Automated Abstraction Methodology for Genetic Regulatory Networks*

Hiroyuki Kuwahara[1], Chris J. Myers[1],
Michael S. Samoilov[2], Nathan A. Barker[1], and Adam P. Arkin[2]

[1] University of Utah, Salt Lake City, UT 84112, USA
{kuwahara, myers, barkern}@vlsigroup.ece.utah.edu
[2] Howard Hughes Medical Institute, University of California, Berkeley
Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
{mssamoilov, aparkin}@lbl.gov

Abstract. In order to efficiently analyze the complicated regulatory systems often encountered in biological settings, abstraction is essential. This paper presents an automated abstraction methodology that systematically reduces the small-scale complexity found in genetic regulatory network models, while broadly preserving the large-scale system behavior. Our method first reduces the number of reactions by using rapid equilibrium and quasi-steady-state approximations as well as a number of other stoichiometry-simplifying techniques, which together result in substantially shortened simulation time. To further reduce analysis time, our method can represent the molecular state of the system by a set of scaled Boolean (or n-ary) discrete levels. This results in a chemical master equation that is approximated by a Markov chain with a much smaller state space providing significant analysis time acceleration and computability gains. The genetic regulatory network for the phage λ lysis/lysogeny decision switch is used as an example throughout the paper to help illustrate the practical applications of our methodology.

1 Introduction

Numerous methods have been proposed for modeling genetic regulatory networks [1,2]. While many traditional approaches have relied on a differential equation representation as inferred from a set of underlying biochemical reactions, there has been a growing appreciation of their limitations [3,4,5,6]. In particular, differential equation analysis of genetic networks generally assumes that the number of molecules in a cell is high and their concentrations can be viewed as continuous quantities, while the underlying reactions are presumed to occur deterministically. However, in genetic networks these assumptions frequently do not hold. For example, DNA molecules are typically present in single digit quantities while some promoters can lead to substantial fluctuations in transcription/translation rates and essentially non-deterministic expression characteristics [7,8].

* This material is based upon work supported by the National Science Foundation under Grant No. 0331270.

C. Priami and G. Plotkin (Eds.): Trans. on Comput. Syst. Biol. VI, LNBI 4220, pp. 150–175, 2006.

In these situations, accurate genetic regulatory network modeling requires the use of a discrete and stochastic process description such as the *master equation formalism* [9]. This approach describes well-stirred (bio)chemical systems at the individual reaction level by exactly tracking the quantities of each molecular species and treating each reaction as a separate random event. It further allows the exact discrete simulation of system behavior via Gillespie's *Stochastic Simulation Algorithm* (SSA) [10]. Unfortunately, the computational requirements for such simulations are generally practical only for relatively small systems with no major reaction time-scale separations. Even then, the computational demands can be very high. For example, Arkin et al.'s numerical analysis of the phage λ decision network using a hybrid stochastic kinetic and statistical-thermodynamic model required a 200-node supercomputer [3].

Several other techniques for reducing the computational costs of stochastic simulations have been proposed [11]. For instance, Gibson and Bruck [12] developed a method to improve Gillespie's first reaction method [10] by reducing the random numbers generated allowing them to simulate the λ network on 10 desktop workstations [13]. While this method does improve efficiency significantly for systems with many species and reaction channels, every reaction event must still be simulated one at a time. Ultimately, given the substantial computational requirements of stochastic simulations and comparative complexities of *in situ* genetic regulatory networks, abstraction is absolutely essential for efficient analysis of any realistic system. This abstraction can be achieved either during the simulation or the model generation stage. An example of abstraction during simulation is Gillespie's explicit τ-leaping method [14]. This method approximates the number of firings of each reaction in a pre-defined interval rather than executing each reaction individually. While this and similar methods [15,16,17] are very promising, they may not perform well for systems with rapidly changing reaction rates, such as those driven by small molecular counts as frequently encountered in gene-regulatory systems. Rao and Arkin performed model abstraction by using (bio)chemical insight in combination with the *quasi-steady-state assumption* to remove fast reactions (thus reducing the problem dimensionality), and then applied a modified version of SSA to the simplified model [18]. Since this method is not automated, it is difficult to apply to reactions within complex biological networks. Furthermore, the quasi-steady-state approximation only represents one of many approximations that can be used to estimate the dynamics of a biological reaction network.

This paper presents a generalized and automated model abstraction methodology that systematically reduces the small scale complexity found in *REaction-Based* (REB) models of genetic regulatory networks (i.e., models composed of a set of chemical reactions), while broadly preserving the large scale system behavior. Notably, though many model reductions of (bio)chemical networks have traditionally been performed manually (i.e., "by hand") this practice is time-consuming and is susceptible to errors in large model transformations. For example, whereas the quasi-steady-state approximation has long been carried out to reduce the complexity of biochemical networks, this practice can result in inaccuracies when the underlying hypothesis of the approximation is ignored and the application of the

quasi-steady-state approximation is inappropriate [19]. Thus, by automating and systematizing this process, our method is able to significantly ameliorate the problem of computability by lowering the model translation error rate and improving simulation times by reducing system complexity.

Our methodology—outlined in Figure 1—begins with a REB model, which could be simulated using SSA or one of its variants though at a substantial computational cost. To reduce the cost of simulation, this paper describes an automatic method that we have implemented for simplifying the original REB model by applying several abstraction methods that leverage the *rapid equilibrium* and quasi-steady-state assumptions. The result is a new abstracted model with less reactions and species which substantially lowers the cost of stochastic simulation. To further reduce the complexity of the system as well as analysis time, this simplified REB model can be automatically translated into a *Stochastic Asynchronous Circuit* (SAC) model by encoding chemical species molecule counts into Boolean (or n-ary) levels. This model can then be efficiently analyzed using the Markov chain analysis method within the asynchronous circuit tool `ATACS` [20].

This paper further exemplifies the applicability of our model abstraction methodology to genetic regulatory networks by applying it to the phage λ developmental decision circuit [3]. Hence, various abstractions in this work target the specific features found in genetic regulatory networks to reduce the analysis time. For example, one abstraction method is used to reduce the low-level description of the interactions of transcription factors and *cis*-regulatory elements which, as noted earlier, are elements that are typically present in small quantity and therefore susceptible to the discrete-stochastic effect. This abstraction substantially accelerates the computation that otherwise would require an expensive non-deterministic treatment. Moreover, the ON-OFF switching behavior often found in genetic regulatory networks is suitable for our abstraction methodology, especially for a SAC model generation. Thus, whereas we believe that our model abstraction methodology can in theory be applied to any (bio)chemical networks, this paper concentrates on genetic regulatory networks as a proof of concept to evaluate our methodology which includes abstraction methods tailored for such networks.

