible in some way for specifying enzymes. It was also around this time that glycolysis, the first metabolic pathway, was being elucidated and the beginnings of the idea that enzymes formed linked sequences called pathways. It is even more remarkable therefore that given the infancy of these concepts, Sewell Wright should attempt to give a physiological explanation for the occurrence of genetic dominance and recessivity [101]. Wright argued that the explanation for the origin of dominance lay with the properties of catalytic networks, and laid out an initial mathematical theory which described the properties of enzyme networks[1]. In the 1940s, as the first digital computers were being built, pioneering individuals such as Garfinkel, Higgins and Chance began investigating the possibility of modeling the subtle behavior of biochemical pathways. Even before the advent of the digital computer, the same group had been using analog computers to model simple biochemical pathways for almost 15 years [30, 37, 14].

Since the work of the pioneers in the 1950s, there have been many small groups that have continued this line of inquiry and that together laid the foundation for many of the techniques and theory that we use today and take for granted, in contemporary systems biology. It should be noted that there is a large body of literature, particularly in the Journal of Theoretical Biology, dating back fifty years that many newcomers to the field will find useful to consider.

[1] This early work later became significant during the development of metabolic control analysis [45]

2.1 What is Systems Biology?

There are many conflicting opinions today on what exactly systems biology is. Historically the answer seems clear. The chief aim of systems biology is to understand how individual proteins, metabolites and genes contribute quantitatively to the phenotypic response. Lee Hood, president of the Institute of Systems Biology in Seattle, US, defines it similarly as "the identification of the elements in a system and the analysis of their interrelationships to explain the emergent properties of the system". Even so, some believe systems biology to be concerned with the collection of high-throughput data while others consider the elucidation of protein-protein networks and gene networks to be its hallmark. Certainly, both are vital prerequisites for understanding systems but neither alone can offer great *insight* into how networks operate dynamically.

Systems biology is the natural progression of classical molecular biology from a descriptive to a quantitative science and is concerned with the dynamic response of biological networks.

2.2 Statement Of Problem

Building models is not an entirely new approach to biology. If one examines any text book on molecular biology or biochemistry, virtually every page has a diagram of a model. These models, which are often termed cartoon based models, represent the culmination of years of painstaking research; they serve as repositories of accepted doctrine and the starting point for the generation of new hypotheses. There are, however, limits to what can be done with these models, their predictive value tends to be poor, and the ability to reason using qualitative models is limited. In other sciences these limitations are avoided through the use of quantitative models, models which are described not just pictorially but also mathematically. Quantitative models by their nature have much better predictive value compared to qualitative models, but their real usefulness stems from the capacity to carry out precise reasoning with them.

2.3 Quantitative Approaches

There is a wide range of mathematical representations that one can use to build quantitative models, the choice of approach depending on the type of biological question, the accessibility of experimental data and the tractability of the mathematics. Probably the most successful and widely used kind of model are those based on differential equations (both ordinary and partial). These models assume a continuum of concentrations and rates. In reality of course, cellular systems are discrete at the molecular level, however, since the numbers of molecules is very large, the continuum approximation turns out to be very good. When the number of molecules drops to below a certain threshold

the continuum model can break down and in these cases one must revert to stochastic simulation. The disadvantage of a stochastic simulation is that all the analytical methods available for continuous models no longer apply. One should therefore only use stochastic simulation if it is necessary and not in cases where an ODE based model adequately describes the data. This problem highlights the need to develop a new set of mathematical approaches in order to understand the dynamics of stochastic systems. There are other approaches, which include boolean, Bayesian, formal logic and connectivity studies but these have yet to show any overwhelming advantage over continuum based models.

This chapter will be primarily concerned with models based on differential equations and to a lesser extent stochastic equations.

List of Modeling Representations

Boolean: One of the simplest possible modeling techniques is to represent a network using Boolean logic [21]. This approach has been used to model gene networks.

Ordinary differential equations (ODEs): This is the most common and arguably most useful representation. Although based on a continuum model, ODE models have proved to be excellent descriptions of many biological systems. Another advantage to using ODEs is the wide range of analytical and numerical methods that are available. The analytical methods in particular provide a means to gain a deeper insight into the workings of the model.

Deterministic hybrid: A deterministic hybrid model is one which combines a continuous model (*e.g* ODE model) with discrete events. These models are notoriously difficult to solve efficiently and require carefully crafted numerical solvers. The events can occur either in the state variables or parameters and can be time dependent or independent. A simple example involves the division of a cell into two daughter cells. This event can be treated as a discrete event which occurs when the volume of the cell reaches some preset value at which point the volume halves.

Differential-algebraic equations (DAEs): Sometimes a model requires constraints on the variables during the solution of the ODEs. Such a situation is often termed a DAE system. The simplest constraints are mass conservation constraints, however these are linear and can be handled efficiently and easily using simple assignment equations (see equation 2). DAE solvers need only be used when the constraints are nonlinear.

Partial differential equations (PDEs): Whereas simple ODEs model well stirred reactors, PDEs can be used model heterogeneous spatial models.

Stochastic: At the molecular level concentrations are discrete, but as long as the concentrations levels are sufficiently high, the continuous model is perfectly adequate. When concentrations fall below approximately one hundred molecules in the volume considered (*e.g.* the cell or compartment) one has to consider using stochastic modeling. The great disadvantage in this approach is that one looses almost all the analytical methods that are available for continuous models, as a result stochastic models are much more difficult to interpret.

2.3.1 Quantitative Models Based on Differential Equations

It is probably fair to say that most of the successful models to be found in the literature are based on ordinary differential equations. Many researchers will express these models using the following equation:

$$\frac{\boldsymbol{dS}}{\boldsymbol{dt}} = \boldsymbol{N}\boldsymbol{v}(\boldsymbol{S}(\boldsymbol{p}), \boldsymbol{p}) \tag{1}$$

Where $\boldsymbol{S}$ is the vector of molecular species concentrations, $\boldsymbol{N}$, the stoichiometry matrix; $\boldsymbol{v}$ the rate vector and $\boldsymbol{p}$ a vector of parameters which can influence the evolution of the system. Real cellular networks have an additional property that is particularly characteristic of biological networks, this is the presence of so-called moiety conserved cycles. Depending on the time-scale of a study, there will be molecular subgroups conserved during the evolution of a network, these are termed *conserved moieties* [78]. The total amount of a particular moiety in a network is time invariant and is determined solely by the initial conditions imposed on the system[2].

In metabolism, conserved cycles act as common conveyors of energy (ATP) or reducing power (NAD); in signaling pathways they occur as protein phosphorylation states while in genetic networks, they occur as bound and unbound protein states to DNA. These conserved cycles will often have a profound effect on the network behavior and it is important that they be properly considered in computational models.

From the full set of molecular species in a model, it is customary to divide the set into two groups, the dependent ($\boldsymbol{S_d}$) and independent set ($\boldsymbol{S_i}$). This division is dependent entirely on the number and kind of conserved cycles in the network. If there aren't any conserved cycles in a model then the dependent set is empty and the size of the independent set equals the number of molecular species in the model. For details on how to compute $\boldsymbol{S_d}$ and $\boldsymbol{S_i}$ the reader should

[2]There are rare cases when a conservation relationship arises out of a non-moiety cycle. This does not affect the mathematics but only the physical interpretation of the relationship. For example, A $\rightarrow$ B + C; B + C $\rightarrow$ D has the conservation, $B - C =$ constant.

consult [81] or refer to Box 1 in this chapter. In many cases it is vital that this separation into dependent and independent species be made. For simple time course simulations the separation is not so important, but for most other analyses it is critical and for stiff integration methods highly desirable. The reason is that many numerical methods, including the stiff integrators, employ a measure called the Jacobian matrix as part of the numerical calculation. If the separation is not carried out, the Jacobian becomes singular and thereby rendering most analyses (*e.g.* steady state location, bifurcation analysis, certain optimization methods and sensitivity methods etc.) numerically unstable if not impossible. Even when carrying out simple time course simulations, the separation is also useful because it enables the number of differential equations to be reduced in number and thereby improve computational efficiency.

Equation 1 is therefore better expressed as:

$$\begin{aligned} \boldsymbol{S_d} &= \boldsymbol{L_0 S_i} + \boldsymbol{T} \\ \frac{\boldsymbol{dS}_i}{\boldsymbol{dt}} &= \boldsymbol{N_R v}(\boldsymbol{S_i}(p), \boldsymbol{S_d}, \boldsymbol{p}) \end{aligned} \tag{2}$$

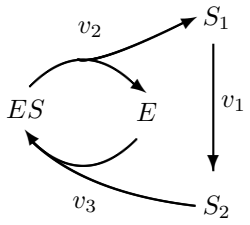
In these equations, $\boldsymbol{S_i}$ is the vector of independent species, $\boldsymbol{S_d}$, the vector of dependent species, $\boldsymbol{L_0}$ the link matrix, $\boldsymbol{T}$ the total mass vector, $\boldsymbol{N_R}$ the reduced stoichiometry matrix, $\boldsymbol{v}$ the rate vector and $\boldsymbol{p}$ the vector of parameters. Equation 2 constitutes the most general expression of an ODE based temporal model [38][36]. The symbolism used in equation 2 is the standard notation used by many in the systems biology community.

Although mathematically, reaction based models are given by equations 1 and 2, many researchers are more familiar with expressing models in the form of a reaction scheme. For example, the following describes part of glycolysis:

```
Glucose-6-P -> Fructose-6-Phosphate
Fructose-6-Phosphate + ATP -> Fructose-1-6-Bisphosphate + ADP
Fructose-1-6-Bisphosphate -> DHAP + GAP
```

For brevity, the rates laws that accompany each reaction have been left out. Such notation is well understood by biologists. It is not straight forward however to convert this representation to the representation give by equation 2. However, many software tools will permit users to enter models as a list of reactions and then automatically generate the mathematical model [80, 79, 82].

Box 1. Reaction Network Consider the simple reaction network shown on the left below:



$$\begin{array}{c} \\ ES \\ S_1 \\ S_2 \\ E \end{array} \begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{bmatrix} 0 & -1 & 1 \\ -1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} \end{array}$$

The **stoichiometry matrix** for this network is shown to the right. This network possesses two conserved cycles given by the constraints: $S_1 + S_2 + ES = T_1$ and $E + ES = T_2$. The set of independent species includes: $\{ES, S_1\}$ and the set of dependent species $\{E, S_2\}$.

The $\boldsymbol{L_0}$ matrix can be shown to be:

$$\boldsymbol{L_0} = \begin{bmatrix} -1 & -1 \\ -1 & 0 \end{bmatrix}$$

The complete set of equations for this model is therefore:

$$\begin{bmatrix} S_2 \\ E \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} ES \\ S_1 \end{bmatrix} + \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}$$

$$\begin{bmatrix} dES/dt \\ dS_1/dt \end{bmatrix} = \begin{bmatrix} 0 & -1 & 1 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Note that even though there appears to be four variables in this system, there are in fact only two independent variables, $\{ES, S_1\}$, and hence only two differential equations and two linear constraints.

