Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

MULTISTATE MODELING AND SIMULATION FOR REGULATORY NETWORKS

Zhen Liu

Department of Computer Science
Virginia Tech
Blacksburg VA, 24061

Clifford A. Shaffer

Department of Computer Science
Virginia Tech
Blacksburg VA, 24061

Umme Juka Mobassera

Program in Genetics, Bioinformatics,
and Computational Biology
Virginia Tech
Blacksburg VA, 24061

Layne T. Watson

Department of Computer Science
Department of Mathematics
Virginia Tech
Blacksburg VA, 24061

Yang Cao

Department of Computer Science
Virginia Tech
Blacksburg VA, 24061

ABSTRACT

Many protein regulatory models contain chemical species best represented as having multiple states. Such models stem from the potential for multiple levels of phosphorylation or from the formation of multiprotein complexes. We seek to support such models by augmenting an existing modeling and simulation system. Interactions between multistate species can lead to a combinatorial explosion in the potential state space. This creates a challenge when using Gillespie's stochastic simulation algorithm (SSA). Both the network-free algorithm (NFA) and various rules-based methods have been proposed to more efficiently simulate such models. We show how to further improve NFA to integrate population-based and particle-based features. We then present a population-based scheme for the stochastic simulation of rule-based models. A complexity analysis is presented comparing the proposed simulation methods. We present numerical experiments for two sample models that demonstrate the power of the proposed simulation methods.

1 INTRODUCTION

As systems biologists attempt to improve on their models for the physiological properties of cells, they require the ability to model more complex forms of protein interactions. In this paper we discuss protein regulatory models involving interactions between multistate chemical species, and their simulation using stochastic techniques. Such multistate species naturally arise from the possibility of a chemical species undergoing multiple levels of phosphorylation, or from the formation of complexes of proteins connected at various potential binding sites (Hlavacek et al. 2003). Since different phosphorylation levels or different binding configurations lead to different biochemical properties, each variation must be represented by separate state variables. The large number of potential bindings can yield a large state system, which can result in inefficient stochastic simulations.

In this paper, we consider two aspects of the problem. The first is how to provide to modelers of regulatory systems a mechanism for defining multistate reactions. The second is how to efficiently simulate such models using stochastic simulation. The outline of this paper is as follows. In Section 2 we discuss methods for supporting the definition of multistate models within the JigCell system (Shaffer et al. 2009). In Section 3, we present an overview of the simulation challenge, and briefly review Gillespie's stochastic simulation algorithm, rule-based modeling, and the network-free algorithm (NFA). Section 4 presents the improved NFA and the full-scale SSA (FSSSA) along with a brief complexity analysis. Numerical experiments are presented in Section 5. Our conclusions are in Section 6.

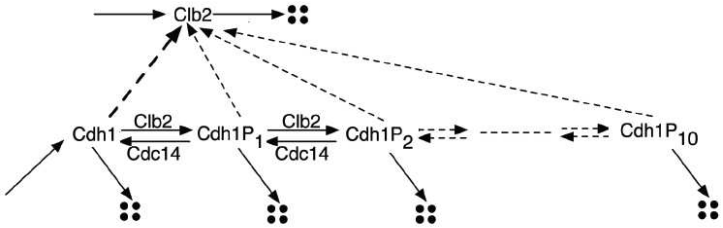


Figure 1: Clb2-Cdh1 antagonistic interaction, forming a bistable switch with Cdc14 as the control variable.

2 REPRESENTING MULTISTATE MODELS

A simple example of a multistate reaction involves a chemical species gaining a level of phosphorylation, perhaps under the influence of some catalyst. For example, the equation $S \rightarrow SP$ might represent a chemical species S undergoing one level of phosphorylation to become a complex represented as SP, meaning "S with one level of phosphorylation". In a standard systems biology modeling system with no support for multistate modeling, to indicate that a species undergoes a second level of phosphorylation would require that the modeler write explicitly a second equation such as $SP \rightarrow SP2$ (in essence, introducing an additional chemical species into the system). Of course, the rate of each reaction depends both on the amount of the the species S (or SP in the second reaction), a rate constant, and perhaps the amount of an enzyme catalyzing the reaction.

Typically, it is possible for the species S to acquire more phosphorylation levels than just one or two. Our first equation would better have been expressed as $SP0 \rightarrow SP1$, since we could view the initial state of S as having zero levels of phosphorylation. Depending on the amount of phosphorylating catalyst, the system will tend to push the bulk of the molecules of S toward either high or low levels of phosphorylation, and this behavior could change over the life of the system as the amount of catalyst changes. We would like some mechanism for writing reactions that expresses this behavior, such as $S\{i\} \rightarrow S\{i+1\}$, where the expression in the braces indicates an index for a multistate species. To be complete, we must also define an index range for the state variable S, and independently a range for the index over which any given reaction operates (since a given reaction could operate on only part of the state variable's potential range). If we wish to support ten levels of phosphorylation, then we would require ten separate reactions be written in a system that provides no support for multistate modeling.