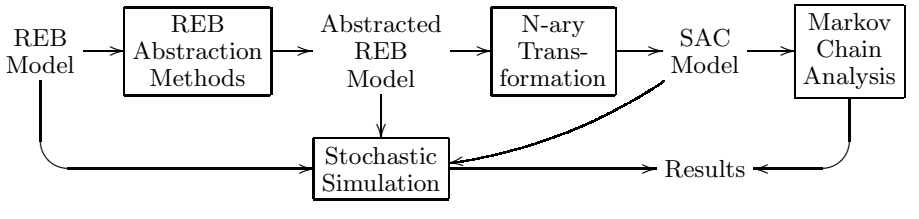

Fig. 1. Automated model abstraction tool flow

2 Reaction-Based Abstraction Methods

In chemical and biological molecular systems, including genetic regulatory networks, reaction-based representations typically provide the most detailed level

of specification for the underlying system structure and dynamics [21]. Reaction-based abstraction methods are used to reduce a REB model's size by merging reactions, removing irrelevant reactions, etc. We have implemented several such techniques, each traversing the graph structure of the REB model and applying transformations to it when the respective conditions are satisfied. The result is a new REB model with fewer reactions and/or species. This section first describes the REB model formally, and then presents several abstraction methods as well as discusses how they are applied to our λ model. Finally, this section presents simulation results for the λ model before and after these abstractions.

2.1 Reaction-Based Model

The REB model is a bipartite weighted directed graph that connects species based on the interactions that they can have via a set of reaction channels.[1]

Definition 1. *A REB model is specified with a 5-tuple* $\langle \mathbf{S}, \mathbf{R}, \mathbf{R_{rev}}, \mathbf{E}, \mathbf{K} \rangle$ *where* $\mathbf{S}$ *is the set of species nodes,* $\mathbf{R}$ *is the set of reaction nodes,* $\mathbf{R_{rev}} \subseteq \mathbf{R}$ *is the set of reversible reactions,* $\mathbf{E} : ((\mathbf{S} \times \mathbf{R}) \cup (\mathbf{R} \times \mathbf{S})) \rightarrow \mathbb{N}$ *is a function that returns the stoichiometry of the species with respect to a reaction, and* $\mathbf{K} : \mathbf{R} \rightarrow (\mathbb{R}^{|\mathbf{S}|} \rightarrow \mathbb{R})$ *are the kinetic rate laws for the reactions.*

For example, a REB model with a single reversible reaction r_1 of the form:

$$2s_1 + s_2 \underset{k_{-1}}{\overset{k_1}{\rightleftarrows}} s_1 + s_2 + s_3$$

contains the following sets:

$$\begin{aligned}
\mathbf{S} &= \{s_1, s_2, s_3\}, \\
\mathbf{R} &= \{r_1\}, \\
\mathbf{R_{rev}} &= \{r_1\}, \\
\mathbf{E} &= \{((s_1, r_1), 2), ((s_2, r_1), 1), ((s_3, r_1), 0), \\
&\quad ((r_1, s_1), 1), ((r_1, s_2), 1), ((r_1, s_3), 1)\}, \\
\mathbf{K} &= \{(r_1 \rightarrow (([s_1], [s_2], [s_3]) \rightarrow (k_1[s_1]^2[s_2] - k_{-1}[s_1][s_2][s_3])))\}.
\end{aligned}$$

In the kinetic rate law, $[s]$ is a variable that represents the state of species s. Note that $[s]_0$ is used to denote the initial number of molecules of species s. Also note that the user can specify a set of *interesting species*, (i.e., $\mathbf{S_i} \subseteq \mathbf{S}$), which should never be abstracted, so that they can be analyzed.

If a reaction consumes a species, then that species is a *reactant* for that reaction. If a reaction produces a species, then that species is a *product* for that reaction. If a species is neither produced nor consumed by a reaction, then it is a *modifier* to that reaction. $\mathbf{R_s^r}$, $\mathbf{R_s^p}$, and $\mathbf{R_s^m}$ are the sets of reactions in which

[1] A REB model can be described via an emerging standard, the *Systems Biology Markup Language* (SBML) [22]. Graphical user interface tools such as BioSPICE's PathwayBuilder [23] allow researchers a convenient mechanism to create such models.

species s appears as a reactant, product, and modifier, respectively. Similarly, $\mathbf{S_r^r}$, $\mathbf{S_r^p}$, and $\mathbf{S_r^m}$, are the sets of species that appear in reaction r as a reactant, product, and modifier, respectively. These sets are defined formally below:

$$\begin{aligned}
\mathbf{R_s^r} &= \{r \in \mathbf{R} \mid \mathbf{E}(s,r) > \mathbf{E}(r,s)\},\\
\mathbf{R_s^p} &= \{r \in \mathbf{R} \mid \mathbf{E}(r,s) > \mathbf{E}(s,r)\},\\
\mathbf{R_s^m} &= \{r \in \mathbf{R} \mid \mathbf{E}(s,r) > 0 \wedge \mathbf{E}(s,r) = \mathbf{E}(r,s)\},\\
\mathbf{S_r^r} &= \{s \in \mathbf{S} \mid \mathbf{E}(s,r) > \mathbf{E}(r,s)\},\\
\mathbf{S_r^p} &= \{s \in \mathbf{S} \mid \mathbf{E}(r,s) > \mathbf{E}(s,r)\},\\
\mathbf{S_r^m} &= \{s \in \mathbf{S} \mid \mathbf{E}(s,r) > 0 \wedge \mathbf{E}(s,r) = \mathbf{E}(r,s)\}.
\end{aligned}$$

Note that in these definitions a species is considered a reactant of a reaction only if the net change of the state of that species by a reaction is negative. Similarly, it is a product only if the net change of the state of that species by a reaction is positive. Finally, it is considered a modifier if it is used in a reaction but the state of that species is not changed by that reaction. Our example includes the following nonempty sets:

$$\begin{aligned}
\mathbf{R_{s_1}^r} &= \{r_1\},\\
\mathbf{R_{s_3}^p} &= \{r_1\},\\
\mathbf{R_{s_2}^m} &= \{r_1\},\\
\mathbf{S_{r_1}^r} &= \{s_1\},\\
\mathbf{S_{r_1}^p} &= \{s_3\},\\
\mathbf{S_{r_1}^m} &= \{s_2\}.
\end{aligned}$$

Sections 2.2 to 2.5 describe general abstraction methods that substantially reduce the model complexity. Section 2.6 describes how these abstraction methods can be combined together and applied to reduce the complexity of REB models. Finally, Section 2.7 shows the improvements gained by our approach.

