The JigCell Model Builder: A Tool for Modeling Intra-Cellular Regulatory Networks

Marc Vass* Clifford A. Shaffer* John J. Tyson** Naren Ramakrishnan* Layne T. Watson***

Departments of Computer Science, Biology**, and Mathematics****
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0106
E-mail: mvass@vt.edu

Abstract

A number of approaches to inputting models for intra-cellular regulatory networks are described. The JigCell Model Builder (JCMB) allows users to describe such models using a spreadsheet paradigm. Spreadsheets are familiar to most expected users of the system, and they match the mental model that users have of this type of model. The spreadsheet also is efficient in its use of screen space, providing a great deal of information in a compact form. JCMB is described in detail, with particular emphasis on its effectiveness in reducing the number of errors made by modelers.

1. Introduction

The use of differential equations for cellular modeling is becoming a common means for both describing and computing cellular activity. Traditionally, many stages in the modeling cycle have been done by hand. This cycle typically begins with the modeler creating a diagram of their model, then converting the diagram to reaction equations, and finally to differential equations. This process presents two problems. First, it consumes a great amount of time and effort on the part of the researchers. Secondly, there are many opportunities within the process for errors to creep in. It was for this reason that the JigCell Model Builder (JCMB) tool was designed and implemented as part of a broader problem solving environment for the eukaryotic cell cycle.

This paper first describes a number of alternative user interface paradigms for describing models. We then describe JCMB in detail, with particular emphasis on how JCMB addresses these two problems.

2. Background and related work

At the core of any description for a regulatory system in cellular biology is the reaction network. A chemical reaction can be defined as a conversion of substrates $S_{1,\ldots,i}$ and products $P_{1,\ldots,j}$, written as $S_1+S_2+\cdots+S_i \rightarrow P_1+P_2+\cdots+P_j$ where a rate v describes the velocity at which the reaction occurs. The rate v can be a simple constant or a more complicated formula involving the substrates and products.

A reaction network consists of a set of chemical reactions and can be represented in two ways: as a graph or as a set of equations. The graph form uses vertices representing substrates and products, collectively referred to as species, and labeled directed edges connecting vertices to represent the velocities of the reactions. An accompanying text typically is used to specify the reaction velocity since it is difficult to fit this information onto a single edge in a graphical representation. This graphical presentation has the problem that arbitrarily complex graphs can result as the reaction network grows in size. There is no standard definition for how a graph of this sort should be drawn and labeled, so ambiguities cause many problems when a graph is converted to a computational model.

The second method of representing a reaction network is by explicitly writing out the chemical reactions and their associated velocities. This approach loses some of the intuitive nature provided by the diagrammatic approach, but allows for a more compact definition of the reaction network. Normally, the modeler prior to this step has already made a hand-or CAD-drawn version of the network in graphical form, showing the interactions in a qualitative sense but without the quantitative information of the velocity equations or the parameter values.

Once the graph or set of reaction equations has been defined the next step is to form the computational model. An ordinary differential equation must be created for each species in the model. Each species ordinary differential equation will include a negative term corresponding to each reaction where the species is consumed by the reaction, and a positive term for each reaction where the species is created by the reaction. In terms of the graph representation each vertex has an equation, with a negative term corresponding to an exiting edge and a positive term

corresponding to an entering edge.

Our model representation must also account for conservation equations. Conservation equations involve sets of species whose combined amounts remain constant throughout a simulation. These equations can simplify simulation of the models, and try to provide extra information on the properties of the models [8-9].

Next, the model representation must allow user-defined rate laws. This allows the modeler to avoid repeatedly defining velocities for similar reactions, or to define a rate law that is not available by default in the system.

Finally, the models used by the Computational Cell Biology Lab at Virginia Tech specify that an event is to occur in the model under given conditions. As an example from the budding yeast cell cycle, cellular division, represented by a decrease in mass, should occur when a given function involving a group of species reaches a certain amount during a simulation. Neither a network diagram, nor a chemical reaction can represent such events.

Several tools have been built to address both the modeling and simulation of reaction networks. The state of the art in model building tools for cell biology is described next.

2.1. Virtual Cell

Virtual Cell [3,11] provides a user interface for specifying a model as a graphical network. The graphical network is designed through use of a model workspace where a species is represented as a circle and a chemical reaction is represented by a barbell. Substrates are connected to the left end of the barbell and products to the right end of the barbell. Catalysts are connected to the middle of the barbell. Right clicking on the barbell generates a dialog box that allows the user to enter the velocity of the reaction. The mass action rate law is given as a default, with a locally defined rate law being allowed, but no user-defined rate law may be specified.

As the model grows, an increasing fraction of the graph's edges cross one another, leading to confusion for the user. Users then typically fall back to an alternative interface in the form of a textual representation for the partial differential equations. Unfortunately, a change in the mathematical workspace is not reflected in the graphical representation, so once the user switches to the math workspace, there is no further opportunity to return to the graphical interface. A strength of Virtual Cell is that both spatial and temporal relationships are modeled, resulting in partial differential equations in up to three space dimensions.

The Virtual Cell model workspace calculates conservation equations automatically and does not allow the user to specify the species that are regarded as dependent in simulation calculations. This may be specified in the mathematical workspace. Also, Virtual Cell does not provide supports for specifying events. In general the graphical tool provided by Virtual Cell does not support large models adequately.