A small but realistic example of a motif in regulatory models shows an antagonistic relationship between Clb2 and Cdh1, mediated by the amount of catalyst Cdc14 (Kapuy et al. 2009). Figure 1 illustrates this system. This can be modeled with reaction equations as

$$\phi \xrightarrow{k_{s1}} Clb2, \tag{1}$$

$$Clb2 + Cdh1P\{i\} \xrightarrow{k_{d1}} Cdh1P\{i\}\ , \tag{2}$$

$$Cdh1P\{i\}\ +\ Clb2 \xrightarrow{k_p} Clb2\ +\ Cdh1P\{i+1\}, \qquad 0 \le i \le MAX-1, \tag{3}$$

$$Cdh1P\{i\}\ +\ Cdc14 \xrightarrow{k_{dp}} Cdc14\ +\ Cdh1P\{i-1\}, \qquad 1 \le i \le MAX, \tag{4}$$

$$\phi \xrightarrow{k_{s2}} Cdh1P\{0\}, \tag{5}$$

$$Cdh1P\{i\} \xrightarrow{k_{d2}} \phi, \qquad 0 \le i \le MAX. \tag{6}$$

All equations here are assumed to operate using the simple mass action rate law. The first two equations represent synthesis and degradation of Clb2 at rates dictated by rate constants (with degradation also dependent on the amount of Clb2 currently existing – see Section 5.1). $Clb2$ acts as an enzyme for the phosphorylation of $Cdh1$. On the other hand, $Cdh1$ activates the degradation of $Clb2$, where the catalytic rate associated with the unphosphorylated form of $Cdh1$ is 10 times larger than those phosphorylated forms of $Cdh1$. Therefore, these two species are in opposition to one another, represented by the third and fourth equations. The last equation represents degradation of Cdh1 in some given state of phosphorylation. With careful selection of its parameters, the model can exhibit bistable behaviors. In particular, the amount of Cdc14 acts as a control variable mediating the interaction of Clb2 and Cdh1. When the amount of Cdc14 is high, the typical phosphorylation level of Cdh1 becomes lower, thus reducing the amount of Clb2. This switch in turn can be used to control other parts of the system. Thus this motif appears within a broader control system, such as the stochastic cell cycle model of budding yeast shown in Figure 2. The cell cycle is the sequence of events by which a growing cell replicates all its components and divides them more or less evenly between two daughter cells, so that each daughter cell receives the information and machinery necessary to repeat the process. Figure 2 shows an important part of the control diagram in cell cycle. The key

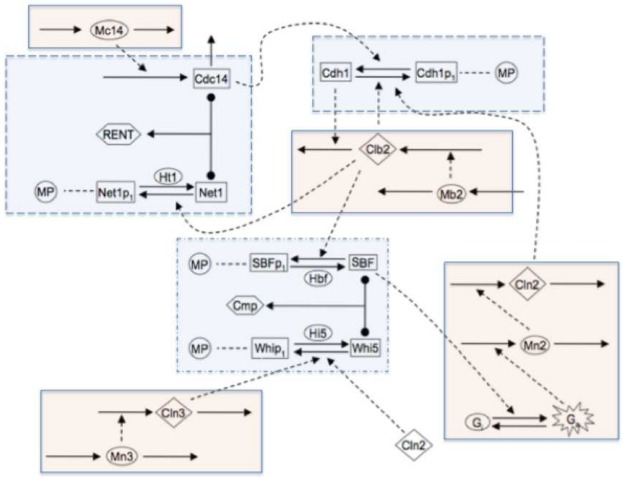


Figure 2: Cell cycle model of the budding yeast involving multiple site phosphorylation. The reactions within the dashed lines involve multistate phosphorylations and are indicated in the diagram by "MP". The portion of the model in the center within the dash-dot line involves a complex of two multistate species (indicated as "Cmp"). Cmp is represented by a two-dimensional state variable in a simulation.

mechanism is the switch between *Clb*2 and *Cdh*1. The whole process starts from the left bottom corner in the diagram. When cell grows, *Cln*3 increases. *Cln*3 activates *SBF* through the motif indicated in the block that contains *SBF* and *Whi*5. *SBF* activates the gene expression of *Cln*2, and *Cln*2 switches the motif between *Cdh*1 and *Clb*2 so that *Clb*2 is activated. The activation of *Clb*2 indicates the G1-S phase, where a cell reproduces its DNA and protein components. Meanwhile, *Clb*2 represses *SBF* (thus reduces the level of *Cln*2 and slowly activates *Cdc*14. Both effects lead to the switch of the motif so that *Clb*2 goes back to the lower level, which indicates the S-G2-M phase, where a cell aligns its duplicated DNA and divides into two daughter cells.

While only six equations are needed in a modeling system that supports multistate reaction equations, this model would require dozens of explicit single-state reactions. Thus we see that explicit support for multistate modeling greatly reduces the burden on modelers.

A more complicated example involves complexes of chemical species. Imagine species A and B, each of which might undergo up to nine levels of phosphorylation, and which can react (each at any level of phosphorylation) to form a complex. We could represent this as

$$A\{i\} + B\{j\} \rightarrow AB\{i,j\}; \qquad 0 \le i \le 9, \quad 0 \le j \le 9.$$

This means that there could be as many as 100 distinct states for the complex AB. As we will see in the remaining sections, performing efficient simulations on such a system can be difficult due to this combinatorial explosion of states. Depending on the amounts of A and B, most of the potential states of complex AB will likely never be realized, or not all coexist.

We are presently working on revisions to the JigCell Model Builder (JCMB) (Vass et al. 2006) to support multistate reactions in models. Figure 3 shows the interface for representing our example of the bistable switch. Since JCMB's native representation language is SBML (Hucka et al. 2003), and the current version of SBML has no multistate native support, it will be necessary for us to develop modifications to SBML to support multistate modeling as well. We are guided in our design by the ideas currently being discussed in proposals to support multistate modeling in the forthcoming SBML Level 3 (Le Novere and Oellrich 2010).