2.2 Michaelis-Menten Approximation

Consider the following elementary enzymatic reactions where E is an enzyme, S is a substrate, C is a complex form of E and S, and P is a product.

$$\mathrm{E} + \mathrm{S} \underset{k_{-1}}{\overset{k_1}{\rightleftarrows}} \mathrm{C} \xrightarrow{k_2} \mathrm{E} + \mathrm{P} \tag{1}$$

If the complex C dissociates into E and S much faster than it is converted into the product P (i.e., $k_{-1} >> k_2$), then the substrate can be assumed to be in rapid equilibrium with the complex. In this case, these reactions can be transformed to the following *Michaelis-Menten* (MM) form:

$$\mathrm{S} \xrightarrow{\frac{k_2 E_{tot} K_1 [S]}{1 + K_1 [S]}} \mathrm{P}\ , \tag{2}$$

where E_{tot} is the total concentration of E and C, and K_1.

Using this approximation, a REB model can be reduced by searching for patterns matching the reaction given by Expression 1 above with rate laws that satisfy the conditions. When such a pattern is found and the species matched to C only appears in the reactions in this pattern, all the reactions in the pattern are removed from the model along with the complex C. Finally, a new reaction is introduced with the form given in Expression 2 above.

The rapid equilibrium approximation has two advantages: (1) the state space of the process is reduced since intermediate species are eliminated, (2) simulation time is reduced by removing fast reactions. Figure 2 shows a graphical representation of a more complex competitive enzymatic reaction from Arkin et al.'s phage λ model [3]. In Figure 2(a), proteins *CII* and *CIII* compete for binding to protease *P1*—producing complexes *P1·CII* and *P1·CIII*, respectively. (Note that each reaction node that is connected to a species with a double arrow is a shorthand way of showing a reversible reaction.) In this complex form, this protease acts to degrade *CII* and *CIII*. An extended form of the rapid equilibrium approximation can be applied to this network to remove this protease, its complex forms, and the reactions that form these complexes. Importantly, this also clarifies the essential biological meaning of the process by removing intermediate steps, which may otherwise obscure the functional logic of the mechanism. The resulting abstracted reaction model is shown in Figure 2(b).

The algorithms shown in Figure 3 implement the rapid equilibrium approximation for multiple alternative substrate systems which is a generalization of the complete characterization of enzyme-substrate and enzyme-substrate-competitor reactions [24]. First, Algorithm 1 considers each species, s, as a potential enzyme. Each species is checked using Algorithm 2. If s is an interesting species or does not occur as a reactant in any reaction, then s is not considered further (line 2). Otherwise, each reaction, r_1, in which s is a reactant is considered in turn. If r_1 is not reversible, does not have two reactants, or does not have a rate law of the right form (i.e., $k_f[s][s_1] - k_r[s_c]$), then again s is not considered further (line 4). Reaction r_1 combines s and s_1 into a complex, s_c. If the initial molecule count of this complex is not 0, s_c is an interesting species, s_c does not occur as a reactant or product in exactly one reaction, or occurs as a modifier in any, then again this approximation is terminated for s (lines 5 and 6). The reaction r_2 converts s_c into a product and releases the enzyme s. If this reaction is reversible, does not have exactly one reactant and no modifiers, does not have s as a product, has more than two products, or does not have a rate law of the form, $k_2[s_c]$, then s is not considered further (lines 7-9). Finally, it checks the validity of the rapid equilibrium assumption by comparing the ratio of the product dissociation rate constant and the substrate dissociation rate constant to the predefined threshold constant T_1 (line 10). For each reaction, a configuration is formed that includes the substrate s_1, complex s_c, equilibrium constant k_f/k_r, production rate k_2, complex forming reaction r_1, and product forming reaction r_2 (line 11). If Algorithm 2 terminates successfully (i.e., returns a nonempty set of configurations, $\mathbf{C}$), then Algorithm 3 is called to apply the transformation to the REB model. First,

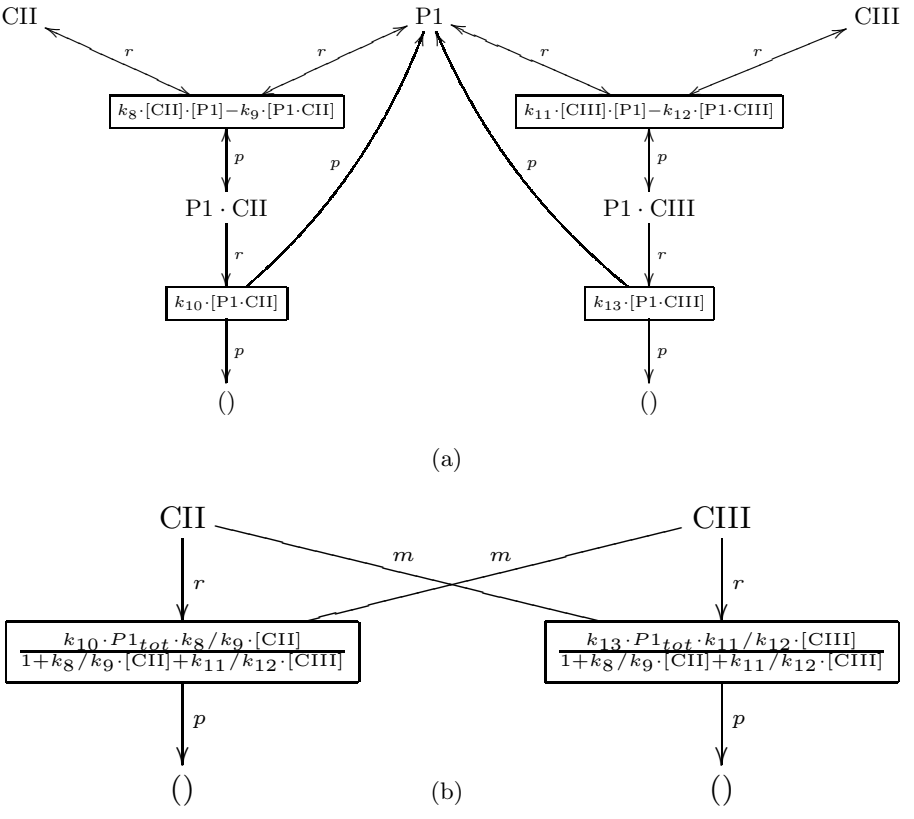

Fig. 2. Rapid equilibrium approximation: (a) original model, and (b) abstracted model

