Simulating Cellular Reaction Networks with Moleculizer 1.0

Larry Lok
The Molecular Sciences Institute
2168 Shattuck Ave.
Berkeley, CA 94704
voice: 510-981-8740
fax: 510-647-0699
email: lok@molsci.org

Roger Brent
The Molecular Sciences Institute
2168 Shattuck Ave.
Berkeley, CA 94704
voice: 510-981-8744
fax: 510-647-0699
email: brent@molsci.org

April 20, 2004
Keywords: stochastic simulation

Abstract

This paper presents *Moleculizer,* an open-source, stochastic simulator for intracellular biochemical systems, with special treatment for protein complexes. In a nutshell, *Moleculizer* starts from descriptions of monomeric proteins and other simple components, along with specifications for binding, unbinding, and other classes of reactions between them. Then it uses a simple scheme to generate new species of complexes and extrapolate new reactions between them, but only as they are called for in the course of the stochastic simulation. Thus it avoids using the network of all possible complexes and reactions involving the given monomeric components, a network which may be catastrophically large. Instead, *Moleculizer* produces a realistic, useful, exportable reaction network consisting of the species and reactions demanded by an actual stochastic simulation run. Even this "minimal" reaction system is frequently a large, confusing network of reactions between closely related complex species when expressed in a typical "flat" species-and-reaction style. By contrast, the specification of the *Moleculizer* simulation that generates the reaction system is manageable and reasonably congruent with biologists' intuition.

1 Introduction

1.1 Simulation

A *dynamical theory* models the behavior of a system over a period of time. A useful dynamical theory is usually predictive in the strong sense of telling how the system that it describes will behave in the future, given the system's state in the present. A *simulation* is a (usually computerized) application of a dynamical theory to some particular system to display the system's behavior over a span of time. A *simulator* is a computer program that does simulation.

1.2 Uses of simulation

Cell biologists can use a simulation to display the behavior of a cellular subsystem, provided the subsystem can be described in terms compatible with the simulation's underlying dynamical physical theory. This description amounts to a hypothesis. Running the simulation amounts to drawing a non-obvious conclusion, the predicted behavior, from the hypothesis and the simulation's underlying physical theory. If the simulation results can be observed experimentally, then the simulation has provided a highly nontrivial link between the hypothesis and an experimental test. Since the computer "does all the work," many different possibilities can be tried out computationally before attempting difficult experimental verification. In effect, the simulator can provide a non-experimental platform for "sanity checking," comparing, and generally playing with multiple hypotheses.

1.3 Simulation in cell biology

Two forms of simulation exploited in cell biology are molecular dynamics [FS96] and simulation of chemical reaction systems. Molecular dynamics displays the behavior of one or a few (usually large, complex) bio-molecules by using the physical theory of inter-atomic interaction and bonding [Pau60].

Chemical reaction system simulation displays the behavior of (usually large) systems of chemical species and reactions between them by means of the dynamical theory of chemical reactions, coming from statistical mechanics. It can describe reaction systems at different levels of detail, depending on what is needed and how much computational effort can be expended. *Moleculizer* incorporates a chemical reaction system simulator. In the next subsection, we classify examples of previous and current efforts to simulate intracelluar chemical networks according to the amount and kind of physical detail that they display.

1.4 Ways of simulating chemical reaction systems

Two properties are useful for describing a chemical reaction system simulator:

1. Whether the simulator represents the amount of each chemical species as a whole number of molecules or as a concentration.

   If the simulator keeps integer counts of all the species, then it can change species' populations only by integer amounts. These integer changes are given by the stoichiometry of the responsible reactions. Such discrete changes must take place at discrete times, as *reaction events.* The times at which reaction events occur are affected by the random motions of the molecules and thus are not deterministic. So, if whole-number molecule counts are maintained in a realistic way, then the simulation is effectively forced to be *stochastic;* that is, to involve modeling random processes.

   For cellular reaction systems, one frequently assumes that the reaction volume is sufficiently small that it is *well-mixed* by diffusion, and in this case, the dynamical theory is summed up in the *chemical master equation* discussed in [Gil92a], [Gil92b], and [Gil76]. This chemical master equation fully describes the temporal part of the stochastic process of reaction events between individual molecules, but because the reactants are assumed well-mixed, the positions of molecules and reaction events are not simulated.

   Experiments showing the importance of precise molecular counts and stochastic effects (noise) in gene expression were reported in [EL$^+$02], while similar effects were treated via simulation in [MA97].

   If the reaction system may be assumed not only to be well mixed, but also sufficiently large in the sense of both volume and the populations of molecular species, then a real-valued approximation that is valid in the "thermodynamic limit" of infinite volume and infinite populations becomes useful: the *mass-action* equation of physical chemistry [DB02, §1.3]. This differential equation describes the way concentrations evolve in the infinite volume, infinite population approximation and was an empirical formula originating in the nineteenth century. It can now be realized as the thermodynamic limit of the more-detailed and theoretically better-grounded chemical master equation [Gil92a, §???].

2. Whether or not the simulator represents the spatial distribution of each chemical species.

   As noted above, if the reaction volume (cell) is small enough relative to the rate of diffusion of each species, then it may be acceptable to approximate the true spatial distribution of each species with a uniform distribution over the reaction volume. If to the contrary the reaction volume is large or spatial inhomogeneities are of great importance, then one must keep track of where species are, and the simulation will be called *spatial.*

   Assuming spatial homogeneity results in a drastic comptutational simplifcation. In the differential equations approach, this assumption means that the simulator solves ordinary differential equations, instead of partial differential equations. In the stochastic approach, the simulation's

state at any time can be summarized by listing all the species and their populations, rather than giving the species and location of every single molecule.

Simulators that model amounts by concentrations and assume that concentrations are uniform are usually general ordinary differential equation solvers applied to the mass-action equation. One well-known implementation with a full set of accessory programs is *Gepasi* [Men94]. This style of simulation has been applied to significant biological problems, in [CCNG+00] and [CAMK02], for example.

Simulators that approximate the amount of each species as a concentration but permit different concentrations at different points in space must solve a partial differential equation that is the manifestation of the mass-action equation in this context. This partial differential equation is sometimes called the *reaction-diffusion equation,* and this type of continuous, spatial simulation is treated in [FW95] and implemented in the *Virtual Cell* simulator [Sch].

Spatial, stochastic simulations are at the highest level of detail. The simulation tracks moleculules' locations and notes the location and time at which each individual reaction event between molecules takes place. Examples of spatial, stochastic simulators are *MCell* [SB00] and *ChemCell* [Pli].

