# Model Composition for Macromolecular Regulatory Networks

Ranjit Randhawa, Clifford A. Shaffer, and John J. Tyson

**Abstract**—Models of regulatory networks become more difficult to construct and understand as they grow in size and complexity. Large models are usually built up from smaller models, representing subsets of reactions within the larger network. To assist modelers in this composition process, we present a formal approach for model composition, a wizard-style program for implementing the approach, and suggested language extensions to the Systems Biology Markup Language to support model composition. To illustrate the features of our approach and how to use the JigCell Composition Wizard, we build up a model of the eukaryotic cell cycle "engine" from smaller pieces.

**Index Terms**—Modeling, composition, fusion, flattening, SBML.

## 1 Regulatory Network Modeling

The physiological properties of cells are governed by macromolecular regulatory networks of great complexity [1]. Understanding the dynamical properties of these networks is facilitated by mathematical modeling of the biochemical reactions [1], [2], [3], [4]. These models are often implemented deterministically, as sets of nonlinear differential equations, or probabilistically by Gillespies' stochastic simulation algorithm. In either case, the modeler is faced with the problem of specifying the reactions involved in a large complex network of interacting species, the rate laws describing each reaction, and numerical values for the rate constants involved in each rate law. Building regulatory network models is a little like putting together a complicated jigsaw puzzle with many interlocking pieces. This complex modeling challenge is best broken down into smaller components that can later be joined together into a larger whole. The Systems Biology Markup Language (SBML) [5] was created to support the modeling of biochemical reaction networks, but the present version (Level 2) does not support any notion of model composition. In an earlier publication [6], we presented an algorithm and a wizard-tool for model "fusion," which is an irreversible process for putting submodels together. In this paper, we describe a reversible process that we call model "composition."

Throughout this paper, we illustrate the process of model composition with the protein interaction network controlling the cell division cycle [7]. In eukaryotes, the cell cycle is controlled by a set of cyclin-dependent protein kinases (CDKs) that phosporylate specific target proteins and thereby initiate the events of DNA replication, mitosis, and cell division. CDK activity is controlled by interactions with a variety of regulatory proteins, including the anaphase promoting complex (APC), which, in combination with two auxiliary proteins (Cdc20 and Cdh1), degrades the cyclin component of CDK, and a suite of cyclin-dependent kinase inhibitors (CKIs), which bind to and inhibit CDKs. A simple model of these interactions (Fig. 1) is sufficient to reproduce (in simulation) many features of cell cycle regulation in budding yeast [8]. This model shows how progress through the cell cycle can be thought of as irreversible transitions (Start and Finish) between two stable states (G1 and S-G2-M) of the regulatory system.

Such networks are often represented as graphs where vertices represent substrates and products (collectively referred to as species), and labeled directed edges connecting vertices represent the reactions. Chemical reactions cause the concentrations of the chemical species $(C_i)$ to change in time according to the equation

$$\frac{dC_i}{dt} = \sum_{j=1}^{R} b_{ij} v_j, i = 1, \ldots, N,$$

where $R$ is the number of reactions, $v_j$ is the velocity of the $j$th reaction in the network, and $b_{ij}$ is the stoichiometric coefficient of species $i$ in reaction $j$ ($b_{ij} < 0$ for substrates, $b_{ij} > 0$ for products, $b_{ij} = 0$ if species $i$ takes no part in reaction $j$).

The full set of rate equations is a mathematical representation of the temporal behavior of the regulatory network. A realistic model of the budding yeast cell cycle consists of over 30 differential equations and 100 rate constants [9]. The parameters are estimated from the cell-cycle behavior of more than 100 mutants defective in the regulatory network. A model of such complexity (10-100 equations) is approaching the limit of what a dedicated modeler can produce and analyze with the tools available today. Beyond this size, it becomes difficult to code the differential equations without error, to manage the simulation of hundreds of separate

- *R. Randhawa is with the Computational Sciences Center of Emphasis, Pfizer Global Research & Development, 620 Memorial Drive, Cambridge, MA 02139. E-mail: ranjit.randhawa@pfizer.com.*
- *C.A. Shaffer is with the Department of Computer Science, Virginia Tech, Blacksburg, VA 24061-0106. E-mail: shaffer@cs.vt.edu.*
- *J.J. Tyson is with the Department of Biological Sciences, Virginia Tech, Blacksburg, VA 24061-0106. E-mail: tyson@vt.edu.*

*Manuscript received 6 Nov. 2007; revised 11 Mar. 2008; accepted 2 June 2008; published online 19 June 2008.*
*For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-2007-11-0152. Digital Object Identifier no. 10.1109/TCBB.2008.64.*

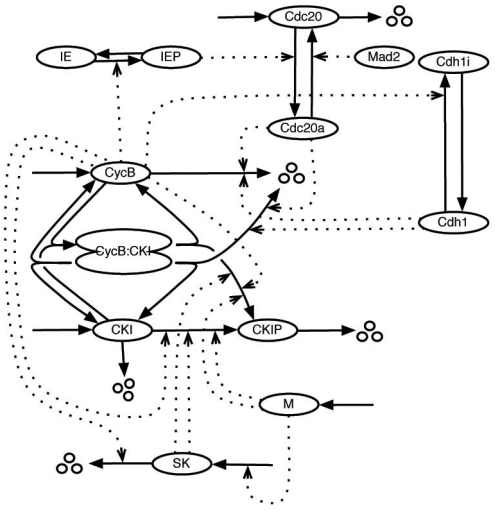



Fig. 1. Reaction network for cell cycle control in yeast. Icons are proteins, solid arrows are chemical reactions, and dotted arrows represent enzymatic catalysis.

experiments, or to comprehend the significance of the results. To adequately describe fundamental physiological processes (such as the control of cell division) in mammalian cells will require models with 100-1,000 equations. Efforts such as the DARPA BioSPICE initiative [10] and the DOE Genomes to Life project [11] aspire to support models at least one order of magnitude larger than are currently used. What sort of software support do mathematical biologists need to build models of such complexity?

## 2 Building Large Networks

Over the last 20 years, molecular biologists have amassed a great deal of information about the genes and proteins that carry out fundamental biological processes within living cells; processes such as growth and reproduction, movement, signal reception and response, and programmed cell death. The full complexity of these macromolecular regulatory networks is too great to tackle mathematically at the present time. Nonetheless, modelers have had success building dynamical models of restricted parts of the network. For example, for budding yeast cells, there have been recent successful efforts to model the cell cycle [9], the pheromone signaling pathway [12], the response to osmotic shock [13], and the morphogenetic checkpoint [14]. Systems biologists need tools now to support composition of "submodels" (like these) into more comprehensive models of integrated regulatory networks. Merging submodels is currently done manually and is an error-prone process. Our work is a step to providing computational assistance for model composition.

We assume that each of the submodels used in creating the larger model is a validated model itself, with experimental data that fixes its parameters. The main motivation for creating a larger model is that there exists experimental information on the interaction of the subsystems—interactions that the submodels cannot account for. By composing validated submodels, we mitigate the problem of searching through large parameter spaces. The parameter estimation problem is now to ensure that the composed model is consistent with the original data used to validate the submodels (for which we already have good initial guesses, inherited from the submodels) and also the new data relevant to the interactions of the subsystems (which are governed by the new parameters describing how the submodels fit together).