it loops through the set of configurations to form an expression that is used in the denominator in each new rate law as well as forming a list of all the substrates that bind to the enzyme s (lines 1-6). Next, for each configuration $(s_1, s_c, K_1, k_2, r_1, r_2)$, it makes the substrate s_1 a reactant for r_2, makes all other substrates modifiers for r_2, creates a new rate law for r_2, and removes species s_c and reaction r_1 (lines 7-13). Finally, this algorithm removes the enzyme, s (line 14).

When k_{-1} is not much greater than k_2, the rapid equilibrium approximation cannot be applied. In such cases, however, if the total concentration of the enzyme is much less than the sum of the initial concentration of the substrate and the MM constant (i.e., $[E]_0 << [S]_0 + K_M$ where $K_M = (k_{-1} + k_2)/k_1$), then the *standard quasi-steady-state approximation* can be used instead to transform an enzymatic one-substrate reaction to the MM form with a somewhat more complex kinetic rate law (i.e., K_1 in Expression 2 is replaced with ${K_M}^{-1}$).

Algorithm 1. *Rapid equilibrium approximation*
Model RapidEqApprox(Model M)

1: ***for all*** $s \in \mathbf{S}$ ***do***
2: $\quad \mathbf{C} \leftarrow RapidEqConditionSatisfied(M, s)$
3: $\quad$ ***if*** $\mathbf{C} \neq \emptyset$ ***then*** $M \leftarrow RapidEqTransform(M, s, \mathbf{C})$
4: ***end for***
5: ***return*** M

Algorithm 2. *Check the conditions for rapid equilibrium approximation*
Configs RapidEqConditionSatisfied(Model M, Species s)

1: $\mathbf{C} \leftarrow \emptyset$
2: ***if*** $(s \in \mathbf{S_i}) \vee (|\mathbf{R^r_s}| = 0)$ ***then return*** $\emptyset$
3: ***for all*** $r_1 \in \mathbf{R^r_s}$ ***do***
4: $\quad$ ***if*** $(r_1 \notin \mathbf{R_{rev}}) \vee (|\mathbf{S^r_{r_1}}| \neq 2) \vee (\mathbf{K}(r_1) \neq$ "$k_f[s][s_1] - k_r[s_c]$"$)$ ***then return*** $\emptyset$
5: $\quad$ ***if*** $([s_c]_0 \neq 0) \vee (s_c \in \mathbf{S_i})$ ***then return*** $\emptyset$
6: $\quad$ ***if*** $(|\mathbf{R^r_{s_c}}| \neq 1) \vee (|\mathbf{R^m_{s_c}}| \neq 0) \vee (\mathbf{R^p_{s_c}}| \neq 1)$ ***then return*** $\emptyset$
7: $\quad \{r_2\} \leftarrow \mathbf{R^r_{s_c}}$
8: $\quad$ ***if*** $(r_2 \in \mathbf{R_{rev}}) \vee (|\mathbf{S^r_{r_2}}| \neq 1) \vee (|\mathbf{S^m_{r_2}}| \neq 0)$ ***then return*** $\emptyset$
9: $\quad$ ***if*** $(s \notin \mathbf{S^p_{r_2}}) \vee (|\mathbf{S^p_{r_2}}| \notin \{1, 2\}) \vee (\mathbf{K}(r_2) \neq k_2[s_c])$ ***then return*** $\emptyset$
10: $\quad$ ***if*** $k_2/k_r > T_1$ ***then return*** $\emptyset$
11: $\quad \mathbf{C} \leftarrow \mathbf{C} \cup \{(s_1, s_c, k_f/k_r, k_2, r_1, r_2)\}$
12: ***end for***
13: ***return*** $\mathbf{C}$

Algorithm 3. *Perform the rapid equilibrium approximation*
Model RapidEqTransform(Model M, Species s, Configs $\mathbf{C}$*)*

1: *kinetic law expression* $Z \leftarrow 1$
2: $\mathbf{L} \leftarrow \emptyset$
3: ***for all*** $(s_1, s_c, K_1, k_2, r_1, r_2) \in \mathbf{C}$ ***do***
4: $\quad Z \leftarrow Z + (K_1 * [s_1])$
5: $\quad \mathbf{L} \leftarrow \mathbf{L} \cup \{s_1\}$
6: ***end for***
7: ***for all*** $(s_1, s_c, K_1, k_2, r_1, r_2) \in \mathbf{C}$ ***do***
8: $\quad M \leftarrow addReactant(M, s_1, r_2, \mathbf{E}(s_1, r_1))$
9: $\quad \forall m \in \mathbf{L} \setminus \{s_1\}.\ M \leftarrow addModifier(M, m, r_2)$
10: $\quad \mathbf{K}(r_2) \leftarrow (k_2 * [s]_0 * k_e * [s_1])/Z$
11: $\quad M \leftarrow removeSpecies(M, s_c)$
12: $\quad M \leftarrow removeReaction(M, r_1)$
13: ***end for***
14: $M \leftarrow removeSpecies(M, s)$
15: ***return*** M

Fig. 3. Algorithms to perform rapid equilibrium approximation

2.3 Operator Site Reduction

REB models of genetic networks generally include multiple operator sites which transcription factors may occupy. It is often the case that the rates at which transcription factors bind and unbind to these operator sites are rapid with

respect to the rate of *open complex formation* (i.e., initiation of transcription). It is also typically the case that the number of operator sites is much smaller than the number of *RNA polymerase* (RNAP) and transcription factor molecules. Therefore, a method similar to rapid equilibrium approximation called *operator site reduction* can be used to systematically merge reactions and remove operator sites and their complexes from REB models. Note that this method may also be applicable to other molecular scaffolding systems such as those found in signal transduction networks.