3 Standards

A standard is defined as a uniform set of specifications applied toward an activity or product that encourages interoperation and cooperation. Ideally, a standard is clearly and unambiguously defined and while remaining easy to interpret and implement. In the modern world, standards have been applied to nearly everything from electronics cables and audio formats to paper sizes and telephone numbers. In the field of systems biology, the desire to facilitate interoperability and reuse of computational models and data was the motivation for

developing standardized digital annotations and representations.

3.1 Minimum Information (MI)

At the turn of the millennium, with the sequencing of the human genome and the rise of DNA microarray technologies, standards development for systems biology reached its first major milestone. Given increasingly large and complex experiments involving numerous biological samples and different experimental conditions, researchers within the DNA microarray community quickly realized that if one was to make sense of the results from any such analysis, a new way of storing and retrieving this complex information was needed. Scientists struggled to coordinate the outputs of different software platforms, identify the ancillary information needed to interpret results, and define the data necessary to enable reproduction of results. Through discussions between interested members of the community, public presentations, and workshop meetings, the Microarray Gene Expression Data Society (MGED) outlined the Minimum Information About a Microarray Experiment (MIAME) specification [11] and Microarray Gene Expression Markup Language (MAGE-ML) [90]. As we will discuss in the following sections, in many ways, this has become the prototype [74] for subsequent data annotation guidelines in systems biology.

The early success of MIAME and its widespread adoption led to the development of many domain-specific extensions and variations. MIAME is now accompanied by a myriad of "minimum information" reporting standards groups that cover practically every corner of the biomedical field [91]. The Minimum Reporting Guidelines for Biological and Biomedical Investigations (MIBBI) [93] project has arisen as a comprehensive source of these reporting "checklists". MIBBI maintains a web-based and freely accessible resource for minimum information standards (http://www.mibbi.org/), providing access to existing checklists, complementary data formats, controlled vocabularies, tools, and databases. This resource thereby enhances both transparency and accessibility of experimental results to the wider bioscience community.

3.1.1 Minimum Information Required in the Annotation of Models (MIRIAM)

Extending beyond the laboratory, the Minimum Information Requested In the Annotation of biochemical Models (MIRIAM) standard [68] [54] was developed for describing quantitative models of biochemical systems and bring together the new standards in computational systems biology, SBML and CellML (both are which discussed later in this chapter). By unifying different modeling subdomain standards under the same requisites, and thus ensuring that models are easily testable, reproducible, and comparable, the utility of quantitative modeling may be enhanced for the benefit of biomedical research. However, the ultimate impact of these standards depends on their adoption throughout the

community and the number of software tools that are developed to facilitate its use (more issues that we will discuss later in this chapter).

Traditionally, a challenge that MIRIAM faces is that encoded models in scientific publications or online are not in a standard format. And, of those that are encoded in a standard format, many actually fail compliance and validation tests developed for the standards. Failures may occur for a variety of reasons, ranging from minor syntactic errors to significant conceptual problems. Further semantic inaccuracies may lie within the model structure. With models that are not annotated, users are faced with ambiguous reaction specifications, such as species "A" and "B" producing "C".

Thus, to address these quality issues, MIRIAM comprises of the following guidelines:

Reference correspondence

- The model must be encoded in a public, standardized, machine-readable format (SBML, CellML, GENESIS, ...).
- The model must comply with the standard in which it is encoded.
- The model must be clearly related to a single reference description. If a model is composed from different parts, there should still be a description of the derived/combined model.
- The encoded model structure must reflect the biological processes described by the reference description.
- The model must be instantiable in a simulation: all quantitative attributes must be defined, including initial conditions.
- When instantiated, the model must be able to reproduce all results given in the reference description within an epsilon (algorithms, round-up errors).

Attribution annotation

- The model has to be named.
- A citation to the reference description must be provided (complete citation, unique identifier, unambiguous URL). The citation should identify the authors of the model.
- The name and contact information for model creators must be provided.
- The date and time of model creation and last modification should be specified. A history is useful but not required.

- The model should be linked to a precise statement about the terms of it's distribution. MIRIAM does not require "freedom of use" or "no cost".

External resource annotation

- The annotation must unambiguously relate a piece of knowledge to a model constituent.
- The referenced information should be described using a triplet {collection, identifier, qualifier}:
- The annotation should be written as a Uniform Resource Identifier (URI).
- The identifier should be considered within the context of the framework of the collection.
- Collection namespace and identifier are combined into a single URI, such as: http://identifiers.org/collection/identifier. For example: http://identifiers.org/uniprot/P62158.
- Qualifiers (optional) should refine the link between the model constituent and the piece of knowledge: "has a", "is version of", "is homolog to", etc.
- The community has to agree upon a set of standard valid URIs. A database and the associated API (Web Services) have been developed at the EBI to provide the generation and interpretation of URIs.

MIRIAM applies these guidelines to a wide range of quantitative models, which may use a variety of different mathematical representations, such as ODE, PDE, or DAEs. However, the ultimate test is to compare simulation results from a model representation with the reference description of the model.

3.1.2 Minimum Information About a Simulation Experiment (MIASE)

While the MIRIAM guidelines promote the inclusion of many crucial pieces of information within a computational model, it is vague about the advanced numerical algorithms and modeling workflows that are used in a modern computational setting. Without this information on the model context in which the original simulations were performed, reproducibility of the model results is still ambiguous. Thus, the Minimum Information About a Simulation Experiment (MIASE) guidelines [99] describe the minimal set of information that a model description must provide regarding the implementation of its simulation. This includes the list of models that were used, any modifications that were made,

simulation procedures that were applied, and how the raw numerical results were processed to produce the final output.

The MIASE guidelines are comprised of the following:

All models used in the experiment must be identified, accessible, and fully described.

- The description of the simulation experiment must be provided together with the models necessary for the experiment, or with a precise and unambiguous way of accessing those models.
- The models required for the simulations must be provided with all governing equations, parameter values and necessary conditions (initial state and/or boundary conditions). If a model is not encoded in a standard format, then the model code must be made available to the user.
- If a model is not encoded in an open format or code, its full description must be provided, sufficient to re-implement it.
- Any modification of a model (pre-processing) required before the execution of a step of the simulation experiment must be described.

A precise description of the simulation steps and other procedures used by the experiment must be provided.

- All simulation steps must be clearly described, including the simulation algorithms to be used, the models on which to apply each simulation, the order of the simulation steps, and the data processing to be done between the simulation steps.
- All information needed for the correct implementation of the necessary simulation steps must be included, through precise descriptions, or references to unambiguous information sources.
- If a simulation step is performed using a computer program for which source-code is not available, all information needed to reproduce the simulation, and not only repeat it, must be provided, including the algorithms used by the original software and any information necessary to implement them, such as the discretization and integration methods.
- If it is known that a simulation step will produce different results when performed in a different simulation environment or on a different computational platform, an explanation of how the model has to be run with the specified environment/platform in order to achieve the purpose of the experiment must be given.

All information necessary to obtain the desired numerical results must be provided.

- All post-processing steps applied on the raw numerical results of simulation steps in order to generate the final results have to be described in detail. That includes the identification of data to process, the order in which changes were applied, and also the nature of changes.
- If the expected insights depend on the relation between different results, such as a plot of one against another, the results to be compared have to be specified.

By providing the information specified by these guidelines, modelers can be reasonably assured that the simulation experiment corresponds with those of the original authors. Thus, as adoption of MIASE spreads, the quality of scientific reporting will increase, and collaborative efforts in computational modeling and simulation of biological systems is encouraged.

3.2 Ontologies

The value of any kind of data is greatly enhanced when it can be easily integrated and interpreted by other systems or third parties. Towards this goal, the MIRIAM and MIASE guidelines state that a model's constituents and simulation procedure must be unambiguously annotated. Thus, a common language is necessary for different models to describe the same physical entity or biological process. One approach is through the annotation of multiple bodies of data using common controlled and structured vocabularies or "ontologies".

For example, one successful biomedical ontology is the Gene Ontology (GO) [87]. GO defines specific gene products across different species. All terms are organized in a hierarchical structure, where there are three main branches: biological process, cellular component, or molecular functions. GO terms have been used in millions of annotations relating to gene products described in protein databases.

3.2.1 Open Biomedical Ontologies (OBO)

The success of the ontology approach has led to dizzying number of different ontologies, the sheer number which may create an obstacle to integration. OBO (`http://obofoundry.org`) [88] was created in 2001 to address this issue by serving as an umbrella body for the developers of life-science ontologies. The key principles behind OBO [88] ontologies are that they must be *open* and

orthogonal. Onotologies within OBO are *open* in the sense that its usage should be available without any constraints, and new applications may build upon OBO without restriction. Ontologies within OBO are *orthogonal* such that vocabulary is non-overlapping with other ontologies.

3.2.2 Model Annotations

One of the ways that modelers directly interact with ontologies is through the annotations within a model. This section will present an overview of the major components of the model and some of the most commonly used external resources and controlled vocabularies used for annotating them.

Model Metadata In the model metadata, the information for what the model is describing, a number of different ontologies may be used. In biological pathway models, Reactome (http://www.reactome.org/ReactomeGWT/entrypoint.html) [44] and KEGG Pathway (http://www.genome.jp/kegg/) [69] are comprehensive, human-curated, pathway databases that are often referenced. Information regarding the taxonomy of the biological pathway can be referenced in the UniProt Taxonomy database (http://www.uniprot.org/) [3].

Mathematics When describing the mathematics in a model, the Systems Biology Onotology (SBO) (http://www.ebi.ac.uk/sbo/main/) [54] is a recently developed vocabulary used for specifying the roles of biochemical species, parameters, kinetic laws, and other model components in relation to a systems biology model. For instance, SBO annotations can denote the substrate, products, and Michaelis-Menten constant in a model. GO and Reactome may also be used to describe what biological process a kinetic rate law is describing.

Physical Entities When it comes to describing the biophysical constituents, or species, in a model, several different ontologies can be referenced, sometimes in combination. GO and UniProt are often referenced for annotating proteins, and KEGG and ChEBI (http://www.ebi.ac.uk/chebi/) [20] may be used for annotating small molecules and chemical compounds that are related to biological processes.

3.2.3 Simulation Experiment Annotations

Ontologies for describing simulation procedure and numerical results are relatively newer than the previously described model annotations, and is currently an active field of work and new changes. The Kinetic Simulation of Algorithm Ontology (KiSAO) (http://biomodels.net/kisao/) is currently being

developed to describe the precise numerical steps and procedures taken in a simulation experiment. When looking at the numerical output of a simulation experiment, the Terminology for the Description of Dynamics (TEDDY) (http://www.ebi.ac.uk/compneur-srv/teddy/) [18] is being designed to describe the observed dynamical behavior in a simulation.

