Use of Kinetikit and GENESIS for modeling signaling pathways

By UPINDER S. BHALLA

National Centre for Biological Sciences,
GKVK Campus
Bangalore 560065
INDIA

Phone: 080-856-1663 or 1658
Fax : 080-856-1662
email: bhalla@ncbs.res.in

Table of contents

Use of Kinetikit and GENESIS for modeling signaling pathways

By UPINDER S. BHALLA

Introduction

Computer modeling is a way of doing experiments *in silico*, to bridge the gap between precise theories of simplified systems and the complexity of experimental biology. Modeling harnesses computer power to scale theoretical calculations up from equations describing individual reactions, to more life-like systems involving thousands of interacting molecules. The start and end-point of this process is experiment: from data to predictions to further data. The ideal of the modeling process is to gain insights into system function along the way.

Modeling has tended to be the preserve of the mathematician or computer scientist, but its greatest value is in the hands of the experimentalist. Modern simulation systems seek to reach their users in two ways: first, by providing a simple graphical interface suitable for non-programmers, and second, by tying closely into advances in web-based databases to tap into better data. There are currently several systems which strive toward this goal in the field of signaling pathways and genetic networks. These include Vcell[1], MCell[2], Dbsolve[3] , Jarnac[4], GENESIS/Kinetikit[5], and others.

There are several levels of detail one can consider in models of signaling pathways. In increasing order of complexity and realism, these are:

1. Logical/Boolean models of signaling and genetic cascades, which treat activity as either on or off and regards signaling interactions as boolean operations.

2. The 'well-stirred cell' approach, which assumes perfect diffusional access to all components, and models signaling pathways in terms of chemical kinetics.

3. The diffusional approach, which treats everything in terms of reaction-diffusion systems.

4. The Monte-Carlo approach, which considers the stochastic interactions and movements of individual molecules.

The limiting factor in all these approaches is data. Computational requirements also increase steeply with increasing realism. The classical test-tube biochemical experiment remains the staple data source for most models. It is still unusual for sufficient experimental information to be available to justify diffusional or stochastic models, although it is clear that such factors are indeed important in biology.
This chapter is based on the 'well-stirred' approach, with compartmentalization as a slight elaboration. We will use G-protein coupled receptor signaling through the cAMP pathway as an example. We will work through the process of determining reaction mechanisms and parameters, incorporating them into a functioning model, and assessing its behavior. In developing a research simulation, these stages would occur at a very fine grain: the user might add a single reaction at a time, test out the model for various parameters, and then study which behaves most realistically. Here we will go through the simulation process in the same sequence, but at a much coarser level since we must describe an entire signaling pathway. It may be helpful to build the model in tandem with reading the chapter, and skip back and forth through the stages described in the chapter to get a better feel for the overall modeling process.

REACTION MECHANISMS IN SIGNALING PATHWAYS.

Data sources

Block diagrams are the basal level information required to set up a model of a signaling pathway. If all else fails, one can essentially transcribe a block diagram into equivalent reactions and try to fit these to the kinetic data as described below. Information of the form "A is an upstream activator of B" comes from many sources, typically from tissue-culture, genetic and molecular biology methods. More detailed mechanistic data rely on

pharmacology and on test-tube biochemistry. Such data can provide details even beyond what a typical model might require[6].

The generic block diagram for a signaling pathway specifies inputs, outputs and regulators for constituent pathways (Figure 1 A). The starting point for devising a model is to expand the block diagram to precisely specify it in terms of individual reactions. The guiding rule for this is to choose the simplest set of reactions that embody the known interactions. As is clear from the expanded version of the block diagram into a precise reaction scheme, there is an alarming proliferation of molecules and reactions even in this highly simplified system (Figure 1 B-D). This complexity is the central problem in developing realistic models of signaling pathways. Kinetikit is therefore designed specifically to provide a friendly interface for managing complex reactions and large amounts of data.

We will use the example of G-protein mediated signal transduction to describe how to create and connect up the reaction and enzyme steps. Later sections will consider parameterization, running models, and analysis. In actual simulation development these steps take place iteratively and at a much finer grain. One would typically add a reaction at a time and test its effects, then go on to another reaction, and so on.

Software and hardware

GENESIS and Kinetikit are freely available from a number of Internet sites as well as on the CD-ROM available as part of the GENESIS reference textbook[5]. At the time of writing, the current versions are GENESIS 2.2 and Kinetikit 7. The examples in this chapter refer to Kinetikit 7, but earlier versions are functionally equivalent except for the unit conversions. Kinetikit 7 will run on GENESIS 2.1 as well. The model script files used in the examples in this chapter are available at the same GENESIS sites[5].

The computational demands of kinetic models are small, and a modern PC can run all but the largest models hundreds of times faster than real time. GENESIS runs on a wide range of UNIX machines, including PCs running the freely available Unix clone, Linux.

Setting up reaction mechanisms for simulation.

To illustrate, we will start off with our example of receptor-G protein signalling (Figure 1). Conversion of the block diagram into reactions should proceed in a modular manner. Modules are typically well-defined signaling molecules or pathways, each individually constrained by experimental data. In this pathway, there are experiments describing ligand-receptor interactions[7],[8] and the closely-coupled receptor-G protein interactions[9]. These collectively describe the receptor-G-protein interaction nicely, and it is a clear conceptual unit. The next item in the block diagram is Adenylyl Cyclase (AC), which is also clearly defined in terms of its rates and interactions[10]. The block diagram can thus be reduced to a set of modules which can each be individually tackled to build up a library of signaling components. In our example, we consider receptor/G-protein, AC, and Protein Kinase A (PKA) as distinct modules. In addition to the basics of setting up the reaction scheme, each module illustrates modeling design decisions and potential pitfalls. The integration of multiple modules is described in a later section.

Insert Figure 1 here.