3 STOCHASTIC MODELING AND SIMULATION OF MULTISTATE SYSTEMS

Gillespie's SSA (Gillespie 1976, Gillespie 1977) is one of the most popular stochastic algorithms for simulating chemical reaction systems. It is an asymptotically exact (as the number of sampled trajectories approaches infinity) simulation scheme because it directly simulates the chemical master equation. It is a *population-based* method since the state variables in the SSA represent the populations of species. However, as models of biological systems become more complicated and realistic,

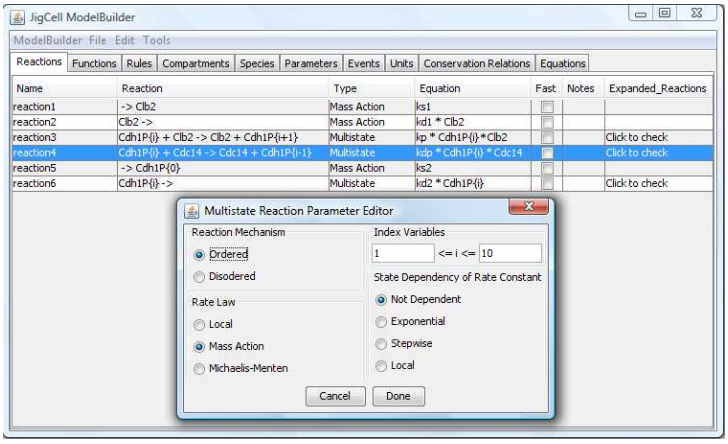


Figure 3: Screenshot from the JigCell Model Builder illustrating the bistable switch model. The dialog box shows how additional details for a given reaction are set.

the traditional SSA faces many challenges, one of which is the combinatorial complexity that can result when attempting to simulate multistate chemical species (Hlavacek et al. 2003, Bray 2003).

Particle-based modeling (Regev and Shapiro 2002, Regev et al. 2004) was proposed to address this challenge. StochSim, developed by Morton-Firth and Bray, treats all reacting molecules as individual objects with their own properties including configurations of different binding sites. In each step, StochSim uses two uniform random numbers to select two objects. Then it checks a pre-calculated probability table to see if these two objects could fire a reaction. If they do, a uniform random number is generated to decide whether this reaction will really fire. StochSim has been successfully applied to the stochastic modeling and simulation of bacterial chemotaxis (Morton-Firth and Bray 1998, Morton-Firth et al. 1999, Shimizu and Bray 2001). It has an advantage for multistate variables as its efficiency is not affected by the number of different reactions present in the system (Novere and Shimizu 2001). However, the probability table is generated with a fixed time step. In order to maintain its accuracy it has to take small time steps in systems containing large numbers of molecules (Liu and Cao 2008, Pettigrew and Resat 2005). Moreover, StochSim often generates null events where no reaction will fire as either the two randomly selected objects cannot react or the reacting probability is smaller than the generated random number.

Another way to model multistate reactions is *rule-based modeling*, proposed by Hlavacek et al. (2006), Harris et al. (2009). The idea is to generalize a group of reactions into one rule if all reactions in the group share the same binding properties of their reactants and products. The resulting rule-based model can be much smaller and simpler than conventional models that describe explicitly each individual chemical reaction. This can simplify both the conceptual complexity of the model for the modeler (who can think in terms of a small number of multistate reactions) and of the simulation. Since rule-based models are similar to models already represented and simulated by StochSim (StochSim supports reactions with configurations for binding sites), the simple rule-based models described in this paper could be simulated by StochSim too. However, StochSim often has low efficiency when required to achieve sufficient accuracy, since it sets the step size at initialization and does not readjust to larger sizes when possible (Liu and Cao 2008). Based on a formal language specification for biochemical modeling, Danos et al. (2007) developed a stochastic simulation method specificly for rule-based models. It modifies Gillespie's original SSA by substituting rules for reactions and using a particle-based simulation scheme. Later, Yang et al. (2008) generalized this method so that a rule can have different rate constants for different states of reactants. Following their work, Sneddon et al. (2008) developed a new rule-based model simulator called the Network-Free Stochastic Simulator, which uses the methods presented in Danos et al. (2007) and Yang et al. (2008). We refer to this method as the network-free algorithm (NFA).

Although NFA shows remarkable advantages over StochSim and Gillespie's SSA for systems with multistate features, it creates objects for all molecules in the system, which is not desirable for a system with large populations of single-state species. Treating every molecule as an object results in both greater space requirements and extra time in data structure maintenance.

For those who are not familiar with the literature, below we now briefly describe some of the modeling and simulation methods we mentioned above. In Section 4 we explain how to modify NFA to treat single-state species with a population-based scheme. This changes NFA from a particle-based scheme to a hybrid scheme and reduces its space and time complexity. To gain further improvements, we propose the full-scale SSA (FSSSA), a population-based simulation algorithm on rule-based models. In numerical experiments, the hybrid NFA and the FSSSA are compared with Gillespie's SSA based on two

realistic biochemical models. One is for the bistable switch motif described in Section 2. The other is a more complete cell cycle model of budding yeast involving multiple phosphorylation states that was recently developed by John Tyson's group (Barik et al. 2010).

3.1 SSA

Suppose the system involves N molecular species $\{S_1, \ldots, S_N\}$. The state vector is denoted by $X(t) = (X_1(t), \ldots, X_N(t))$, where $X_i(t)$ is the number of molecules of species S_i at time t. M reaction channels $\{R_1, \ldots, R_M\}$ are involved in the system. A reaction channel is any chemical event that causes changes to the state variables. Assume that the system is well stirred and in thermal equilibrium. The dynamics of reaction channel R_j is characterized by the *propensity function* a_j and the *state change vector* $\nu_j = (\nu_{1j}, \ldots, \nu_{N,j})$: $a_j(x)dt$ gives the probability that one R_j reaction will occur in the next infinitesimal time interval $[t, t+dt)$, and ν_{ij} gives the change in the S_i molecule population induced by one R_j reaction.