3.3 Physiological Models

In these following sections, the most popular and influential standards for encoding and exchanging physiological models will be discussed. The relative merits of each format will be surveyed and compared. But first, why are model standards useful?

Over the years, there has been an ever increasing list of wide ranging cellular models published in the literature. For most of scientific publishing history, each author has a particular notation that they use to publish the model. Some authors will publish the model as a reaction scheme (see the example in Section 2.3.1), much like the notation given in scheme. Others will itemize the actual mathematical representation in the form of a list of differential equations. Some authors do not publish the model at all but provide the model as supplementary information. Until recently, there has been no way to publish models in a standard format. Without a standard format it has proved very difficult if not impossible in many cases to implement and use published models without considerable effort.

Thus, as a result of this serious issue, a number of groups set out to gather community support to develop a standard that model developers would be happy to use. There was an early effort in 1998 by the BTK (BioThermoKinetics) group to standardize on a practical format for exchanging models between Gepasi [62] and SCAMP [80], both tools were widely used at the time. Around the same time, bioengineers at the University of Auckland began investigating the role that Extensible Markup Language (XML) [35] could play in defining a standard for exchanging computational models in order to reduce errors that appeared frequently in published models. From the Auckland team emerged CellML [58]. Members from the BTK group subsequently took their experience and contributed significantly to the other major model exchange standard, called SBML [40].

3.3.1 CellML

CellML [58] represents cellular models using a mathematical description similar to equation 2. CellML also has provisions for metadata annotations to allow MIRIAM compliance. In addition, CellML represents entities using a component based approach where relationships between components are represented by connections. In many ways CellML represents a literal translation of the

mathematical equations, except that the relationship between dependent and independent species is implied rather than explicit. The literal translation of the mathematics however goes much further, in fact the representation that CellML uses is very reminiscent of the way an engineer might wire up an analog computer to solve the equations (though without specifying the integrators). As a result CellML is very general and in principle could probably represent any system that has a mathematical description (and not just the kind indicated by equation 1). CellML is also very precise in that every item in a model is defined explicitly. However, the generality and explicit nature of CellML also results in increased complexity especially for software developers. Another side effect of the increased complexity is that models that are represented using CellML tend to be quite large. On average, a sample from the CellML repository (http://models.cellml.org/cellml) indicates that each reaction in a model requires about 5 kilobytes of storage.

Owing to the complexity of CellML, one unfortunate side effect is that there are substantially fewer tools which can read and write CellML compared to SBML. The CellML team (http://cellml.sourceforge.net/) also provides their own software tools to third-party developers, including the CellML API (http://cellml-api.sourceforge.net/), which is a library much like libSBML (discussed later in Section 3.3.2) that allows software developers to read and write CellML models.

3.3.2 Systems Biology Markup Language (SBML)

SBML was developed in 2000 at Caltech, Pasadena, as a result of funding received from the Japanese ERATO program. Both CellML and SBML are today viewed as the main standards for exchanging cellular network models. There are however fundamental differences between the approaches that CellML and SBML take in the way models are represented.

Whereas CellML attempts to be highly comprehensive, SBML was designed to meet the immediate needs of the modeling community and is therefore more focused on a particular problem set. One result of this is that the standard is much simpler and much less verbose. Like CellML, SBML is based on XML, however unlike CellML, it takes a different approach to representing cellular models. The way SBML represents models closely maps the way existing modeling packages represent models. Whereas CellML represents models as a mathematical wiring diagram, SBML represent models as a list of chemical transformations. Since every process in a biological cells can ultimately be broken down into one or more chemical transformations this was the natural representation to use. However SBML does not have generalized elements such as components and connections, SBML employs specific elements to represent spatial compartments, molecular species and chemical transformations. In addition to these, SBML also has provision for rules which can be used to represent constraints, derived values and general math which for one reason or another cannot be transformed into

a chemical scheme. Like CellML, the dependent and independent species are implied.

The development of SBML is stratified in order to organize architectural changes and versioning. Major editions of SBML are termed Levels and represent substantial changes to the composition and structure of the language. Models defined in lower Levels of SBML can always be represented in higher Levels, though some translation may be necessary. The converse (from higher Level to lower Level) is sometimes also possible, though not guaranteed. The Levels remain distinct; a valid SBML Level 1 document is not a valid SBML Level 2 document. Minor revisions of SBML are termed Versions and constitute changes within a level to correct, adjust, and refine language features. Finally, specification documents inevitably require minor editorial changes as its users discover errors and ambiguities. Such problems are corrected in new Releases of a given SBML specification.

Extensibility It was realized early on by the authors of SBML that as systems biology developed there would be pressure from the community to make additional functionality available in SBML. To address this issue, SBML has a formal means for adding extensions in the form of annotations. There now exist a number of annotations that are used by software developers. Some of these address issues such as providing visualization information to allow software tools to render the model in some meaningful way (two examples of these will be given in a later section). Other extensions provide a means to store information necessary for flux balance analysis or to provide information for stochastic simulations. Ultimately some of the extensions will most likely be folded into the official SBML standard. This mechanism, a sort of Darwinian evolution, permits the most important and popular requests to be made part of SBML. It makes the process of SBML evolution more transparent and permits users to be more involved in the development of SBML.

The current generation of SBML, Level 3 (http://sbml.org/Documents/Specifications#SBML_Level_3_Packages) [41], is modular in the sense of having a defined core set of features and optional packages adding features on top of the core. This modular approach means that models can declare which feature-sets they use, and likewise, software tools can declare which packages they support. It also means that the development of SBML Level 3 can proceed in a concurrent manner, where each module is developed relatively independently. SBML Level 3 package development is today an ongoing activity, with packages being created to extend SBML in many areas that its core functionality does not directly support. Examples include models whose species have structure and/or state variables, models with spatially non-homogeneous compartments and spatially dependent processes, and models in which species and processes refer to qualitative entities and processes rather than quantitative ones.

SBML Development Tools Early on in the development of SBML, the original authors decided to provide software tools almost immediately for the community. Since XML at the time was not well understood by many software developers the provision of such assistance was crucial. In hindsight, this is probably one reason why SBML has become a popular standard. Initially the original authors provided a simple library for the Windows platform since the bulk of biology based users tend to be Windows users. Today this library is still used by a number of tools including Gepasi, Jarnac and JDesigner (discussed later). With the growing popularity of SBML, the community has since developed a comprehensive cross platform tool libSBML (http://sbml.sourceforge.org) which is now the recommended SBML toolkit to use. LibSBML was developed in C/C++, with bindings to a number of different languages, for maximum portability.

Practical Considerations Whereas CellML is very general, SBML is more specific, as result, the storage requirement for SBML is much less. It takes on average roughly 1.5 kilobytes to store a single chemical transformation in SBML Level 2 (compared to 5 kilobytes for CellML). Interestingly it only takes roughly 50 to 100 bytes to store single transformations in raw binary format where there is minimal extraneous syntax. Some readers may feel that with todays cheap storage technologies, that discussions on storage requirements is unnecessary. Indeed for small models it is not an issue. However, in future very large models are likely to be developed. There is, for example a serious attempt (http://www.physiome.org) now underway to model in the long term entire organs and even whole organisms. The amount of information in these cases is huge and the question of efficient storage is not so trivial. Obviously XML is highly compressible and large models can be stored in this way. However, inefficient storage also increases the time taken to manipulate the models. Furthermore, in a modeling environment, model authors tend to generate hundreds of variants while developing the model. For a large model this clearly would generate huge amounts of XML based data. One of the things that has yet to be addressed by either standard is the how model variants can be efficiently stored.

Usage Both SBML and CellML have been taken up by many software developers and implemented in their software. Over the past decade, SBML has become the *de facto* standard for systems biology models. As December 2012, 251 different software packages are officially listed on the SBML Software Guide (http://sbml.org/SBML_Software_Guide). In addition, SBML is the official model interchange format for the SBW project (http://sys-bio.org/), the international *E. coli* alliance, and the receptor tyrosine kinase consortium.

3.3.3 NeuroML

Paralleling efforts in SBML and CellML in molecular pathway and cell physiology modeling, NeuroML [33] provides a common data format for defining and exchanging descriptions of neuronal cell networks. Level 1 (MorphML), Level 2 (ChannelML), and Level 3 (NetworkML) describe neuronal systems to different levels of biological granularity.

A number of software of packages are written to work with NeuroML, NEURON [13], GENESIS [10], MOOSE [76], NEST [23], and PSICS [12]. These different environments were successfully able to reproduce the same model simulation (including a reconstruction of the 3D structure of a neural pathway) [32], using NeuroML as the exchange format.

There are also recent efforts to convert NeuroML into SBML [46], which may allow NeuroML models and modelers access to the vast library of SBML compliant software tools.

3.4 Simulation

The share and reuse of biological models are primary challenges in the field of computational biology. While the previous discussed model exchange formats address issues of reproducing the structural components of the model, there are still missing elements in the computational procedure to unambiguously generate or reproduce relevant simulation results. This section will cover several standards that implement the MIASE guidelines and enable the transmission and sharing of simulation experimental procedures and results

3.4.1 Simulation Experiment Description Markup Language (SED-ML)

SED-ML (`http://sed-ml.org/`) [49] is an XML format that enables the storage and exchange of part of the information required to implement the MIASE guidelines. It covers information about the simulation settings, including information about the models, changes on them, simulation settings applied to the models and output definitions. SED-ML is independent of the formats used to encode the models as long as they are expressed in XML, and it is independent of the software tools used to run the simulations. The community believes that providing detailed information about simulation recipes will highly improve the efficient use of existing models, encoded in widely accept formats of model structure, such as SBML and CellML.

3.4.2 Systems Biology Results Markup Language (SBRML)

SBRML (http://www.comp-sys-bio.org/SBRML) [19] is another standardization effort that essentially associates a model with one or more datasets. Each dataset is represented as a series of values associated with the model variables. SBRML and SED-ML are complementary. While the main purpose of SBRML is to encode the simulation results, or even experimental data and the context in which it was obtained, SED-ML is used as a more detailed description of the operations that generate simulation results. This means that a SED-ML document can be used to describe the specific operations that led to the data contained in an SBRML document

3.4.3 Numerical Markup Language (NuML)

NuML (http://code.google.com/p/numl/) aims to standardize the exchange of numerical results, and is planned to be used by SED-ML and SBRML. NuML is designed to support any type of numerical result through the powerful coupling of ontology terms with one or more result components. Ontology terms reference external resources that define the vocabulary and terms used to describe the results in the NuML file. The results of the NuML file contains two principle components, a description of the results and the results themselves. Further details can be found by consulting the NuML specification (http://numl.googlecode.com/svn/trunk/numl-spec-l1v1.pdf).