Stochastic simulators that do not model the distribution of molecules in space were the first class of stochastic simulators to be formulated and investigated, mainly through the pioneering work of Gillespie in [Gil76]; see also [Gil92a] and [Gil01]. *Moleculizer* belongs to this class of stochastic, non-spatial simulators and uses a queued form of Gillespie's *first reaction* algorithm [Gil76, p. 419]. For other more efficient and complex algorithmic variants of this method, see [Gib00, chapter 5] and [GB99]. Other non-spatial stochastic simulators include *StochSim* [MF98] and *Stochastirator* [EL]. Some applications of this kind of simulation to gene expression are [MA97] and [MA98, p.215].

To maintain exact counts of all molecular species at all times in the simuation, every single reaction event between molecules must be simulated, and this represents a substantial computational burden. In [Gil01], Gillespie proposed an accelerated, approximate method of simulation in the stochastic domain, subsequently refined in [GP03]. This technique, called *tau-leaping*, moves forward in time in essentially fixed time steps, instead of working through all the individual reaction events that happen during the time steps. The integer changes in species populations from one time step to the next are summaries of the effects of many intervening reaction events of different kinds. The numbers and kinds of reaction events during a "leap" are approximated by using technology for solving ordinary differential equations. This kind of stochastic simulator is called *approximate,* as opposed to an *exact* stochastic simulator, which must model every reaction event.

1.5 Cellular reaction networks can be very large

The most apparent hurdle in setting up a non-spatial reaction-system simulation of cellular chemistry is the proliferation of species and reactions. This proliferation is partly due to the fact that there are many truly different proteins [HPG98] in cells. Moreover, each protein may interact (i.e. participate in mutual reactions) with many other proteins [DIP]. But the databases of proteins and their interactions generally record only simple, monomeric proteins, so they do not show the true vastness of the possible reaction systems that they imply: even when there are only a few simple proteins, biochemical systems frequently involve thousands of species and reactions because of the many ways that simple proteins may combine to form complexes. That relatively astronomical numbers of distinct complexes can be formed from combinations of only a few simple proteins was noted by Morton-Firth [MF98], who termed it a "combinatorial explosion."

These vast combinatorial networks pose two problems. The first is that they are hard for a researcher to understand and specify. The reactions in such networks are not only numerous, but also in a sense, highly redundant: what a biologist might regard as a single reaction, say the dimerization of two simple proteins, may be approximately replicated over and over as dimerizations of protein complexes that contain the two simple proteins and expose the appropriate binding sites. The second problem is that the full combinatorial network of possible species of complexes usually demands a huge amount of computer memory to accommodate it. It is relatively easy to cook up simple artificial examples that exhaust all memory.

1.6 Moleculizer manages the explosion

Moleculizer tries to help the researcher work with large combinatorial reaction networks by reducing the simulation specification to what a biologist normally regards as distinct reactions and by automatically extrapolating the large network of elementary reactions as a by-product of simulation. At the other end of the simulation, displaying and interpreting simulation results from a vast, confusing reaction network is also difficult for the researcher. *Moleculizer* eases some of this burden as well. For example, *Moleculizer* provides simple ways to plot the populations of many related species of complexes all in one trace. A biologist can easily arrange that a single trace on an output plot gives the total population of all those species of complexes that contain a particular simple protein, say *Ras*. The trace would then be "total *Ras*." This sort of specific, multi-species trace is generally more useful from a biological viewpoint than a plot of one single chemical species in the system. By making parallel simplifications in setting up the simulation and displaying its results, *Moleculizer* buffers the reasearcher against the full blast of the explosion of species and reactions.

As for demands on computer memory, *Moleculizer* tames the proliferation of *possible* species by using only the species that are required in the course of simulation. The complex species that actually occur in the simulation are

usually only a small fraction of the possible species of complexes that could be formed from the given simple proteins. The "memory footprint" of a *Moleculizer* job is still sometimes quite large, as real simulations frequently require thousands of reactions and species, but the unrequired species and reactions would be many times more bulky.

2 Reaction network generation

2.1 *Moleculizer's* network generation scheme

Moleculizer generates its expanding reaction network by means of a cyclic process (Fig. 1) that is attached to, but largely independent of the core stochastic simulation machinery that generates reaction events. This fact promises to make it relatively easy to port the reaction network generation technology to stochastic simulators of other kinds. For example, investigators at The Molecular Science Institute are providing technical assistance in the development of the *ChemCell* spatial stochastic simulator at Sandia National Laboratory [Pli].

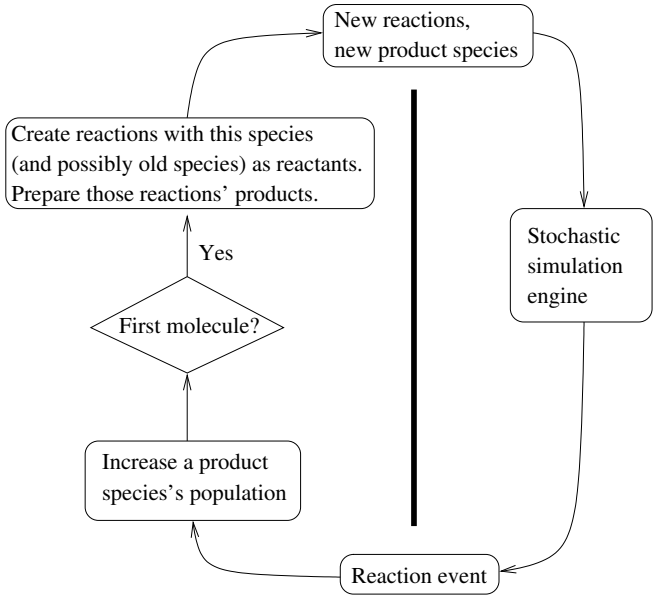

Figure 1: Reaction network generation cycle.