The dynamics of the system can be simulated by the SSA (Gillespie 1976, Gillespie 1977). With $X(t) = x$, let $a_0(x) = \sum_{j=1}^{M} a_j(x)$. On each step, the SSA generates two random numbers r_1 and r_2 in $U(0,1)$, the uniform distribution on the interval $(0,1)$. The time for the next reaction to occur is given by $t+\tau$, where τ is given by

$$\tau = \frac{1}{a_0(x)} \log\left(\frac{1}{r_1}\right). \tag{7}$$

The index j for the next reaction is given by the smallest integer satisfying

$$\sum_{l=1}^{j} a_l(x) > r_2 a_0(x). \tag{8}$$

The system states are updated by $X(t+\tau) = x + \nu_j$. The simulation proceeds to the next occurring time, until it reaches the final time.

3.2 Rule-Based Modeling

Rule-based modeling (Hlavacek et al. 2006, Harris et al. 2009) uses rules to replace reaction channels. A rule defines how a molecular particle reacts with other particles, depending on the states of their binding sites. Suppose protein species A has three binding sites and the first site can either be open or occupied by a protein of species B. A rule might include all reactions that bind a B protein onto the first site of an A protein, regardless of the other two sites:

$$A_{open,?,?} + B \xrightarrow{k_{rule}} A_{B,?,?}\,, \tag{9}$$

where the subscripts of a species describe the configuration of all its binding sites. A question mark indicates all possible states for a certain site. k_{rule} is the rule rate constant. To represent this single rule, the SSA must assign one reaction channel to every qualified state variable of species A and therefore may have a large system size.

Rule-based modeling is particularly convenient for representing biochemical systems with complex chemicals that may bind to each other. Suppose there is only one species with three binding sites, and each empty binding site can be bound to an empty binding site of another particle from the same species. When these two particles bind together, the remaining empty sites can still be bound by other particles. So eventually a complex may have many particles that collectively have thousands of empty binding sites remaining. Rule-based modeling was proposed to represent this type of system, which has many reactions that follow relatively few rules.

3.3 Network-Free Algorithm

Due to combinatorial complexity, when represented by the SSA structure the converted rule-based model usually has a large system size. A particle-based method called the network-free algorithm (NFA) (Sneddon et al. 2008) was proposed for rule-based models. NFA calculates the propensities for rules in a similar way as in the SSA. The major difference is that, instead of using populations of the reactant species, it uses populations of all particles that are candidates to be the reactants of a rule. For example, bimolecule rule (9)'s propensity is calculated by the reaction rate k_{rule} times the number of particles of species A with the first site open, then times the number of particles of species B. The firing time and index of the next reaction are still calculated as in (7) and (8). For each rule, reactant candidate lists are created to store all particles that satisfy the rule conditions. Thus when a rule is selected as in the SSA procedure, reacting particles in the selected rule are selected from corresponding candidate lists by generating uniform random numbers to calculate the indices of the reacting particles. The general procedure of the network-free algorithm for our implementation is as follows.

Initialization. In general, every rule entails the following information.

1. The description of the reactants and products with their species and states of all the binding sites, including the binding site conditions for this rule to fire.
2. Rule rate constant, by default each rule has its own rate.
3. Lists of candidate reactants. To facilitate selecting reacting particles after the rule index is determined, the method maintains a reactant list for each reactant in each rule. A reactant list contains pointers to all the candidate particles that meet the corresponding binding site conditions in its rule.
4. Populations of reactants. To calculate propensities quickly, populations of reactants are recorded and updated when necessary, so that the counting of particles in reactant lists is not repeated in every step.

The algorithm first creates rules by initializing items (1) and (2). Then objects for all molecular particles in the system are created. Upon the creation of each object, all rules are scanned. If the created object qualifies as a reactant of some rule, a pointer to the object is added to the corresponding reactant list and the reactant population is increased by one. After all particle objects are created and assigned, initialization of items (3) and (4) is completed.

Simulation. NFA repeats the following simulation procedure until a stopping criterion is reached.

1. Calculate propensities for all rules, using the rules' reactant populations and rate constants.
2. Calculate the next event time with equation (7).
3. Select a rule index with equation (8).
4. Select reacting objects (particles) from the corresponding reactant lists of the selected rule. In this step, the index of each reacting particle in the corresponding list is calculated by generating a uniform random number and multiplying it with the total population in the list.
5. Update the system by executing the selected rule. Destroy, create, or modify the states of reacting objects (reactants and products), if necessary, as described by the rule. For a reactant object, first check through all rules for which it originally qualifies. If it no longer satisfies the firing condition of a rule, then remove its pointer from the corresponding reactant list for that rule and decrease the corresponding reactant population by one. Then search all rules for the modified or created object. If it qualifies for a rule, its pointer is inserted into the corresponding reactant list and the reactant population is increased by one.
6. Update time, and go to Step 1.

The major advantage of the NFA is that it does not generate small time steps as StochSim often does, and it does not need to generate the huge reaction network that Gillespie's SSA does. The disadvantage is the extra cost for the storage of a large number of particle objects, and the maintenance of the reactant lists of all rules in every time step.

4 NEW RULE-BASED SIMULATION METHODS

Since NFA creates objects for all molecular particles and keeps reactant lists for each rule, if the total population in the system is large, NFA incurs a severe memory and time cost for the maintenance of those long reactant lists. With these concerns in mind, we modify NFA to reduce its space and time complexity as much as possible.

4.1 Population-based NFA (PNFA)

Here we propose a hybrid population-based NFA, which stores single-state species as populations instead of creating multiple individual objects. In other words, PNFA creates only one object for each single-state species with a population. Objects of this type are called *population objects*. The difference between population objects and normal molecular objects is that population objects store the populations of the species while molecular objects store the state of binding sites. With this change, the space complexity of NFA is greatly reduced for those systems involving simple chemical species with high populations. Of course, changes to the state variables require corresponding modifications in the simulation steps. We highlight the changes from the NFA to the PNFA procedure.