Modeling languages and tools help modelers construct their models by providing a computational environment that minimizes the amount of human error during the construction step. While modelers are currently able to construct small and medium-sized models by hand, the process is simplified by using computational tools that decrease the time taken to input a model and provide error-testing services along the way. In this paper, we describe a representation and a tool that enable modelers to create large models by (sub)model composition.

Our prior work has identified several distinct processes related to model composition [15]. In this paper, we briefly describe model fusion and flattening to provide some background for composition. Fusion and flattening have previously been implemented, in the Fusion Wizard application [6] and the Model Flattener algorithm, respectively. The present paper focuses on the SBML extensions needed to describe composition, and presents a prototype tool that enables composing models together. We provide a meaningful example of composition using Tyson and Novak's models from [8].

*Model Fusion* combines two or more models in an irreversible manner. In fusion, the identities of the original (sub)models being combined are lost. The result of fusion is a model in the same language as the submodels (in our case, standard SBML [5]), meaning that the same simulation analysis tools can be applied. Beyond some size, fused models will become too complex to grasp and manage as single entities. In this case, it may be more useful to represent large models as compositions of distinct components. Thus, while model fusion is a useful tool for manipulating small to mid-sized models, it does not seem to be a viable solution in the long run.

*Model Flattening* converts a composed model with some hierarchy or connections (discussed later) to one without such connections. The result is equivalent to fusing the submodels. The relationship information provided by the composition process must be sufficient to allow the flattening to take place without any further human intervention. The relationships used to describe the interactions among the submodels are lost, as the composed model is converted into a single large (flat) model. Flattening a model allows us to use existing simulation tools, which have no support for composition.

*Model Composition* provides a potential solution to our goal to build models of large reaction networks. With composition, one can think of models not as monolithic entities, but rather as collections of smaller components (submodels) joined together. A composed model is built from two or more submodels by describing their redundancies and interactions. Composition is a reversible process, in that removing the intermodel interaction description that holds the composed model together recovers the individual submodels.

## 3 Context and Prior Work

The XML-based SBML [5] has become widely supported within the network modeling community. Thus, we choose to present concrete implementations for the various modeling processes through added SBML constructs that express the necessary glue that connects submodels together. It is not necessary that our proposals be implemented in SBML but doing so provides clear reference implementations in the same way as expressing an algorithm in a particular programming language.

SBMLmerge [16] is a tool for building large models from smaller components, but does not support model composition. Process Modeling Tool (ProMoT) [17] and E-Cell [18] are modeling packages that use some kind of modularity. Modules in ProMoT are logical, encapsulated groupings of modeling elements that represent compartments which contain reactions, species, and special signaling parts. ProMoT provides support for modularity and hierarchical modeling. It uses object-oriented models, composed from modules, and has its origins in process engineering. It provides support for importing/exporting standard SBML (Level 2). E-Cell uses an architecture where the complete model may be modularized through compartments. In this sense, modules must have some physical border and are not only logical or functional groupings but represent an object in the physical topology of the cell.

Snoep et al. [19] showed it was possible to construct a large model in a bottom-up manner by manually linking together smaller modules. They demonstrated this by combining a glycolysis pathway model with a glycoxylate pathway model. Bulatewicz et al. [20] suggested an interface for model coupling and provided a number of solutions, from a brute force technique to using frameworks specifically designed to support coupling. A number of authors from domains outside systems biology find that successful composition (or model "reuse") requires components that are specifically designed for the purpose [21], [22], [23], [24], [25]. Within the context of regulatory models, we distinguish this approach with the term "aggregation," which we will discuss briefly in Section 7.

Proposals have been made within the SBML community [26], [27], [28] that describe the mechanics of composition through additional SBML features, as we will do. However, none of these proposals have been published in the peer-reviewed literature, nor to our knowledge have any been implemented. While some commercial tools might have more or less support for various forms of composition, we are unaware of any nonproprietary implementations for model composition in this application domain, or any publications describing proprietary features in commercial applications. Model composition for pathway models remains very much an open problem. Our work focuses primarily on how the composition process can be achieved. The SBML extensions we propose and put into context with respect to the composition process elaborate on those originally proposed in [26].

Our implementation differs from [26] in a number of ways. The notion of <instance> structures which enabled model reuse using XLink [29] to instantiate any number of models to access them has been replaced by adding a new XPointer [29] attribute to the <model> structure. In this way, we minimize the number of additional SBML constructs needed while still allowing for model reuse. The appendix provides an example using SBML syntax of two models located in different SBML documents. The <link> structure also differs significantly from the one proposed in [26]. Previous proposals have highlighted the need for imposing restrictions on linkages in a composed model without identifying these restrictions. Our implementation incorporates the notion of direct links [26] by forcing level-by-level linkages and also checks for circular linkages (discussed in Section 5). Restricting the linkages does not mean that components further than a level away cannot be linked together, it simply means that linking them will be done indirectly and automatically (level-by-level). Our implementation allows modelers to link components of different types (species, compartment, or parameter) together, unlike previous proposals. For example, a modeler can quickly and easily promote a parameter to a species in cases where the parameter was originally only used as an approximation before more information was discovered of the system or pathway under investigation. Our implementation also deals with the notion of replacing or deleting unwanted or unused components within submodels (discussed in Section 5). Finally, we add an additional attribute to the <link> structure called a merge/replacement attribute which specifies the linkage type (discussed in the Appendix).

## 4 Model Fusion

Model Fusion is an iterative process to build large models by merging two or more submodels. Unlike composition (where submodels are referenced but not modified), fusion irreversibly changes the submodels in the process of combining them together. The goal of fusion is to combine submodels into a single unified model containing the aggregated information (without redundancies) across the original collection.

During fusion, the modeler produces a mapping table for the various SBML component types (compartments, species, reactions, etc.). During processing, we must deal with dependencies across component types. For example, we must resolve the identities of compartments (which represent the bounded space in which species are located) before species, since each species stores a reference to its containing compartment. Fortunately, the following ordering for the eight SBML component types has no conflicting dependencies:

1. compartments,
2. species,
3. function definitions,
4. rules,
5. events,
6. units,
7. reactions, and
8. parameters.

In other words, by fusing the component types in this order, we resolve all dependencies before they are encountered.

Fusion can be implemented by "mapping tables." Each column in a mapping table represents a model, and each row represents an SBML component in that model. Duplicate names within a model are not allowed. For example, a species name may occur only once per column. The first column in the mapping table is reserved for the fused model. The two actions available to the modeler during fusion are

1. define two or more SBML components to be equivalent and
2. remove the equivalence definition between two or more SBML components (which have previously been incorrectly equivalent).

We have created a Fusion Wizard application that solicits information from the user in a step-by-step process. Fusion proceeds in two stages: setup and resolution. During setup, the modeler is guided through various steps that initialize the Fusion Wizard. The modeler first assigns a name to the fused model, then selects with a file chooser the submodels to be fused together. The next screen allows the modeler to select a control option for the mapping tables: the user directs the system either 1) to place components of the same name on the same row or 2) to place each component on a different row. The auto-fill screen attempts to minimize the amount of work needed by the modeler by filling up the fused column on rows where there are no naming conflicts. However, the modeler may decide during resolution whether to use these initial choices or to change them (either to another choice from a predetermined list of available components or to provide a more appropriate name for the component by specifying a new name).