The reaction network generation cycle may best be understood through an example: the generation of a typical family of dimerization reactions and the subsequent generation of their reaction products, illustrated in Fig. 2. Reaction generation is triggered when the first molecule of a species appears in the course of simulation. One way that the triggering molecule may appear is in the initial

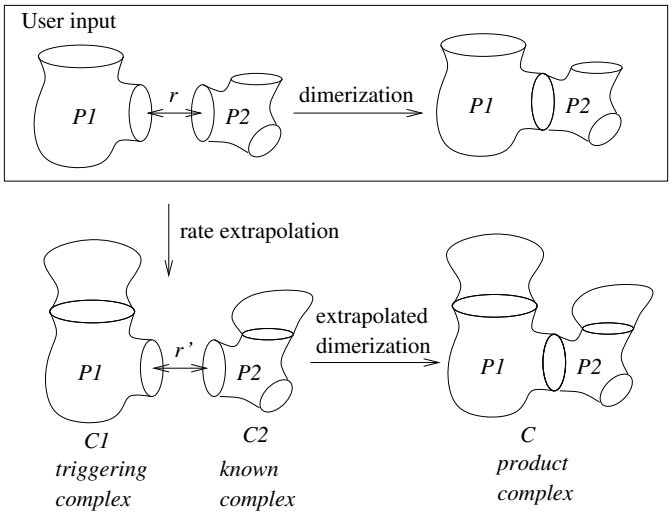

Figure 2: Dimerization example. See §3.2.2 for details of rate extrapolation.

population of the simulation; another way is as a product of some reaction when it occurs for the first time in the simulation. Suppose that the triggering molecule is a complex *C1*, and that this complex contains a simple protein *P1*. Suppose that there is already a known complex *C2* containing another simple protein *P2*. Also, suppose that the user has specified on-rates and off-rates for the two simple proteins *P1* and *P2* at binding sites that are exposed in the complexes *C1* and *C2*. The two main parts of extrapolating the dimerization between *P1* and *P2* to a dimerization between *C1* and *C2* are calculating the reaction rate and preparing the dimerization product species *C*.

The rate r at which the two simple proteins *P1* and *P2* bind together is given in *Moleculizer's* input. This rate is extrapolated to a binding rate r' for the two complexes *C1* and *C2* by making a correction for the larger molecular weights of the complexes, as detailed in §3.2.2. This molecular weight correction ignores many other factors that might change the dimerization rate, particularly geometrical consderations such as obstruction of the binding sites. Improving this method of reaction rate extrapolation is an important next step in *Moleculizer's* development.

"Preparing" the dimerization product species C means making an entry for it in the growing database of all species known to the simulation. This database records the complex species that have been encountered in the simulation up to this time, along with their numbers. A structural description of C is formed from the structures of the dimerizing complexes *C1* and *C2*; the states of the simple proteins making up C are the same as they were in *C1* and *C2*. The database of known complex species is searched for a species matching this description of structure and state. The product species C has many such descriptions,

but the description determines C completely. If the product species C has already been encountered in the simulation, then its entry in the database will be found. If the product species C has not yet appeared in the simulation, then it is entered into the database at this time and assigned population zero. But new reactions having C as a reactant are not created at this time. Rather, the above reaction generation process is triggered when the first *molecule* of C appears, that is, when the population of species C first becomes non-zero. This might happen because the just-constructed dimerization reaction of *C1* and *C2* occurs for the first time. If *Moleculizer* did not temporize in this way, the network of all possible reactions and reactants would be generated all at once at the beginning of the simulation. But by waiting until reactant molecules appear, automatically generated reactions are created at the last instant before they might be demanded by the simulation.

2.2 Other reaction network generation tools

Other interesting efforts at generating reaction networks involving protein complexes, either automatically or with user supervision, are [BL97] and [BF$^+$03]. These are separate tools from the simulator. By contrast, *Moleculizer's* reaction network is generated as a by-product of a simulation and can be dumped at any time(s) during the simulation. The reaction network can then be used in other, perhaps faster, simulation tools.

3 *Moleculizer* models

3.1 Species

3.1.1 Small molecules

Some simple but useful *Moleculizer* species do not participate in reaction network generation. These are called *stoch-species* in honor of *Stochastirator* [EL], a stochastic, non-spatial simulator written by Drew Endy and Eric Lyons at the Molecular Sciences Institute. "Small molecules" such as ATP, GTP, phosphate, etc. are generally modeled as stoch-species in *Moleculizer.* When bound to a protein, such small molecules are modeled as *modifications,* see §3.1.2. Stoch-species are also useful for importing reaction networks from other tools, as only a name and molecular weight need be given to specify a stoch-species.

3.1.2 Complexes

The monomeric constituents of a protein complex are not treated as species in *Moleculizer*. The protein species corresponding to one of these monomeric constituents is modeled as the complex containing just the one constituent. This avoids there being two ways of representing the simple protein species. Simple protein constituents of complexes are called *mols* in *Moleculizer's* input.

The most common kind of mol is a *mod-mol,* which can be modified (phosphorylated, methylated, sumoylated, etc.) at many sites. The modification sites on a mod-mol are named by the user when the mod-mol is defined. The modifications themselves (phosphorylated, sumoylated, methylated, etc.) are described in a special section of input, where the user gives their names and the changes in molecular weight associated with them. For two complexes built out of mod-mols to be the same species, both of the following conditions must hold:

1. The complexes must have the same combinatorial structure. That is, they must consist of the same mod-mols bound together in the same way.

2. Corresponding mod-mols must have the same modifications.

For future extensibility, each complex species retains references to complete abstract state information (modification state in the case of a mod-mol) about all of its simple monomeric constituents. In general, complexes are of the same species when they are combinatorially the same and their corresponding mols are in the same state. The state descriptions of future kinds of mols, such as nucleic acids, are not limited *a priori* in any way. This prepares the way for new kinds of simple constituents of complex species.

Naturally, reactions can change the states of monomeric constituents of their reactant complexes. For example, a phosphatase may dephosphorylate one member of a protein complex at one specified site. Perhaps more interesting is how molecular state is treated in reactions, such as dimerizations and decompositions, that simply recombine the molecular constituents of their reactant complexes. The general rule in such reactions is that the states of the simple constituents remain unchanged: the product complexes of the reaction inherit the states of their constituents from the reactant complexes.

3.1.3 Influence of molecular state on reaction rate, allostery

Simple proteins in *Moleculizer* bind to other simple proteins at binding sites. Each binding site on a simple protein can assume a number of different *shapes* that affect on- and off-rates. These "shapes" are really just symbolic tags, bookkeeping conveniences like "obstructed-shape" or "GTP-bound-shape," and do not carry any geometric or chemical information at this time. But the shapes of the binding sites do affect reaction rates: the specification of a dimerization can give different rates for different combinations of binding site shapes, for example. The dependency of a binding site's shape on its molecular environment is referred to in *Moleculizer* as *allostery.*

The shape of a binding site can be affected by the state of the simple protein where it is found. A binding site on a mod-mol (§3.1.2) can be given a special shape for any combination of modifications, for example.