3.5 Visualization

Circuit diagrams and Unified Modeling Language diagrams are just two examples of standard visual languages that help accelerate work by promoting regularity, removing ambiguity and enabling software tool support for communication of complex information. Ironically, despite having one of the highest ratios of graphical to textual information, biology still lacks standard graphical notations. The recent deluge of biological knowledge makes addressing this deficit a pressing concern. For many users, the ability to visualize models and to build models using visual tools is an important feature. There are currently a number of visualization formats that are in common use. One of the earliest, most comprehensive, and freely available formats is the molecular interaction maps developed by Kohn [51] and more recently by Mirit Aladjem [50]. The Kohn format emerged from the need to represent complex signaling networks in a compact way. Unlike metabolic networks, signaling networks can be extremely complex with multiple protein states and interactions and therefore an alternative and more concise approach is desirable.

Kitano [48] took a traditional approach where different molecular entities (such as proteins, ions, transporters etc.) have particular pictorial representations.

The software tool CellDesigner [27], which will be discussed later in this chapter, implemented this proposed format.

3.5.1 Systems Biology Graphical Notation (SBGN)

SBGN (http://www.sbgn.org/) [57] has arisen in recent years as one of the most widely supported and comprehensive visual languages, developed by a community of biochemists, modelers and computer scientists. SBGN consists of three complementary languages: process diagram, entity relationship diagram and activity flow diagram. Together they enable scientists to represent networks of biochemical interactions in a standard, unambiguous way. The process diagram draws its inspiration from process-style notations, borrowing ideas from the work of Kitano. In contrast, the entity relationship diagram is based to a large extent on Kohn's notation. The SBGN activity flow diagram depicts only the cascade of activity, thus making the notation most similar to the representations often used in the current literature to describe signaling pathways.

SBGN-ML While SBGN defines how biological information should be visualized, it does not specify how the mapping should be stored electronically. SBGN-ML (http://www.sbgn.org/LibSBGN/Exchange_Format) [55] is a dedicated file format that can be used to store and transfer the information necessary for software to faithfully render the corresponding SBGN diagram. The software library libSBGN (http://www.sbgn.org/SBGN_Software/LibSBGN) complements the file format. It consists of two parallel implementations [43] in Java and C++, which can be easily ported to different programming languages.

The plugin cySBGN (http://www.ebi.ac.uk/saezrodriguez/cysbgn/) [34], through use of libSBGN and SBGN-ML, allows SBGN diagrams to be imported, modified, and analyzed within Cytoscape (http://www.cytoscape.org/), a popular network visualizer. Coupled with the cySBML (http://sourceforge.net/projects/cysbml/) [52] plugin, which allows SBML models to be imported into Cytoscape, SBGN maps can be generated from SBML models directly.

3.6 Other Standards

Apart from the mentioned model interchange formats, there are many other mediums for representing models. This section will briefly cover several additional formats.

3.6.1 Human Readable Formats

In addition to visualization approaches and the use of XML to represent models, there has been a long tradition in the field to describe models using human

readable text based formats. Indeed the very first simulator BIOSSIM, [29], allowed a user to describe a model using a list of reaction schemes. Variants of this have been employed by a number of simulators since, including, SCAMP [80], Jarnac [79], E-Cell [94] and more recently PySCeS [71]. Being able to represent models in a human readable format offers many advantages, including, conciseness, easily understood and manipulated using a simple editor, flexible, portable and above all extremely easy to include commenting and annotation.

Jarnac Users of Jarnac also have access to an advanced graphical user interface (GUI) (Figure 1)

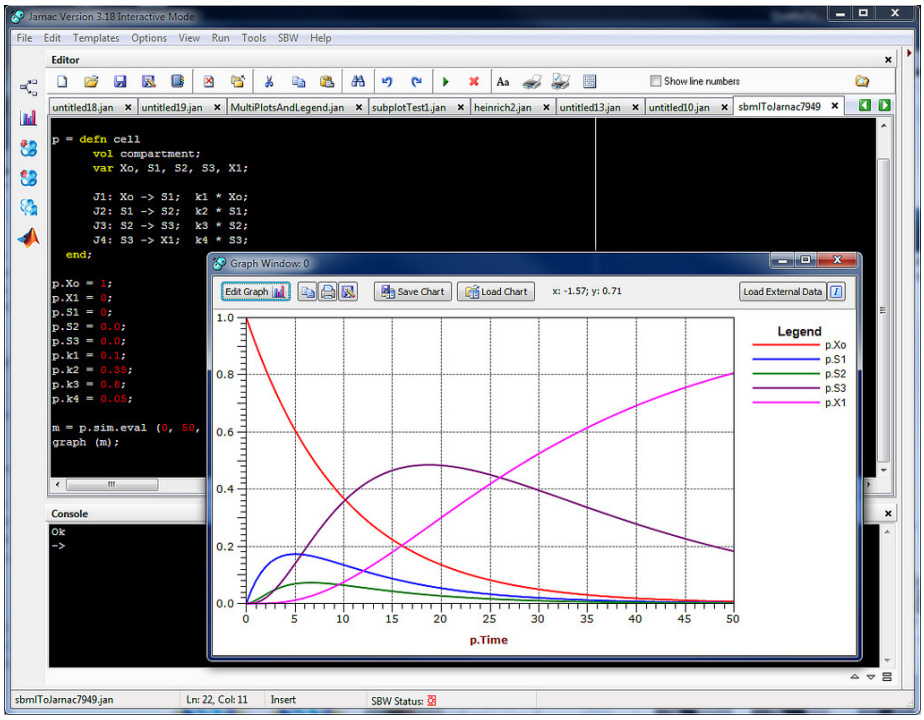


Figure 1: User interface for Jarnac

Mathematical Modeling Language (MML) MML is a text-based format that is the primary form of model representations in the JSim platform [77] (discussed in Section 4.1.2). Unlike SBML and CellML which are based on XML, MML uses its own a C-styled language for model declaration. MML models are often expressed generally in terms of mathematical equations, any mixture of ordinary and partial differential equations, implicit equations, integrations, discrete events, and even external programming code, such as Java, C, or MAT-

LAB. One feature that sets MML apart from other modeling languages is its awareness of physical units when run through JSim's MML compiler [15].

PySCeS PySCeS is a console based application written in the Python (http://www.python.org) programming language that runs on all major computing platforms [71]. Although users interact with PySCeS via Python scripts, many of the underlying numerical capabilities are provided by well established C/C++ or FORTRAN based numerical libraries through the SciPy package [70]. While it is possible to build models directly in Python using SciPy, PySCeS was written by team experienced with biochemical modeling to provide a high-level modeling interface that saves the modeler from needing to work with low level numerical algorithms.

SBML-Shorthand The SBML community has also developed a human readable script called SBML- shorthand [31]. This notation maps directly on to SBML but is much easier to hand write compared to SBML. The shorthand is also much less verbose and uses infix to represent expressions rather than MathML.

Antimony Antimony (http://antimony.sourceforge.net/main.html) [89] is a more recent scripting language that combines the relative simplicity of languages like Jarnac, PySCeS, and SBML-shorthand with the modularity of languages like little-b (http://www.littleb.org/), without forcing the modeler learn a general programming language. Antimony- formatted files may be read by other software packages, like PySCeS using the libAntimony library. Furthermore, Antimony to SBML converters extend Antimony's usefulness by allowing users to convert their modules into a form usable with the vast number of SBML-compliant software.

3.6.2 Biological Pathway Exchange (BioPAX)

BioPAX (http://www.biopax.org/) [22] [92] is another proposed standard based on XML. BioPAX aims to integrate many of the incompatible pathway related databases (such as BioCYC, BIND, WIT, aMAZE, KEGG and others) so that data from any one of these databases can be easily interchanged. In future it should be possible to extract data from many of the pathway databases and integrate the data directly into SBML (or CellML) via BioPAX. The BioPAX group proposes to embed BioPAX elements onto SBML or cellML for unambiguous identification of substances (metabolites, enzymes) and reactions. However, it is possible for to convert, or map, from SBML to BioPAX (http://www.ebi.ac.uk/compneur-srv/sbml/converters/SBMLtoBioPax.html).

3.7 Model Databases

High throughput experimental techniques have led to the population of web-accessible databases with vast amounts of biological data. Mathematical models of biological systems are playing an essential role in the interpretation of this data. The scientific community now faces the challenge of the mathematical models themselves becoming increasingly complex and numerous. There is a need for centralized databases to store all these models in standard formats to make them easily accessible and reusable by the research community. Publishing the models in a standard format, concurrent with the submission of a written paper, will eliminate many of the errors introduced into the model during the publication process.

3.7.1 BioModels

BioModels Database (http://www.ebi.ac.uk/biomodels/) [56] is one largest open- access databases in systems biology. Part of the international initiative BioModels.net, BioModels provides access to peer-reviewed and published models. Each model is manually curated by the database maintainers to verify that it corresponds to the reference publication and gives the expected numerical results. Curators also annotate the components of the models with terms from controlled vocabularies (Taxonomy, Gene Ontology, ChEBI, etc.) and links to other databases (UniProt, KEGG, Reactome, etc.). This annotation is a crucial feature of BioModels Database in that it permits the unambiguous identification of molecular species or reactions and enables effective search algorithms in finding model and model components of interest. As of December 2012, the database contains 142,973 models, comprising of 923 models published in literature, of which roughly half are manually curated by BioModels, and 142,050 models automatically generated from the Path2Models project (http://code.google.com/p/path2models/), an effort aimed at automatically converting biological pathway databases (such as KEGG) into corresponding SBML models

3.7.2 CellML Repository

The CellML Model Repository (http://www.cellml.org/models) [59] [6] is a similar effort, which contains hundreds of biochemical pathway models that have been described in peer-reviewed publications. CellML and the CellML Model Repository are part of the IUPS Physiome Project effort to create a virtual physiological human [42]. The explicit representation of modularity, together with the flexible nature of the CellML language which allows the description of a diverse range of cellular and sub-cellular systems.

The CellML Model Repository contains over 330 freely available, quantitative models of biological processes taken from the peer-reviewed literature. In con-

trast with other databases, such as BioModels, which focus on specific areas such as systems biology pathway models or computational neuroscience, the CellML Model Repository contains models describing a wide range of biological processes, including: signal transduction pathways, metabolic pathways, electrophysiology, immunology, the cell cycle, muscle contraction and mechanical models and constitutive laws. This wide scope exemplifies CellML's ability to describe much of the biochemistry, electrophysiology and mechanics of the intracellular environment. Lumped parameter models dealing with systems physiology (e.g. blood pressure control, fluid retention, electrolyte balance, endocrine function, etc.) are also within the scope of CellML.

3.7.3 Physiome Repository

The Physiome Model Repository (http://www.cellml.org/tools/pmr) [102] is an offshoot of the CellML repository, which is unique in that it allows users to make their own copies of CellML and keep track of model changes using a distributed version control system, Mercurial [73]. One of the primary goals of this platform is to facilitate collaboration between several researchers, a common occurence during model development.