Initialization. The algorithm first creates rules by initializing their definitions of reactants, products, and rate constants. Then objects for all molecular particles in the system are created, species by species. For a single-state species, a population object is created with its initial population. For a multistate species, normal molecular objects are created with binding site states. Upon the creation of each object, all rules are scanned. If the created object qualifies as the reactant of some rule, a pointer to the object is added to the corresponding reactant list and the reactant population is increased by one if the created object is a normal molecular object (multistate species). Population objects increase the reactant population by the amount of the object's population. After all objects are created, initialization is completed.

Simulation. The simulation procedure is similar to that for NFA, except that PNFA treats population objects and normal molecular objects differently. We replace the original Step 4 with the following. 4*: Select reacting objects from

the corresponding reactant lists of the selected rule. In most cases, the reactant list contains either a population object or molecular objects, but not both, in which case selection is easy. When both exist in the list, the number of molecular objects is calculated and treated as a pseudo population object. We first search for which (population) object is selected. If the selection is a normal population object, this object is chosen. Otherwise we use a uniform random number to select among the molecular objects of the chosen pseudo population object.

We replace Step 5 with the following. 5*: Update the system by executing the selected rule. Destroy, create, or modify reacting objects (reactants and products) as described by the rule. For a molecular object, the operation is the same as in the original Step 5. A population object needs only to decrease or increase its population value and reactant populations for related rules.

4.2 Full-scale SSA (FSSSA)

PNFA is a hybrid scheme combining both population-based and particle-based strategies. It is natural to extend the idea and ask: "Is the particle-based scheme always efficient for simulating rule-based models? Under what condition will a pure population-based method perform better for a rule-based model?" We believe that, besides single-state species, any multistate species with a large population and a small number of states would be better represented by a population instead of as many particles. On the other hand, the particle-based scheme is preferred when a small population of a multistate species is sparsely distributed into a large number of states.

To study the population-based simulation method, we propose the full-scale SSA (FSSSA), a totally population-based algorithm for simulating rule-based models. FSSSA can also be considered as a new implementation of the SSA on rule-based models. Here is the FSSSA simulation procedure.

Initialization. Like the NFA, rules are first initialized with their first two data items. Since the method is not particle based, it does not create particle objects for molecules and the reactant lists for rules are not needed either. Species are initialized one by one. A single-state species is simply treated as a multistate species with one state. For each species, an object with a multidimensional array is created to store the populations of all the possible states, and is initialized state by state. Then all the rules are searched. The populations of all states that qualify as the reactant of a rule are summed and saved as the qualified population for that rule.

Simulation. The simulation procedure is similar to that for the NFA, but without molecular objects and pointers to objects. As a result, we replace the original NFA Steps 4 and 5 with the following two steps.

4**. For a selected rule, select a particular state for each reactant. The search is done within the multidimensional array but only those states that satisfy the selected rule are searched. The selection procedure is similar to SSA equation (8).

5**. Update the populations of the selected states of the reactant species and product species according to the descriptions of the selected rule. Then search all the rules. If the original state of the reactant species does not satisfy a rule, but after the reaction the modified state happens to satisfy the rule, increase the reactant population of that rule by one. Similarly, if the state of the reactant species satisfies a rule but the modified state does not satisfy it any more, decrease the reactant population of that rule by one.

4.3 Complexity Analysis

Except for StochSim, the algorithms described in this paper can all be considered as different implementation strategies for the SSA, directly simulating the chemical master equation. The key difference is in their efficiency. Below we offer a complexity comparison of these algorithms.

Suppose N is the number of species, M is the number of reaction channels, N_R is the number of rules in the system, N_S is the maximum number of states for each multistate species S, $N_{m:obj}$ is the total number of molecular objects in the system, and $N_{pm:obj}$ is the total number of pseudo molecular (molecular and population) objects in the system. Table 1 gives the time and space complexities of four simulation methods for rule-based models. Note that since these four methods all use (7) to generate the next reaction time, their average numbers of time steps are asymptotically the same. Therefore, the computational cost of each time step can be divided into three parts: reactant selection, system update, and propensity calculation. For the original SSA, the first and third part are usually achieved in time $O(M)$. Since in most cases for rule-based models $M \gg N_R$, the original SSA exhibits lower efficiency than the other three methods.

Comparing NFA and PNFA, both algorithms use uniform random numbers to calculate the indices of the reactant objects. The time complexity of this calculation is $O(1)$. Thus the first parts of their time complexities are dominated by selecting the rule to fire, which is $O(N_R)$. For the system update part, there is a good chance for PNFA to select population objects and not have to update the corresponding reactant pointer lists, so PNFA often has a smaller runtime constant than NFA, and therefore the total runtime for PNFA is often lower than that for NFA. $N_{pm:obj}$ for PNFA is different from $N_{m:obj}$ for NFA. The space complexity of PNFA is lower than that of NFA because of the population objects, which can make $N_{pm:obj} \ll N_{m:obj}$.

	Time Complexity of Each Step			*Space*
	reactant selection	*system update*	*propensity calculation*	*Complexity*
SSA	$O(M)$	$O(1)$	$O(M)$	$O(N)$
NFA	$O(N_R)$	$O(N_R)$	$O(N_R)$	$O(N_{m:obj})$
PNFA	$O(N_R)$	$O(N_R)$	$O(N_R)$	$O(N_{pm:obj})$
FSSSA	$O(N_R)+O(\hat{N})$	$O(N_R)$	$O(N_R)$	$O(N\hat{N})$

Table 1: Complexity of four methods for rule-based models, $\hat{N} = \max_{s\in S} N_s$.