The shape of a binding site can also be affected by the larger molecular environment of the simple protein that bears the site. The shape of any binding sites in a complex, either bound or unbound, can be set by an allostery specification for the whole complex.

Particular sub-complexes can be specified in the simulation input and detected in complex species as they appear. The shapes of binding sites appearing in these sub-complexes can be set by an allostery specification for the sub-complex.

Since these influences on binding site shape can interact (interfere) with one another, the order in which they are applied to binding sites in a complex is significant: First, allostery specifications for simple protein that bears the binding site change the site's shape depending on the simple protein's state are applied. Second, allostery specifications for particular sub-complexes that contain the site are applied in arbitrary order. Finally, allostery specifications for the whole complex are applied.

3.2 Reactions

3.2.1 Explicit reactions

While most of the reactions in a *Moleculizer* simulation are usually automatically generated, the user may include any desired reactions in explicit stoichiometric form. The only prerequisite to specifying reactions in this way is that the reactant and product species must be described and named. *Moleculizer's* input provides the means to describe and name particular species of complex molecules, including the states of the monomeric constituents. Stoch-species (§3.1.1), which are always named and need no further description aside from their molecular weight, can be mixed freely with complex species in these reactions. Molecular weights do not enter into the operation of an explicit reaction; the rate is given explicitly in the reaction's specification. Hence, reaction networks from simulation tools that do not understand complexes can be imported as explicit reactions between stoch-species, whose molecular weights can be arbitrarily, harmlessly assigned if necessary.

3.2.2 Reaction rate extrapolation

In the dimerization example, the rate of binding between the two simple proteins *P1* and *P2* was extrapolated to give a rate of binding between two complexes containing these simple proteins. In this extrapolation, only the increased mass of the two complexes is taken into account. Needless to say, this is a very crude approximation, especially because it disregards the geometry of the two complexes.

The mass correction for dimerization is done by reference to the following formula

$$c_\mu = V^{-1}\pi d_{12}^2(8kT/\pi m_{12})^{1/2}\exp(-u_\mu^*/kT)$$

from Gillespie's original exposition of the Stochastic Simulation Algorithm in [Gil76, §2, eq. 6] and treated further in [Gil92b]. Basically, this expression relates a binary reaction rate to several physical properties of the two reacting molecules, along with environmental factors such as temperature. We will not

attempt to explain this formula more fully or recapitulate its development, but the pertinent factor here is

$$m_{12}^{1/2} = \sqrt{\frac{m_1 m_2}{m_1 + m_2}}$$

where m_1 and m_2 are the masses of the two molecules. *Moleculizer's* correction of the dimerization rate r to obtain a dimerization rate r' between larger complexes of mass m_1' and m_2' is obtained by assuming that

$$r' \sqrt{\frac{m_1 m_2}{m_1 + m_2}} = r \sqrt{\frac{m_1' m_2'}{m_1' + m_2'}}.$$

In other words, all the rest of the factors in Gillespie's formula above are assumed to be the same for for the "primed" dimerization. This assumption is especially bad for the ideal molecular diameters involved in d_{12}.

3.2.3 Bogus reactions and unrolling loops

Here is an example of an interesting way that *Moleculizer's* simplistic reaction rate extrapolation method can produce unexpected species and reactions: It is easy to imagine a simple protein with two binding sites so configured as to naturally form a homo-trimeric complex, as illustrated in Fig. 3. *Moleculizer* is only told that there are two binding sites on the protein and that they can bind to one another, and it knows nothing about the geometry of the protein. Hence, it generates polymeric chains, as illustrated, in addition to the "correct" homo-trimer. This phenomenon was also noted in [BL97].

Note also that the problem of correctly extrapolating reaction rates in the face of such geometric constraints naturally segues into what appears here to be a problem of incorrectly generated reactions. It would be difficult to say whether the reaction that makes the first four-mer in the figure was impossible due to geometric constraints or that it was merely very slow and the inverse decomposition reaction was very fast.

This "runaway polymerization" phenomenon can sometimes be avoided by carefully modifying the input specification to avoid the possibility of a cycle, a sequence of simple proteins in a complex, each bound to the next, that starts and ends with the same simple protein. By avoiding complexes containing loops, the unrolling of loops to form "runaway" polymeric complexes is avoided. This is a bad solution, since such cyclic complexes are involved in real cooperative binding, and working out practical, realistic ways to address this difficulty is an important next step in *Moleculizer's* development.

4 A system model in *Moleculizer*

Moleculizer was developed at The Molecular Sciences Institute in the context of the Alpha Project, a comprehensive experimental and computational program

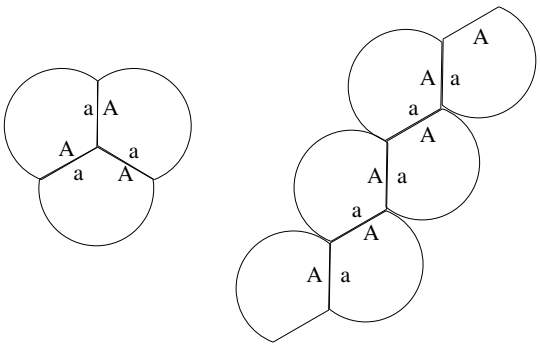

Figure 3: "Runaway" polymerization.

to study the molecular biology of the yeast mating pheromone signal transduction pathway. *Moleculizer* therefore provides special generators for reactions in this cellular subsystem to aid in its simulation. In this section, the signal transduction pathway and the specialized reaction generators for simulating it are described. Alpha pathway chemistry is only summarized, and the reader is encouraged to delve into a more thorough treatment such as [DT01].

4.0.4 Cartoon of the alpha pathway

At some times and under some circumstances, brewer's yeast has two different sexes, "a" and "α." These forms normally reproduce by budding, but when the opposite sexes find one another, they can mate. Mutual recognition is accomplished by each of the two sexes' secreting a pheromone and each having receptors on the cell membrane for the opposite sex's pheromone. When the pheromone of the opposite sex binds to these receptors, a cascade of biochemical reactions ensues, resulting in the yeast's ceasing to bud, moving/growing toward the mating partner, and finally fusing with it. This cascade of activity has been studied for years, and it has reached a point where a finer, quantitative study of it is now possible and has been undertaken in the Alpha Project. Currently, *Moleculizer's* special reaction generators serve the modeling of two parts of the signal transduction cascade: the first part receives the extracellular signal, and the second part is an intracellular signal-processing complex.