3.7.4 JSim Repository

The JSim (discussed in Section 4.1.2) group provides a repository of 370 freely-accessible MML models (http://www.physiome.org/jsim/db/). These may be either downloaded to the desktop as JSim project files, or directly simulated in the web browser using the JSim Java applet.

3.7.5 JWS Online

JWS Online (http://jjj.biochem.sun.ac.za/) [72] is a repository of kinetic models, describing biological systems, which can be interactively run and interrogated over the Internet. As of December 2012, JWS Online contains 131 models, downloadable to SBML, while also providing a web browser interface to a simulation server.

3.8 Future Considerations

With the success of Minimum Information guidelines and standardized representations of biological models, quantitative modeling has surged in popularity. However, the ever-growing number of published dynamic models published also presents a significant challenge in terms of model reuse and integration. While there is currently no agreed upon way to merge smaller sub- models into larger models, MIRIAM and MIASE annotations enable model composition software

to make use of the semantic information and enable algorithms parse through models, or parts of models, of interest to the user. Recent efforts in this arena include semanticSBML [53], SemSim [67], and the SBML Reaction Finder [66].

4 Platforms

This section will focus on the different modeling and simulation platforms that are available, which implement the systems biology standards highlighted in the previous section. While this section is certainly not an exhaustive list of all modeling platforms, highly influential or previously mentioned software packages will be discussed.

The first systems biology simulation package, BIOSIM, was written in the 1960s [29]. Especially in recent years, there has been a boom of software applications for the systems biology community. While, most software projects have ended development over the years, for a variety of reasons, such as lack of funding or maintainers, there are still far too many different modeling platforms to possibly be covered in this chapter. Thus, in this section will highlight specific software tools that have had a significant impact on the community, or some unique features that set them aside.

Furthermore, discussion will be focused specifically on modeling and simulation tools, and not more advanced analytical techniques, such as metabolic flux balance analysis. For more detailed discussion on flux balance software, please refer to a recent review [17] and a very comprehensive listing of SBML compatible software provided by the SBML consortium (http://sbml.org/SBML_Software_Guide/SBML_Software_Matrix).

4.1 Modeling

On the whole, many of these applications provide very similar functionality. The distinguishing feature among them is how easy they are to install and use. The more mature applications tend to be easier to install and have a much richer repertoire of functionality. Many of the applications are simple wrappers around standard ODE or Gillespie solvers and provide a simple means to load models and run time courses. Some of the applications fall by the wayside because the author has lost interest or funding has stopped. It is important therefore that what ever tool one uses, that the ability to export and import a recognized standard (or at least a documented format) such as SBML and/or CellML be available.

4.1.1 CellDesigner

CellDesigner (http://www.celldesigner.org/) [28] [26] is a structured diagram editor for drawing gene-regulatory and biochemical networks. Networks are drawn based on the process diagram, with graphical notation system that influenced the development of SBGN, and are stored using the Systems Biology Markup Language (SBML), a standard for representing models of biochemical and gene-regulatory networks. CellDesigner supports simulation and parameter scanning through a selection of different simulation engines, SBML ODE Solver, COPASI, or SBW.

4.1.2 JSim

JSim [77] is a Java-based simulation system for building quantitative numeric models and analyzing them with respect to experimental reference data. JSim was developed primarily for generating model solutions for use in designing experiments and analyzing data in physiological and biochemical studies, but its computational engine is general and equally applicable to solving equations in physics, chemistry, and mechanics. JSim has been under development at the National Simulation Resource for Mass Transport and Metabolism (NSR) since 1999. JSim uses a model specification language, MML (for Mathematical Modeling Language) which supports ordinary and partial differential equations, implicit equations, integrals, summations, discrete events, and allows calls to external procedures. JSim's compiler translates MML into Java code in which the numeric results are calculated. Within the JSim GUI users adjust parameter values, initiate model runs, plot data, and perform behavioral analysis, sensitivity analysis, parameter optimization for curve fitting. Alternatively one can use JSim's command line interfaces (jsbatch and jsfim). JSim source code, binaries (for Windows, Macintosh and Linux) and documentation are available free for non-commercial use at http://physiome.org/.

4.1.3 JDesigner

One of the first visualization tools, JDesigner [82] [79] implements a traditional way to depict networks (see Figure 2) using a pictorial representation to indicate substances and reactions. JDesigner also employs Bezier curves to represent arcs in an attempt to make the diagrams similar to the notation found in many molecular biology text books. JDesigner can use Jarnac's extensive simulation capabilities (both ODE and stochastic) via SBW.

4.1.4 PySCeS

While PySCeS has already been mentioned earlier as a human readable format for expressing biological models, the software package warrants mention again

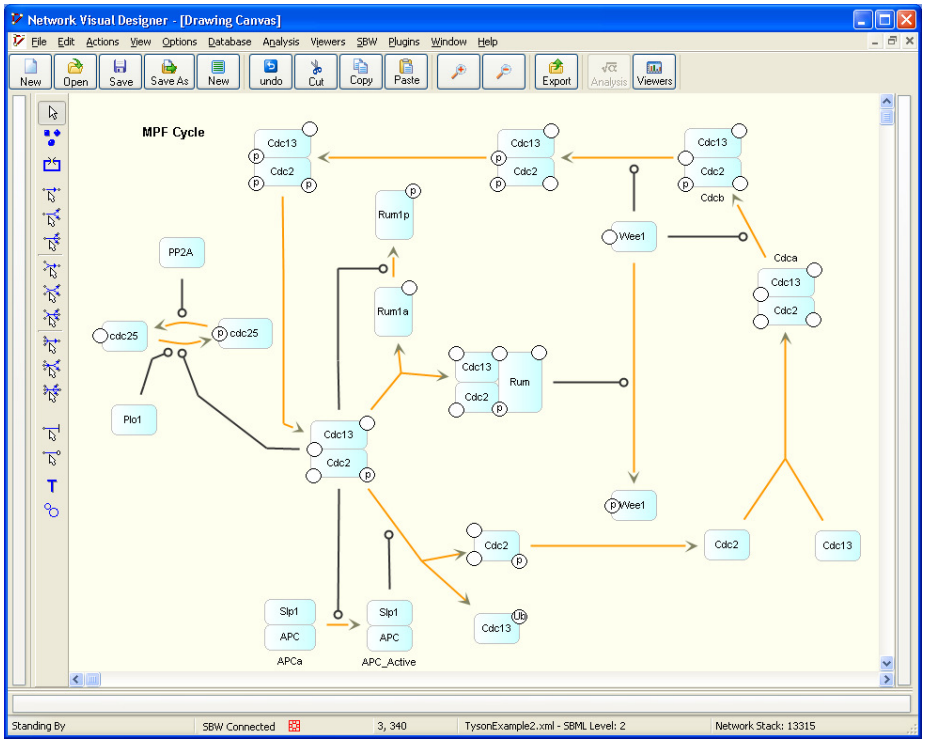


Figure 2: Example of JDesigner's visual format

as a full-featured modeling platform that can be used interactively and as a library. along with its scripted model description language, PySCeS is SBML compatible, and provides a full set of simulation tools, including stoichiometric, simulation, steady state, metabolic control, and Eigen analysis. Two and three-dimensional graphing is also made available through utilizing additional Python libraries.

4.1.5 Systems Biology Workbench (SBW)

SBW (http://www.sys-bio.org) [8] is an extensible software framework that is both platform and language independent. Its primary purpose is to encourage code reuse among members of the systems biology community. Developers can run SBW on Linux, Windows or Mac OS and can develop software in a variety of different languages including C/C++, Java, Delphi, FORTRAN, MATLAB, Perl, Python and any .NET language (*e.g.* Visual Basic or C#). The SBW was originally developed in parallel with SBML as part of the Symbiotic Systems Project ERATO project at Caltech, Pasadena.

The central component of SBW is the broker, which is responsible for coordinating interactions among the different resources connected to it. These resources include simulation engines, model editors, SBML translators, databases, visualization tools and a variety of analysis packages. All modules in SBW connect via defined interfaces, which allows any one of the modules to be easily replaced if necessary. The key concept in SBW is that any new module may exploit resources provided by other modules; this dramatically improves productivity by allowing developers to build on existing tools rather than continuously reinvent.

An SBW module (the client) provides one or more interfaces or services. Each service provides one or more methods. Modules register the services they provide with the SBW Broker. The module optionally places each service it provides into a category. By convention, a category is a group of services from one or more modules that have a common set of methods.

One of the key advantages of SBW is its language and OS neutrality. At a stroke this eliminates the irrational language and operating systems wars that often plague software development. In addition to providing support for multiple languages there is also the facility to automatically generate web services from any SBW module.

4.1.6 VCell

The Virtual Cell (VCell; http://vcell.org/) [60] [64] is a client/server based tool that specializes in three-dimensional whole cell simulations. It is unique in that it provides a framework for not only modeling biochemical networks but also electrophysiological, and transport phenomena while at the same time considering the subcellular localization of the molecules that take part in them.

This localization can take the form of a three-dimensional arbitrarily shaped cell, where the molecular species might be heterogeneously distributed. In addition, the geometry of the cell, including the locations and shapes of subcellular organelles, can be imported directly from microscope images. VCell is written in Java but has numerical analysis carried out by C/C++ and FORTRAN coded software to improve performance. Currently, modeling must be carried out using the client/server model which necessitates a connection to the internet. In addition models are generally stored on the VCell remote server rather than the clients desktop. This operating model is not always agreeable to users and as a result the VCell team are reorganizing the software so that it can also be run as a stand-alone application on a researchers machine. The VCell team has incorporated the BioNetGen [9] network generator which allows models to be specified in a rule based manner. VCell is also one of the few tools that can both import and export SBML and CellML. This feature could in principle be used to translate between SBML and CellML models.

4.1.7 Other Modeling Tools

Several other modeling tools will be mentioned in this section but with shorter descriptions. Some basic information on which platforms this software is available on and a URL to find more information will be provided.

BioNetGen [9] Tool for rule-based modeling of biochemical systems. In rule-based models, molecules and their complexes are represented using graphs, and their consequent interactions will cause rewiring of the graph based on rules. Originally, it was developed to study the problem of combinatorial complexity of signal transduction systems, such as antibody receptor binding.

- Open source; Linux, Mac OS, Microsoft Windows
- http://bionetgen.org/index.php/Main_Page

Gepasi [62] This is forms based application was one of the first simulation platforms written for the PC. The tool is particularly adept at carrying out optimizations of ODE based models to data.

- Closed source; Microsoft Windows
- http://www.gepasi.org/

iBioSim [65] Tool for the design and analysis of genetic circuits, with applications to both systems and synthetic biology. Models can be constructed manually or imported from various databases and then analyzed with a variety of ODE and stochastic simulators.