Comparing PNFA and FSSSA, since the latter still uses (8) to select the states of the reacting variables from multistate arrays, the first part of its time complexity has an extra term $O(\hat{N})$, the maximum number of states for any chemical species. However, for the system update part, FSSSA does not maintain reactant pointer lists, so its runtime constant can be smaller than that of PNFA. Therefore, when $\hat{N}$ is not large compared with N_R, FSSSA can need less total time. On the other hand, if $\hat{N}$ is large, FSSSA may not be efficient. For a system with a few molecules that could have great numbers of states, coupled with uni-state species with large populations. The hybrid simulation scheme PNFA will show its advantage over either FSSSA or NFA.

Comparing SSA and FSSSA, the asymptotic space complexity for FSSSA compared to that for SSA depends on $\hat{N}$. If $\hat{N} = O(1)$, FSSSA generally consumes less actual memory than SSA because of the smaller size of the rule-based model. SSA uses elementary single-state rules, resulting in a larger rule network.

5 SIMULATION EXPERIMENTS

To compare the methods for rule-based models with the SSA in both accuracy and efficiency, we present numerical experiments on two realistic rule-based models. So far as the authors know, there are no general benchmark rule-based models available. Thus we choose examples from the modeling practice in stochastic cell cycle model for budding yeast. In the following the experiments were performed on a 2.66 GHz Intel Core 2 Duo iMac with codes written by the authors in C++ languages.

5.1 Bistable Switch Model

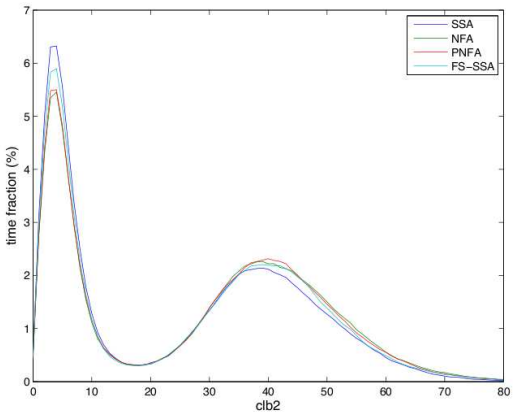
Our first example is the bistable switch motif shown in Figure 1. The integer subscript i represents the level of phosphorylation ranging from 0 to MAX, where $5 \le MAX \le 20$ is a range for typical phosphorylation levels in budding yeast cell. We choose $MAX = 10$ for our experiments. The parameters are $k_{s1} = 0.4$, $k_p = 0.05$, $k_{dp} = 0.05$, $k_{s2} = 0.05$, $k_{d2} = 0.0025$. For reaction (2), the reaction rate is calculated by $k_{d1} = 0.0055 Cdh1P\{0\} + 0.0005 \sum_{i=1}^{MAX} Cdh1P\{i\}$. For initial values, we start with $Clb2$ and $Cdh1Pi$ $i = 1,\ldots,10$ all at 0 and $Cdh1P0$ at 10. As the parameter $Cdc14$ varies, the distribution of $Clb2$ exhibits monostable or bistable behavior.

Figure 4 compares the distributions and time trajectories of $Clb2$ generated by SSA and the rule-based simulation methods. Since all the methods yield nearly identical distributions and similar system behaviors, we expect that they have comparable accuracy.

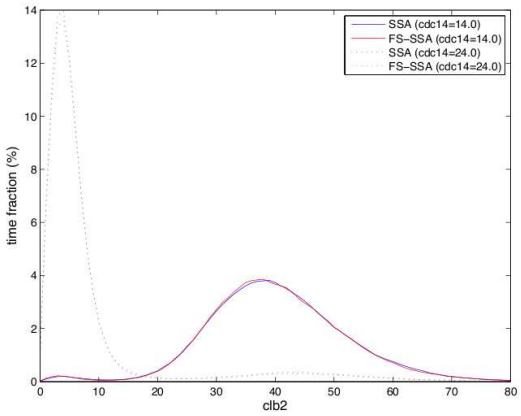
Table 2 shows runtimes for this model. To configure this model for SSA, the six rules in (1)–(6) are translated to 34 reaction channels with 12 state variables involved. NFA performs better than SSA due to the smaller rule-based model. PNFA has slightly better performance than NFA because of the single-state species $Clb2$. Since $Clb2$'s population does not dominate $Cdh1$, the efficiency gain should not be expected to amount to much. FSSSA performs the best, because it spends much less time in system update and data structure maintenance than NFA and PNFA do. However, according to Table 1, FSSSA has an extra term of $O(\hat{N})$ in the reactant selection part. In this model, $\hat{N} = 11$, which is comparable with $N_R = 6$. Therefore, the advantage of FSSSA (that it is free of reactant candidate lists) can be weakened by its extra work in selecting the state of the multistate reactant. We see for this model that it is more efficient to treat the multistate species as populations than as particles.

5.2 Budding Yeast Cell Cycle Model

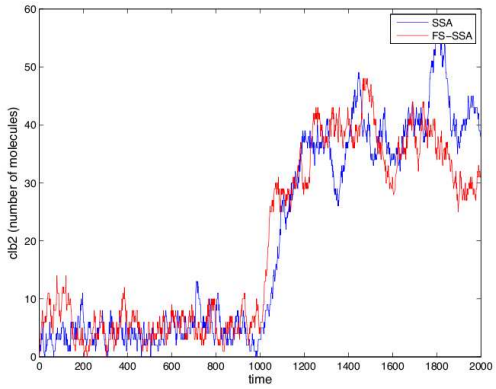
Our second example is a new stochastic cell cycle for budding yeast. Prior budding yeast cell cycle models were often formulated as ODEs and simulated deterministically (Chen et al. 2004). However, they failed to capture the intrinsic noise that is caused by the stochastic effects of the low copy numbers of some species. Recently Tyson's group developed a stochastic model for the budding yeast cell cycle by integrating several functional biochemical motifs (Barik et al. 2010). These motifs



(a) Bistable distribution ($Cdc14 = 19.0$).



(b) Two monostable distributions.