Receptor/G-protein module

It is relatively easy to describe receptor-ligand binding as a simple reversible reaction. The coupling of the receptor to the heterotrimeric G-protein is more complex: does the coupling happen before ligand binding or after ? It turns out that both reactions are observed, and in fact the reaction details are quite complex[9]. This example confines itself to a loop of reactions as in Figure 1 B. Care must be exercised to ensure that the product of Keq around the loop is unity, since there is no net free energy change around the loop. Further steps of this module are dissociation of the heterotrimer, binding of GTP, the hydrolysis of GTP, and the re-association to form the GDP-bound trimer. Many design decisions arise at this stage. In this example we choose to ignore GTP metabolism, as we can treat it fairly accurately as a well-buffered pool. We also simplify the details of the

process of GTP-GDP exchange and release of GTP.Gα. Another simplification pertains to the activation of the downstream pathway (in this case AC). This is at first sight straightforward: it only involves binding of activated Gs to AC to obtain an active Gs.AC complex. The obvious enzyme activity of the complex is cyclization of ATP to form cAMP, as described below. The other, often ignored activity is the hydrolysis of GTP by the G-protein. Since AC levels are very low it may be reasonable to ignore this. On the other hand, recent results show that several G-protein activated molecules also act as GTPase-activating proteins (GAPs) which enhance the rate of hydrolysis of GTP by the bound G-protein[11]. This apparently simple, and easily ignored step, turns out to be potentially quite important for signal flow.

Adenylyl cyclase module

In designing this module we again can choose to ignore the metabolism of ATP. In our example the cyclase is treated as a two-state enzyme: it is either off, or in the Gs-bound form, it is on. One could add in further regulatory interactions (Gi, phosphorylation, CaM binding etc) depending on the AC subtype. Indeed, if we wished to go into details for subtype interactions we could consider some 15 or so isoforms of AC[10]. This is a modeling decision which requires judicious evaluation of likely interactions and the details of the specific system being modeled.

There are at least two ways of treating isoforms: the explicit approach is to model each individual AC variant separately and treat its interactions as distinct. This is more accurate but more tedious, and the proliferation of similar reaction steps increases the need for care in constructing the model. For many purposes it is adequate to lump various isoforms together and treat them as an 'averaged molecule'. Fortunately for this example, we are only interested in the linear pathway of activation of AC by G-proteins, and can ignore the rest.

Protein kinase A module

PKA is one of the best constrained enzymes in the signaling literature[6]. It is fairly straightforward to construct the reaction sequence based on these experiments, but here we encounter a different design decision: should we include all the available experimental data

? For example, it is known that the sharpness of PKA activation is increased by the presence of an inhibitor molecule, which stoichiometrically blocks the kinase till all the inhibitor is bound. Should this interaction be included ? Here we have chosen to do so, at the cost of introducing a further free parameter into the model.

Regulatory and housekeeping functions

We have already bypassed several sources of complexity in the model by treating highly regulated molecules such as GTP and ATP as buffered to a fixed value. A general principle which we cannot so easily bypass is that every enzyme should have a counteracting enzyme. In the case of the cyclase, we must provide a phosphodiesterase for removing the cAMP. This mundane task assumes signaling importance because the PDE itself can be regulated and further activated upon phosphorylation by PKA. (There are other PDEs which we will not discuss here.) Similarly, PKA will need a counteracting protein phosphatase to work upon its substrates. These balancing or housekeeping molecules can, of course, be modeled in their full glory as highly regulated enzymes. A simpler approach which can often be justified is to treat them as a fixed-rate back-reaction. In our example mechanism we have done both: modeled the PDE as an enzyme which itself is regulated by PKA, and modeled the phosphatase as a simple irreversible back-reaction.

Insert Figure 2 here.

Molecules

All reactions are set up in Kinetikit by clicking and dragging objects from the menu bar into the edit window (Figure 2). The creation of an object (a pool, reaction, enzyme or other simulation module) is accompanied by the appearance of a dialog box for editing the object. The dialog box allows complete specification of object parameters including name, kinetic parameters, and display attributes such as color. At this stage of dealing with reaction mechanisms we need only change the name from the default. For this example, we would create distinct molecular pools by dragging the pool icon from the menu bar into

the edit window and assign their names (table 1). For now we can ignore the concentration terms, which are discussed later.

Insert Table 1 here.

Reactions

Reaction objects are also created by dragging reaction arrow icons from the menu bar into the edit window. The bare reaction object is represented by a reversible-arrow symbol, and only two parameters are needed to specify it: the forward and backward rates (Table 2). The reaction arrow symbol only defines a reaction object, but does not set up its substrates or products. This is because Kinetikit maintains a distinction between the objects (pools, reactions, etc) and their interactions. Once the substrate pools, reactions, and products have been set up, the reaction can be completed by dragging the substrate pool onto the reaction symbol and then dragging the reaction symbol onto the product pool. In our example, to complete the binding of ligand to receptor we must first drag the L icon to the L-bind-R icon (a green arrow appears) and then drag the L-bind-R icon onto the R icon (another green arrow appears). The assignment of reaction rates is described in a later section.

Insert Table 2 here.

Enzymes

Enzyme objects in Kinetikit should be regarded as enzyme activities of pre-existing molecules. They can only be created on existing molecular pools. This is done by dragging the enzyme icon from the menu bar onto an already defined molecular pool in the edit window. Any number of enzyme activities can be set up on a single pool. For example, a single protein kinase might have a large number of substrates, each with different kinetics. Each of the substrates would therefore interact with a different enzyme activity object. This makes it possible to specify different substrates with different reaction rates. It does not, of course, imply that the actual structure of the enzyme contains multiple sites. As with the

reaction objects, the interactions between substrates and products of the enzyme are set up by click-and-drag operations within the edit window. To set up the enzymatic reaction, the substrate is dragged onto the enzyme activity, and the enzyme site onto the product. Enzyme rate assignment is discussed in a later section. The only enzymes we have to consider in this example are AC, PKA and cAMP-PDE (Table 3).

Insert Table 3 here.