2.2. Gepasi

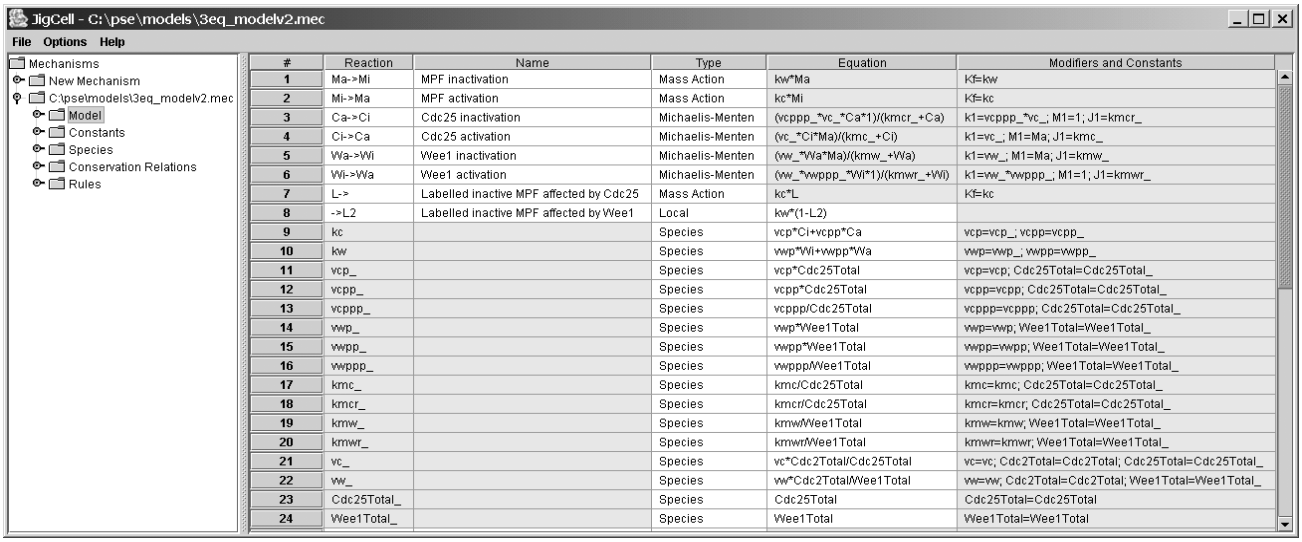
Gepasi [4-6] is a purely chemical-reaction-centered approach to designing reaction networks. It uses a "wizard" metaphor with a series of dialog boxes to lead the user through the various stages of creating a reaction network. First, the user must enter the chemical equations for their network by selecting the reactions button. Next, to specify the velocities of the equations, the user must click the kinetics button and select the rate law to use for the chemical equation. To add a user-defined rate law the user must go back to the previous screen and click the kinetic types button and choose the add button. User defined functions are specified in a similar manner by clicking on the functions button and choosing the add button.

Gepasi breaks the model representation (and the model building process) across many screens. Thus, large models are difficult to visualize using Gepasi due to the limited space available. Events are not specifiable in Gepasi. Gepasi provides the user with the conservation equations found in the model, but does not allow the user to specify what the independent and dependent species are within those relations. The ordinary differential equations resulting from the chemical network are unavailable to the user.

2.3. Jarnac

Jarnac [10] is based on the chemical reaction approach, but it also includes an application known as JDesigner for creating graphical reaction networks. Jarnac uses a text-based scripting language for describing models that is compact and similar to the specification of chemical equations. A shortcoming of this tool is that one cannot generate a graphical design network from the scripting language. User-defined functions are allowed, but globally defined rate laws are not. A large model may be specified quickly and be compartmentalized due to support in the scripting language for network data types. This allows the user to be able to more easily visualize or consider a larger model.

The text-based scripting language is not a natural approach to describing reactions for may users, who are biologists and not programmers. Models are specified in a programming language-like form. For example, a simple branch system whose chemical equations are written as:



#	Reaction	Name	Type	Equation	Modifiers and Constants
1	Ma->Mi	MPF inactivation	Mass Action	kw*Ma	Kf=kw
2	Mi->Ma	MPF activation	Mass Action	kc*Mi	Kf=kc
3	Ca->Ci	Cdc25 inactivation	Michaelis-Menten	(vcppp_*vc_*Ca*1)/(kmcr_+Ca)	k1=vcppp_*vc_; M1=1; J1=kmcr_
4	Ci->Ca	Cdc25 activation	Michaelis-Menten	(vc_*Ci*Ma)/(kmc_+Ci)	k1=vc_; M1=Ma; J1=kmc_
5	Wa->Wi	Wee1 inactivation	Michaelis-Menten	(vw_*Wa*Ma)/(kmw_+Wa)	k1=vw_; M1=Ma; J1=kmw_
6	Wi->Wa	Wee1 activation	Michaelis-Menten	(vw_*vwppp_*Wi*1)/(kmwr_+Wi)	k1=vw_*vwppp_; M1=1; J1=kmwr_
7	L->	Labelled inactive MPF affected by Cdc25	Mass Action	kc*L	Kf=kc
8	->L2	Labelled inactive MPF affected by Wee1	Local	kw*(1-L2)	
9	kc		Species	vcp*Ci+vcpp*Ca	vcp=vcp_; vcpp=vcpp_
10	kw		Species	vwp*Wi+vwpp*Wa	vwp=vwp_; vwpp=vwpp_
11	vcp_		Species	vcp*Cdc25Total	vcp=vcp; Cdc25Total=Cdc25Total_
12	vcpp_		Species	vcpp*Cdc25Total	vcpp=vcpp; Cdc25Total=Cdc25Total_
13	vcppp_		Species	vcppp/Cdc25Total	vcppp=vcppp; Cdc25Total=Cdc25Total_
14	vwp_		Species	vwp*Wee1Total	vwp=vwp; Wee1Total=Wee1Total_
15	vwpp_		Species	vwpp*Wee1Total	vwpp=vwpp; Wee1Total=Wee1Total_
16	vwppp_		Species	vwppp/Wee1Total	vwppp=vwppp; Wee1Total=Wee1Total_
17	kmc_		Species	kmc/Cdc25Total	kmc=kmc; Cdc25Total=Cdc25Total_
18	kmcr_		Species	kmcr/Cdc25Total	kmcr=kmcr; Cdc25Total=Cdc25Total_
19	kmw_		Species	kmw/Wee1Total	kmw=kmw; Wee1Total=Wee1Total_
20	kmwr_		Species	kmwr/Wee1Total	kmwr=kmwr; Wee1Total=Wee1Total_
21	vc_		Species	vc*Cdc2Total/Cdc25Total	vc=vc; Cdc2Total=Cdc2Total; Cdc25Total=Cdc25Total_
22	vw_		Species	vw*Cdc2Total/Wee1Total	vw=vw; Cdc2Total=Cdc2Total; Wee1Total=Wee1Total_
23	Cdc25Total_		Species	Cdc25Total	Cdc25Total=Cdc25Total
24	Wee1Total_		Species	Wee1Total	Wee1Total=Wee1Total

Figure 1. Frog egg extract model in JCMB

X0→S1, S1→X1, S1→X2

is defined in Jarnac as:

```
function Mult (a, b)
    return a * b;
end;
p = defn Branch
    [J1] $X0 -> S1; Mult(k1,X0);
    [J2] S1 -> $X1; k2*S1;
    [J3] S1 -> $X2; k3*S1;
end;
```

Most biologists are unfamiliar with such syntax and only a template of the code (the reactions such as S1->$X1) is provided as output from the graphical reaction network created in JDesigner.