(c) Time trajectories show change in *Cdc*14 from 24.0 to 14.0.

Figure 4: Figures (a) and (b) show distributions of *Clb*2 by SSA, NFA, PNFA and FSSSA under different values of *Cdc*14. Figure (c) shows the transition from the lower steady state to the upper steady state due to a sudden drop in the level of *Cdc*14. All results are obtained with a simulation of 5×10^6 time units.

are based on multiple site phosphorylation, which has been discovered to generate nonlinear dynamics (Qu et al. 2003) and form bistable switches. Figure 2 illustrates this model.

As we can see, the bistable switch model discussed in the previous subsection is one of these motifs. Since these motifs are well represented by a rule-based modeling technique, we convert the elementary reaction model with 184 reaction channels (single-state reactions) into a rule-based model with 64 rules. Since the rule-based model still contains many species that are single-state and highly populated, according to our previous analysis FSSSA should be faster than SSA. In the simulation of this model, the volume of the cell is fixed and the division event is not introduced. With this configuration, the simulation is able to show the steady state for the components of the cell. Figure 5 shows the distribution plot for the key protein species *Clb*2 from SSA and FSSSA when the volume of the cell equals 26.0. FSSSA generates a nearly identical distribution to that of SSA. Table 3 shows the runtimes of the simulation. Since the number of rules for FSSSA is much smaller than the number of reactions for SSA, FSSSA saves a significant amount of CPU time in calculating the propensities. On the other hand, FSSSA spends around twice the time in system updating as SSA does.

6 CONCLUSION

To deal with the combinatorial complexity caused by multistate species in many biological models, rule-based modeling techniques have shown advantages in representing systems involving multistate species (Faeder et al. 2008). NFA is a particle-based method for stochastic simulation of rule-based models. In this paper, we have proposed a new hybrid method PNFA combining the population-based and particle-based schemes, and then proposed a new population-based method FSSSA. Both methods can be successfully used for rule-based model simulation.

	CPU time in seconds				
Method	*reactant selection*	*system update*	*propensity calculation*	*other*	*total*
SSA	*33.1*	*12.8*	*47.6*	*6.6*	*100.1*
NFA	*12.3*	*58.5*	*12.3*	*5.2*	*88.3*
PNFA	*13.1*	*54.2*	*10.6*	*5.3*	*83.2*
FSSSA	*30.3*	*20.8*	*10.9*	*4.5*	*66.5*

Table 2: CPU times used by four simulation methods on a Mac Core 2 Duo 2.0 GHz machine for a bistable switch model. Results were obtained with a simulation of 5×10^6 time units.

	CPU time in minutes				
Method	*reactant selection*	*system update*	*propensity calculation*	*other*	*total*
SSA	*12.5*	*8.0*	*64.6*	*1.1*	*86.2*
FSSSA	*10.2*	*14.8*	*24.9*	*2.0*	*51.9*

Table 3: CPU times used by four simulation methods on a Mac Core 2 Duo 2.0GHz machine for a budding yeast model. Results were obtained with a simulation of 5×10^5 time units.

We have provided complexity results for our new methods. Numerical results on two biochemical models with multistate species have shown that PNFA and FSSSA are accurate, but have efficiency differences depending on problem characteristics. Following this work, our current research goal is to develop the best simulation strategy for every multistate species in a rule-based model. PNFA treats all multistate species as particles while FSSSA treats them all as populations. An even better strategy might be a hybrid one that treats some of them as particles and the others as populations as appropriate. The work presented here is the first step towards that goal.

As shown in Figure 3, our modeling interface provides support for a simple indexing mechanism, allowing users to express models with notation nearly identical to that of equations (1)–(6) in Section 2. Note that our notation, while supporting behavior such as phosphorylation, is not completely general. We indicate (for example) level of phosphorylation rather than specific phosphorylation bindings, or specific bindings of protein complexes, such as in Equation (9). We have not yet devised notation for expressing the concept of bindings between proteins at one of several explicit sites. In other words, we cannot (for example) support explicit binding at the third, fifth, and sixth phosphorylation sites. There is no fundamental difficulty to support those features. But the current progress in biological modeling areas does not demand this extra complexity, yet. We will extend our system as modelers develop a need for additional features such as this.

REFERENCES

Barik, D., W. Baumann, M. Paul, B. Novak, and J. Tyson. 2010. A model of yeast cell cycle regulation based on multisite phosphorylation. *Mol. Sys. Biol.* Submitted.

Bray, D. 2003, February. Molecular prodigality. *Science* 299:1189–1190.

Chen, K., L. Calzone, A. Csikasz-Nagy, F. Cross, B. Novak, and J. Tyson. 2004. Integrative analysis of cell cycle control in budding yeast. *Mol. Biol. Cell* 15:3841–3862.

Danos, V., J. Feret, W. Fontana, and J. Krivine. 2007. Scalable simulation of cellular signaling networks. *Lect. Notes Comput. Sci.* 4807:139–157.

Faeder, J., M. Blinov, and W. Hlavacek. 2008. Rule-based modeling of biochemical systems with bionetgen. In *Methods in molecular biology: Systems biology*, ed. I. V. Maly, Volume 500, 113–167. Humana Press.

Gillespie, D. 1976. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comput. Phys.* 22:403–434.

Gillespie, D. 1977. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* 81:2340–2361.

Harris, L. A., J. S. Hogg, and J. R. Faeder. 2009. Compartmental Rule-Based Modeling of Biochemical Systems. In *Proc. 2009 Winter Simulation Conf. (WSC)*.

Hlavacek, W., J. Faeder, M. Blinov, A. Perelson, and B. Goldstein. 2003. The complexity of complexes in signal transduction. *Biotechnol. Bioeng.* 84:783–794.