Groups

A useful organizational feature of Kinetikit is that closely coupled sets of reactions and molecules can be placed on 'groups'. Each signaling module is typically represented by a distinct group. A group is created in the same way as any other simulation object, by dragging it from the menu window to the reaction edit window. Assignment of reactions and pools to a group is done by dragging them onto the group icon. Once grouped, the objects in a signaling module can be saved, displayed, moved, deleted and duplicated as a unit. In our example there are three groups: for the receptor/Gs, for AC and for PKA.

Modeling regulation.

In modeling terms there are two principal approaches to describing enzyme regulation. The first (illustrated here by AC) is to regard the enzyme as either on or off. In this situation the only enzyme site required is on the activated form of the enzyme. Activation level here is determined purely by the amount of the enzyme in the activated state. The second approach is to consider all possible forms of activation and represent each by an individual enzyme site with different activity. For example, PDE has a basal activity, but can also be stimulated further by phosphorylation. An enzyme with complex regulation might need several such enzyme sites. If the enzyme has multiple substrates the number of potential sites grows rather rapidly.

There are different philosophies regarding reaction regulation. Kinetikit is built around the assumption that a given molecular species has a fixed interaction rate with any other

molecule. Regulation is therefore regarded as the formation of a different molecule which has its own interaction rate. This is perhaps more correct, but does lead to a proliferation of reaction sites as mentioned above. A different and commonly used approach is to vary the reaction rates themselves. Kinetikit permits this, but grudgingly. This and other rarely used options are available under the 'Options' menu item. In some situations (e.g., voltage gated channels) this formalism is more appropriate.

Correcting errors

Objects can be renamed and their parameters changed at any time. The objects themselves can be removed (deleted) by dragging them onto the dinosaur icon in the menu bar. Interactions between objects obviously vanish when one of the objects is deleted. Interactions can also be removed by repeating the click-and-drag that created the interaction.

PARAMETERS.

Parameter specification is the most difficult part of biological modeling. Signaling pathway models can draw upon a large body of experimental work, particularly on the biochemistry of signaling molecules, so the bench work will often have been taken care of. A careful reading of the methods and conditions and a good deal of informed judgment are required to extract useful parameters from typical papers. Most published kinetic data are not in a form that can directly be plugged into a model. It is necessary therefore work backwards by simulating the experimental conditions, comparing results to the measurement, and adjusting parameters till they match. In this section we will examine common strategies for setting parameters for different kinds of signaling reactions.

Volume and concentration units

GENESIS internally models all reactions in terms of numbers of interacting molecules. This approach is independent of local volume and localization issues, but has the drawback that the units are unfamiliar. Kinetikit therefore provides conversions to commonly used units. These are selected through the **units** menu item. All reaction pools are initialized with the volume specified here, which defaults to 1e-15 m^3, corresponding to a cube of 10 micron sides or a sphere of 12.79 microns diameter. These defaults can be set to other values if desired. Concentration and time units default to μM (micromolar) and seconds respectively. We will retain all these defaults for our example.

Setting pool concentrations

The pool parameter dialog can now be filled in for each of the molecules in the simulation. Note that it provides its own local volume which can differ from the default setting. The pool parameter window displays concentrations both in terms of numbers of molecules, and also in terms of the selected concentration units. The most important fields in the parameter window are current concentration (**Co**) and initial concentration (**CoInit**). The counterparts of these values in terms of numbers of molecules are **n** and **nInit**. **Co** and **n** are calculated by the simulator, and are not normally set by the user. Most pools in the model are formed as intermediates in the signaling pathway, and should have a **CoInit** of zero. Only a few key reactants will need to have this value assigned. For example, the concentration of the unbound receptor, the unstimulated AC, and the GDP-bound Gαβγ trimer should be set up to known total values. Provided that the key reactants have the correct total levels, and the rates are reasonable, the system will settle to a reasonable steady state. Metabolites such as ATP which are tightly regulated can be 'buffered' to a known level using the **Buffering** toggle. This is a facility which should be used with care, since it implies an infinite source (or sink) of the molecule. Enzymes are an interesting case, since their concentrations in vivo are usually estimated by successive purifications and activity assays. Thus an impure enzyme will overestimate the concentration but

underestimate the activity in proportion. In modeling terms, the total cellular enzyme activity may be correct even if the levels are not. Unfortunately the regulation of the enzyme is likely to depend on its exact levels. In our example, the rate of binding of Gs to AC depends on exact enzyme levels, and would be affected by such an error.

Setting reaction kinetics

By far the most common experimental approach to quantitation of reaction parameters is the concentration-effect curve. This, in conjunction with time-course information, is enough to tightly constrain most binding reactions. If one assumes a perfect match to the theoretical reaction

$$A + B \underset{kb}{\overset{kf}{<====>}} C \qquad (1)$$

the half-max of the curve define the Kd for binding. Kd is related to the rate constants by

$$Kd = kb/kf. \qquad (2)$$

Since the time-course is approximately

$$tau \sim= 1/(kb + kf) \text{ (at fixed B)} \qquad (3)$$

we could in principle solve for kf and kb. In practice, the binding curve is usually more complex and the time-course not as well defined. Here the calculated values of kf and kb could be used as a starting point for modeling the reaction, and progressive refinement could be done using the model itself to match the observed curves. This process of refinement is also useful when the experimental input is several steps removed from the

measured response. In our example, we might have data on the formation of cAMP as a function of ligand concentration. If we can approximate some of the steps as linear responses, the remainder can be constrained by the results. In this system we might be able to treat cAMP response as proportional to the number of free Gs molecules, and use this to improve our estimates for rates in the G-protein module.

Some care is required in working out units, since this depends on the order of reaction. The **kf** and **kb** dialogs use units of numbers of molecules and seconds, and do not require scaling when additional substrates or products are added to the reaction. **Kf** and **Kb** are somewhat friendlier and use the concentration and time units specified by the user. These do, however, depend on the order of reaction, and will be rescaled if this is changed. **Kd** is also calculated in terms of user-specified units. This may need some care in relating to the experimental value if one or more reactants is assumed fixed. **Tau** is in user-specified time units, and is really only applicable for first order reactions.