Once the environment has been initialized, the modeler starts resolving the naming conflicts in the various submodels. Resolving names is vital to be able to identify unambiguously all the distinct entities within a model and their relationships with each other. Once the SBML component mapping tables have been generated, the application uses this information to automatically merge the submodels together. The fused model is now created from the reaction networks of the submodels.

## 5 Model Composition

We now explain in detail how to generate a hierarchy of models that produce what we call a *composed* model. Composed models can be thought of as a collage of smaller submodels. Within the context of SBML, we add new language features to describe the relationships among submodels. We refer to such constructs as "glue." The language additions for SBML described in the appendix allow modelers to compose models from submodels, and include support for multiple instances of a given submodel. The features both define the hierarchy of the submodels and represent the interactions, relationships, links, and reactions between the submodels.

It turns out that there are significant similarities between model fusion and model composition, as we discovered during the process of developing the fusion tool. The fusion process described in Section 4 defines a series of steps taken to merge two or more models together. This series of steps can be viewed as an "audit trail" used in generating the necessary mapping tables. Precisely, this same information can be used to describe the set of instructions needed to connect/link the submodels for composition. Thus, both composition and fusion produce the same results. This is ensured by the fact that fusion and composition are equivalent processes that take the same information to create larger models. The processing applied to both approaches guarantees the same set of transforms are used to create fused models on one hand, and composed-then-flattened models on the other. As an additional check to confirm proper implementation, we have compared the simulation results of the fused model versus the flattened version of the composed model. By restricting the type of information required to generate the fused and composed forms, we can say with some certainty that both fusion and composition will indeed produce the same results. At a first glance, the two models in Figs. 5 and 7 look vastly different, however they represent two representations for the same regulatory network (see the caption in Fig. 7 for more details). While fusion combines submodels together in an irreversible way, composition simply references submodel components by defining the "glue" that holds the submodels together. A major difference is that in fusion, the explicit description of relationships between entities within submodels is lost, while composition keeps a "record" of how models were composed/connected together. Further details of the language features required to implement composition within SBML are given in the Appendix.

Model composition is an iterative process, as models are usually constructed in increments, with modelers switching back and forth between adding components to a model and fine tuning models through simulations. Constructing composed models is a bottom-up process as smaller models are first composed together to create larger models which in turn can be used to create even more complex models, as previously demonstrated in [19]. A composed model can be created from a combination of flat and/or composed models. Model composition generates a composition tree that describes the hierarchical relations among the various submodels. Changing the connections between submodels in a composed model results in a different composition tree structure. Since model composition is a combination of constructing submodels and generating composition trees, a tool for composition should take into consideration the iterative nature of the process.

When using our tools, the composition process is similar to the fusion process. The main difference is in how the mapping tables are set up and used. Composition mapping tables must deal with combining both composed and flat models. The first column of a composition mapping table is reserved for the root of the composition tree (the composed model), followed by its children (submodels). The mapping tables are able to identify equivalent components across the submodels in order to determine which components to add (and link) in the composed model. The various lists of SBML components (species, reactions, parameters, etc) in each submodel are treated as distinct sets, and the intersection of these sets represents components that occur across all the submodels. For example, only those species that occur within all the submodels will be present in the intersection set and therefore added to the composed model. To signify

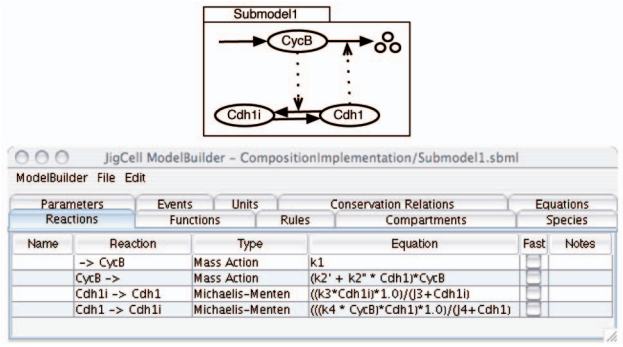



Fig. 2. Submodel 1 wiring diagram and reaction definitions in the JigCell ModelBuilder.

equivalence, links are automatically created from components in the composed model to components in the submodels. The composed model will also contain modeler-defined components (selected during name resolution) from its children, which will also be linked together. Adding and linking components at each level in a composition hierarchy ensures that a composed model will only directly reference components at most one level below itself, but can indirectly reference components across many levels in the composition tree. The four actions applied to the composition mapping tables during resolution are as follows:

1. Automatically add components that occur within the intersection (initially components with the same name) of the immediate submodels to the composed model only, and create links between equivalent components from the composed model to its submodels.
2. Add new user-defined/selected components from the immediate submodels to the composed model and create links to the equivalent components in the submodels.
3. Define two or more SBML components within the immediate submodels to be equivalent, add them to the composed model, and create links to the equivalent components in the submodels.
4. Remove the link between a component in the composed model and a component within the submodels (which have previously been incorrectly linked together).

The JigCell Composition Wizard (http://jigcell.biol.vt.edu) restricts the linking mechanism to ensure components can only be linked together level-by-level and not across many levels. For example, suppose model $X$ contains a submodel $Y$ which itself contains another submodel $Z$. Both models $X$ and $Z$ contain the same species $S$. Then, $X.S$ cannot be linked directly to $Z.S$ as there is more than a single level of distance between the two models in the composition hierarchy. In this case, the Composition Wizard automatically adds a new species $S$ to model $Y$ and creates two links, one from $X.S$ to $Y.S$ and the other from $Y.S$ to $Z.S$. This mechanism ensures that components can be indirectly linked across any number of levels in a composition hierarchy and prevents the linking of components not defined in the namespace they are used in.

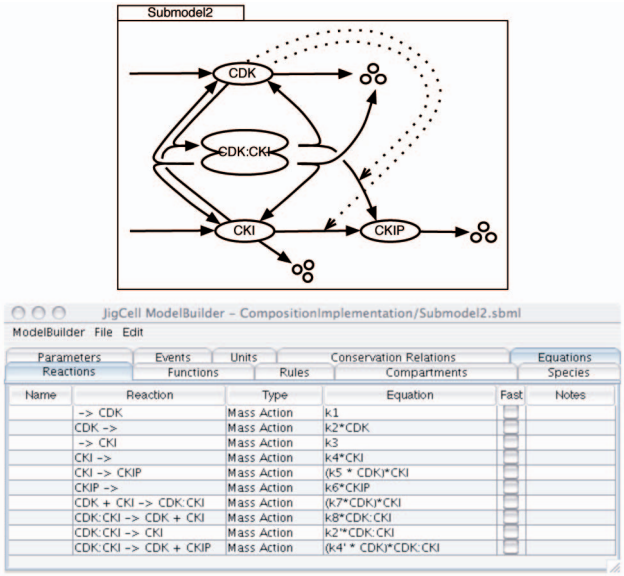



Fig. 3. Submodel 2.