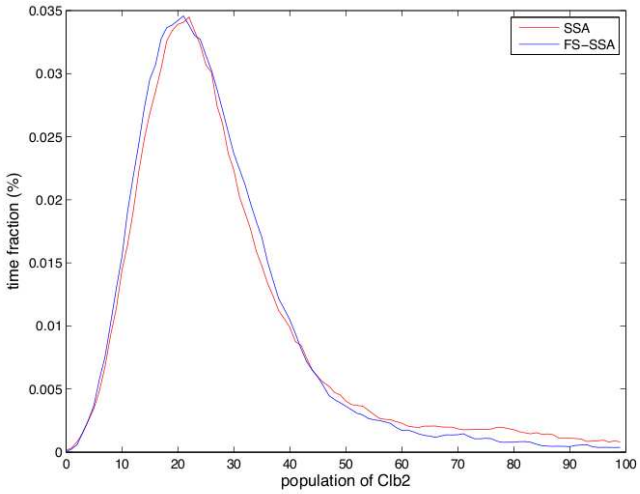


Figure 5: Distributions for *Clb*2 generated by SSA and FSSSA on budding yeast model with fixed cell volume of 26.0.

Hlavacek, W., J. Faeder, M. Blinov, R. Posner, M. Hucka, and W. Fontana. 2006, July. Rules for modeling signal-transduction systems. *Science STKE* 344:re6.

Hucka, M., A. Finney, H. Sauro, and 40 additional authors. 2003. The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics* 19 (4): 524–531.

Kapuy, O., D. Barik, M. Sananes, J. Tyson, and B. Novak. 2009. Bistability by multiple phosphorylation of regulatory protein. *Prog. Biophys. Mol. Biol.* 100:47–56.

Le Novere, N., and A. Oellrich. 2010. Multistate and multicomponent species: A proposal for SMBL Level 3. http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Multistate_and_Multicomponent_Species_Proposal.

Liu, Z., and Y. Cao. 2008. Detailed comparison between the StochSim and SSA. *IET Sys. Bio.* 2:334–341.

Morton-Firth, C., and D. Bray. 1998. Predicting temporal fluctuations in an intracellular signalling pathway. *J. theor. Biol.* 192:117–128.

Morton-Firth, C., T. Shimizu, and D. Bray. 1999. A free-energy-based stochastic simulation of the rat receptor complex. *J. Mole. Bio.* 287:1059–1074.

Novere, N. L., and T. Shimizu. 2001. StochSim: Modeling of stochastic biomolecular processes. *Bioinf.* 17:575–576.

Pettigrew, M., and H. Resat. 2005. Modeling signal transduction networks: A comparison of two stochastic kinetic simulation algorithms. *J. Chem. Phys.* 123:114707.

Qu, Z., J. Weiss, and W. MacLellan. 2003. Regulation of the mammalian cell cycle: A model of the G1-to-S transition. *Am. J. Physiol. Cell Physiol.* 284:C349C364.

Regev, A., E. Panina, W. Silverman, L. Cardelli, and E. Shapiro. 2004. Bioambients: An abstraction for biological compartments. *Theoret. Comp. Sci.* 325:141–167.

Regev, A., and E. Shapiro. 2002. Cellular abstractions: Cells as computation. *Nature* 419:343.

Shaffer, C., J. Zwolak, R. Randhawa, and J. Tyson. 2009. Modeling molecular regulatory networks with JigCell and PET. In *Methods in molecular biology: Systems biology*, ed. I. Maly, Volume 500, 81–111. Humana Press.

Shimizu, T., and D. Bray. 2001. Computational cell biology - the stochastic approach. In *Foundations of Systems Biology*, ed. H. Kitano. Cambidge, MA: MIT Press.

Sneddon, M., W. Pontius, J. Faeder, and T. Emonet. 2008. NFsim: Managing complexity in stochastic simulation of reaction networks. In *The 2nd q-bio conference*. Santa Fe, NM: Los Alamos National Laboratory.

Vass, M., C. Shaffer, N. Ramakrishnan, L. Watson, and J. Tyson. 2006, Apr–Jun. The JigCell Model Builder: a spreadsheet interface for creating biochemical reaction network models. *IEEE/ACM Trans. Comp. Bio. Bioinf.* 3 (2): 155–164.

Yang, J., M. Monine, J. Faeder, and W. Hlavacek. 2008. Kinetic monte carlo method for rule-based modeling of biochemical networks. *Phys. Rev.* 78:031910.

AUTHOR BIBLIOGRAPHIES

ZHEN LIU is currently a PhD student in the Department of Computer Science at Virginia Tech. His email address is <zhenliu@vt.edu>.

CLIFFORD A. SHAFFER is a Professor in the Department of Computer Science at Virginia Tech. He received his PhD from University of Maryland in 1986. His current research interests include problem solving environments, bioinformatics, component architectures, visualization, algorithm design and analysis, and data structures. His email address is <shaffer@cs.vt.edu>.

UMME JUKA MOBASSERA is a PhD student in the Program in Genetics, Bioinformatics, and Computational Biology at Virginia Tech. Her email address is <mobassera@vt.edu>.

LAYNE T. WATSON is a Professor in the Department of Computer Science at Virginia Tech. He received his Ph.D. in mathematics from the University of Michigan, Ann Arbor, in 1974. His current research interests include numerical analysis, nonlinear programming, mathematical software, solid mechanics, fluid mechanics, image processing, parallel computation, and bioinformatics. Dr. Watson is a Fellow of the IEEE and the National Institute of Aerospace. His email address is <ltw@cs.vt.edu>.

YANG CAO is an Assistant Professor in Computer Science at Virginia Tech. He received his Ph.D. degree in computer science from the University of California, Santa Barbara in 2003. His research focuses on the development of multiscale, multiphysics stochastic modeling and simulation methods and tools. His email address is <ycao@cs.vt.edu>.