Setting Enzyme constants.

Enzyme interactions in Kinetikit follow the classical Michaelis-Menten irreversible scheme:

$$\text{Sub} + \text{Enz} \underset{k2}{\overset{k1}{\Longleftrightarrow}} \text{Sub.Enz} \xrightarrow{k3} \text{Prd} + \text{Enz} \qquad (4)$$

The enzyme object is therefore really just a shortcut way of implementing two reactions and a molecular pool (the Substrate.Enzyme complex). Enzyme reactions require three parameters: k1, k2 and k3. Most experiments report the two parameters for Michaelis-Menten enzymes, that is, Km and Vmax. These are related as follows:

$$Km = (k2 + k3)/k1 \qquad (5)$$

$$Vmax = k3 \text{ (allowing for enzyme levels).} \qquad (6)$$

These two Michaelis-Menten parameters do not fully constrain the requisite three rate parameters. The third parameter may be fixed by assuming a fixed ratio of k2 to k3. It turns out that most models are rather insensitive to this ratio. Then,

$$k3 = Vmax \quad (7)$$

$$k2 = ratio * Vmax \quad (8)$$

$$k1 = (1 + ratio) * Vmax / Km. \quad (9)$$

Based on the fact that typical enzymes do not accumulate much enzyme-substrate complex, a ratio of 4 is usually satisfactory. Either of these alternative representations of enzyme rates can be used in the enzyme dialog to set up rates.

Further constraints.

One benefit of using detailed reaction schemes for representing biochemical reactions is that experimental manipulations can usually be quite faithfully replicated. This proves very useful in constraining parameters based on indirect data. For example, it is common to have data for steady-state levels of various reactants but no direct rate information. In the example there were useful numbers of this kind for numbers of GTP.Gα complexes following stimulation[12]. This was used to improve estimates for rates of binding of the L.R complex to the GDP.Gαβγ complex. A simulation of the reaction system could predict reactant levels for given reaction rates. By iteration one can work backwards to obtain rates which fit the observed levels.

Parameters vs. Mechanisms.

In any reasonably complex signaling network, one is likely to find situations where the obvious reaction mechanisms do not appear to yield the observed responses. At some point parameter adjustment must give way to a reformulation of the reaction scheme. This is usually a tedious operation, as all reaction rates may need recalculation. Nevertheless, it is

critical to the iterative process of model development. It closes the cycle involving specification of rate, concentration and mechanistic details. This process is likely to involve many cycles: 50 versions of models for a single pathway are common.

MODELING.

We finally discuss the actual running of the model, though this will normally have begun as soon as there were a couple of pools and reactions to simulate. The process of modeling includes running, displaying, saving, and other aspects of controlling the simulation.

Running simulations

There are three appropriately coloured buttons just under the menu bar: **Start**, **Stop** and **Reset** (Figure 2). **Start** runs the simulation. **Stop** halts the simulation cleanly while it is running, so that it can be resumed by another click of the Start button. **Reset** re-initialises the simulation and sets the time back to zero, losing any information from the previous run. The user specifies the desired duration of the simulation in the **Runtime** dialog. The current simulation time is continually updated in the **Current Time** dialog.

Displaying simulation output

The current level of any pool can be checked by double-clicking on the pool. This will pop up the pool parameter window. The current pool level (**n** as well as **Co**) updates during the course of the simulation.

It is usually more useful to display the levels of several pools at a time using plots. Plots, as the name implies, generate time-course plots of reactant levels. These are set up by clicking on any reactant in the edit window, and dragging it to a graph window. This causes the appearance of the name of the plot in the graph window. Plotting will not start until the **Reset** button has been clicked. The plot name can be dragged to the **Delete** icon in the menu window to delete it. Double-clicking of the plot name opens a plot editor window offering various self-explanatory options. One of the most important is the **Save to file**

option, which dumps all the data points in the plot into a text file with time and concentration value pairs on each line. Several global plot options are available in the **Graphs** menu item. These include the **Overlay** facilty for overlaying new runs on previous plots for comparison.

Plots of long simulations can consume lots of memory. The number of data points is equal to simulation duration divided by time-step for the plot. Therefore it is advisable to use a fairly long time-step for the plots. One normally only needs to update the plot every second or so, unless the reactant is varying exceptionally rapidly. This is controlled by setting the plot timestep in the **Options** menu.

Timesteps and accuracy

GENESIS/Kinetikit uses the exponential Euler method, which is well suited to handling differential equations which exhibit an exponential time-course[13]. Kinetic equations are of this form. The method is explicit, so it requires some care in selecting integration timesteps to ensure accuracy. In general, a timestep 10 times faster than the fastest rates in the system should be quite accurate. The timestep is specified in the **Options** menu. A quick way of selecting a timestep is to successively halve timesteps and compare output plots for convergence. The process is then repeated with longer timesteps till one attains a good balance between accuracy and speed. Given that typical reaction parameters have fairly large error bars, a numerical accuracy of 1% should be quite adequate.

Many systems have rapid initial kinetics followed by much slower, prolonged responses. Furthermore, stimulus delivery in the simulation can be instantaneous, which is difficult to handle accurately. A simple solution to this is to set a very fine timestep for initial transients, and then switch over to a slower timestep for the smoother part of the simulation. This feature can be set up in the **Options** menu. The initial timesteps can be selected using the same convergence criterion described above. This procedure is a crude but effective version of more complex variable timestep methods.

Volumes and units

All computations in the simulator actually use numbers of interacting particles. This makes it possible to specify different volumes for pools (for example, in distinct cellular compartmentals) without incurring scaling errors. Therefore, all concentration terms are actually scaled by the volume for the purposes of calculation.

The default volume for reactions is 10e-15 m^3, or a 10 micron cube. A diameter for a sphere can also be used. These terms can be set in the **Units** menu item. All subsequently created pools will be set to this volume upon creation. The volume can be edited as required within the pool edit box. Note that enzymes sites too have a volume term which applies to the enzyme-substrate complex. It is always set equal to the volume of the parent enzyme molecule. The **Units** menu also provides for altering the units used for display and data entry purposes.