3. JigCell Model Builder

The JigCell Model Builder is based on the chemical reaction approach discussed earlier with special considerations given to the needs of the modelers at Virginia Tech (conservation equations and events). We also attempt to build in enough flexibility to support the needs of future modelers. The interface was designed with a spreadsheet metaphor with reactions, functions, and rate laws being defined on individual rows of the spreadsheet.

3.1. Design rationale

JCMB was designed primarily to reduce the number of errors generated in the cellular modeling process and to closely match the mental model of the modeler. The spreadsheet interface allows the modeler to visualize the entirety of their model and express it in the language of their domain. Modelers can see and specify a chemical equation and its associated rate laws, constants, and the generated velocity equation on the same line.

JCMB attempts to reduce errors by forcing the user to use a reaction-centered approach that separates the reaction equations from rate law specification. This approach allows the computer to apply the specified rate law to discover the velocity for a particular reaction, which is then shown to the user in a separate column.

Preventing errors is important because an error in the model leads to meaningless simulation results. Even worse, the errors may be difficult to detect and may cost the modeler hours searching for an error in their logic when the actual error is a minor syntactic slip made at model-building time.

JCMB attempts to disallow inconsistencies in a model while it is being entered. When an inconsistency or error is detected, the spreadsheet will highlight the problem cell in orange and will propagate the error throughout the spreadsheet by highlighting other problem cells as orange. When the error has been fixed, all cells now made consistent will return to their normal color. Many columns in a given row show derived data, and thus are "grayed-out" and not editable.

Most model builders (biologists) might be expected to prefer a graphical interface, such as Virtual Cell provides. This concept was carefully considered as a future layer for JCMB. This would move JCMB from being a hybrid tool - incorporating textual modeling concepts into a graphical, spreadsheet-like environment - to an almost purely visual programming environment. However, two confounding issues prevented us from implementing a graph-based interface.

The first issue arises from the problem of converting bi-

directionally between functional representation and diagrammatic representation. Diagrammatic construction is certainly possible, and such concepts have been effectively applied to other domains, such as electronics. Unfortunately, users also would require the ability to go "under the hood" and tweak the model on a formula level. An automated system could have no systematic way of knowing how to appropriately adjust the diagrammatic model to reflect the formulaic changes. Any change could have the effect of making the diagram less readable. Although allowing the user to lay out the changed portions of the diagram might alleviate this problem, the complications introduced by this editing would likely outweigh the benefits reaped by the original diagrammatic construction.

The second issue is more closely tied to the primary purpose for which JCMB was designed (error reduction). There is no current standard notation for computational cell modeling on a diagrammatic level. In fact, field experts use many widely different notation schemes in their publications. In some cases, the symbol used by one author has a very different meaning in the diagram of another author. An additional problem is that a practical diagrammatic notation scheme for describing a large model in a graphical interface has yet to be designed. This presents a major problem for design and user understanding of diagram layouts.

3.2. Model spreadsheet

JCMB uses a spreadsheet interface with access to the model sections done through selection from a tree view (Figure 1). Only the *Model* spreadsheet has differing row types.

3.2.1. Reaction row

A reaction specifies what species are involved in a chemical reaction, the chemical equation for the reaction, the rate law, and the modifiers and constants needed by the rate law. A reaction consists of a list of substrates separated by +s, an arrow "->", and a list of products also separated by +s. Substrate and product names must begin with a letter and may be followed by any combination of letters, numbers, or apostrophes. The stoichiometry of a species is specified by placing the value directly in front of the name, or by writing the value separated from the name with a '*'. For example, one can write "2X + Y -> 3Y".

A reaction name can be any character string. A reaction name may be duplicated in other reactions. The name has no meaning except as a cue to the user for what reaction is contained in the row.

The *Type* column specifies the rate law to be applied for the reaction given in the reaction column for this row. The three predefined rate laws are mass action, michaelis-menten and local (we present in the next section how to define new rate laws). Mass action is defined as $k\Pi_i S_i$ where the arguments to the rate law are the substrates (S_i) and a constant k. Michaelis-menten is defined as $(k1*M1*S_1)/(J1+S_1)$, where both $k1$ and $J1$ are constants. Local is defined by the user in the equation column and may contain any algebraic expression that uses species given in the reaction column, any constant/modifier defined in the *Modifiers and Constants* column, any function defined by the user in the current model, or any predefined function. Any variable that is not defined in the model will be regarded as a constant/modifier and will appear in the modifiers and constants column for this reaction. Locals are useful if the equation is likely to not be used again within a model, as it avoids the definition of a new rate law for a every reaction.

The *Equation* column is not editable by the user unless the row is a Local rate law. Otherwise, the column will display the expression derived from substituting the values for the constants, modifiers, and species into the rate law given in the *Type* column.

The *Modifiers and Constants* column lists the modifiers and constants that exist for the given rate law. Where the user has specified the values of the modifier or constant to be an argument to the rate law, it is shown on the right hand side of the "=" sign next to the name of the rate law argument. To specify the values for the rate law arguments, the user can click on this column, which will display a window for editing the values.

Figure 2. Modifier/Constant editor

Figure 2 shows the editor specifying that the constant *Kf* has value *kw*. The user could have specified an algebraic expression. Any variable used in the expression that is not defined in the model will be regarded as a constant and will appear in the *Constants* spreadsheet.

3.2.2. Rate law row

A rate law (Figure 3) specifies the velocity of a chemical reaction through specification of the new rate law's name and the associated equation for the rate of the reaction. Once a new rate law is defined, it will become available within the current model for use in other reaction rows.

The *Reaction* column has no meaning in a rate law. It

#	Reaction	Name	Type	Equation	Modifiers and Constants
1		degradation	New	S1*k1*(E1+k2*E2)	
2		catalyzed	New	S1*k*E1	

Figure 3. Rate law rows

must remain empty. The rate law name column can be any character string. Other rate laws may not duplicate a rate law name.

The user must choose *New* to be the type of the *Type* column. This specifies that the row defines a new rate law to be used in this model.

The *Equation* column specifies the algebraic equation defining the rate law. Substrates and Products for the rate law are specified as Si or Pi respectively, where i is the order in which the species appears as a substrate or product. The first substrate is specified as S1 and the first product is specified as P1. User defined functions and predefined functions may be used in this column. Any variable other than a Si or Pi is treated as an argument to the rate law, and will be shown in the *Modifiers and Constants* column for any reaction using this rate law.

The modifiers and constants column has no meaning in a rate law and is not editable by the user.

3.2.3. Function row

A function row specifies an algebraic function that takes a list of arguments and returns a value.