The first step in this transformation is to identify operators within the REB model. This is done by assuming that an operator is a species small in number that is neither produced nor degraded. Suppose our algorithm has identified an operator O, and there are $N + 1$ configurations in which transcription factors and RNAP can bind to it. Let O_i, K_i, and X_i with $i \in [1, N]$, be the i-th bound complex of the operator O, the equilibrium constant for forming this configuration, and the product of the concentrations of the substrates for each component of the complex in this configuration, respectively. Let O_0 be the operator in free form (i.e., not bound to anything). Let C_i with $i \in [0, N]$ be each of the operator configurations. Then, assuming rapid equilibrium, the probability of this operator being in each configuration is:

$$Pr(C_i) = \begin{cases} \frac{1}{Z} & \text{if } i = 0 \\ \frac{K_i \cdot X_i}{Z} & \text{if } 1 \leq i \leq N \end{cases}$$

where $Z = 1 + \sum_{j=1}^{N} K_j \cdot X_j$. This probability is the same as the equilibrium statistical thermodynamic model when $K_i = exp(\Delta G_i / RT)$ where ΔG_i is the relative free energies for the i-th configuration, R is the gas constant, and T is the absolute temperature [25]. Assuming that $O_{tot} = [O_0]_0$, then $[O_i] = Pr(C_i)O_{tot}$ is the fraction of operators in the i-th configuration.

Figure 4(a) shows the graphical representation of a portion of a REB model for the P_{RE} promoter from the phage λ decision network [3]. The top three reactions involve the binding of *RNAP* and *CII* to P_{RE} and the bottom two reactions result in the production of 10 molecules of the protein *CI*. In this example, there are 4 configurations of the operator, namely, *PRE*, *PRE·RNAP*, *PRE·CII·RNAP*, and *PRE·CII*. Assuming that the operator-binding and unbinding rates are much faster than those of open complex formation, our method can apply operator site reduction. Figure 4(b) is the result of applying this abstraction method to Figure 4(a). The result has only three species and two reactions. The transformed model represents the probability of P_{RE} being in a configuration that results in production of *CI* instead of modeling every binding and unbinding of transcription factors and *RNAP* to the promoter precisely. After performing operator site reduction on all of the operators, the resulting two reactions are found to be structurally similar and can be further combined into a single reaction by applying our *similar reaction combination* method. Also, after performing operator site reduction on all of the operators, *RNAP* only appears as a modifier in every reaction. In this case, another abstraction method—known as *modifier constant propagation*—can be applied. Namely, in each kinetic law

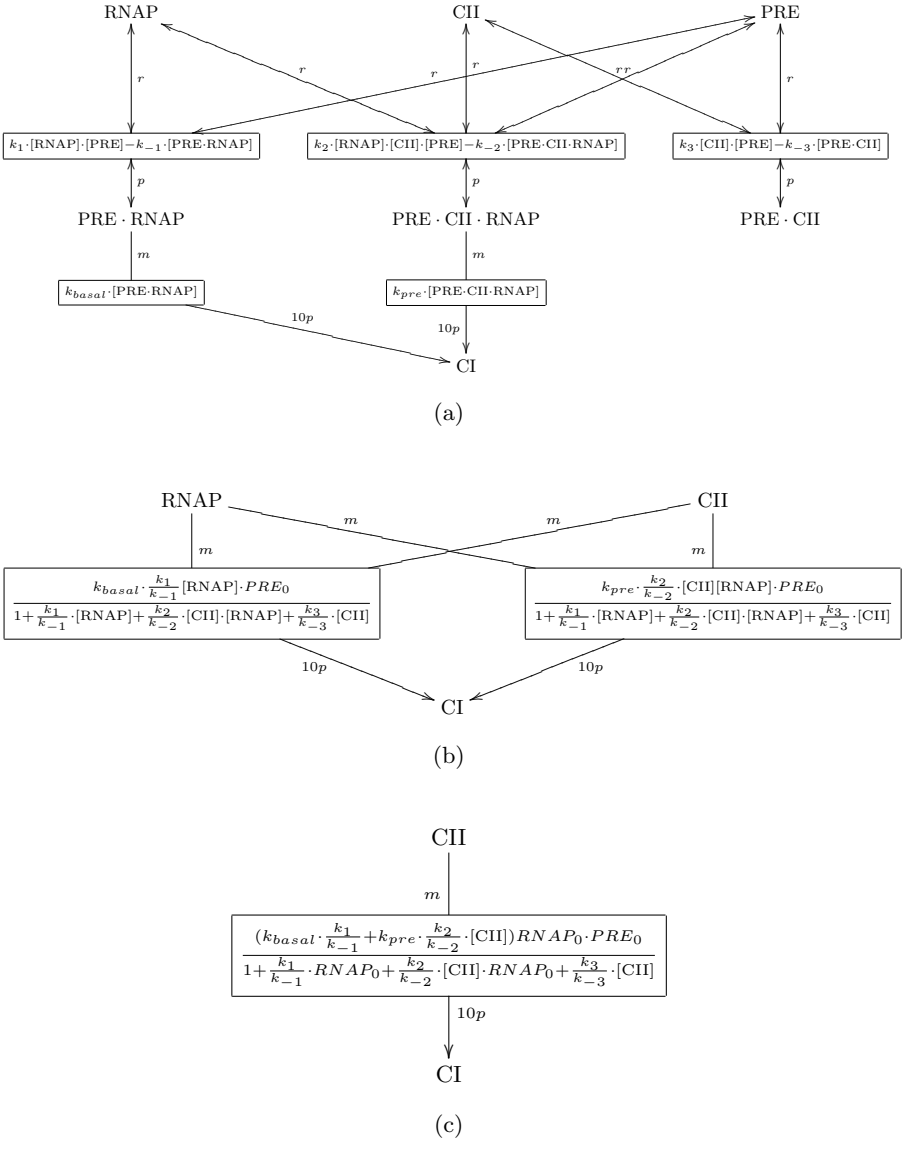

Fig. 4. Operator site reduction: (a) original model, (b) abstracted model, and (c) after similar reaction combination and modifier constant propagation

in which $[RNAP]$ appears, it can be replaced with the constant, $RNAP_0$, which is the initial molecule count of *RNAP*. The result after both of these steps is shown in Figure 4(c).