- Open source; Linux, Mac OS, Microsoft Windows
- (http://www.async.ece.utah.edu/iBioSim/)

JigCell [97] [98] A suite of computational tools with graphical user interfaces that includes model building, simulation, and parameter estimation. A unique feature of JigCell is its display of data and simulation experiments in a spreadsheet format and run in batch mode.

- Closed source; Linux, Mac OS, Microsoft Windows
- http://jigcell.cs.vt.edu/

ProMot [63] Short for "process modeling tool", this is a software package for simulating and manipulating models. Some key features are its support of modular models, modeling libraries, its own modeling language MDL, and graphical capabilities.

- Open source; Linux, Microsoft Windows
- http://www.mpi-magdeburg.mpg.de/projects/promot

SBSI [2] Provides fitting of model parameters to experimental data, especially on models with oscillatory components, as well as standard model editing and simulation capabilities.

- Open source; Linux, Mac OS, Microsoft Windows
- http://www.sbsi.ed.ac.uk/

WinSCAMP [**@SauroF9**] [83] A script based GUI application, which like Gepasi has a long tradition. Specialized for time course, steady state and metabolic control analysis of ODE based models.

- Source available upon request; Windows
- http://sbw.kgi.edu/software/winscamp.htm

4.2 Simulation Engines Libraries

Simulation and modeling is one of the standard approaches to understanding complex biochemical processes. Therefore, there is a growing need for software tools that allow access to diverse simulation and modeling methods as well as support for the usage of these methods. These software libraries should be compatible, *e.g.* via file standards, platform independent and user friendly to avoid time-consuming conversions and learning procedures. In addition, the software should be actively maintained and updated by its authors.

This section will cover some of the most widely used, open source, simulation libraries that many modeling platforms depend on for computation. These libraries all support the simulation of SBML models, and have been validated against an online suite of SBML test cases (http://sbml.org/Facilities/Online_SBML_Test_Suite) provided by the SBML consortium.

4.2.1 COPASI

COPASI [39] (http://www.copasi.org/tiki-view_articles.php) is the successor to Gepasi and comes in two versions: a graphical and a command line interface. The command line version is designed for batch jobs where a graphical user interface is unnecessary and where runs can be carried out without human supervision. COPASI uses its own file format to store models, however like all the tools discussed here, it can import and export SBML. One of its undoubted strengths is optimization and parameter fitting which it inherited from its predecessor. It has a unique ability to optimize on a great variety of different criteria including metrics such as eigenvalues, transient times etc. This makes COPASI extremely flexible for optimization problems. Installation is very simple and entails using a one-click installer. Although the source code to COPASI is available and can be freely used for research purposes in academia, owing to the way in which the development of COPASI was funded there are restrictions on commercial use.

The graphical user interface is based on a menu/dialog approach, much like its immediate predecessor, Gepasi. COPASI has capabilities to simulate deterministic as well as stochastic models and includes a wide range of analyzes. It correctly takes into account conservation laws and has very good support for metabolic control analysis amongst other things. COPASI is without doubt one of the better simulators available. Although the user interface is graphical, it does, due to its particular design, require some effort to master but with the availability of the COPASI the source code, there is the opportunity to provide alternative user interfaces. Finally there is a version that has an SBW interface (Systems Biology Workbench) which allows SBW enabled tools access to COPASI's functionality (currently available at http://sys-bio.org/).

SimpleCOPASI SimpleCOPASI (http://code.google.com/p/copasi-simple-api/) is a C interface to the C++ COPASI library. The core functionality of reading, writing, simulating, and numerical analysis of SBML models is retained from COPASI. In addition, Antimony scripts can be used to load models. The structural library (http://libstruct.sourceforge.net/) is also included within this library for analyzing the stoichiometric networks.

4.2.2 LibSBMLSim

LibSBMLSim (http://fun.bio.keio.ac.jp/software/libsbmlsim/) is a relatively newer simulation library and available on Unix and Windows based operating systems. It features a number of explicit and implicit ODE solvers and a relatively straightforward interface for producing simulation results from an SBML input file. However, compared to the other simulators in this chapter that have been around longer, the API contains fewer features. As of this writing, there are no functions for steady state analysis (useful for flux balance analysis) and single time step simulations (useful for real-time and interactive simulations).

LibSBMLSim is written in C which makes it "portable" in the sense that it is relatively easy to port to a wide variety of different programming languages and environments through the use of the SWIG software tool (http://swig.org/). Indeed, this feature of C to serve as a least common denominator of sorts is an attractive reason systems biology software developers to program in to expand the reach of their software to a user base that runs numerous different computing environments.

4.2.3 RoadRunner

RoadRunner is a powerful and portable simulation engine that is used in SBW, a resource sharing framework that allows applications to share functionality with each other [8]. Jarnac and Roadrunner are already included with an SBW installation. COPASI and SBML ODE Solver may also be used as simulation engines alongside RoadRunner within SBW [7].

Instead of interpreting model equations, RoadRunner compiles the model equations dynamically, which results in much improved performance when compared with traditional simulators. RoadRunner uses the integrator CVODE and NLEQ for steady state analysis [16]. To further speed up the simulation, the model is separated into a system of independent and dependent variables. This separation process is described in detail in [96]. Thus, all major operating systems are supported given that RoadRunner only depends on CVODE and NLEQ being available on the platform.

4.2.4 SBMLSimulator

SBMLsimulator (http://www.cogsys.cs.uni-tuebingen.de/software/SBMLsimulator/) [24] is a simulation library and accompanying GUI that is implemented in Java. In particular, the Java developer community benefits greatly from Java software to ease the ability to implement third party dependencies. Analogous to the way COPASI and RoadRunner are built from libSBML (C/C++), SBMLsimulator depends on JSBML, an SBML document manipulation library written entirely in Java (http://sbml.org/Software/JSBML).

4.3 MATLAB

MATLAB (http://www.mathworks.com/products/matlab/) is one of the most widely used numerical platforms in science and engineering. MATLAB contains excellent numerical and data analysis methods useful for systems biology. Many add-ons, referred to as "toolboxes" are available commercially or open-source to extend the functionality of MATLAB.

4.3.1 SimBiology

MathWorks offers a specialized toolbox called SimBiology (http://www.mathworks.com/products/simbiology/) which offers many useful capabilities. SimBiology provides graphical and programmatic tools to model, simulate, and analyze dynamic biological systems. SimBiology also includes a library of common pharmacokinetic/pharmacodynamic models. Users may use a graphical block diagram editor for building models, or directly import existing SBML models. Models within SimBiology can then use MATLAB's powerful scripting interface and extensive set of built-in ODE and stochastic solvers for simulation.

4.3.2 Systems Biology Toolbox (SBToolBox2) and PottersWheel

SBToolBox2 (http://www.sbtoolbox2.org/main.php) [86] is a very extensive, open- source, MATLAB tool box developed by Henning Schmidt. The tool box has a wide range of capabilities. In addition, PottersWheel [61], is a very comprehensive parameter fitting tool that works well with the SBToolBox2 but can also be used alone. In a number of cases it is better than COPASI's capabilities particularly in the area of generating nonlinear confidence limits on parameter fits and analyzing the resulting fit. The experimental data input formats are also very flexible. The tool provides a number of optimization algorithms including genetic and simulated annealing approaches.

4.3.3 SBMLToolbox

SBMLToolbox (http://sbml.org/Software/SBMLToolbox) [47] is an open source MATLAB toolbox developed by the SBML Team. SBMLToolbox ports functionality from libSBML into MATLAB, by creating MATLAB structures that mirror the functionality of libSBML. SBMLToolbox is also compatible with Octave (http://www.gnu.org/software/octave/), a free and open source computing environment that is similar to MATLAB.

4.3.4 SBML to MATLAB Translation

It is also possible to export SBML models into MATLAB scripts without the need for any additional toolboxes. SBML2MATLAB (http://sysbio-online.org/sbml2matlab/) is a cross-platform tool for performing such conversions. The SBML model structures and mathematics are mapped to MATLAB functions and structures, allowing users to easily manipulate the models through additional MATLAB scripting. SBML2MATLAB has also been integrated as a standalone web application that provides a user friendly interface for using SBML2MATLAB without any need for installing software.

In addition, a web application for viewing, editing, and simulating SBML models is also actively being developed (http://sysbio-online.org) which would allow modelers work on any platform that supports a web browser, circumvent the need to install any software, and not be limited by the local computer hardware power by performing computationally intensive calculations on a remote server.

5 Applications

This section will provide an overview for some of the different computational techniques and applications that are used with quantitative models in systems biology.

5.1 Model Analysis

As a user, one of the most important aspects that is considered is the range of techniques that are available for analyzing a model. The purpose of building a model is not simply to generate a predictive tool, if that was solely the case, than one could probably use empirical statistical techniques or machine learning approaches rather than the mechanistic models discussed in this chapter. An additional important role of model building is to also gain a deeper understanding into the properties of the model and to how the structure of the model leads

it to behave the way it does. In order to answer these kinds of questions one needs techniques that can interrogate the model in a variety of different ways.

The list below outlines some of the most important techniques that are available for analyzing models. Without these techniques, a model will often be as difficult to understand as the real system it attempts to model; the application of these techniques is therefore important.

Approach and Description

Connectionist Theory: Connectivity studies are centered around the search for patterns in the way cellular networks are physically connected. [5]

Structural Analysis: There are a wide range of useful techniques which focus on the properties of the networks that depend on the mass conservation properties of networks. These include, conservation analysis, flux balance and elementary mode analysis [36].

Cellular Control Analysis: CCA (also known as metabolic control analysis) is a powerful technique for analyzing the propagation of perturbations through a network. There exists a very large literature describing applications and theory [25].

Frequency Analysis: Closely related to CCA is the analysis of how signals propagate through a network [84, 75].

Bifurcation Analysis: Bifurcation analysis is concerned with the study of how the qualitative behavior of steady state solutions change with fluctuations in the model parameters [95].

All these techniques are extremely useful in gaining insight into how a model operates. The connectionist and structural analyses focus on the network properties of the model. They do not explicitly consider the dynamics of the model but rather how the network connectivity sets the stage for generating the dynamics of the model. The last three techniques, CCA, frequency analysis and bifurcation analysis focus on the dynamical aspects of a model and are crucial to gaining a deep insight into the model [4, 95].

5.2 Model Fitting and Validation

An important activity in systems biology modeling is the need to fit experimental data to models. While the scope of this chapter does not permit covering this topic to any great detail, as time series data from microarray, proteomic, and metabolomic data becomes more readily available, the need to fit models to experimental data will become more acute. There are a number of issues related to this topic, one such concern is the nature of the data that is generated by many of the current experimental techniques. In particular, most current techniques generate normalized data, that is absolute values are not given. This poses a number of problems to a fitting algorithm, since the underlying model is in terms of absolute quantities. A number of solutions are potentially available, however none are entirely satisfactory and, ultimately, the models generated by normalized data will may only be capable of reproducing trends in the data. Whether such models will have great predictive value is open to question and much research remains to be done in this area.