## 6 An Example

To illustrate how composition is implemented, we follow the approach used by Tyson and Novak when building their basic model of cell cycle control in yeast cells [8]. They built their model in stages starting from a simple model and then adding new pieces until they obtained a satisfactory representation of the cell cycle control system. Their starting model (which we will call *Submodel 1*) deals with the antagonistic interactions between cyclin B-dependent kinase (*CycB*) and a cyclin B-degrading factor (*Cdh1*), as shown in Fig. 2. The next step was to create a model (which we call *Submodel 2*) of the interaction between the cyclin-dependent kinase (now called *CDK*) and a *CKI*, as shown in Fig. 3. Finally, they built a second version of the simple model (which we call *Submodel 3*) of the interaction between "mitosis promoting factor" (*MPF*) and a different form of the cyclin-degrading factor (*Cdc20*), as shown in Fig. 4. Note that different names have been used here for the same *CycB* in all three models (*CycB*, *CDK*, and *MPF*) to better highlight implementation details of composition.

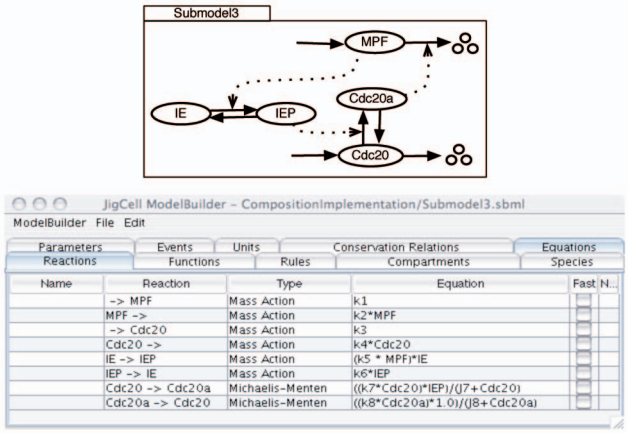



Fig. 4. Submodel 3.

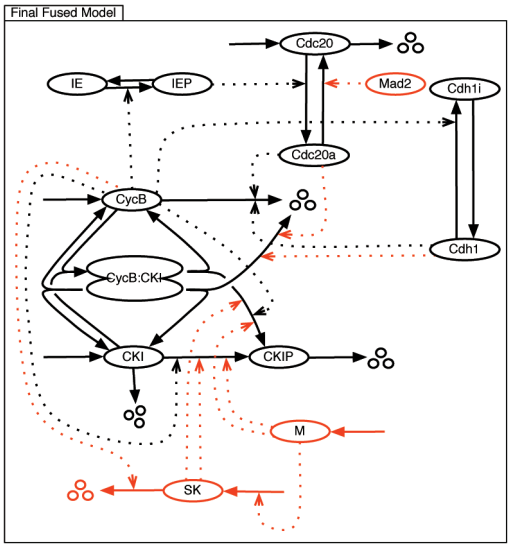



Fig. 5. Fusion of Submodels 1-3. Components in red were added after the fusion process was completed.

The three submodels can be combined using the fusion process to create a *Fused Model* (Fig. 5) or composed together to create a *Composed Model* (Fig. 6). Species, reactions, and modifier effects in red are new components that have been added after completing fusion or composition.

The *Composed Model* shown in Fig. 6 is only a conceptual version, showing how the user envisions the composed model but not how the computer represents it. The *Composed Model* produced by the JigCell Composition Wizard is better represented in Fig. 7, which contains duplicated species, reactions, and parameters (not shown in Fig. 6). In both figures, color is used to indicate equivalent species that have been linked together across submodels. Construction of the *Composed Model* requires multiple rounds of both model construction (using the JigCell ModelBuilder [31]) and composition (using the JigCell Composition Wizard). The three steps needed to create the *Composed Model* are as follows:

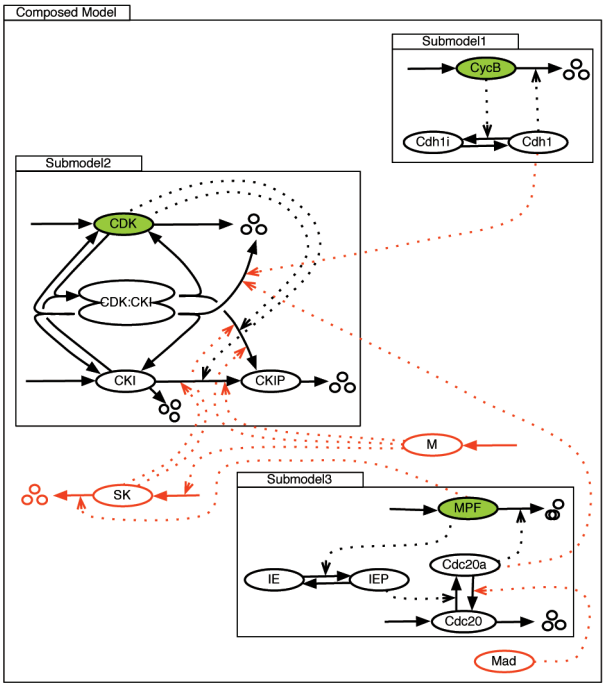



Fig. 6. Conceptual representation of the composition of Submodels 1-3. Colored species indicate equivalences across submodels. Components in red were added to the model after the composition process was completed.

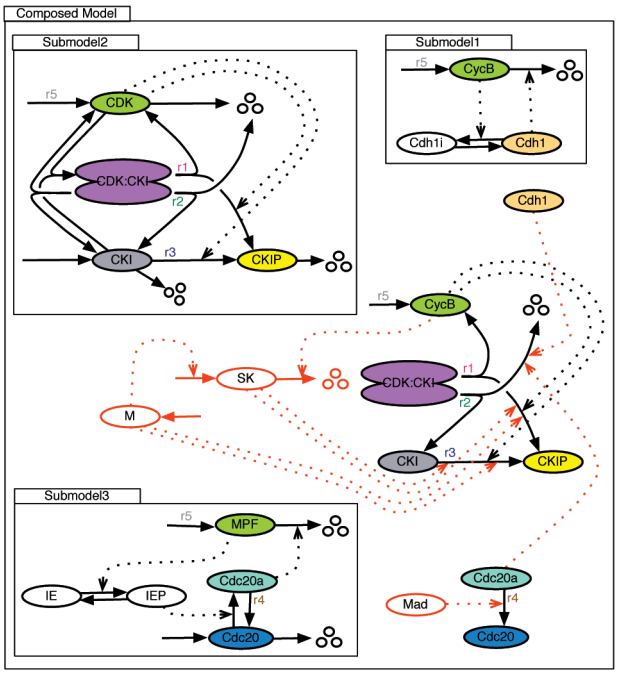
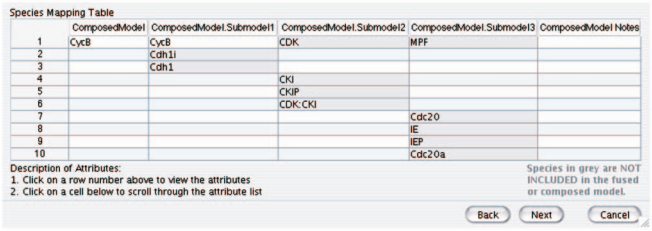



Fig. 7. Actual representation of the composition of Submodels 1-3 by the JigCell Composition Wizard and the JigCell ModelBuilder. Colored species and numbered reactions ($r1-r5$) indicate equivalences across submodels. For example, while there are four reactions labeled $r5$, only one (reaction $r5$ in the top-level composed model) is "used," the others are linked to the top-level reaction and remain "unused." Similarly with species, while the model contains four green species representing *CycB* only the top-level *CycB* is used, and the others become placeholders. Flattening this model will produce the model in Fig. 5.