The algorithms shown in Figure 5 implement operator site reduction. First, Algorithm 4 considers each species, s, as a potential operator site. Each species

Algorithm 4. *Operator site reduction*
Model OpSiteReduction(Model M)

for all $s \in \mathbf{S}$ *do*
  $\mathbf{C} \leftarrow OpSiteConditionSatisfied(M, s)$
  if $\mathbf{C} \neq \emptyset$ *then* $M \leftarrow OpSiteTransform(M, s, \mathbf{C})$
end for
return M

Algorithm 5. *Check the conditions for operator site reduction*
Configs OpSiteConditionSatisfied(Model M, Species s)

$\mathbf{C} \leftarrow \emptyset$
if $[s]_0 > maxOperatorThreshold$ *then return* $\emptyset$
if $(s \in \mathbf{S_i}) \vee (|\mathbf{R_s^p}| \neq 0)$ *then return* $\emptyset$
for all $r_1 \in \mathbf{R_s^r}$ *do*
  if $(r_1 \notin \mathbf{R_{rev}}) \vee (|\mathbf{S_{r_1}^r}| < 2) \vee (|\mathbf{S_{r_1}^p}| \neq 1)$ *then return* $\emptyset$
  if $(\mathbf{K}(r_1) \neq$ "$k_f \prod_{s' \in \mathbf{S_{r_1}^r}} [s']^{\mathbf{E}(s', r_1)} - k_r[s_c]$") *then return* $\emptyset$
  if $(s_c \in \mathbf{S_i}) \vee (|\mathbf{R_{s_c}^p}| \neq 1) \vee (|\mathbf{R_{s_c}^r}| \neq 0)$ *then return* $\emptyset$
  for all $r_2 \in \mathbf{R_{s_c}^m}$ *do*
    if $(|\mathbf{S_{r_2}^r}| \neq 0) \vee (|\mathbf{S_{r_2}^m}| \neq 1) \vee (|\mathbf{S_{r_2}^p}| \neq 1) \vee (\mathbf{K}(r_2) \neq k_2[s_c])$ *then return* $\emptyset$
  end for
  $e \leftarrow (k_f/k_r) \prod_{s' \in (\mathbf{S_{r_1}^r} \setminus \{s\})} [s']^{\mathbf{E}(s', r_1)}$
  $\mathbf{C} \leftarrow \mathbf{C} \cup \{(s_c, e, r_1)\}$
end for
return $\mathbf{C}$

Algorithm 6. *Perform transformation for operator site reduction*
Model OpSiteTransform(Model M, Species s, Configs $\mathbf{C}$*)*

kinetic law expression $Z \leftarrow 1$
$\mathbf{L} \leftarrow \emptyset$
for all $(s_c, e, r_1) \in \mathbf{C}$ *do*
  $Z \leftarrow Z + e$
  $\mathbf{L} \leftarrow \mathbf{L} \cup \mathbf{S_{r_1}^r}$
end for
for all $(s_c, e, r_1) \in \mathbf{C}$ *do*
  for all $r_2 \in \mathbf{R_{s_c}^m}$ *with* $\mathbf{K}(r_2) = k_2[s_c]$ *do*
    $\forall m \in \mathbf{L}$. $M \leftarrow addModifier(M, m, r_2)$
    $\mathbf{K}(r_2) \leftarrow (k_2 * [s]_0 * e)/Z$
  end for
  $M \leftarrow removeSpecies(M, s_c)$
  $M \leftarrow removeReaction(M, r_1)$
end for
$M \leftarrow removeSpecies(M, s)$
return M

Fig. 5. Algorithms for operator site reduction

is checked using Algorithm 5. First, it is assumed that the molecule count of operator sites is small, so if s has an initial molecule count greater than a given threshold, then it is assumed not to be an operator site (line 2). Next, if s is an interesting species or occurs as a product in any reaction, then s is not considered further (line 3). Otherwise, each reaction, r_1, in which s is a reactant is considered in turn. If r_1 is not reversible, has less than two reactants, does not have exactly one product, or does not have a rate law of the right form, then again s is not considered further (lines 5 and 6). Each reaction, r_1, combines the potential operator site, s, with RNAP and/or transcription factors forming a complex, s_c. If s_c is an interesting species, s_c does not occur as a product in exactly one reaction, or occurs as a reactant in any, then again this approximation is terminated for s (line 7). The species s_c may appear as a modifier in any number of reactions that result in the transcription and translation of proteins. Each of these reactions, r_2, is checked that it has no reactants, only one modifier, only one product, and a rate law of the right form (lines 8-10). For each complex, s_c, a configuration is formed that includes the complex s_c, an equilibrium expression for this configuration, and the complex forming reaction r_1 (lines 11 and 12). If Algorithm 5 terminates successfully, then Algorithm 6 is called to apply the transformation to the REB model. This algorithm is very similar to the one for rapid equilibrium. Again, it loops through the set of configurations to form an expression that is used in the denominator in each new rate law as well as forming a list of all the transcription factors that bind to the operator site s (lines 1-6). Next, it considers each configuration, (s_c, e, r_1). For each reaction r_2 in which s_c appears as a modifier, it adds all the transcription factors as modifiers and creates a new rate law for r_2 (lines 8-11). It then removes species s_c and reaction r_1 from the model (lines 12-13). Finally, at the end, this algorithm removes the operator site, s (line 15).