#	Reaction	Name	Type	Equation	Modifiers and Constants
1		BB	Function	A2-A1+A3*A2+A4*A1	

Figure 4. Function row

The reaction column has no meaning in a function (Figure 4). It must remain empty. The name column specifies the function name. A function name may not be duplicated by other functions or species equations or species in a reaction. The user must choose *Function* to be the type for the *Type* column.

The *Equation* column specifies the algebraic equation defining the function. Arguments for the function are named as Ai, where i is the order in which the argument appears in the list of arguments to the function. User defined functions and pre-defined functions may be used in this column. Any variable other than Ai is treated as a constant or modifier, and will be shown in the *Constants and Modifiers* column for this function. This column lists the modifiers and constants that exist for the given equation and follows the same rules as in the *Reaction* row.

3.2.4. Species equation row

A species equation row specifies the equation for a species that does not appear as a substrate or product in any reaction. The name in the reaction column is the name of the species. This name may also represent an intermediate variable for use in computation. The name column has no meaning in a species equation. The user must choose *Species* to be the type for the *Type* column. The equation column is editable by the user as in the case of the Local rate law for a *Reaction* row. The *Modifiers and Constants* column lists the modifiers and constants that exist for the given equation and follows the same rules as in the *Reaction* row.

3.3. Constants spreadsheet

The constants spreadsheet (Figure 5) contains all the variables that have not been recognized as species and who need their values to be specified. Users do not add constants on this spreadsheet; they are generated automatically from the *Model* spreadsheet.

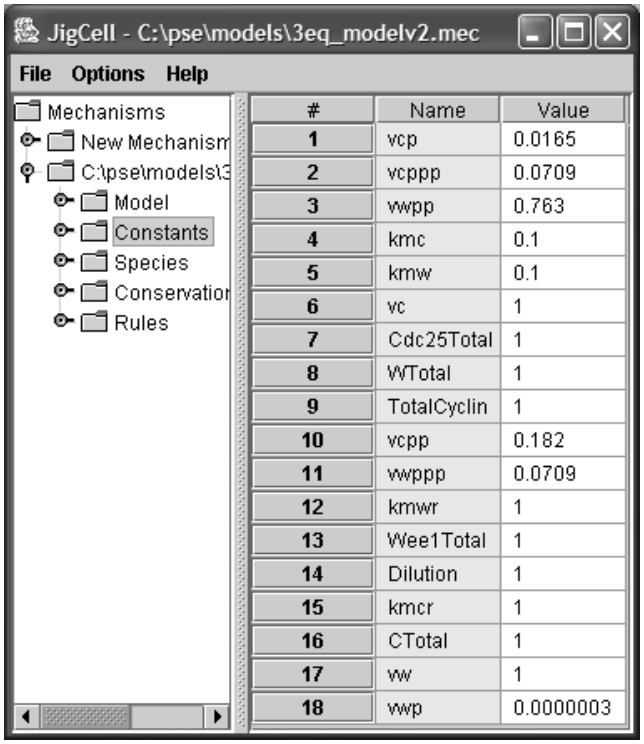


Figure 5. Constants spreadsheet for Figure 1

3.4. Species spreadsheet

The species spreadsheet (Figure 6) contains all the species from the reaction rows specified in the Model spreadsheet. It allows the user to specify the initial conditions for the species.

3.5. Conservation relation spreadsheet

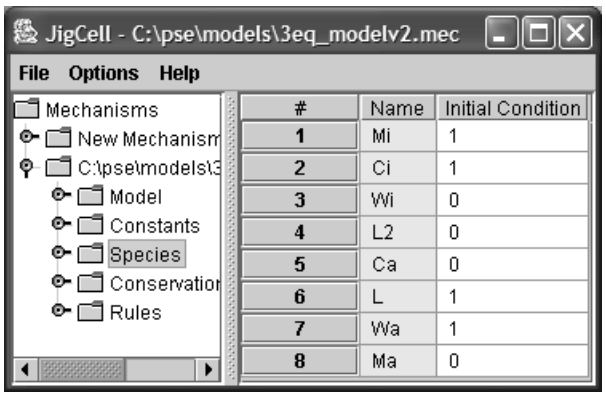


Figure 6. Species spreadsheet for Figure 1

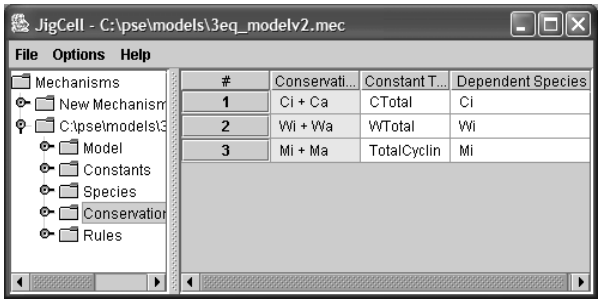


Figure 7. Conservation relations spreadsheet for Figure 1

Conservation relations (Figure 7) appear as a separate node of the tree view (Figure 1). The *Conservation Relation* column shows the various conservation equations that exist in the model and is filled in automatically by the model builder using Reder's method [8-9]. The C*onstant Total Name* column is editable by the user and is for giving a name to the constant total for the conservation relation. This name will appear as a constant in the *Constants* spreadsheet.

The *Dependent Species* column is editable by the user and must contain the name of a species from this relation that is to be treated as dependent. This means that JCMB will not generate a differential equation for this species, and will instead use a linear combination of other species to generate its concentration. JCMB automatically chooses one of the species to be dependent by default.

3.6. Rules spreadsheet

Rules (Figure 8) appear as a separate node of the tree view (Figure 1). They are conditions that when met, trigger certain user-defined actions to occur. The *Name* column can be any character string. The name has no meaning except as a cue to the user for what rule is contained in the row.

The *Sign* column represents the directional crossing of the boolean axis for a condition to trigger an action. "+1" means that the condition has become true after being false. "0" means that the condition has either become true or has become false. "-1" means that the condition has become false after being true. Users can also enter any constant expression, if the expression evaluates to be greater than 0 it is true, otherwise it is false.

#	Name	Sign	Condition	Actions
1	Mass	-1	CLB2-KEZ	MASS=f*MASS; ORI=0; BUD=0; SPN=0; Ite1I

Figure 8. Rule row

The *Condition* column may contain any combination of algebraic expressions and boolean operators that evaluate to a boolean value.

The *Actions* column specifies a list of species and constants that are to be changed, along with their new values, when the desired condition has been met. The user enters these actions in the following example format:

species=a+b+c ; species2=k9 + f(c)

The list uses semi-colons as a separator, and an equal sign must follow the species name. Any variable on the right hand side of the equals will be treated as a constant if its name is not a species or constant previously defined.