It is instructive to monitor the value of **n** (number of molecules per cell) in the pool parameter window. In small cellular compartments, several pools may exist in single-molecule quantities. The continuous approximation used here breaks down under such situations, and stochastic calculations must be used. This is one of the key areas for future simulator developments.

Given a preferred set of concentration and time units, rate constants are scaled from the more fundamental but less friendly *#/cell/second* units used internally. The dimensions of rate terms depend on the order of the reaction and care is needed to match up with the experimental conditions so that this dimension is handled correctly.

Save, Restore and merging.

One of the most important operations in modeling is frequent saving of the model state, with annotations. The **File** menu option is used to store the complete current state of the model and interface (with the exception of plotted data points). If a signaling module is

encapsulated within a group, it can also be saved as an option in the group parameter window. The data file in both cases is a standard GENESIS script file. This has two implications. First, the simulation can be resumed simply by entering

```
genesis <filename>
```

Second, for more sophisticated users, complex batch jobs using the model can be set up using the GENESIS scripting language and by simply including the simulation file name. If the simulation is included with the option `nox` it will not display the interface, and only load in the numerical part of the model:

```
include <filename> nox
```

Previously stored models can also be opened from the GENESIS command line. If Kinetikit is not yet running, the simulator will automatically load it and then open the model.

```
genesis> include model.g  // loads in model file model.g
```

There is a special situation when Kinetikit is already running with a previously loaded model. In such situations the system will attempt to 'merge' the two models. In doing so it always loads up the most recent model in its entirety. If there is overlap with an existing model, it does not duplicate existing pools but it does change their parameters to that of the most recent model. Similarly, existing linkages between pools, reactions and enzymes are not duplicated. This behavior is designed to allow merging of interacting signaling pathways. In our example, there are three signaling modules which interact as follows. First, the GTP.Ga pool from the Gs module binds to AC to produce the Gs.AC active enzyme. Second, cAMP from the AC module binds to and activates PKA in several steps. Third, PKA itself phosphorylates the PDE from the AC module in a negative feedback loop. These molecules are highlighted in Figure 1 B-D. The merging facility in Kinetikit allows all these interactions to be set up by simply loading in the three modules in sequence. In order to do this, the only requirement is that the signaling molecule involved

in an interaction between two modules must be defined (with the same name) in both modules. In our example, GTP.Ga is produced by the Gs module and is also defined in the AC module as an input molecule. The second messenger cAMP was present as an end-product in the model of AC, and as a regulator of PKA in the PKA model. PKA itself phosphorylated the PDE present in the AC model. The following sequence of commands was used to merge the three modules:

```
genesis> include pka.g
genesis> include ac.g
genesis> include gs.g
```

Note that the parameter values are those of the most recently loaded module. This usually does not matter. In rare cases, there is a shared pool or reaction whose parameter values differ between modules. This (inadvisable) situation can lead to a dependence on order of loading. The graphical layout of objects in the edit window is also determined by the most recently loaded module, and this is often a more practical reason for choosing a particular sequence of module loading.

Miscellaneous features

There are a wide range of advanced features in Kinetikit which are beyond the scope of this chapter, but which are documented in detail in its **Help** menu. A brief road-map of some of the capabilities of the options is provided in table 4.

Insert Table 4 here

INTERPRETING MODELS

Detail

One of the hardest aspects of modeling is deciding how much detail to include. Even in this small example, we have had to take many decisions about simplifying and excluding

interactions. These decisions determine the kinds of interpretation we can put on model results. For example, our model explicitly considers the association of G-protein subunits. If we were now to include a second Gα which bound the same βγ subunits, the simulation would calculate the rate at which free βγ subunits bound to the first or second Gα. Thus we would be justified in using our model to interpret βγ mediated cross-talk between the two Gα subtypes.

A different aspect of detail, which again we have encountered, is that every molecule seems to interact with several others. In our example, we chose to stop the reaction cascade at PKA, just at the point where we would have had to deal with half a dozen different substrates. Where does one draw the line in terms of scope of the model ? One answer is to explicitly model every interaction that is part of a measurement which the model seeks to replicate. In the simulator we can readily monitor PKA activity by plotting the concentration of its active form. In an experiment, we would have had to put in a test substrate. It would therefore be more accurate to model this substrate explicitly as well. By doing so the model itself would incorporate the effects of enzyme saturation, etc, and the output would be directly equivalent to the experimental measurement of phosphorylation. Similar issues of scope crop up when considering isoforms. In our model we have ignored the ten or so G-protein and AC isoforms, let alone the hundreds of G-protein coupled receptors. There is a happy medium level of detail where we have enough information to reasonably represent the system but not so much that interpretation becomes difficult.

Reliability

The error bars in model output are at least as large as those of the model parameters. We can, however, be fairly confident that they are not much larger if the model is able to quantitatively predict experimental outcomes. In our example, we could assess the reliability of most of the model by examining the time-course of cAMP stimulation by agonist. This is not a result which the model has been fine-tuned to predict, so a reasonable match to experiment means that our model mechanism and parameters are probably

reasonable as well. Good experimental papers frequently overconstrain a model. That is, they conduct more than the minimal number of experiments required to specify the model parameters. If this is the case then the model can be set up based on the first few experiments, and tested for prediction of the remainder.

Robustness

Biological systems are usually robust. Signaling pathways reflect this by usually having a wide operating range over which their quantitative behavior may change moderately, but their qualitative behavior does not. In modeling terms, one should be able to vary the rates, concentrations etc. of the model over a range comparable to the experimental error, without affecting the basic model properties. A factor of 2 above and below the basal level is common. A single 'fragile' parameter might be interpreted as a a sensitive regulator of the system. Several such parameters usually indicate that the model mechanisms need reworking. An analysis of robustness (or parameter sensitivity analysis) proceeds by varying the selected parameters in turn and measuring a key response of the system as a whole. In the GENESIS/Kinetikit context, this is usually done by writing a GENESIS script program which loads in the model, systematically adjusts the selected parameters, runs the simulation and monitors the response. In the cAMP signaling example, we might choose to vary such parameters as receptor affinity, GTP exchange rates, levels of key molecules and so on. The obvious response to monitor would be PKA activation as a function of ligand concentration.