2.4 Dimerization Reduction

Dimerization is another type of reaction, which often involves only regulatory molecules and could thus frequently proceed very rapidly compared to the rate of transcription initiation. Therefore, it might also be useful to abstract away these reactions whenever possible by using a version of rapid-equilibrium constraints. The dimerization reduction method is used to express dimer and monomer forms of species in terms of their total concentration as follows [26]. Let us consider a species s_m that forms a dimer s_d. If $[s_t]$ is the total number of the monomer molecules, then it is defined as follows:

$$[s_t] = [s_m] + 2[s_d]. \tag{3}$$

Let K_e be the equilibrium constant for dimerization (i.e., $K_e = k_+/k_-$), then, by assuming s_m and s_d are in rapid equilibrium, we have

$$[s_d] = K_e[s_m]^2. \tag{4}$$

Using Equations 3 and 4, we can derive the following equation:

$$K_e[s_t]^2 - (4K_e[s_t] + 1)[s_d] + 4K_e[s_d]^2 = 0. \tag{5}$$

Solving Equation 5, we can express $[s_m]$ and $[s_d]$ in terms of $[s_t]$ as follows:

$$[s_m] = \frac{1}{4K_e}\left(\sqrt{8K_e[s_t]+1}-1\right), \tag{6}$$

$$[s_d] = \frac{[s_t]}{2} - \frac{1}{8K_e}\left(\sqrt{8K_e[s_t]+1}-1\right). \tag{7}$$

As an example, consider the reactions for the species CI from the phage λ decision network shown in Figure 6(a). This species can only effectively degrade in the monomer form (reaction r_1), but it is transcriptionally active (reactions r_3 and r_4) only as a dimer (reaction r_2). Using Equations 6 and 7, the reactions r_1, r_3, and r_4 can be transformed to $r_{1'}$, $r_{3'}$, and $r_{4'}$, respectively, with kinetic laws that are now all expressed in terms of total amount of CI as shown in Figure 6(b). Note that the dimerization reaction r_2 is eliminated completely.

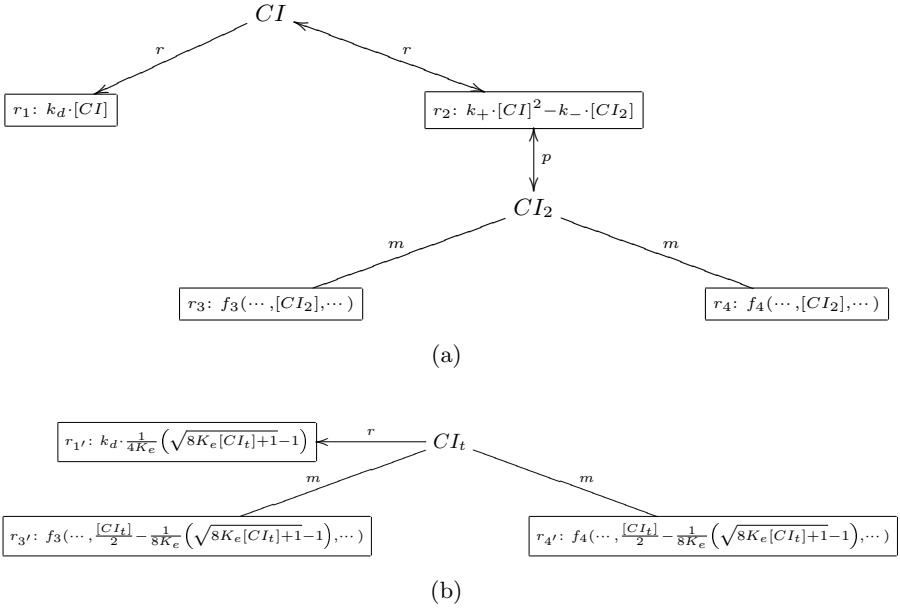

Fig. 6. Dimerization reduction: (a) original model, and (b) abstracted model

The algorithms to perform dimerization reduction are shown in Figure 7. First, Algorithm 7 is used to identify a dimerization reaction. It checks each reaction, r, using Algorithm 8. A dimerization reaction must include exactly one reactant, one product, and no modifiers (line 1). It must also have a rate law of the right form (line 2). The dimerization reduction also requires that the monomer is never used as a modifier, and that there is only one reaction (this one) which produces the dimer (lines 3-4). If these conditions are met, a record is made of the monomer species s_m, dimer species s_d, and equilibrium constant

k_+/k_- (line 5). The transformation is performed by Algorithm 9. First, a new species s_t is introduced into the model with an initial concentration $[s_m]_0+2[s_d]_0$ (lines 1-2). Next, s_m is replaced by s_t in each reaction in which s_m is a reactant, and the rate law is updated as described in Equation 6 (lines 3-6). The dimer s_d is also replaced with s_t in the reactions that it appears as a reactant or modifier, and the rate law is updated using Equation 7 (lines 7-11). Finally, the species s_m, the species s_d, and reaction r are all removed from the model (lines 12-14).

Algorithm 7. *Dimerization reduction*
Model DimerReduction(Model M)

1: ***for all*** $r \in \mathbf{R_{rev}}$ ***do***
2: $\quad C \leftarrow DimerConditionSatisfied(M, r)$
3: $\quad$ ***if*** $C \neq$ **nil** ***then*** $M \leftarrow DimerTransform(M, r, C)$
4: ***end for***
5: ***return*** M

Algorithm 8. *Check the conditions for the dimerization reduction*
Record DimerConditionSatisfied(Model M, Reaction r)

1: ***if*** $(|\mathbf{S_r^r}| \neq 1) \vee (|\mathbf{S_r^p}| \neq 1) \vee (|\mathbf{S_r^m}| \neq 0)$ ***then return*** **nil**
2: ***if*** $(\mathbf{K}(r) \neq$ *"*$k_+[s_m]^2 - k_-[s_d]$*"*$)$ ***then return*** **nil**
3: $\{s_m\} \leftarrow \mathbf{S_r^r}$ *and* $\{s_d\} \leftarrow \mathbf{S_r^p}$
4: ***if*** $(|\mathbf{R_{s_m}^m}| \neq 0) \vee (|\mathbf{R_{s_d}^p}| \neq 1)$ ***then return*** **nil**
5: ***return*** $\langle s_m, s_d, k_+/k_- \rangle$

Algorithm 9. *Perform the dimerization reduction transformation*
Model DimerTransform(Model M, Reaction r, Record $\langle s_m, s_d, K_e \rangle$*)*