3.7. Output Formats

JCMB currently has full support for several output formats, including: xpp (used by the simulation engines XPPAUT and WinPP); Fortran90 files for use with LSODAR; and a tab delimited text file that serves as a report for what has been entered into the model.

4. JCMB experimental study

JCMB has been evaluated in a recent study [12] to determine its effectiveness in reducing errors in converting network diagrams to differential equations and to classify the types of errors made in the use of JCMB.

To address the concerns listed above, a threefold set of experiments were designed. Five computational cell biology researchers at Virginia Tech were recruited for the study. The participants had varying experience levels with symbolic cellular modeling in general. Some had a very small amount of experience with the JCMB tool, and one had a background in computer science.

Participants were given two diagrammatic cellular models of medium-difficult complexity, called model A and model B (Figures 9,10). The first two segments of the study were completed in tandem. Participants were given one model and asked to fully represent it as differential equations by hand. Observers watched for critical incidents and mistakes, noting each that was seen. Participants were

asked to "think out loud" while working. After participants completed the first model, they were interviewed about their experiences. Of particular interest were critical incidents, signs of confusion, and moments when subjects realized they had been proceeding in an inappropriate direction. If mistakes had been made, subjects were asked to try to find them. They were asked about their normal debugging strategies when in similar situations. Together researchers and subjects determined whether the debugging strategies suggested would have been effective or successful in locating the particular bug.

After the first model and follow-up questioning were completed, participants were given a second model to represent symbolically, this time using the JCMB environment rather than traditional pencil and paper methods, and the follow-up questioning procedure was repeated. To ensure that observed patterns of errors were related to the method, rather than the model, some subjects used paper and pencil with Model A and JCMB with Model B, while others completed Model B by hand and Model A with JCMB.

After both models and follow-ups were complete, subjects were interviewed with regard to their likes, dislikes, experiences, and frustrations when using JCMB. What aspects of the JCMB environment made their task (whether construction or debugging) more difficult? What aspects removed them from their central task? Problems which suddenly make apparent the fact that a user is programming, rather than building a cell model (for example) are likely to distract the user's concentration enough that errors are introduced into the process.

4.1. Experimental results

Results were organized on a per-subject basis into four error categories: typographical, omission, incomplete, and computational. Typographical errors were simply a typo or a copying mistake. Omission errors stem from cases where an equation should have been written for a species, but the entire equation was left out (or the species was effectively ignored in the model). Incomplete errors would fail to identify all of the reactions governing the species' behavior. Computational errors generally involve either incorrect mathematical representation of a term in a differential equation or evidence that the diagrammatic model was incompletely understood.

4.2. JCMB versus pencil and paper

JCMB is substantially different from traditional paper and pencil modeling methods on a conceptual

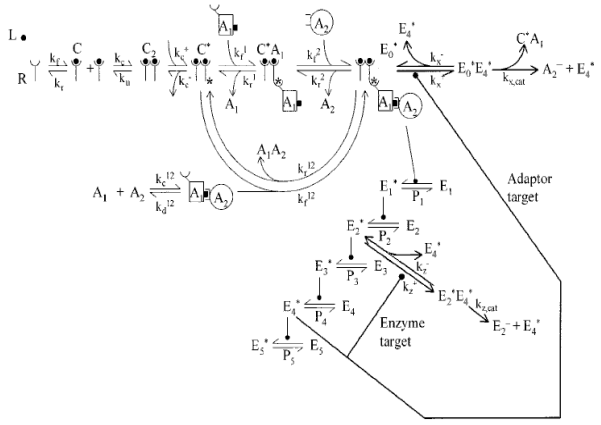


Figure 9. Model A [1]

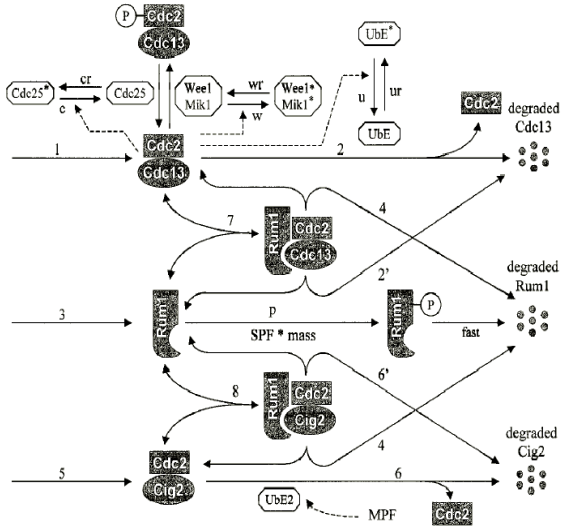


Figure 10. Model B [7]

level. Normal modeling with paper and pencil is protein-centric, meaning that each differential equation derived from the diagrammatic model represents the interactions affecting a single protein. JCMB, conversely, is reaction-centric, meaning that each formula/equation (or line of code) entered into the JCMB environment represents a single reaction. The reaction-centric paradigm is much less removed conceptually from the diagrammatic model, which should make the translation process from diagram to JCMB easier than more traditional methods. It was interesting to note instances when subjects were suddenly distracted by other people or events in the offices. Invariably, recovery times were much shorter when subjects used the JCMB model, regardless of the model being coded. Utterances of, "Now where was I…" (with the characteristic pause that follows) were very brief when JCMB was used, but could last up to several minutes when the subject was working

only with pencil and paper. This might simply reflect a false sense of confidence when the user is relying on a computer to do the hard work.

In both methods, subjects caught the majority of their own errors during the process of building the models. Many errors were minor and involved typographical errors. Occasionally, errors were introduced when the subject copied an item (a reaction rate constant, for example) from the wrong part of the diagram into the symbolic model. Other times, component proteins of an equation/formula were omitted. Sometimes this was traceable to an incorrect understanding of the diagrammatical notation, but most often, these instances represented what would generally be termed "carelessness". Both of these errors (copying errors and omission errors) could often be explained by the fact that the subject's strategy for traversing the diagram, in an effort to avoid duplicating or omitting sections, did not lend itself well to the diagram's layout. While the layout problems are unfortunate, they do represent real-world situations quite well. Diagram traversal strategies were varied and included top-down left-to-right, top-down right-to-left, counter-clockwise, and cluster-based.