1. *Submodels 1-3* are loaded into the Composition Wizard, and initial mapping tables for species and reactions are produced. From these mapping tables, the user generates an intermediate composed model.
2. The intermediate composed model is opened in the ModelBuilder and additional species and reactions are added (shown in red in Fig. 7).
3. The intermediate composed model is opened in the Composition Wizard and a second round of name resolution occurs, producing the final mapping tables.

The Composition Wizard resolves components based on their names and automatically adds and links the intersection of components within the submodels to the composed model. Each (sub)model in this example contains only one compartment, and they are all identified as a single compartment in the composed model in the compartment mapping table. The compartment mapping table can also handle models with more than one compartment. As the CDK has been named differently in the three submodels, the first step in the initial species mapping table is to set


Species Mapping Table

| | ComposedModel | ComposedModel.Submodel1 | ComposedModel.Submodel2 | ComposedModel.Submodel3 | ComposedModel Notes |
|---|---|---|---|---|---|
| 1 | CycB | CycB | CDK | MPF | |
| 2 | | Cdh1i | | | |
| 3 | | Cdh1 | | | |
| 4 | | | CKI | | |
| 5 | | | CKIP | | |
| 6 | | | CDK:CKI | | |
| 7 | | | | Cdc20 | |
| 8 | | | | IE | |
| 9 | | | | IEP | |
| 10 | | | | Cdc20a | |

Description of Attributes:
1. Click on a row number above to view the attributes
2. Click on a cell below to scroll through the attribute list

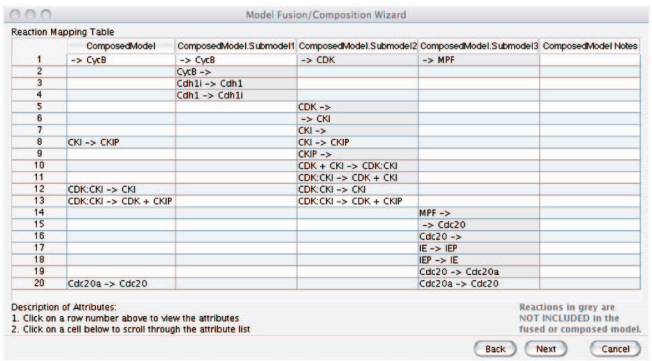
Species in grey are NOT INCLUDED in the fused or composed model.

Back Next Cancel

Fig. 8. Initial species mapping table for the *Composed Model* in the JigCell Composition Wizard.

these species to be equivalent to each other by placing them on the same row. Next, the CDK is added to the composed model column and called *CycB*. This causes three links to be automatically created: one from *Composed Model.CycB* to *Submodel 1.CycB*, the second from *Composed Model.CycB* to *Submodel 2.CDK*, and the third from *Composed Model.CycB* to *Submodel 2.MPF*. In this way, all the instances of CDK are linked together in the initial species mapping table and thus represent the same species (Fig. 8). The next step is to create the initial reaction mapping table in Fig. 9. The five reactions marked $r1{-}r5$ in Fig. 7 are added and linked in the composed model. Reactions ($r1{-}r4$) will be modified as a result of the composition. The *CycB* synthesis reaction ($r5$) occurs in all submodels with the same rate law and is thus linked to ensure that only one copy of the reaction will be used during simulation/flattening. However, the reverse reaction (*CycB* degradation) is not added/linked in the composed model as it contains different rate laws (and modifier effects) in the three submodels.

Once the first round of composition is finished, an intermediate composed model is generated and opened in the JigCell ModelBuilder for editing. Three additional reactions ($\rightarrow SK$, $SK \rightarrow$, and $\rightarrow M$) are added, which correspond to the red reactions in Fig. 7. $SK$ refers to a "starter kinase" and $M$ represents cell "mass." $M$ increases according to a logistic rate equation, and $M$ is decreased by a factor of 2 each time the cell divides. Cell division is triggered when *CycB* drops below a certain threshold, as cyclin B is degraded by *Cdc20* and *Cdh1* and is best represented as an event in the model (see [8] for details). Next, the four novel reactions in the composed model ($r1{-}r4$) are updated to reflect their new kinetic laws. This intermediate composed model is saved and loaded into the Composition Wizard. The components are once again


Model Fusion/Composition Wizard

Reaction Mapping Table

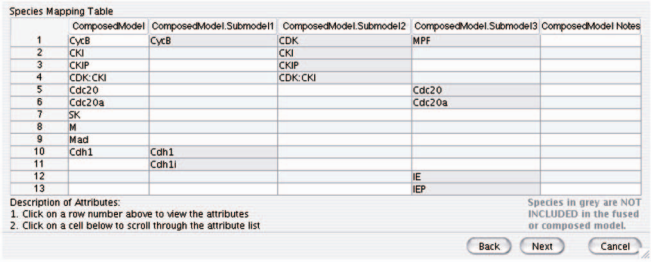
| | ComposedModel | ComposedModel.Submodel1 | ComposedModel.Submodel2 | ComposedModel.Submodel3 | ComposedModel Notes |
|---|---|---|---|---|---|
| 1 | -> CycB | -> CycB | -> CDK | -> MPF | |
| 2 | | CycB -> | | | |
| 3 | | Cdh1i -> Cdh1 | | | |
| 4 | | Cdh1 -> Cdh1i | | | |
| 5 | | | CDK -> | | |
| 6 | | | -> CKI | | |
| 7 | | | CKI -> | | |
| 8 | CKI -> CKIP | | CKI -> CKIP | | |
| 9 | | | CKIP -> | | |
| 10 | | | CDK + CKI -> CDK:CKI | | |
| 11 | | | CDK:CKI -> CDK + CKI | | |
| 12 | CDK:CKI -> CKI | | CDK:CKI -> CKI | | |
| 13 | CDK:CKI -> CDK + CKIP | | CDK:CKI -> CDK + CKIP | | |
| 14 | | | | MPF -> | |
| 15 | | | | -> Cdc20 | |
| 16 | | | | Cdc20 -> | |
| 17 | | | | IE -> IEP | |
| 18 | | | | IEP -> IE | |
| 19 | | | | Cdc20 -> Cdc20a | |
| 20 | Cdc20a -> Cdc20 | | | Cdc20a -> Cdc20 | |

Description of Attributes:
1. Click on a row number above to view the attributes
2. Click on a cell below to scroll through the attribute list

Reactions in grey are NOT INCLUDED in the fused or composed model.

Back Next Cancel

Fig. 9. Initial reaction mapping table for the *Composed Model*.


Species Mapping Table

| | ComposedModel | ComposedModel.Submodel1 | ComposedModel.Submodel2 | ComposedModel.Submodel3 | ComposedModel Notes |
|---|---|---|---|---|---|
| 1 | CycB | CycB | CDK | MPF | |
| 2 | CKI | | CKI | | |
| 3 | CKIP | | CKIP | | |
| 4 | CDK:CKI | | CDK:CKI | | |
| 5 | Cdc20 | | | Cdc20 | |
| 6 | Cdc20a | | | Cdc20a | |
| 7 | SK | | | | |
| 8 | M | | | | |
| 9 | Mad | | | | |
| 10 | Cdh1 | Cdh1 | | | |
| 11 | | Cdh1i | | | |
| 12 | | | | IE | |
| 13 | | | | IEP | |

Description of Attributes:
1. Click on a row number above to view the attributes
2. Click on a cell below to scroll through the attribute list

Species in grey are NOT INCLUDED in the fused or composed model.