The first part of the signal transduction cascade that requires special reaction generators is the G-protein coupled receptor [GK$^+$02, chapter 4] complex, consisting of the "7-spanning" [GK$^+$02, chapter3, p. 51] alpha receptor Ste2 and the tripartite G-protein complex of Gpa1, Ste4, and Ste18, which are called the G_α, G_β, and G_γ subunits respectively. Several different combinatorial structures have been asserted for the Ste2, Gpa1, Ste4, Ste18 complex and modeled with *Moleculizer;* for convenience, the rough functional description here uses just one of them, illustrated in Fig. 4. The β and γ parts, Ste4 and Ste18, seem to function as a unit. On the other hand, the α part, Gpa1, functions by dissociating

from the complex when the alpha signal is being received, then reassociating with it when the alpha signal fades. Gpa1 binds GDP in its "base" state but also binds GTP. Gpa1 catalyses hydrolysis of GTP to which it is bound, reverting to its "base" GDP-bound state. When in the GDP-bound state, Gpa1 has a high affinity for the β-subunit Ste4, but in its GTP-bound state, Gpa1 has low affinity for Ste4. When the alpha pheromone binds to Ste2, a conformational change occurs that is propagated through the complex. It causes Gpa1, which is bound to GDP and complexed with Ste4, to change its affinities for GTP and GDP so that it strongly prefers GTP. When Gpa1 binds GTP, its loses most of its affinity for Ste4, and it dissociates from the G-protein complex until Gpa1 reverts to its GDP-bound form by the auto-hydrolysis reaction noted above. Another enzyme, Sst2, (not shown) also "helps" this hydrolysis reaction along, hastening the reversion of Gpa1 to its GDP-bound form, thereby bringing about rebinding of Gpa1 to Ste4 and eventually shutting off the pheromone response.

The second portion of the alpha pathway that has special *Moleculizer* reaction generators is the scaffolding complex, consisting of the protein Ste5, along with the protein kinases Ste11, Ste7, and Fus3. The scaffolding complex does not concentrate near the membrane until Gpa1 dissociates from Ste4, because Ste5 binds to Ste4 competitively with Gpa1. When Gpa1 is in its GDP-bound form and has high affinity for Ste4, Ste5 is unable to compete with it, and Ste5 floats free in the cytosol along with the protein kinases. But when Gpa1 dissociates from Ste4 because of an alpha signal, Ste5 binds to Ste4, bringing the kinases along with it into proximity with the membrane. There, a membrane-associated kinase, Ste20, phosphorylates Ste11. When Ste11 is sufficiently phosphorylated, it becomes active as a kinase and phosphorylates Ste7. In turn, Ste7 phosphorylates the MAP-kinase Fus3, activating it as a kinase. When active, Fus3 back-phosphorylates Ste7 in a negative feedback. When doubly phosphorylated, Fus3 tends to dissociate from the scaffold complex; it goes on to perform its nominal MAP-kinase function of phosphorylating the transcription-activating protein Ste12 (not shown) as well as other kinase activities. One or more phosphatases dephosphorylate the scaffold kinases, damping the response as a whole.

4.1 *Moleculizer* provides special tools for alpha pathway simulation

At a frontier of research such as the Alpha Project, "the model" varies on a day to day basis, so flexibility in the repertoire of modeling constructs provided by this simulation tool is critical. Real programming is needed to wrap a simulation tool around any biological concepts and hypotheses that were not programmed into it initially.

One way to use real programming to create new models is to provide a full-scale programming language within the model-building language. An example of this strategy in the simulation of digital electronics is VHDL, which makes more or less all of the general-purpose ADA programming language available in the model building language. But this strategy requires sophisticated compilation

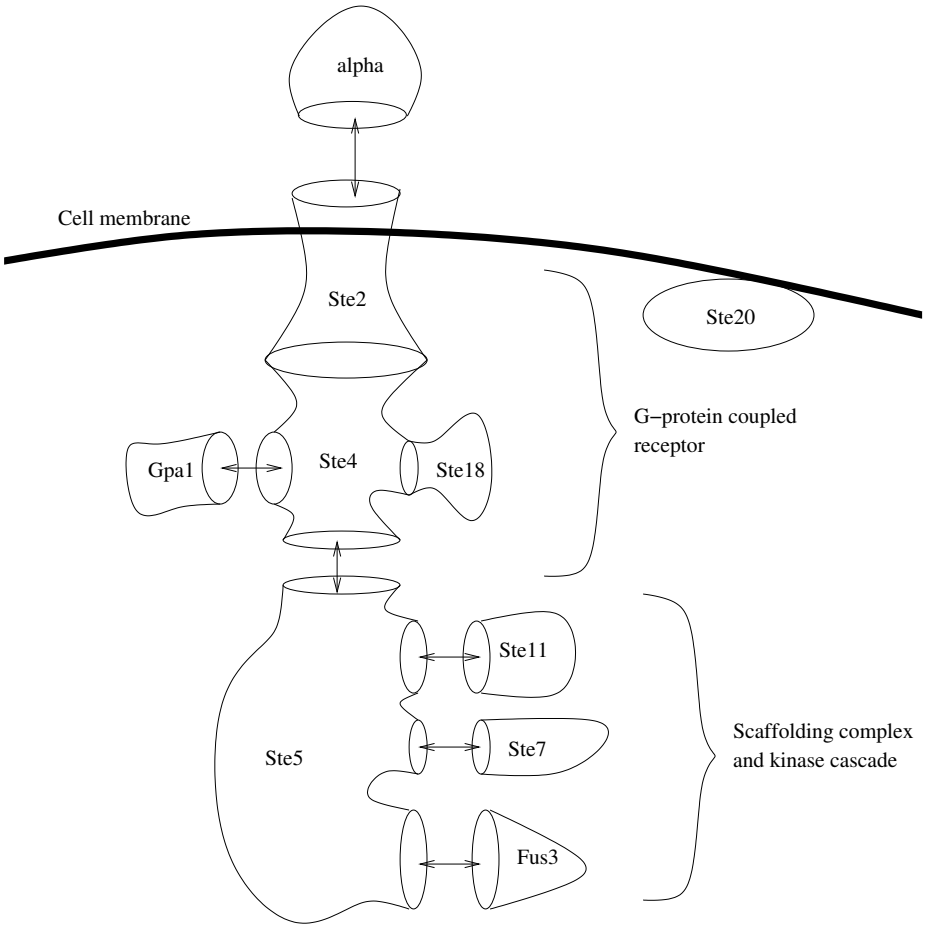

Figure 4: Cartoon of one model of an alpha pathway fragment.

of the model code, and most biologists are not interested in learning or applying a general-purpose programming language like this anyway.