No subject completed a paper and pencil model translation without mistakes. The subject who made the fewest mistakes with paper and pencil (generating mistakes only one-sixth as often as the "average" participant) used a model traversal strategy whereby they physically crossed off sections of the diagram as they completed them, to ensure that every part of the model was addressed and that no section was revisited. The subject used this strategy both with paper-pencil and JCMB phases. Ironically, even though this subject used their strategy successfully, they used it inconsistently. When questioned about this strategy after completing the JigCell model, the subject realized that some of their characteristic marks were missing. At this point they nervously rechecked a substantial part of their work only to realize that it was already correct. This may suggest that if support for a similar strategy is incorporated into JigCell, it may not need to be rigorously enforced in order to be effective.

4.3. Paper and pencil mistakes

A breakdown of the overall mistake rate and the relative frequency of each type of mistake is shown in Table 1. The most common errors made when a subject used paper and pencil to generate differential equations were incomplete equations. In these cases, the subject would begin an equation for the appropriate species, but would fail to identify all of the reactions governing the species' behavior. Each reaction manifests itself as a term in the equation. This error generally seemed to be caused by distraction or an incomplete search of the diagram model. The fact that a complete search is required demonstrates the fact that the differential equation model is too far removed from the diagram model. The diagram model represents the simplest, high-level means of understanding the model. Conversion to the very different mathematical model is bound to be error prone.

Unfortunately, mistakes such as this are not easily preventable. This problem can manifest itself in several ways, which could be as simple as reversing the sign (+/-) on a term, or complex enough to demonstrate perhaps that the math itself was not understood.

Table 1. Breakdown of observed mistakes

	Paper/Pencil	JCMB
Mistake Rate	48.43%	4.67%
Computation Errors	30.88%	0.00%
Incomplete Equation	37.80%	8.33%
Typographical Errors	8.31%	83.33%
Omission Errors	23.01%	8.33%
Omitted Equations	15.77%	7.14%
Total Errors	51	8

Omission errors are almost always attributed to incomplete traversal of the diagram model. The errors (both conversion and omission types) cannot be completely prevented by any automated system because they generally lead to valid models (even if not the intended model).

Typographical errors are different in that an automated system can identify possible (or even probable) points where a mistake has been made. This is the smallest portion of overall errors for the paper and pencil method, but are the most directly preventable.

4.4. JCMB mistakes

Since JCMB is a reaction-centric, it alleviates many of the problems that were observed in paper and pencil experiments. This is an indirect solution. By more closely mimicking the easily understood diagram model, less effort is required of the user to generate a symbolic model. This reduced effort pays off in a greater ability to notice errors as they are generated. Simply put, the task of error detection (on the part of the user) becomes one of matching (diagram to JCMB) rather than computation (requiring searches, mathematical double-checking, etc.).

The overall mistake rate in JCMB (4.67%) was approximately one-tenth of that observed with paper and pencil modeling set of experiments. This rate represents the average over all subjects of the total number of mistakes divided by total number of equations. Percentages for types of mistakes dropped significantly for every error type

except typographical. Furthermore, a single subject, a novice, was responsible for all non-typographical errors generated using JCMB, suggesting that in most cases, these errors are negligible. At the very least, JCMB significantly addresses these problems, although there is certainly room for improvements to the JCMB system. It is worth remembering here that none of the subjects had substantial previous experience with JCMB.

The large increase in percent of typographical errors for JCMB is due to the fact that the number of occurrences of every other type of error was substantially reduced. In fact, examination of the raw experiment data shows that the same number of typographical errors (six) occurred in both phases. While this is a small number, it does demonstrate that problems related to typographical errors are either insufficiently addressed by JCMB or are not addressed at all. In other words, while JCMB helps users to more easily convert a diagram model to a symbolic notation, and also helps users to notice and correct mistakes before the symbolic model is committed, JCMB does not prevent the most basic of mistakes: a simple slip of the mind or the finger. Redesign of the system has the potential to address problems of this type.

The fact that other errors, while significantly reduced, continue to confound some efforts is reason enough to believe that the JCMB system still can be improved. The reduction in mental effort, which affords closer attention to the modeling task, is certainly responsible for at least some of JCMB's success in error reduction. If steps can be taken to further increase attention to the task, rather than to the system, these errors might be virtually eliminated. Suggestions for such improvements were obtained in interviews with subjects (see Section 6).

5. JCMB in practice

Currently, the JigCell Model Builder is being used as part of the JigCell problem solving environment for modeling the eukaryotic cell cycle. Models of the cell cycle in fission yeast and frog eggs have currently been entered using the JCMB. Figure 1 represents a current working version of a frog egg extract model in JCMB entered by Jason Zwolak with the corresponding diagram appearing in Figure 11. Figure 12 represents an unpublished budding yeast model in JCMB by Andrea Ciliberto.

JCMB may also be used to model more than just the cell cycle. Any intra-cellular regulatory network may be modeled. This is best shown in Model A of the experimental study of JCMB. As part of the DARPA BIOSPICE project we hope to facilitate use of JCMB by many different modelers.