Back Next Cancel

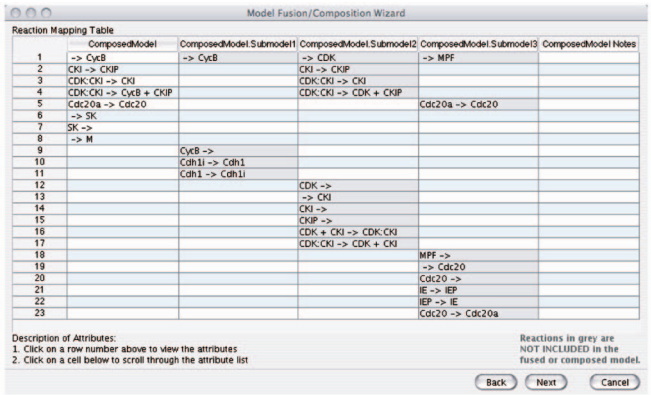
Fig. 10. Final species mapping table for the *Composed Model*.

resolved, and the final species and reaction mapping tables are produced (Figs. 10 and 11, respectively).

These steps produce the final composed model (Fig. 7) which must then be simulated to verify that its dynamic properties represents the observed behavior of growing-dividing yeast cells in expected ways. Since our current simulators require standard SBML (Level 2) input, composed models must be flattened before they can be simulated. This flattening is done by removing the additional constructs used to describe the composition. The JigCell flattening algorithm automates this process and produces a single flat model that can then be sent to a simulator. Simulating the flattened *Composed Model* produces the simulation output shown in Fig. 12, which closely matches the simulation output from Tyson and Novak's model [8, Fig. 8].

## 7 Future Plans

We are currently investigating a number of different problems in composition, such as determining the feasibility of automating the fusion/composition process, applying composition to large-scale modeling efforts, and looking at alternative approaches to composition in order to build larger models. As models increase in size and complexity the approaches discussed in this paper would greatly benefit from some form of automation. While an automated form of fusion or composition is not in the scope of this paper, efforts within the systems biology community that focus on model curation and annotation of quantitative models of biological systems, such as Minimum Information Requested in the Annotation of biochemical Models


Model Fusion/Composition Wizard

Reaction Mapping Table

| | ComposedModel | ComposedModel.Submodel1 | ComposedModel.Submodel2 | ComposedModel.Submodel3 | ComposedModel Notes |
|---|---|---|---|---|---|
| 1 | -> CycB | -> CycB | -> CDK | -> MPF | |
| 2 | CKI -> CKIP | | CKI -> CKIP | | |
| 3 | CDK:CKI -> CKI | | CDK:CKI -> CKI | | |
| 4 | CDK:CKI -> CycB + CKIP | | CDK:CKI -> CDK + CKIP | | |
| 5 | Cdc20a -> Cdc20 | | | Cdc20a -> Cdc20 | |
| 6 | -> SK | | | | |
| 7 | SK -> | | | | |
| 8 | -> M | | | | |
| 9 | | CycB -> | | | |
| 10 | | Cdh1i -> Cdh1 | | | |
| 11 | | Cdh1 -> Cdh1i | | | |
| 12 | | | CDK -> | | |
| 13 | | | -> CKI | | |
| 14 | | | CKI -> | | |
| 15 | | | CKIP -> | | |
| 16 | | | CDK + CKI -> CDK:CKI | | |
| 17 | | | CDK:CKI -> CDK + CKI | | |
| 18 | | | | MPF -> | |
| 19 | | | | -> Cdc20 | |
| 20 | | | | Cdc20 -> | |
| 21 | | | | IE -> IEP | |
| 22 | | | | IEP -> IE | |
| 23 | | | | Cdc20 -> Cdc20a | |

Description of Attributes:
1. Click on a row number above to view the attributes
2. Click on a cell below to scroll through the attribute list

Reactions in grey are NOT INCLUDED in the fused or composed model.

Back Next Cancel

Fig. 11. Final reaction mapping table for the *Composed Model*.

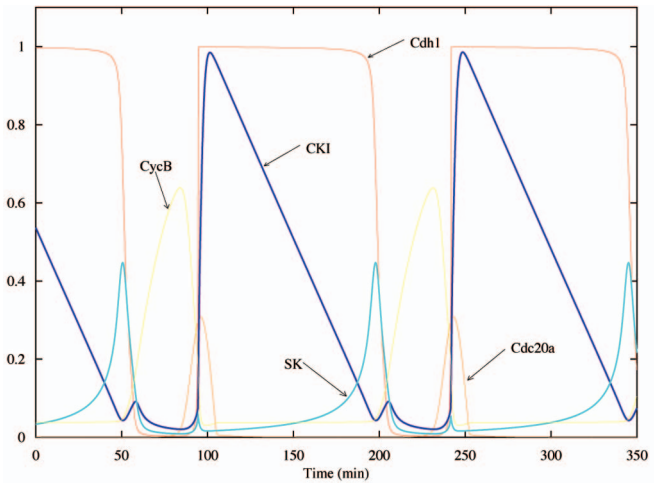



Fig. 12. Simulation of the *Composed Model* using XPP.

(MIRIAM) [31] and BioModels [32], might someday make this a very real possibility.

Current modeling efforts in Tyson's Group at Virginia Tech involve challenging issues in large-scale modeling. One such effort is focused on the "morphogenesis checkpoint" in budding yeast. Ciliberto et al. [14] developed a model of the morphogenesis checkpoint that was "hooked up" to a very primitive cell cycle engine in budding yeast. We have successfully combined the morpho-checkpoint module with the full cell cycle engine proposed by Chen et al. [9] and will publish the results in the near future.

While it is appealing in the short term to build larger models from preexisting models, we believe that ultimately it will become necessary to build large models from components that have been designed for the purpose of combining them. This matches the experience of the broader modeling community [21], [22], [23], [24], [25]. We distinguish this approach from model composition as described in this paper. We define *Model Aggregation* as a restricted form of composition that represents a collection of model elements as a single entity (a "module"). A module contains a definition of predetermined input and output ports. These ports link to internal species and parameters. They define the module's interface, which provides restricted access to the components in the module. The process of aggregation (connecting modules via their interface ports) allows modelers to create larger models in a controlled manner. Our future work will explore the model aggregation process.