Another way to create new models that require real programming is to incorporate the new material directly to the simulation engine. For the time being, at least, we chose this approach for *Moleculizer*. Close adherence to the standard programming precept of modularity is called for with this approach: New material is organized in modules so that existing code remains largely unchanged when new simulation constructs are added, simplifying the upgrade process and making maintenance easier. The following paragraphs describe content from three *Moleculizer* extension modules that support research into the alpha pathway:

Module	Purpose
Kinase-phosphatase	Supports "generic" kinases and phosphatases, as well as simple nucleotide binding and unbinding.
Nucleotide exchange	Supports the special nucleotide binding and autohydrolysis properties of G-proteins.
Scaffold kinase	Supports the localization of the action of scaffold kinases to the scaffolding complex.

4.1.1 Receptor models

The main specialized reaction of the receptor complex occurs when alpha factor binds to the receptor protein, Ste2, causing a conformational change in the whole receptor complex. This conformational change causes Gpa1's affinity for GTP to increase and its affinity for GDP to decrease, and GDP is exchanged for GTP. This exchange of nucleotides causes the affinity of Gpa1 for Ste4 to drop and causes Gpa1 to dissociate from the G-protein complex.

Conditioning the nucleotide exchange reaction on the presence of an "enabling" subcomplex (Gpa1, Ste4, Ste2, alpha) that can propagate the conformational change is the main reason that special programming is needed in this case. *Moleculizer* supports this reaction with two slightly different reaction generators:

The "more precise" version of nucleotide exhange changes the relative affinity of Gpa1 for GTP and GDP depending on whether or not Gpa1 is in an "enabling" complex. This version depends on nucleotide binding and unbinding reactions from the kinase-phosphatase module to actually effect the exchange of the nucleotides.

The "less precise" version of nucleotide exchange just unbinds the nucleotide-binding protein Gpa1 from GDP and binds it to GTP, provided Gpa1 is in an "enabling" complex. It does not depend on independent reactions to unbind GDP and bind GTP.

4.1.2 Scaffold kinase cascade model

An important hypothesis connected with the scaffolding complex is that, by assembling all of the scaffold kinases in close proximity to one another, complex formation increases their activity on one another. The proximity enhancement of these reactions is modeled by special scaffold kinase reactions that operate only in species of complexes where the kinase and its substrate are both bound to the scaffolding protein Ste5. Once again, the reason that these reactions require special programming is that they are conditioned on a particular ambient subcomplex, just as in the receptor model.

The scaffold kinases' activities are also regulated in complicated, and so far poorly-understood, ways by their own phosphorylation states.

The phosphorylation of Ste11 by the membrane-associated kinase Ste20 is thought to occur slowly or not at all unless Ste11 has been "recruited" to the membrane by the binding of the scaffold protein Ste5 to Ste4. A special reaction is provided in which phosphorylation of Ste11 by Ste20 only occurs when Ste11 is in a Ste11, Ste5, Ste4 subcomplex and, of course, where Ste11 has free target phosphorylation sites for Ste20.

With minor modifications, these reactions could be adapted to other kinase cascades, such as those occurring in the osmolarity or filamentation pathways in yeast, which have close organizational connections with the alpha pathway.

Using a combination of these "localizing" reactions with generic kinase reactions can arrange that the kinase cascade reactions also take place outside of the scaffolding complex, perhaps at lower rates.

4.1.3 Illustrative simulation output

Figure 5 shows an example plot, made with *gnuplot,* of *Moleculizer* output. It shows how *Moleculizer* can combine multiple species in a single trace. This plot was produced from a test simulation of a portion of the alpha pathway as described above; the simulation generated 41,033 reactions and 16,886 species. Reaction rates and molecular populations were *not* realistic.

The plot focuses on the receptor complex. When alpha factor is added at time 2.0, Gpa1 undergoes nucleotide exchange and dissociates from the receptor complex. The green trace "four-one-GDP" shows the population of all species of complexes containing Ste4 and Gpa1 where Gpa1 is bound to GDP. This trace gives the total population of 6 complex species, and dissociation of Gpa1 is marked by its rapid drop after time 2. The magenta trace "four-five" shows the population of all species of complexes containing Ste4 bound to Ste5, about 1,300 species. The binding of Ste5 to Ste4 made possible by Gpa1's dissociation from the receptor complex is marked by the rapid rise in this trace after alpha factor addition.

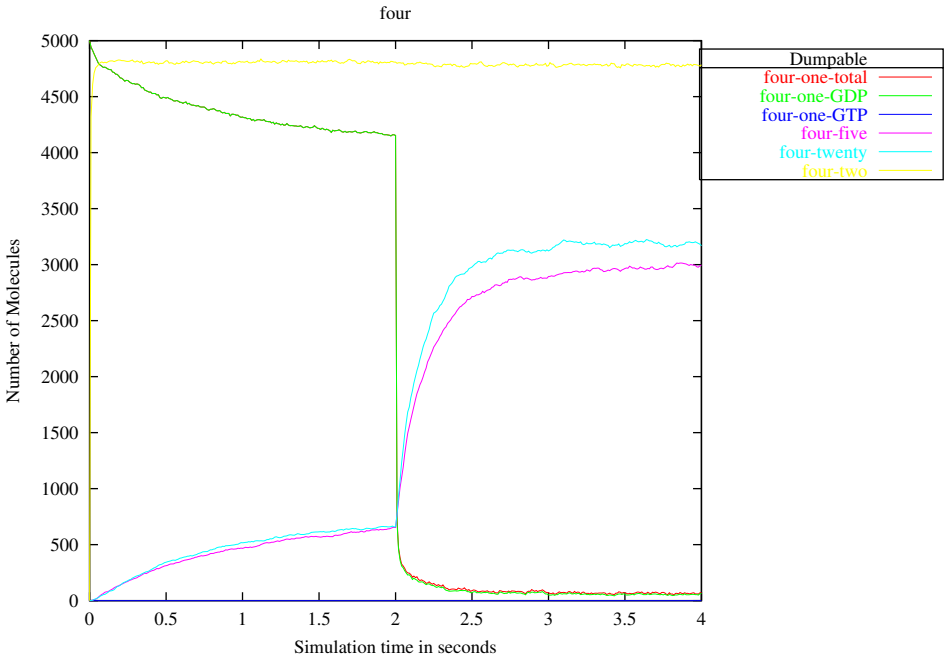

Figure 5: Illustrative simulation output.