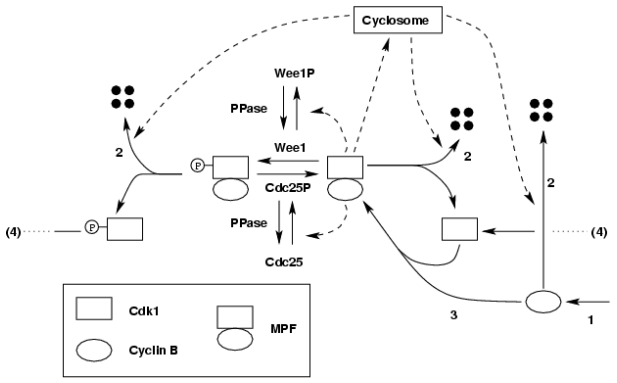


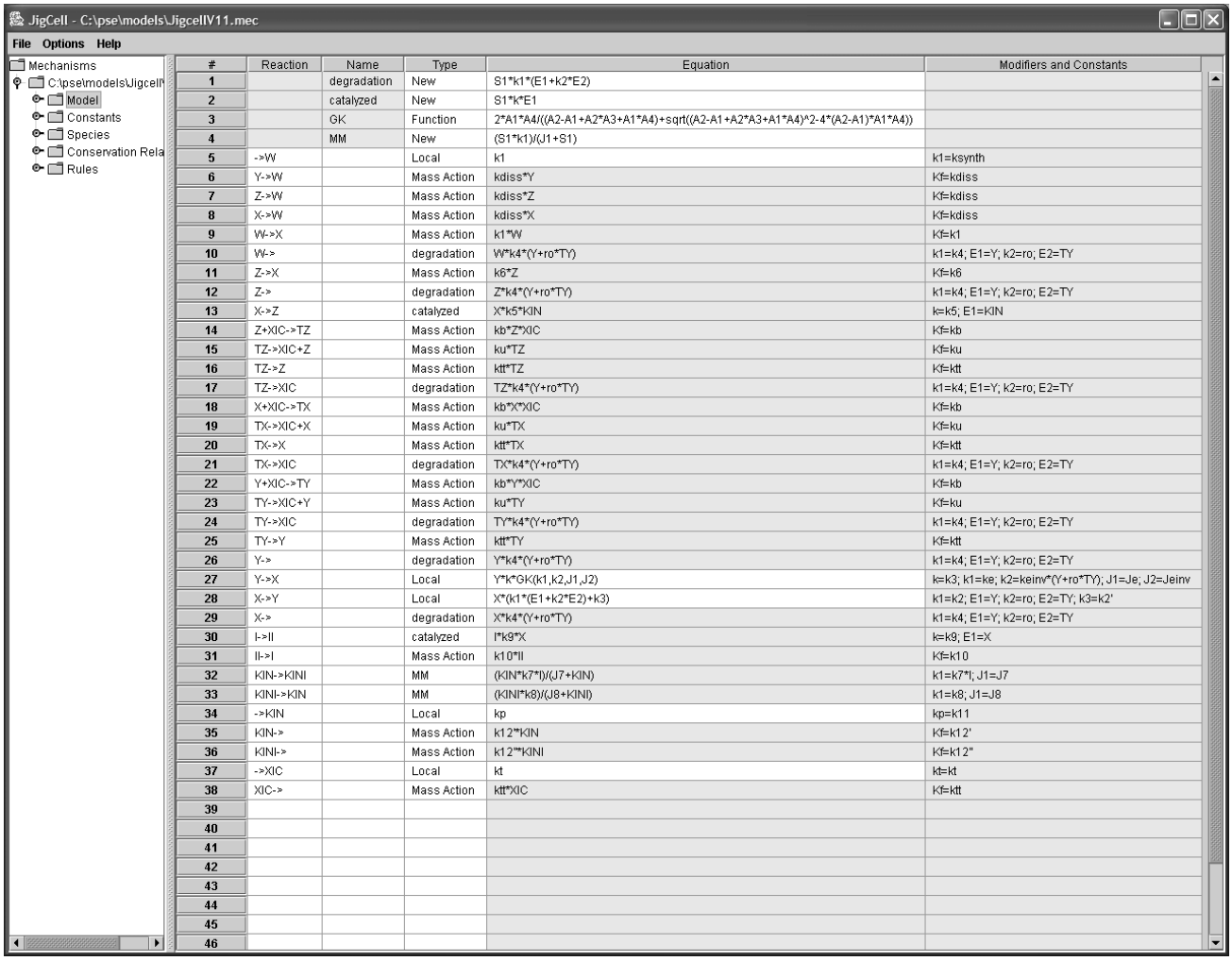
Figure 11. Diagram of cell cycle in frog eggs

6. Discussion and future work

Two areas for future expansion of JCMB include support for modularization, commenting, and annotation. Participants made attempts at both in this study, but explicit support for these needs were not incorporated into the tool. Such support would afford end-users the ability to export the responsibility for keeping track of details and organization to the computer, and allow the user to free up mental resources and memory for problem solving.

JCMB also needs a comprehensive copy and paste facility. This would aid users in reorganizing sections of their model, which would in turn support modularization. This facet would also prove useful for reproducing similar sub-structures within the model. The fact that most subjects in this study attempted to do complex copy-paste operations suggests that users have made two assumptions, and that JCMB would benefit from taking advantage of these assumptions. First, JCMB's spreadsheet metaphor, though not explicitly stated is understood. JCMB is laid out in a manner similar to spreadsheets, and the study participants invariably first attempted to copy and paste using the same gestures accepted by normal spreadsheets. Second, users have become so accustomed to having copy-paste functionality available in other applications that they complain when it is missing, whether or not it was promised.

Repeated comments from study subjects compel the addition of a protein name-matching feature to the JCMB system. Users constantly commented on their extreme caution in following their own naming conventions, and their fear of either referring to a single species with two different names, or using the same name to denote two distinct proteins. Either mistake has the potential to change the model computationally with far-reaching effects. A simple feature to highlight matching names as a species is entered, identify when a new (unmatched) species name is entered, and find matching species on user demand would unload the user's concentration on programming-related concerns and reduce the number of typographical errors –



#	Reaction	Name	Type	Equation	Modifiers and Constants
1		degradation	New	S1*k1*(E1+k2*E2)	
2		catalyzed	New	S1*k*E1	
3		GK	Function	2*A1*A4/((A2-A1+A2*A3+A1*A4)+sqrt((A2-A1+A2*A3+A1*A4)^2-4*(A2-A1)*A1*A4))	
4		MM	New	(S1*k1)/(J1+S1)	
5	->W		Local	k1	k1=ksynth
6	Y->W		Mass Action	kdiss*Y	Kf=kdiss
7	Z->W		Mass Action	kdiss*Z	Kf=kdiss
8	X->W		Mass Action	kdiss*X	Kf=kdiss
9	W->X		Mass Action	k1*W	Kf=k1
10	W->		degradation	W*k4*(Y+ro*TY)	k1=k4; E1=Y; k2=ro; E2=TY
11	Z->X		Mass Action	k6*Z	Kf=k6
12	Z->		degradation	Z*k4*(Y+ro*TY)	k1=k4; E1=Y; k2=ro; E2=TY
13	X->Z		catalyzed	X*k5*KIN	k=k5; E1=KIN
14	Z+XIC->TZ		Mass Action	kb*Z*XIC	Kf=kb
15	TZ->XIC+Z		Mass Action	ku*TZ	Kf=ku
16	TZ->Z		Mass Action	ktt*TZ	Kf=ktt
17	TZ->XIC		degradation	TZ*k4*(Y+ro*TY)	k1=k4; E1=Y; k2=ro; E2=TY
18	X+XIC->TX		Mass Action	kb*X*XIC	Kf=kb
19	TX->XIC+X		Mass Action	ku*TX	Kf=ku
20	TX->X		Mass Action	ktt*TX	Kf=ktt
21	TX->XIC		degradation	TX*k4*(Y+ro*TY)	k1=k4; E1=Y; k2=ro; E2=TY
22	Y+XIC->TY		Mass Action	kb*Y*XIC	Kf=kb
23	TY->XIC+Y		Mass Action	ku*TY	Kf=ku
24	TY->XIC		degradation	TY*k4*(Y+ro*TY)	k1=k4; E1=Y; k2=ro; E2=TY
25	TY->Y		Mass Action	ktt*TY	Kf=ktt
26	Y->		degradation	Y*k4*(Y+ro*TY)	k1=k4; E1=Y; k2=ro; E2=TY
27	Y->X		Local	Y*k*GK(k1,k2,J1,J2)	k=k3; k1=ke; k2=keinv*(Y+ro*TY); J1=Je; J2=Jeinv
28	X->Y		Local	X*(k1*(E1+k2*E2)+k3)	k1=k2; E1=Y; k2=ro; E2=TY; k3=k2'
29	X->		degradation	X*k4*(Y+ro*TY)	k1=k4; E1=Y; k2=ro; E2=TY
30	I->II		catalyzed	I*k9*X	k=k9; E1=X
31	II->I		Mass Action	k10*II	Kf=k10
32	KIN->KINI		MM	(KIN*k7*I)/(J7+KIN)	k1=k7*I; J1=J7
33	KINI->KIN		MM	(KINI*k8)/(J8+KINI)	k1=k8; J1=J8
34	->KIN		Local	kp	kp=k11
35	KIN->		Mass Action	k12*KIN	Kf=k12'
36	KINI->		Mass Action	k12"*KINI	Kf=k12"
37	->XIC		Local	kt	kt=kt
38	XIC->		Mass Action	ktt*XIC	Kf=ktt
39					
40					
41					
42					
43					
44					
45					
46					