Complexity

How does one interpret results from a model which itself is too complex to intuitively comprehend ? The modeling process itself frequently gives rise to insights at the abstract level, well above the tedious details of parameter specification. The process of decomposing the example block diagrams into detailed reaction schemes will probably

have suggested to the modeler that certain reactions could be sensitive points of regulation. GTP exchange and hydrolysis might be such points of regulation in the cAMP pathway.

A more concrete application of modeling is as a counterpart to experiment. In our example, we could ask mechanistic questions: does the receptor bind the G-protein before or after the ligand ? The model would predict quantitatively different responses, which could be tested. We could also consider quantitative issues: What would happen if AC increased the GTPase activity of $G\alpha_s$? In each case the model encapsulates all the complex detail of reaction mechanisms and rates, and outputs a simple experimentally verifiable quantity.

Emergent properties.

One of the most important functions of a model is to predict something that the user does not expect. There are unlikely to be many surprises in a pathway as well-established as cAMP signaling. Despite this, it would be an instructive exercise to work out whether the modeled pathway could generate oscillations or bistable behavior. On theoretical grounds this might occur if the PKA phosphorylated the receptor and inhibited or activated it, respectively. This behavior would be emergent in the sense that it would be almost impossible to predict the actual outcome without doing an experiment, whether *in vivo* or *in silico*. It would also be a response of a qualitatively different (and more interesting) kind than the conventional linear flow of signaling information.

The above process of setting up a model is really designed to make sure it behaves as expected in situations which are experimentally well understood. It follows that the best way of getting interesting predictions from the model is to try it out in unknown territory: complex interactions, novel signaling contexts, or simply in systems where the role of the pathway is uncertain. All of these situations are likely to be true as experiments and models scale up to describe biology more completely. At this point the model truly goes from descriptive to predictive..

Concluding remarks

There are many levels of analysis of cellular signaling, but the starting point remains the definition of reactions, rates, and concentrations. Biology poses many interesting problems at finer levels of detail such as cytoskeletal or single-molecule interactions. Nevertheless there is an immediate flood of data at the bulk, non-diffusive level of detail we have considered. Modeling in the age of proteomics will have to contend with single proteomics experiments which can provide a thousand data-points, and genomics assays which yield a hundred times that. Kinetikit and this chapter represent an early evolutionary step towards systems which combine the scalability of databases and the predictive and interpretive capacity of models.

Acknowledgments

This work was supported by the National Centre for Biological Sciences.

Figure legends

Figure 1: cAMP signaling pathway example. A: Block diagram. B: Reaction details for Receptor/G-protein module. C: Reaction details for AC/PDE module. D: Reaction details for PKA module. In B-D some molecules may be represented more than once for illustrative reasons, but they are modeled as a single pool. The molecules highlighted within a box are common to more than one module and are used when merging modules to form the composite simulation.

Figure 2: Kinetikit interface layout. The top row of buttons are the menu options, followed by several buttons for controlling the running of the model. The menu bar with the library of prototypes is below this. The edit window is the large window to the bottom left and is shown displaying the reaction scheme for the Receptor/G-protein module. The graph windows on the upper right show the results of a simulation with step increases in ligand levels. The pool parameter window on the lower right displays parameters for the L.R.GDP.Gabc pool.

TABLE 1

MOLECULAR POOLS

Receptor-G-protein module		AC module		PKA Module	
Name	Initial Concentration (μM)	Name	Initial Concentration (μM)	Name	Initial Concentration (μM)
L	Variable	AC	0.015	R2C2	0.5
R	0.1	Gs.AC	0	cAMP.R2C2	0
GDP.Gabc	1	ATP	5000 (buffered)	cAMP2.R2C2	0
R.GDP.Gabc	0	cAMP	0	cAMP3.R2C2	0
L.R	0	AMP	1000 (buffered)	cAMP4.R2C2	0
L.R.GDP.Gabc	0	cAMP-PDE	0.5	cAMP4.R2C	0
GTP.Ga	0	cAMP-PDE*	0	cAMP4.R2	0
Gbg	0			PKA-active	0
GDP.Ga	0			PKA-inhibitor	0.25
				inhibited-PKA	0

TABLE 2

REACTIONS

Receptor-G-protein module			AC module			PKA Module		
Name	Kf (1/sec/ μM)	Kb (1/sec)	Name	Kf (1/sec/ μM)	Kb (1/sec)	Name	Kf (1/sec/ μM)	Kb (1/sec)
L-bind-R	0.1	0.1	dephosph-PDE	0.1 (1/sec)	0	cAMP-bind-B1	54	33
L-bind-R.Gabc	5	0.1	Gs-bind-AC	500	1	cAMP-bind-B2	54	33
R-bind-Gabc	0.002	0.1				cAMP-bind-A1	75	110
L.R-bind-Gabc	0.1	0.1				cAMP-bind-A2	75	32.5
Activate-Gs	1 (1/sec)	0				Release-C1	60	18
GTPase	0.0667 (1/sec)	0				Release-C2	60	18
Trimerize-Gs	6	0				inhib-PKA	60	1

TABLE 3

ENZYMES

Parent molecule	Symbol in model	Km (μM)	Vmax (1/sec)	Ratio
Gs.AC	cyclase	20	18	4
cAMP-PDE	PDE	19.84	10	4
cAMP-PDE*	PDE*	19.84	20	4
PKA-active	phosph-PDE	7.5	9	4