5 Using *Moleculizer*

5.1 Batch mode operation

It is best to be able to run a computationally intensive program like *Moleculizer* on remote machines, on clusters, etc., at times when computers are available. Remote or delayed execution requires the possibility of unattended "batch mode" operation, in which program input is collected into an input file ahead of time. Hence, we chose this mode of operation as superior for our purposes to a graphical user interface permitting direct manipulation of the simulation state, such as might be used in a simulation-based computer game. In alpha testing, the ability to watch the progress of a batch simulation by occasionally plotting its output seemed generally satisfactory.

5.2 Input and output

5.2.1 All input and output is in XML

With the choice of batch mode operation, the issues of usability and interoperability focus on simulation input and output files.

Moleculizer originally had a simple line-based scripting language. The advantages of a simple text input file, such using one's favorite text editor, helped to make the scripting arrangement acceptable and practical. But as time passed, the importance of interoperability with other programs and the inadequacy of the simple scripting solution became more and more apparent. The appearance of Systems Biology Markup Language, SBML [HF+03], [SBM], and the development of MONOD [SG+03] at the Molecular Sciences Institute, both encouraged us to adopt a modern XML-based approach to file formats. In the end, powerful translation facilities, such as XSLT, more than compensate for the verbosity of XML and the need for a special editor.

A simple, repluggable editing solution was needed, and eventually an open-source generic XML editor, *xmloperator* [Dem] was chosen. Written in Java, it runs on many platforms. *xmloperator* provides "guided editing," whereby the editor generally disallows changes that do not conform to the specified syntax of the file and automatically inserts required material. The syntax description that *xmloperator* uses for this, an XML schema file, is easy to write and customizes *xmloperator* to each of the file formats connected with *Moleculizer*. Combined with easy translation into a web page, the upshot is that a user is initially presented with a template file and HTML links into documentation for all of its parts. The user can then edit and expand on this template document with full access to documentation and with guidance from the XML editor.

5.2.2 Executables and file formats

In fact, there are three *Moleculizer* executables with slightly different functions. One is actually called *moleculizer;* it is a conventional simulation run. It reads

Executable	Purpose	Reads	Writes
moleculizer	simulate, generate network	moleculizer-input	moleculizer-state, plot files
continuator	continue interrupted simulation	moleculizer-input, moleculizer-state	moleculizer-state, plot files
parametrizer	consistently adjust reaction network rates	moleculizer-input, moleculizer-state	moleculizer-state
odie	ODE simulator	odie	plot files

Figure 6: Main executables delivered with *Moleculizer 1.0*.

a "moleculizer-input" file, and runs the simulation, starting at time 0 and continuing as long as specified in the input file.

A *Moleculizer* simulation can be caused to dump the state of its reaction network at any time(s) during a simulation. The reaction network is written in an (XML) "moleculizer-state" file. Among other uses, this moleculizer-state file can be processed to provide a reaction network to another simulation tool, such as an ODE solver, or to restart the simulation from which it was dumped.

The second *Moleculizer* executable is called *continuator,* and it reads both a moleculizer-input file and a moleculizer-state file. As its name implies, this program continues the simulation from the point where the state of its reaction network was dumped.

The third *Moleculizer* executable is called *parametrizer,* and it too reads both a moleculizer-input file and a moleculizer-state file. Its purpose is to re-generate the reaction network given by the moleculizer-state file, but with parameters (reaction rates, molecular weights, etc.) given by the moleculizer-input file, which need not be the same as in the original simulation. The reparametrized moleculizer-state file is dumped immediately, and no simulation is actually run. The intent is to provide a way of changing reaction parameters in a *Moleculizer*-generated reaction network without having to run another time-consuming *Moleculizer* simulation. The expected use of a reparametrized reaction network of this kind is to power another, presumably faster simulation tool than *Moleculizer,* for example, an ODE solver. If one had to run a *Moleculizer* simulation to make every change of parameters, then the faster simulation tool would always have to wait for *Moleculizer* and no time savings would be realized.

But note that reaction rates and other parameters can affect the topology of the reaction network. Therefore, one should not attempt to reparametrize a network in this way too drastically: the reparametrized networks produced by *parametrizer* have the same topology as before reparametrization.

File format	Purpose	Input to	Output from
moleculizer-input	set up *Moleculizer* simulation	moleculizer, continuator, parametrizer	user editing with *xmloperator*
moleculizer-state	convey reaction network of complex species	moleculizer, continuator, parametrizer	moleculizer, continuator, parametrizer
odie	reaction network input for ODE solver	odie	conversion from moleculizer-state
SBML 2.0	convey reaction network of simple species	external tools	conversion from moleculizer-state
HTML	"live" documentation of *Moleculizer* files	web browser	conversion from any *Moleculizer* file format

Figure 7: File formats connected with *Moleculizer 1.0*.

5.2.3 Translation tools

Several tools are provided to convert *Moleculizer's* state output into formats useful to other simulators.

One translator generates input for *rk4tau,* an experimental stochastic simulator. Based on Gillespie's "tau-leaping" [Gil01] acceleration concept, *rk4tau* uses parts of a standard high-order adaptive Runge-Kutta solver for ordinary differential equations. To be honest, this program does not work well enough to be practical: It succumbs frequently to the same "stiffness" phenomenon [DB02, §4.1.3], [PT$^+$92] that hinders the use of many standard ("explicit" or "forward") methods of solving ordinary differential equations, but it is provided along with *Moleculizer* as a demonstration target simulator. Recent improvements of the tau-leaping strategy [RPCG03] will be applied in a later *Moleculizer* release.

Another translator converts moleculizer-state documents into input for *odie,* a simulator based on solving ordinary differential equations. The same phenomenon of "stiffness" mentioned above led us to choose the *Bulirsch-Stoer* algorithm for use in this solver, an "implicit" method of ODE solution that does not suffer from stiffness.

Finally, a third translator converts moleculizer-state documents into SBML 2.0 [HF$^+$03] format.

5.3 Web interface

Translatability into HTML makes *Moleculizer* documents web-friendly, and the web provides an interface that is sufficiently flexible to help the user set up batch simulations, check on their progress, etc. relatively painlessly. Many, perhaps a majority of users can do what they want to do with *Moleculizer* using only a web browser. For example, the web interface provides users with a way of

making parameter changes in a simulation without actually getting "hands on" with the simulation input file. The web interface also provides access to the three translation tools described in the last subsection.