Another issue is the intensive computations that are required to fit even a moderately sized model. One of the necessary requirements for fitting a model is estimating the confidence limits on the fitted parameters and the range of parameter space which describes the experimental data. This information is crucial to determine the validity of the model and can be used to design additional experiments to either refute the model or increase the precision of the model parameters. As a result of these requirements, computing a global optimization can take a considerable time. For example, in a recent study, Vijay Chickarmane[3] estimated that the time required to fit a model of approximately three hundred parameters would be of the order of seven years on a normal desktop computer. Luckily, global optimization can be easily parallelized given a suitable optimizer (for example a genetic algorithm based optimizer) and the computation time can be reduced by hosting the problem on a cluster machine. Chickarmane estimates that using a one thousand node cluster, the optimization of a three hundred parameter model can be reduced to approximately two days of computation time. Such a computation can be easily set up using SBW. A single node on the cluster would act as the primary optimizer; this node in turn would farm out the time consuming simulation computations to the remaining nodes on the computer. For very large models, Grid computing [1] may be very appropriate for solving this kind of problem.

6 Future Prospects and Conclusion

The systems biology field has been developing rapidly in recent years but much remains to be done. One of the most useful developments must undoubtedly go to the development of standards, such as SBML and CellML. Indeed one of

[3] Personal correspondence

the highest impact systems biology journals, Molecular Systems Biology, has stipulated that SBML is the preferred format for contributing models. Furthermore, as models are being increasingly better annotated with standardized vocabularies, an exciting horizon for the future of systems biology models is their improved modularity and ease of reuse.

The other area that has received a lot of attention in recent years is the development of tools for systems biology. To avoid the problem of stagnating software projects that may have a short lifetime, the development of reusable software is a potential solution. Examples of reusable software are extensible frameworks, such as SBW, and suites of open-source libraries that can carry out specific functionality. An example of this is libSBML which enables other developers to concentrate on unique features, such as graphical interface or simulation capability, rather than waste unnecessary effort developing their own SBML parser. In terms of other possible libraries, examples include, open-source Gillespie based stochastic solvers and ODE solvers. Further more, hybrid methods combining continuous and stochastic methods is a pressing need at the current time. Many biological systems interface noisy sensory apparatus (*e.g.* ligand binding to the surface of a cell membrane) to internal continuous analog networks [85]. In addition to the core solvers, there is also need for scalable analysis tools, particularly bifurcation and sensitivity analysis tools. On the model validation front, much remains to be done, particularly the relationship between model validation and how this can direct future experimentation. This leads on to the development of new methods and algorithms for analyzing the complex networks in particular methods should be developed to modularize large networks since understanding an entire network is virtually impossible without some recourse to a hierarchical modularization.

In conclusion, while there is certainly much to be done in this field, it is a very exciting time for systems biology and the role that standards and software platforms will play. The development of standards and their adoption in the community have accelerated drastically over the years. Furthermore, the tremendous efforts of many systems biology groups worldwide have made available an unprecedented number of resources and tools for building, sharing, and publishing models. We believe that more modular designs will accelerate the development of standards [41], and promote the re-usability of models, or parts of models [66]. Furthermore, recent advancements in cloud computing and web technologies could potentially lead to many exciting and novel approaches in addressing the current challenges in computational systems biology.

7 Recommended Resources

Three main web sources which are of interest to readers of this chapter include:

`http://www.cellml.org` **:** This is the main CellML site. It has a very rich set

of models expressed in CellML including specifications for the standard and pointers to software toolkits.

http://www.sbml.org : This is the main SBML site. The site as ample documentation, examples illustrating how SBML is and should be used. In addition is has a rich set of software tools, in particular libSBML, which allows developers to easily add SBML support to their tools.

http://www.sys-bio.org : This is the main SBW (Systems Biology Workbench) site. The latest versions for SBW, developer documentation, example models, screen shots, user guides can be obtained from this site. A link to the main sourceforge site is given where all the source code for SBW is made available.

8 Acknowledgments

We would like to acknowledge the generous support from the Japan Science and Technology Agency, DARPA (BAA01-26 Bio-Computation), the US Department of Energy GTL program, and the National Institute of General Medical Sciences (NIGMS GM081070) without which the bulk of the work described in this chapter would not have been carried out. We would also like to acknowledge Mike Hucka, Andrew Finney and Hamid Bolouri for their initial work on the Systems Biology Workbench, Frank Bergmann for his tremendous programming work, Sri Paladugu and Vijay Chickarmane for their critical support in the development of novel computational methods in Systems Biology, and Maxwell Neal for his help and expertise on biomedical ontologies.

References

[1] A. Abbas. *Grid Computing: A Practical Guide to Tech. and App.* Firewall Media, 2004.

[2] R. Adams et al. "SBSI: An extensible distributed software infrastructure for parameter estimation in systems biology". In: *Bioinformatics* (2013).

[3] A. Bairoch et al. "The universal protein resource (UniProt)". In: *Nucleic acids research* 33.suppl 1 (2005), pp. D154–D159.

[4] B. M. Bakker et al. "Glycolysis in bloodstream form Trypanosoma brucei can be understood in terms of the kinetics of the glycolytic enzymes." In: *J. Biol. Chem.* 272 (1997), pp. 3207–3215.

[5] A.L. Barabási and Z.N. Oltvai. "Network biology: understanding the cell's functional organization". In: *Nature Reviews Genetics* 5.2 (2004), pp. 101–113.

[6] D.A. Beard et al. "CellML metadata standards, associated tools and repositories". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 367.1895 (2009), pp. 1845–1867.

[7] F.T. Bergmann and H.M. Sauro. "Comparing simulation results of SBML capable simulators". In: *Bioinformatics* 24.17 (2008), pp. 1963–1965.

[8] F.T. Bergmann and H.M. Sauro. "SBW-a modular framework for systems biology". In: *Proceedings of the 38th conference on Winter simulation*. Winter Simulation Conference. 2006, pp. 1637–1645.

[9] M.L. Blinov et al. "BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains". In: *Bioinformatics* 20.17 (2004), pp. 3289–3291.

[10] J.M. Bower, D. Beeman, and A.M. Wylde. *The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System*. Telos New York, 1995.

[11] A. Brazma et al. "Minimum information about a microarray experiment (MIAME)-toward standards for microarray data". In: *Nature genetics* 29.4 (2001), pp. 365–372.

[12] R.C. Cannon, C. O'Donnell, and M.F. Nolan. "Stochastic ion channel gating in dendritic neurons: morphology dependence and probabilistic synaptic activation of dendritic spikes". In: *PLoS computational biology* 6.8 (2010), e1000886.

[13] N.T. Carnevale and M.L. Hines. *The NEURON book*. Cambridge University Press, 2006.

[14] B. Chance. "The Kinetics of the Enzyme-Substrate Compound of Peroxidase". In: *J. Biol. Chem.* 151 (1943), pp. 553–577.

[15] HJ Chizeck, E. Butterworth, and J.B. Bassingthwaighte. "Error detection and unit conversion". In: *Engineering in Medicine and Biology Magazine, IEEE* 28.3 (2009), pp. 50–58.

[16] S.D. Cohen, A.C. Hindmarsh, et al. "CVODE, a stiff/nonstiff ODE solver in C". In: *Computers in physics* 10.2 (1996), pp. 138–143.

[17] W.B. Copeland et al. "Computational tools for metabolic engineering". In: *Metabolic engineering* (2012).

[18] M. Courtot et al. "Controlled vocabularies and semantics in systems biology". In: *Molecular systems biology* 7.1 (2011).

[19] J.O. Dada et al. "SBRML: a markup language for associating systems biology data with models". In: *Bioinformatics* 26.7 (2010), pp. 932–938.

[20] K. Degtyarenko et al. "ChEBI: a database and ontology for chemical entities of biological interest". In: *Nucleic acids research* 36.suppl 1 (2008), pp. D344–D350.

[21] H. deJong. “Modeling and Simulation of Genetic Regulatory Systems: A Literature Review”. In: *Journal Computational Biology* 9 (2002), 67–???103.

[22] E. Demir et al. “The BioPAX community standard for pathway data sharing”. In: *Nature biotechnology* 28.9 (2010), pp. 935–942.

[23] M. Diesmann and M.O. Gewaltig. “NEST: An environment for neural systems simulations”. In: *Forschung und wisschenschaftliches Rechnen, Beiträge zum Heinz-Billing-Preis* 58 (2001), pp. 43–70.

[24] A. Dräger et al. “JSBML: a flexible Java library for working with SBML”. In: *Bioinformatics* 27.15 (2011), pp. 2167–2168.

[25] D.A. Fell. *Understanding the Control of Metabolism*. London: Portland Press., 1997.

[26] A. Funahashi et al. “CellDesigner 3.5: a versatile modeling tool for biochemical networks”. In: *Proceedings of the IEEE* 96.8 (2008), pp. 1254–1265.

[27] A. Funahashi et al. “CellDesigner: a process diagram editor for gene-regulatory and biochemical networks”. In: *BIOSILICO* 1 (2003), pp. 159–162.

[28] A. Funahashi et al. “CellDesigner: a process diagram editor for gene-regulatory and biochemical networks”. In: *Biosilico* 1.5 (2003), pp. 159–162.

[29] D. Garfinkel. “A Machine-Independent Language for the Simulation of Complex Chemical and Biochemical Systems.” In: *Comput. Biomed. Res.* 2 (1968), pp. 31–44.

[30] D. Garfinkel, J. D. Rutledge, and J. J. Higgins. “Simulation and analysis of biochemical systems: I. representation of chemical kinetics”. In: *Communications of the ACM* 4(12) (1961), pp. 559–562.

[31] C.S. Gillespie et al. “Tools for the SBML Community”. In: *Bioinformatics* 22.5 (2006), pp. 628–629.

[32] P. Gleeson et al. “NeuroML: a language for describing data driven models of neurons and networks with a high degree of biological detail”. In: *PLoS computational biology* 6.6 (2010), e1000815.

[33] Nigel˜H. Goddard et al. “Towards NeuroML: Model Description Methods for Collaborative Modelling in Neuroscience”. In: *Philosophical Transactions of the Royal Society* (2001).

[34] E.J.V. Goncalves, J. Saez-Rodriguez, et al. “CySBGN: A Cytoscape plug-in to integrate SBGN maps”. In: *BMC bioinformatics* 14.1 (2013), p. 17.

[35] E. R. Harold and E. S. Means. *XML in a Nutshell*. 2001.

[36] R. Heinrich and S Schuster. *The Regulation of Cellular Systems*. Chapman and Hall, 1996.

[37] J. J. Higgins. “Kinetic properties of sequential enzyme systems.” PhD thesis. University of Pennsylvania, 1959.

[38] J.-H. S. Hofmeyr. “Metabolic Control Analysis in a Nutshell”. In: *Proceedings of the Second International Conference on Systems Biology*. Caltech, 2001.