TABLE 4

KINETIKIT FEATURES

Feature	Location	Function
kpool	Menu bar	Pool of reactants
kreac	Menu bar	Reversible reaction
kenz	Menu bar	Michaelis-Menten Enzyme
stim	Menu bar	Pulse stimuli
group	Menu bar	Hierarchy and organization of models
xtab	Menu bar	Smooth stimuli
kchan	Menu bar	membrane channel
transport	Menu bar	Molecular transport with delay
delete	Menu bar	Delete objects
Duplicate	Menu bar	Duplicate objects
Postscript	Tools menu	Specify postscript options. Control-P in the edit window will generate postscript output according to these settings.
Plot	Tools menu	Options for graphs
Compare models	Tools menu	Compare model parameters
Tabulate models	Tools menu	Generate text table of model parameters
Timestep control	Options menu	Set timesteps for simulation and display.
Reaction options	Options menu	Numerous options for special reaction situations.
Graph axis limits	Graphs menu	Globally set graph axis limits
Show more graphs	Graphs menu	Display additional graph windows
Plot overlay	Graphs menu	Toggle retention of previous plots after reset.
Save all plots	Graphs menu	Save data points from all plots in text file.

Footnotes/References

[1] http://www.nrcam.uchc.edu

[2] http://www.mcell.cnl.salk.edu

[3] http://websites.ntl.com/igor.goryanin/

[4] http://members.tripod.co.uk/sauro/biotech.htm, http://www.fssc.demon.co.uk

[5] U.S. Bhalla, in The Book of GENESIS, Ed. J.M. Bower and D. Beeman, 2nd ed., Springer Verlag, 1998, http://www.bbb.caltech.edu/GENESIS, http://www.ncbs.res.in/downloads

[6] S.O. Døskeland and D. Øgreid, *J. Biol. Chem.* **259(4),** 2291 (1984)

[7] R. S. Kent, A. De Lean and R. J. Lefkowitz, *Mol. Pharmacol.* **17,** 14 (1980)

[8] U. Gether, S. Lin, B. K. Kobilka, *J. Biol. Chem.* **270(47),** 28268 (1995)

[9] P. Samama, S. Cotecchia, T. Costa, and R. J. Lefkowitz, *J. Biol. Chem.* **268(7),** 4625 (1993)

[10] J. P. Pieroni, O. Jacobowitz, J. Chen, R. Iyengar, *Curr. Opin. Neurobiol.* **3,** 345 (1993)

[11] K. Scholich, J.B. Mullenix, C. Wittpoth, H. M. Poppleton, S. C. Pierre, M. A. Lindorfer, J. C. Garrison, and T. B. Patel, *Science* **283,** 1328 (1999).

[12] S. R. Post, R. Hilal-Dandan, K. Urusawa, L. L. Brunton, and P. A. Insel, *Biochem. J.* **311(1),** 75 (1995)

[13] R. J. MacGregor, *Neural and Brain Modeling,* Academic Press, San Diego (1987).

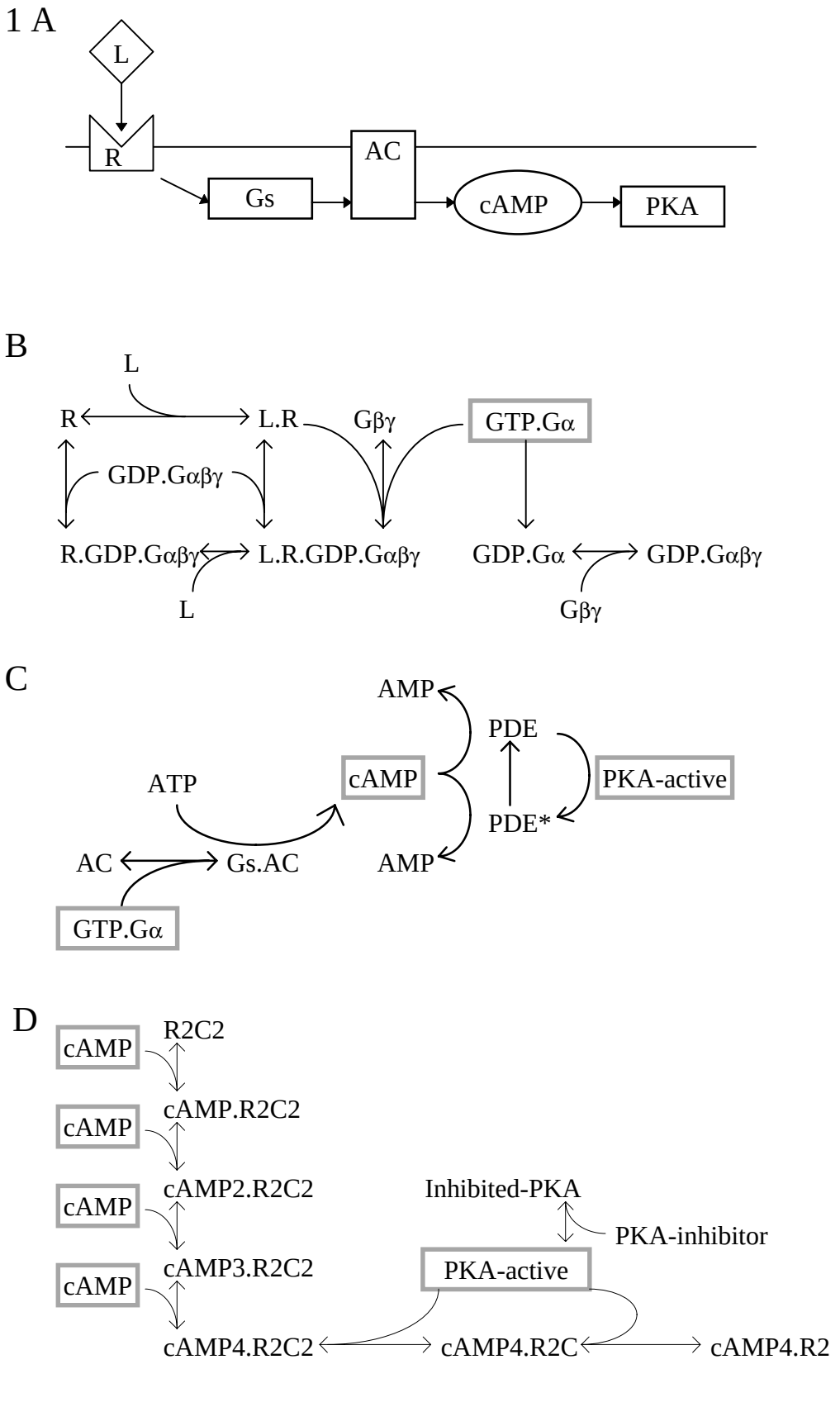
1 A
L
R
AC
Gs
cAMP
PKA
B
L
R
L.R
Gβγ
GTP.Gα
GDP.Gαβγ
R.GDP.Gαβγ
L.R.GDP.Gαβγ
GDP.Gα
GDP.Gαβγ
L
Gβγ
C
AMP
PDE
ATP
cAMP
PKA-active
PDE*
AC
Gs.AC
AMP
GTP.Gα
D
cAMP
R2C2
cAMP
cAMP.R2C2
cAMP
cAMP2.R2C2
Inhibited-PKA
PKA-inhibitor
cAMP
cAMP3.R2C2
PKA-active
cAMP4.R2C2
cAMP4.R2C
cAMP4.R2