## Appendix

## SBML Syntax

The SBML features described below elaborate on those originally proposed in [26] by providing a defined framework with a proof of concept implementation to demonstrate the feasibility of composition. To illustrate the SBML features needed to describe model composition, consider a large model (called *Global*), composed of two submodels ($A$ and $B$). Model $A$ contains the chemical species $x$ and model $B$ contains the species $y$. It is now possible to make a new reaction in *Global* that represents $x \rightarrow y$, by referring to $x$ and $y$ in $A$ and $B$, respectively. *Global* consists of a model

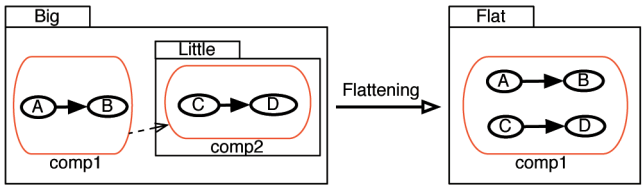



Fig. 13. Composed model showing a link between two compartments in different submodels and its corresponding flattened model.

with only one reaction. The names of reactants and products for that reaction refer to the corresponding species in the two submodels. It should be noted that adding a new reaction (or any new component) is not performed in the fusion or composition tool. Instead, this action is accomplished in a model-building environment used to create the (sub)models.

A composed model can contain one or more submodels within its structure. A submodel contains a valid SBML model (an SBML <model> structure), with its own namespace, and can itself be a composed model. Since there is no restriction on the number of submodels a model can contain, a <model> structure is enclosed in a <listOfSubmodels> structure. A simple example (Fig. 13) shows a composed model (*Big*) and its corresponding flattened model (*Flat*). Model *Big* contains a submodel called *Little*, and each (sub)model contains a single compartment (*comp1* and *comp2*, respectively) and reaction ($A \rightarrow B$ and $C \rightarrow D$, respectively).

Finally, different components (species, reactions, etc) within either the submodels or the global model are connected/accessed using <link> structures.

We adopt a naming convention to enable modelers to uniquely identify an SBML component (e.g., species, parameters, etc) within a model (or submodel). Our format is

```
<component object="ObjectIdentifier">
  <subobject object="SubobjectIdentifier"/>
</component>
```

We will also represent this same information using the syntax *ObjectIdentifier.SubobjectIdentifier*. This convention makes it possible to refer to SBML components with the same name in different models without having to change their names.

Models can be composed of more than one instance of a particular submodel. This is accomplished using the XPointer framework [29] to refer to submodels, as it defines a reference to another location in the current document, or an external document, using an extended pointer notation. An instance of submodel *Little* (called *Submodel_Little*) can be made within model *Big* to access submodel *Little* in model *Big*. The <model> structure contains a new attribute: an *xref*, which is represented using an XPointer string [30] which is used for locating data within an XML document.

A <link> (enclosed in a <listOfLinks> ) links two entities in separate submodels of a composed model. A <link> should be able to link two <species> , <parameters>, <reactions> , or <compartments> to each other. Linking components in composition can be achieved by using mapping tables similar to those created during fusion. Components on the same row in the mapping table will be

linked together. A <link> is composed of two fields, <from> and <to> . The <to> field references an object (the *to object*) whose attribute values will be overridden by the object referenced by the <from> field (the *from object*). The objects referenced by <from> and <to> fields must be of the same type. Only those attribute values that have been declared in the *from object* will be overridden in the *to object*. This is somewhat analogous in *C/C++* to treating the *to object* as a pointer, and the *from object* as its target. However, a *to object* can have attribute values that are retained if no overriding attribute value is declared in the *from object*. Note that if we have two components inside a (sub)model we are still able to link subobjects of the components using our object/subobject naming convention. The following example shows how the two compartments in *Big* and *Little* can be linked together (Fig. 13). (Note that the two models occur as separate SBML files.)

```
<model id="Little">
 <listOfCompartments>
  <compartment id="comp2" size="1"/>
 </listOfCompartments>
</model>

<model id="Big">
 <listOfCompartments>
  <compartment id="comp1" size="1"/>
 </listOfCompartments>
 <listOfSubmodels>
  <model id="Submodel_Little"
   xref="#xpointer
    (Little.sbml/sbml/
    model[@id=%22Little%22])"/>
 </listOfSubmodels>
 <listOfLinks>
  <link merge="true">
   <from object="comp1"/>
   <to object="Submodel_Little">
    <subobject object="comp2"/>
   </to>
  </link>
 </listOfLinks>
</model>
```

The above SBML code shows an example where the submodel *Little* occurs within a different SBML document (named "Little.sbml"). If the submodel *Little* occurred within the same SBML document, the *xref/xpointer* attribute of the <model> structure would not have a filename prepended to it.

The <link> structure contains a *merge* attribute, whose value can be either true (indicating a *merge* link) or false (indicating a *replacement* link). To see the difference, consider models $R$ and $T$ which each contain a chemical species called $S1$ with different attributes. $S1$ in Model $R$ has attribute $InitialSubstance = 1.0$. $S1$ in Model $T$ has attributes $InitialSubstance = 2.0$ and $Constant = true$. Linking $S1$ in $R$ to $S1$ in $T$ with a merge link uses $S1$'s attributes from $T$. $S1$ that have not been declared in $R.S1$. The result is that $S1$ has attributes $InitialSubstance = 1.0$ and $Constant = true$ since it keeps its old value for $InitialSubstance$ and gains the definition for $Constant$. If $S1$ in $R$ is linked to $S1$ in $T$ using a replacement link (i.e., the *merge* attribute is false), then only $R.S1$'s attributes are used. The result will be that $S1$ will only have attribute $InitialSubstance = 1.0$ (it has not value for $Constant$). Specifying the type of link for composition requires adding a new field to the mapping table to specify the merge/replacement attribute.

The <link> structure can link certain combinations of differing SBML component types to each other, such as species $\leftrightarrow$ parameters and rules $\leftrightarrow$ species/parameters. A link can take a <species> structure as the *from object* and a <parameter> structure as the *to object*, and vice versa.

The <link> structure should also be able to link together groups of reactions, thus enabling N to N links. The syntax for the <link> structure should allow zero or more <from> and one or more <to> object references. A link with zero <from> references deletes the <to> object(s). This is useful to remove or ignore a component or a group of components within a particular (sub)model that do not participate in the overall composition. A link cannot contain more than one species or compartment <from> references (otherwise, it would be possible to split a species or compartment). Note that a <link> to nothing is not allowed.

## ACKNOWLEDGMENTS

The authors would like to acknowledge the support of the US National Institutes of Health Grant R01-GM078989-01 and TCNP Grant U54RR022232.