[39] S. Hoops et al. “COPASI - a complex pathway simulator”. In: *Bioinformatics* 22.24 (2006), pp. 3067–3074.

[40] M. Hucka et al. “The Systems Biology Markup Language (SBML): A Medium for Representation and Exchange of Biochemical Network Models”. In: *Boinformatics* 19 (2003), pp. 524–531.

[41] M. Hucka et al. “The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version”. In: (2010).

[42] P. Hunter et al. “Integration from proteins to organs: the IUPS Physiome Project.” In: *Mechanisms of ageing and development* 126.1 (2005), p. 187.

[43] M.P. van Iersel et al. “Software support for SBGN maps: SBGN-ML and LibSBGN”. In: *Bioinformatics* 28.15 (2012), pp. 2016–2021.

[44] G. Joshi-Tope et al. “Reactome: a knowledgebase of biological pathways”. In: *Nucleic acids research* 33.suppl 1 (2005), pp. D428–D432.

[45] H. Kacser and J. A. Burns. “The Molecular Basis of Dominance.” In: *Genetics* 97 (1981), pp. 1149–1160.

[46] S.M. Keating and N. Novère. “Encoding Neuronal Models in SBML”. In: *Computational Systems Neurobiology* (2012), pp. 459–488.

[47] S.M. Keating et al. “SBMLToolbox: an SBML toolbox for MATLAB users”. In: *Bioinformatics* 22.10 (2006), pp. 1275–1277.

[48] H. Kitano. “A graphical notation for biochemical networks.” In: *BIOSILICO* 1 (2003), pp. 169–176.

[49] D. Köhn and N. Le Novere. “SED-ML–an XML format for the implementation of the MIASE guidelines”. In: *Computational Methods in Systems Biology*. Springer. 2008, pp. 176–190.

[50] K. W. Kohn et al. *Cell cycle control: molecular interaction map*. London: Nature Publishing Group (c) 2002 Macmillan Publishers Ltd., 2004, pp. 457–474.

[51] Kurt W. Kohn. “Molecular Interaction Map of the Mammalian Cell Cycle Control and DNA Repair Systems”. In: *Molecular Biology of the Cell* 10.8 (1999), pp. 2703–2734. URL: http://www.molbiolcell.org/content/10/8/2703.abstract.

[52] M. König, A. Dräger, and H.G. Holzhütter. “CySBML: a Cytoscape plugin for SBML”. In: *Bioinformatics* 28.18 (2012), pp. 2402–2403.

[53] F. Krause et al. “Annotation and merging of SBML models with semanticSBML”. In: *Bioinformatics* 26.3 (2010), pp. 421–422.

[54] N. Le Novère. "Model storage, exchange and integration". In: *BMC neuroscience* 7.Suppl 1 (2006), S11.

[55] N. Le Novere. "Report on the status of SBGN ER and proposed extensions". In: (2010).

[56] N. Le Novere et al. "BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems". In: *Nucleic acids research* 34.suppl 1 (2006), pp. D689–D691.

[57] N. Le Novère et al. "The systems biology graphical notation". In: *Nature biotechnology* 27.8 (2009), pp. 735–741.

[58] C. M. Lloyd, M. D. Halstead, and P. F. Nielsen. "CellML: its future, present and past." In: *Prog Biophys Mol Biol.* 85 (2004), pp. 433–50.

[59] C.M. Lloyd et al. "The CellML model repository". In: *Bioinformatics* 24.18 (2008), pp. 2122–2123.

[60] L. M. Loew and J. C. Schaff. "The Virtual Cell: a software environment for computational cell biology." In: *Trends Biotechnol.* 19 (2001), pp. 401–6.

[61] T. Maiwald and J. Timmer. "Dynamical modeling and multi-experiment fitting with PottersWheel". In: *Bioinformatics* 24.18 (2008), pp. 2037–2043.

[62] P. Mendes. "GEPASI: A software package for modelling the dynamics, steady states and control of biochemical and other systems." In: *Comput. Applic. Biosci.* 9 (1993), pp. 563–571.

[63] S. Mirschel et al. "PROMOT: modular modeling for systems biology". In: *Bioinformatics* 25.5 (2009), pp. 687–689.

[64] I.I. Moraru et al. "Virtual Cell modelling and simulation software environment". In: *Systems Biology, IET* 2.5 (2008), pp. 352–362.

[65] C.J. Myers et al. "iBioSim: a tool for the analysis and design of genetic circuits". In: *Bioinformatics* 25.21 (2009), pp. 2848–2849.

[66] M.L. Neal and H.M. Sauro. "SBML Reaction Finder: Retrieve and extract specific reactions from the BioModels database". In: (2012).

[67] M.L. Neal et al. "Advances in semantic representation for multiscale biosimulation: a case study in merging models". In: *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing.* NIH Public Access. 2009, p. 304.

[68] N.L. Novere et al. "Minimum information requested in the annotation of biochemical models (MIRIAM)". In: *Nature biotechnology* 23.12 (2005), pp. 1509–1515.

[69] H. Ogata et al. "KEGG: Kyoto encyclopedia of genes and genomes". In: *Nucleic acids research* 27.1 (1999), pp. 29–34.

[70] BG Olivier, JM Rohwer, and J.H.S. Hofmeyr. "Modelling cellular processes with Python and Scipy". In: *Molecular biology reports* 29.1 (2002), pp. 249–254.

[71] B.G. Olivier, J.M. Rohwer, and J.H.S. Hofmeyr. "Modelling cellular systems with PySCeS". In: *Bioinformatics* (2004).

[72] B.G. Olivier and J.L. Snoep. "Web-based kinetic modelling using JWS Online". In: *Bioinformatics* 20.13 (2004), pp. 2143–2144.

[73] B. O'Sullivan. "Distributed revision control with Mercurial". In: *Mercurial project* (2007).

[74] J. Quackenbush. "Standardizing the standards". In: *Molecular systems biology* 2.1 (2006).

[75] C.V. Rao, H.M. Sauro, and A.P. Arkin. "Putting the control in metabolic control analysis". In: *7th International Symposium on Dynamics and Control of Process Systems, DYCOPS*. Vol. 7. 2004.

[76] S. Ray and U.S. Bhalla. "PyMOOSE: interoperable scripting in Python for MOOSE". In: *Frontiers in neuroinformatics* 2.6 (2008), pp. 1–16.

[77] GM Raymond, E. Butterworth, and JB Bassingthwaighte. "JSIM: Free software package for teaching physiological modeling and research". In: *Experimental Biology* 280 (2003), pp. 102–107.

[78] J. G. Reich and E. E. Selkov. *Energy metabolism of the cell*. London: Academic Press, 1981.

[79] H. M. Sauro. "Jarnac: A System for Interactive Metabolic Analysis". In: *Animating the Cellular Map: Proceedings of the 9th International Meeting on BioThermoKinetics*. Ed. by J-H. S. Hofmeyr, J. M. Rohwer, and J. L. Snoep. Stellenbosch University Press, 2000. ISBN: ISBN 0-7972-0776-7.

[80] H. M. Sauro and D. A. Fell. "SCAMP: A metabolic simulator and control analysis program." In: *Mathl. Comput. Modelling* 15 (1991), pp. 15–28.

[81] H. M. Sauro and B. Ingalls. "Conservation analysis in biochemical networks: computational issues for software writers." In: *Biophys Chem.* 109 (2004), pp. 1–15.

[82] H. M. Sauro et al. "Next Generation Simulation Tools: The Systems Biology Workbench and BioSPICE Integration". In: *OMICS* 7(4) (2003), pp. 355–372.

[83] H.M. Sauro. "SCAMP: a general-purpose simulator and metabolic control analysis program". In: *Computer applications in the biosciences: CABIOS* 9.4 (1993), pp. 441–450.

[84] H.M. Sauro, B. Ingalls, et al. "Conservation analysis in biochemical networks: computational issues for software writers." In: *Biophysical chemistry* 109.1 (2004), p. 1.

[85] H.M. Sauro and B.N Kholodenko. "Quantitative analysis of signaling networks". In: *Progress in biophysics and molecular biology* 86.1 (2004), pp. 5–43.

[86] H. Schmidt and M. Jirstrand. "Systems Biology Toolbox for MATLAB: a computational platform for research in systems biology". In: *Bioinformatics* 22.4 (2006), pp. 514–515.

[87] B. Smith et al. "Relations in biomedical ontologies". In: *Genome biology* 6.5 (2005), R46.

[88] B. Smith et al. "The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration". In: *Nature biotechnology* 25.11 (2007), pp. 1251–1255.

[89] L.P. Smith et al. "Antimony: a modular model definition language". In: *Bioinformatics* 25.18 (2009), pp. 2452–2454.

[90] P. Spellman et al. "Design and implementation of microarray gene expression markup language (MAGE-ML)". In: *Genome biology* 3.9 (2002), research0046.

[91] "Standard operating procedures". In: *Nature Biotechnology* 24.11 (), p. 1299.

[92] L. Strömbäck and P. Lambrix. "Representations of molecular pathways: an evaluation of SBML, PSI MI and BioPAX". In: *Bioinformatics* 21.24 (2005), pp. 4401–4407.

[93] C.F. Taylor et al. "Promoting coherent minimum reporting guidelines for biological and biomedical investigations: the MIBBI project". In: *Nature biotechnology* 26.8 (2008), pp. 889–896.

[94] M. Tomita et al. "E-CELL: software environment for whole-cell simulation." In: *Bioinformatics* 15 (1999), pp. 72–84.

[95] J. J. Tyson, K. Chen, and B. Novak. "Network Dynamics And Cell Physiology". In: *Nature Reviews Molecular Cell Biology* 2 (2001), pp. 908–916.

[96] R.R. Vallabhajosyula, V. Chickarmane, and H.M. Sauro. "Conservation analysis of large biochemical networks". In: *Bioinformatics* 22.3 (2006), pp. 346–353.

[97] M. Vass et al. "The JigCell model builder and run manager". In: *Bioinformatics* 20.18 (2004), pp. 3680–3681.

[98] M.T. Vass et al. "The JigCell model builder: a spreadsheet interface for creating biochemical reaction network models". In: *Computational Biology and Bioinformatics, IEEE/ACM Transactions on* 3.2 (2006), pp. 155–164.

[99] D. Waltemath et al. "Minimum information about a simulation experiment (MIASE)". In: *PLoS computational biology* 7.4 (2011), e1001122.

[100] S. Wright. "Fisher's Theory of Dominance". In: *The American Naturalist* 63 (1929), pp. 274–279.

[101] S. Wright. “Physiological and Evolutionary Theories of Dominance”. In: *The American Naturalist* 68 (1934), pp. 24–53.

[102] T. Yu et al. “The physiome model repository 2”. In: *Bioinformatics* 27.5 (2011), pp. 743–744.