Figure 1

2

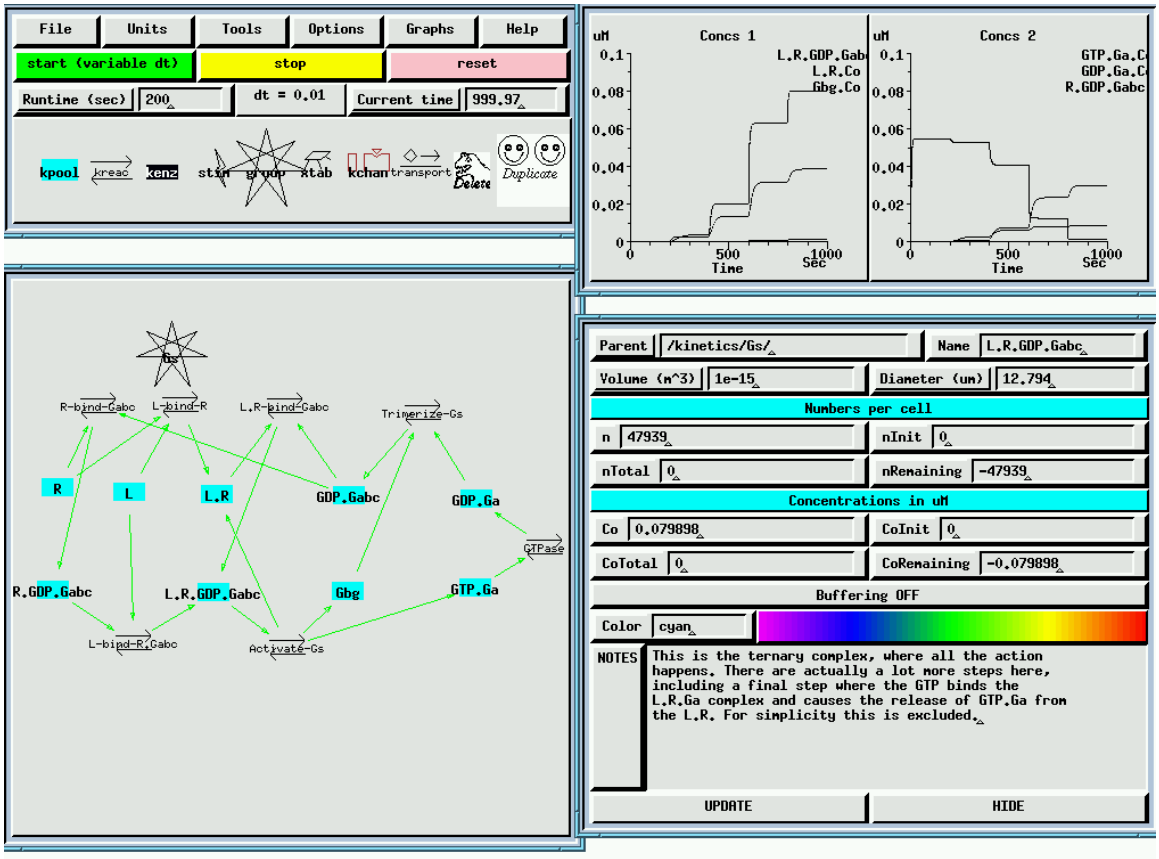
File
Units
Tools
Options
Graphs
Help
start (variable dt)
stop
reset
Runtime (sec)
200
dt = 0.01
Current time
999.97
kpool
kreac
kenz
stim
group
xtab
kchan
transport
Delete
Duplicate
uM
Concs 1
L.R.GDP.Gab
L.R.Co
Gbg.Co
Time
Sec
Concs 2
GTP.Ga.C
GDP.Ga.C
R.GDP.Gabc
Gs
R-bind-Gabc
L-bind-R
L.R-bind-Gabc
Trimerize-Gs
R
L
L.R
GDP.Gabc
GDP.Ga
GTPase
R.GDP.Gabc
L.R.GDP.Gabc
Gbg
GTP.Ga
L-bind-R.Gabc
Activate-Gs
Parent
/kinetics/Gs/
Name
L.R.GDP.Gabc
Volume (m^3)
1e-15
Diameter (um)
12.794
Numbers per cell
n
47939
nInit
0
nTotal
0
nRemaining
-47939
Concentrations in uM
Co
0.079898
CoInit
0
CoTotal
0
CoRemaining
-0.079898
Buffering OFF
Color
cyan
NOTES
This is the ternary complex, where all the action happens. There are actually a lot more steps here, including a final step where the GTP binds the L.R.Ga complex and causes the release of GTP.Ga from the L.R. For simplicity this is excluded.
UPDATE
HIDE

Figure 2.