## REFERENCES

[1] K. Kohn, "Molecular Interaction Map of the Mammalian Cell Cycle Control and DNA Repair Systems," *Molecular Biology of the Cell,* vol. 10, no. 8, pp. 2703-2734, 1999.
[2] B. Novak and J. Tyson, "Modelling the Controls of the Eukaryotic Cell Cycle," *Biochemical Soc. Trans.,* vol. 31, no. 6, pp. 1526-1529, 2003.
[3] J. Sible and J. Tyson, "Mathematical Modeling as a Tool for Investigating Cell Cycle Control Networks," *Methods,* vol. 41, no. 2, pp. 238-247, 2007.
[4] J. Tyson, "Bringing Cartoons to Life," *Nature,* vol. 445, no. 7130, p. 823, 2007.
[5] M. Hucka et al., "The Systems Biology Markup Language (SBML): A Medium for Representation and Exchange of Biochemical Network Models," *Bioinformatics,* vol. 19, no. 4, pp. 524-531, 2003.
[6] R. Randhawa, C. Shaffer, and J. Tyson, "Fusing and Composing Macromolecular Regulatory Network Models," *Proc. High Performance Computing Symp. (HPC '07),* pp. 337-344, 2007.
[7] A. Csikasz-Nagy, D. Battogtokh, K. Chen, B. Novak, and J. Tyson, "Analysis of a Generic Model of Eukaryotic Cell-Cycle Regulation," *Biophysical J.,* vol. 90, no. 12, pp. 4361-4379, 2006.
[8] J. Tyson and B. Novak, "Regulation of the Eukaryotic Cell Cycle: Molecular Antagonism, Hysteresis, and Irreversible Transitions," *J. Theoretical Biology,* vol. 210, pp. 249-263, 2001.
[9] K. Chen, L. Calzone, A. Csikasz-Nagy, F. Cross, B. Novak, and J. Tyson, "Integrative Analysis of Cell Cycle Control in Budding Yeast," *Molecular Biology of the Cell,* vol. 15, pp. 3841-3862, 2004.
[10] DARPA BioSPICE Website, DARPA, http://community.biospice.org, 2005.
[11] US Department of Energy Genomes to Life Website, DOE, http://doegenomestolife.org/, 2005.
[12] B. Kofahl and E. Klipp, "Modelling the Dynamics of the Yeast Pheromone Pathway," *Yeast,* vol. 21, no. 10, pp. 831-850, 2004.
[13] E. Klipp, B. Nordlander, R. Kruger, P. Gennemark, and S. Hohmann, "Integrative Model of the Response of Yeast to Osmotic Shock," *Nature Biotechnology,* vol. 23, no. 8, pp. 975-982, 2005.

[14] A. Ciliberto, B. Novak, and J. Tyson, "Mathematical Model of the Morphogenesis Checkpoint in Budding Yeast," *J. Cell Biology*, vol. 163, pp. 1243-1254, 2003.
[15] C. Shaffer, R. Randhawa, and J. Tyson, "The Role of Composition and Aggregation in Modeling Macromolecular Regulatory Networks," *Proc. Winter Simulation Conf. (WSC '06)*, Dec. 2006.
[16] M. Schulz, J. Uhlendorf, E. Klipp, and W. Liebermeister, "SBMLmerge, a System for Combining Biochemical Network Models," *Genome Informatics*, vol. 17, no. 1, pp. 62-71, 2006.
[17] M. Ginkel, A. Kremling, T. Nutsch, R. Rehner, and E.D. Gilles, "Modular Modeling of Cellular Systems with ProMoT/Diva," *Bioinformatics*, vol. 19, no. 9, pp. 1169-1176, 2003.
[18] K. Takahashi, N. Ishikawa, Y. Sadamoto, H. Sasamoto, S. Ohta, A. Shiozawa, F. Miyoshi, Y. Naito, Y. Nakayama, and M. Tomita, "E-Cell 2: Multi-Platform E-Cell Simulation System," *Bioinformatics*, vol. 19, pp. 1727-1729, 2003.
[19] J. Snoep, F. Bruggeman, B. Olivier, and H. Westerhoff, "Towards Building the Silicon Cell: A Modular Approach," *Biosystems*, vol. 83, pp. 207-216, 2006.
[20] T. Bulatewicz, J. Cuny, and M. Warman, "The Potential Coupling Interface: Metadata for Model Coupling," *Proc. Winter Simulation Conf. (WSC '04)*, pp. 183-190, 2004.
[21] P. Davis and R. Anderson, "Improving the Composability of DoD Models and Simulations," *J. Defense Modeling and Simulation*, vol. 1, no. 1, pp. 5-17, 2004.
[22] D. Garlan, R. Allen, and J. Ockerbloom, "Architectural Mismatch or Why It's Hard to Build Systems Out of Existing Parts," *Proc. 17th Int'l Conf. Software Eng. (ICSE '95)*, pp. 179-185, 1995.
[23] S. Kasputis and H. Ng, "Model Composability: Formulating a Research Thrust: Composable Simulations," *Proc. Winter Simulation Conf. (WSC '00)*, pp. 1577-1584, 2000.
[24] R. Malak and C. Paredis, "Foundations of Validating Reusable Behavioral Models in Engineering Design Problems," *Proc. Winter Simulation Conf. (WSC '04)*, pp. 420-428, 2004.
[25] M. Spiegel, P. Reynolds, and D. Brogan, "A Case Study of Model Context for Simulation Composability and Reusability," *Proc. Winter Simulation Conf. (WSC '05)*, pp. 437-444, 2005.
[26] A. Finney, "Systems Biology Markup Language (SBML) Level 3 Proposal: Model Composition Features," http://www.cds.caltech. edu/afinney/model-composition.pdf, 2003.
[27] M. Ginkel, "Modular SBML Proposal for an Extension of SBML towards Level 2," *Proc. Fifth Forum on Software Platforms for Systems Biology*, 2003.
[28] D. Schroder and J. Weimar, "Modularization of SBML," http://www. sbml.org/workshops/ninth/VortragSBMLForum.pdf, 2003.
[29] P. Grosso, E. Maler, J. Marsh, and N. Walsh, "XPointer Framework Version 1.0 W3C Recommendation," http://www.w3.org/TR/xptr-framework/, 2003.
[30] M.T. Vass, C. Shaffer, N. Ramakrishnan, L. Watson, and J. Tyson, "The JigCell Model Builder: A Spreadsheet Interface for Creating Biochemical Reaction Network Models," *IEEE/ACM Trans. Computational Biology and Bioinformatics*, vol. 3, no. 2, pp. 155-164, Apr.-June 2006.
[31] N.L. Novre, A. Finney, M. Hucka, U. Bhalla, F. Campagne, J. Collado-Vides, E. Crampin, M. Halstead, E. Klipp, P. Mendes, P. Nielsen, H. Sauro, B. Shapiro, J. Snoep, H. Spence, and B. Wanner, "Minimum Information Requested in the Annotation of Biochemical Models (MIRIAM)," *Nature Biotechnology*, vol. 23, no. 12, pp. 1509-1515, 2005.
[32] N.L. Novre, B. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, L.L.L.H. Sauro, M. Schilstra, B. Shapiro, J. Snoep, and M. Hucka, "BioModels Database: A Free, Centralized Database of Curated, Published, Quantitative Kinetic Models of Biochemical and Cellular Systems," *Nucleic Acids Research*, vol. 34, pp. D689-91, 2006.

**Ranjit Randhawa** received the BS degrees in computer science and genetic biology from Purdue University and the MS and PhD degrees from Virginia Tech, focusing on computational biology. He is a senior scientist at the Computational Sciences Center of Emphasis at Pfizer Global Research & Development. His research interests include software design, systems biology, computational biology, bioinformatics, and modeling and simulation.

**Clifford A. Shaffer** received the PhD degree from the University of Maryland. He is a professor in the Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg. His current research interests include problem-solving environments, bioinformatics, component architectures, visualization, algorithm design and analysis, and data structures. He is a senior member of the IEEE.

**John J. Tyson** received the PhD degree in chemical physics from the University of Chicago in 1973. He has been specializing in theoretical cell biology since that time. He is a University Distinguished professor of biological sciences in the Department of Biological Sciences, Virginia Polytechnic Institute and State University, Blacksburg. His current interests include the gene-protein interaction networks that regulate features of cell physiology such as cell division, circadian rhythms, intracellular signaling networks, and programmed cell death.