Three categories of service, with different levels of capability and concomitantly different levels of complexity, are available to users:

1. Users for whom the web interface alone is sufficient need only a web browser.
2. Users who wish to write new simulations, rather than modify parameters in library simulations, need to use the Java program *xmloperator*, probably in conjunction with the web interface.
3. Users who want to run *Moleculizer* from the command-line, automatically run suites of simulations, etc. need to install the whole delivery on their Linux workstations, along with several ancillary programs including a web server like *Apache.* Administrators do essentially the same installation to set up a *Moleculizer* server.

5.4 Obtaining *Moleculizer,* licensing, etc.

Moleculizer, all of its source code and associated programs, including the web server installation, are available at the following URL:

`http://www.molsci.org/~lok/moleculizer/downloads`

The programs are all under open-source license, either the Gnu General Public License or the Gnu Library General Public License. This is intended to maximize the availability of source code (hence "open-source") and thereby encourage its development, not to inhibit commercial activity. Two notable pieces of open-source software written by other parties are delivered along with *Moleculizer*, the editor *xmloperator* [Dem] and the XML parsing library *libxml++* [LIB]. In both cases, the *Moleculizer* bundle includes their entire source delivery.

6 Acknowledgments

Funding for work on *Moleculizer,* among other projects at The Institute, has come from several far-seeing, generous sources:

1. DARPA
2. NIH
3. *Other?*

The authors would like to express grateful appreciation for their financial support.

References

[BF+03] M. Blinov, J. Faeder, et al. Expanding receptor signaling models by removing implicit assumptions restricting complex formation. *Preprint*, 2003. `http://cellsignaling.lanl.gov`.

[BL97] D. Bray and S. Lay. Computer-based analysis of the binding steps in protein complex formation. *Proc.Nat.Acad.Sci.*, 94:13493–13498, 1997.

[CAMK02] F. Cross, V. Archambault, M. Miler, and M. Klovstad. Testing a mathematical model of the yeast cell cycle. *Molecular Biology of the Cell*, 13:52–70, January 2002.

[CCNG+00] K.C. Chen, A. Csikasz-Nagy, B. Gyorffy, J. Val, B. Novak, and J. Tyson. Kinetic analysis of a molecular model of the budding yeast cell cycle. *Molecular Biology of the Cell*, 11:369–391, January 2000.

[DB02] Peter Deuflhard and Folkmar Bornemann. *Scientific Computing with Ordinary Differential Equations*, volume 42 of *Texts in Applied Mathematics*. Springer-Verlag, New York, 2002.

[Dem] Didier Demany. Web site. `http://www.xmloperator.net/`.

[DIP] Database of interacting proteins. Web site. http://dip.doe-mbi.ucla.edu/.

[DT01] Henrik G. Dohlman and Jeremy W. Thorner. Regulation of g protein-initiated signal transduction in yeast: Paradigms and principles. *Annual Review in Biochemistry*, 70, 2001.

[EL] D. Endy and E. Lyons. Stochastirator. Web site. `http://opnsrcbio.molsci.org/stochastirator/stoch-main.html`.

[EL+02] M. Elowitz, A. Levine, et al. Stochastic gene expression in a single cell. *Science*, 297:1183–1186, 2002.

[FS96] D. Frenkel and B. Smit. *Understanding Molecular Simulation*. Academic Press, 1996.

[FW95] T. Fricke and D. Wendt. The markoff automation: a new algorithm for simulating the time-evolution of large stochastic dynamic systems. *International Journal of Modern Physics*, 1995.

[GB99] Michael A. Gibson and Jehoshua Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journal of Physical Chemistry*, 104(9):1876–1889, 1999.

[Gib00] Michael A. Gibson. *Computational Methods for Stochastic Biological Systems*. PhD thesis, California Institute of Technology, April 2000.

[Gil76] Daniel T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22:403–434, 1976.

[Gil92a] Daniel T. Gillespie. *Markov processes: an introduction for physical scientists*. Academic Press, 1992.

[Gil92b] Daniel T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A*, pages 404–425, 1992.

[Gil01] Daniel T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115(4):1716–1733, July 2001.

[GK+02] Bastein D. Gomperts, Ijsbrand M. Kramer, et al. *Signal Transduction*, chapter 4. Academic Press, 2002.

[GP03] Daniel T. Gillespie and Linda R. Petzold. Improved leap-size selection for accelerated stochastic simulation. *Journal of Chemical Physics*, 119(16):8229–8234, 2003.

[HF+03] M. Hucka, A. Finney, et al. The systems biology markup language(sbml): A medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[HPG98] P. Hodges, W. Payne, and J. Garrels. The yeast protein database (ypd): a curated proteome database for saccharomyces cerevisiae. *Nucleic Acids Research*, 26(1):68–72, 1998.

[LIB] libxmlpp. Web site. `http://libxmlplusplus.sourceforge.net/`.

[MA97] H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *Proc.Nat.Acad.Sci.*, 94(3):814–819, 1997.

[MA98] H. McAdams and A. Arkin. Simulation of prokaryotic genetic circuits. *Annu. Rev. Biophys. Biomol. Struct.*, 27:199–224, 1998.

[Men94] Pedro Mendes. *Computer Simulation of the dynamics of biochemical pathways*. PhD thesis, University of Wales Aberystwyth, 1994. `http://www.gepasi.org`.

[MF98] Carl Jason Morton-Firth. *Stochastic Simulation of Cell Signalling Pathways*. PhD thesis, University of Cambridge, September 1998.

[Pau60] L. Pauling. *The Nature of the Chemical Bond and the Structure of Molecules and Crystals*. Cornell University Press, 1960.

[Pli] S. Plimpton. Web site. `http://www.cs.sandia.gov/~sjplimp/`.

[PT+92] William H. Press, Saul A. Teukolsky, et al. *Numerical Recipes in C*, chapter 16.6, page 734 ff. Cambridge University Press, second edition, 1992.

[RPCG03] Muruhan Rathinam, Linda R. Petzold, Yang Cao, and Daniel T. Gillespie. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *Journal of Chemical Physics*, 119(24):12784–12794, 2003.

[SB00] J.R. Stiles and T.M. Bartol. Monte carlo methods for simulating realistic synaptic microphysiology using mcell. In E. de Schutter, editor, *Computational Neuroscience: Realistic Modeling for Experimentalists*. CRC Press, 2000.

[SBM] Systems biology markup language. `http://www.sbml.org`.

[Sch] J. Schaff. Web site. `http://www.nrcam.uchc.edu`.

[SG+03] David Soergel, Brian George, et al. Monod, a tool to support collaborative modeling of biological processes, 2003. `http://monod.molsci.org/docs/Monod-June-2003.pdf`.