Figure 12. Budding yeast model in JCMB

the only ones not currently addressed by JCMB. The unloaded effort, again, would allow the user to more carefully work on the model transformation process, thereby reducing other potential errors. A selectable list of previously entered species would also alleviate this problem, and could be easily equipped with graphic capabilities. The addition of graphic capabilities would necessitate that textual representation continue to operate "under the hood," but the concept could still be beneficial to the user. Some published diagrams use pictures, rather than names, to represent proteins [1]. Allowing the user to specify a species with a quick sketch could conceivably improve the ease with which mapping from diagram to reaction model is achieved.

Incorporating of a simple drawing tool, although not a debugging tool by precise definitions, could help users with the diagram traversal problems. If a diagram could be imported, and the user allowed to arbitrarily mark up the diagram, a simple scheme for preventing partial-formula errors (or omission errors) could be gained. This tool could be logistically separate from the JigCell system. As our study suggests, such a scheme may not have to be enforced to be successful.

Two problems may be addressable by Programming-By-Example (PBE) conventions. The first involves repeated substructures in the diagram model. In such cases, similar reactions, involving similarly named proteins, appear in several places in the model. These repetitions are generally recognized quite quickly [2]. Users in this study attempted to copy and paste these sections, rather than retyping them. Temporary errors (later identified and solved by the participants) were introduced as users made the small changes required from copy to copy. Such repetitive tasks can be automated. When the participants began a complex copy-paste-adjust process, they were no longer transforming the model – they were doing pattern-based replacement. The fact that their focus was removed from their primary goal (model transformation) accounted for the errors generated in this process. If the computer can assume the more repetitive, mundane copy-paste-adjust

responsibilities, the user can maintain focus on modeling and error detection. One user even remarked, while engaging in this process, “This is crazy. The computer should see what I’m obviously doing here and do it for me.”

Another avenue for PBE intervention involves reaction reversals. Many (if not most) reactions in a model have a symmetric opposite reaction that also appears in the model. Users became frustrated by having to type not only the reaction, but also the counter reaction, when conceptually they seemed to view the two as parts of a whole, which could be inferred rather than explicitly written. Some expressed the wish that they could simply tell JCMB to reverse a reaction equation and have the resulting formula added to the next line. This would save time and effort, and allow the user to concentrate on more challenging aspects of the model.

7. Conclusion

The JigCell Model Builder offers both error-reduction benefits, potential for further error-prevention schemes, and a usage model close to the user’s mental model. It has been shown to be highly useful by model builders in practice and will continue to be an integral part of the JigCell problem solving environment for the eukaryotic cell cycle. Further extensions will enhance the ability of modelers to express their models more accurately and effectively.

8. Acknowledgments

We would like to acknowledge the assistance of John Tyson’s Computational Cell Biology Lab in the development of JCMB. This work was supported in part by NSF Grant No. MCB-0083315, as well as National Institutes of Health Grant 1 R01 GM64339-01.

The work reported herein was partly sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL), Airforce Materiel command, USAF, under agreement number F30602-02-0572. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, AFRL, or the U.S. Government.

9. References

[1]Asthargiri, A. and Lauffenburger, D., "A computational study of feedback effects on signal dynamics in a mitogen-activated protein kinase (MAPK) pathway model", Biotechnol. Prog. 17, 2001, pp. 227-239.

[2]Koffka, K., Principles of Gestalt Psychology, Harcourt-Brace, New York, 1935.

[3]Loew, L. M. and Schaff, J. C., "The Virtual Cell: A software environment for computational cell biology", Trends in Biotechnology 19, 2001, pp. 401-406.

[4]Mendes, P., "GEPASI: A software package for modeling the dynamics, steady states and control of biochemical and other systems", Comput. Applic. Biosci. 9, 1993, pp. 563-571.

[5]Mendes, P., "Biochemistry by numbers: simulation of biochemical pathways with Gepasi 3", Trends Biochem. Sci. 22, 1997, 361-363.

[6]Mendes, P. and Kell, D.B., "Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation", Bioinformatics 15, 1998, pp. 869-883.

[7]Novák, B. and Tyson, J.J., "Modeling the control of DNA replication in fission yeast", Proc. Natl. Acad. Sci. USA 94, Aug. 1997, pp. 9157-9162.

[8]Reder, C., "Metabolic control theory: a structural approach", J. Theoret. Biol. 145, 1988, pp. 175-201.

[9]Sauro, H.M., Small, J.R., and Fell, D.A.,"Metabolic control and its analysis. Extensions to the theory and matrix method", Eur. J. Biochem. 175, 1987, pp. 216-221.

[10]Sauro, H.M., “Jarnac: Systems Biology Software”, http://www.cds.caltech.edu/~hsauro/Jarnac.htm.

[11]Schaff, J., Loew, L., "The Virtual Cell", Pacific Symposium on Biocomputing, 4, 1999, pp. 228-239.

[12]Vass, M. and Schoenhoff, P., “Error Detection Support in a Cellular Modeling End-User Programming Environment”, Submitted for review to HCC '02 IEEE Symposium on Empirical Studies of Programmers.