1: $M \leftarrow addSpecies(M, s_t)$
2: $[s_t]_0 \leftarrow [s_m]_0 + 2[s_d]_0$
3: ***for all*** $r' \in \mathbf{R_{s_m}^r}$ ***do***
4: $\quad M \leftarrow addReactant(M, s_t, r', E(s_m, r'))$
5: $\quad$ *replace* $[s_m]$ *with* $\frac{1}{4K_e}\left(\sqrt{8K_e[s_t]+1}-1\right)$ *in* $\mathbf{K}(r')$
6: ***end for***
7: ***for all*** $\forall r' \in (\mathbf{R_{s_d}^r} \cup \mathbf{R_{s_d}^m})$ ***do***
8: $\quad$ ***if*** $r' \in \mathbf{R_{s_d}^r}$ ***then*** $M \leftarrow addReactant(M, s_t, r', E(s_d, r'))$
9: $\quad$ ***if*** $r' \in \mathbf{R_{s_d}^m}$ ***then*** $M \leftarrow addModifier(M, s_t, r')$
10: $\quad$ *replace* $[s_d]$ *with* $\frac{[s_t]}{2} - \frac{1}{8K_e}\left(\sqrt{8K_e[s_t]+1}-1\right)$ *in* $\mathbf{K}(r')$
11: ***end for***
12: $M \leftarrow removeSpecies(M, s_m)$
13: $M \leftarrow removeSpecies(M, s_d)$
14: $M \leftarrow removeReaction(M, r)$
15: ***return*** M

Fig. 7. Algorithms to perform dimerization reduction

2.5 Irrelevant Node Elimination

In a large system, there may be species that do not have significant influence on the species of interest, $\mathbf{S_i}$. Even when all the species in the original model are coupled, after applying abstractions, a species may no longer influence the species of interest. In such cases, computational performance can be gained by removing such irrelevant species and reactions. *Irrelevant node elimination* performs a reachability analysis on the REB model and detects nodes that are not used to influence the species in $\mathbf{S_i}$. For example, in Figure 8(a), s_6 is the only species in $\mathbf{S_i}$. Therefore, the production and degradation reactions of s_6, r_3 and r_2, must be relevant. The reaction r_3 uses s_3 as a reactant and s_2 as a modifier, so these species are relevant too. Since s_2 is relevant, the degradation reaction of s_2, r_1, is also relevant. This reaction uses s_1 as a modifier, so s_1 is relevant. Using these deductions, *irrelevant nodes elimination* results in the reduced model shown in Figure 8(b).

Whereas the irrelevant node elimination guarantees that all the removed nodes are irrelevant to the species in $\mathbf{S_i}$ by statically analyzing the structure of the model, there may still be nodes in the transformed model that can be safely removed without any significant effect on the model. In such cases, a more extensive and expensive dynamic analysis such as sensitivity analysis [27,28] can be applied to further reduce the model complexity.

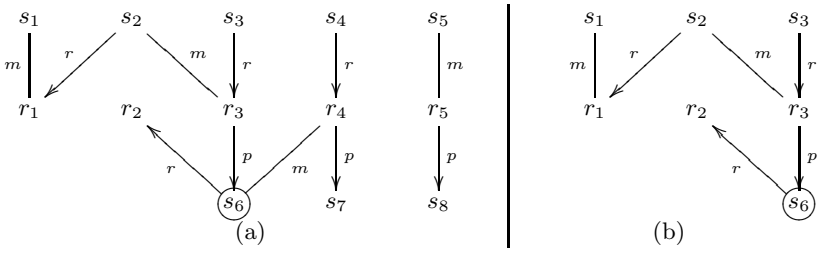

Fig. 8. Irrelevant node elimination: (a) original model and (b) after reduction

2.6 Top Level Abstraction Algorithm

The top level algorithm that combines all the abstraction methods described above is shown in Figure 9, and it is implemented within the tool `REB2SAC` [29]. The seven abstraction methods, irrelevant node elimination (line 3), modifier constant propagation (line 4), rapid equilibrium approximation (line 5), standard quasi-steady-state approximation (line 6), operator site reduction (line 7), similar reaction combination (line 8), and dimerization reduction (line 9), are applied iteratively until there is no change in the model. The irrelevant node elimination and the modifier constant propagation are applied first to reduce the complexity of the model without compromising accuracy. The rapid equilibrium approximation is applied before the standard quasi-steady-state approximation so that, whenever the model contains patterns that match the conditions for

both methods, the former has precedence in order to reduce the complexity of the reaction rate laws. The similar reaction combination is applied right after the operator site reduction to immediately combine the structurally similar reactions that are often generated by operator site reduction. The dimerization reduction is placed after operator site reduction since an operator site with a dimer molecule as a transcription factor cannot be reduced otherwise.

Algorithm 10. *Top level abstraction algorithm*
Model AbstractionEngine(Model M)

1: ***repeat***
2: $M' \leftarrow M$
3: $M \leftarrow IrrelevantNodeElim(M)$
4: $M \leftarrow ModifierConstantProp(M)$
5: $M \leftarrow RapidEqApprox(M)$
6: $M \leftarrow StandardQSSA(M)$
7: $M \leftarrow OpSiteReduction(M)$
8: $M \leftarrow SimilarReactionComb(M)$
9: $M \leftarrow DimerReduction(M)$
10: ***until*** $M' = M$
11: ***return*** M

Fig. 9. Top level abstraction algorithm

2.7 Abstraction Results

As a case study, we built a REB model of the phage λ decision circuit based on the one described in [3]. Phage λ is a virus that infects *E. coli* cells. This virus replicates either by making new copies of itself within the *E. coli* and lysing the cell (i.e., *lysis*) or embedding its DNA into the cell's DNA and replicating through cell division (i.e., *lysogeny*). The goal of the analysis is to determine the probability that lysogeny is chosen under various conditions. For example, it has been shown experimentally that the probability of lysogeny increases as the *multiplicity of infection* (MOI)—the number of phages simultaneously infecting the same cell—increases [30]. The initial REB model includes 55 species and 69 reactions, and the set of interesting species, $\mathbf{S_i}$, includes *CI* and *Cro*. This model is available with the `REB2SAC` tool [29].

After applying the reaction-based abstraction methods as specified in Section 2.6, the REB model is reduced to only 5 species and 11 reactions as shown graphically in Figure 10. This figure shows the biological gene-regulatory network of the phage λ lysis/lysogeny decision circuit, and it is quite similar to the high-level hand-generated diagram in [3]. The structure of this graph, however, is automatically generated using abstractions from the low level model. This highlights the additional benefit of abstraction in facilitating a higher level view of the network being analyzed, since it removes the low level details such as intermediate species and reactions which involve them. This makes it easier to visualize crucial interactions including identification of the key species which

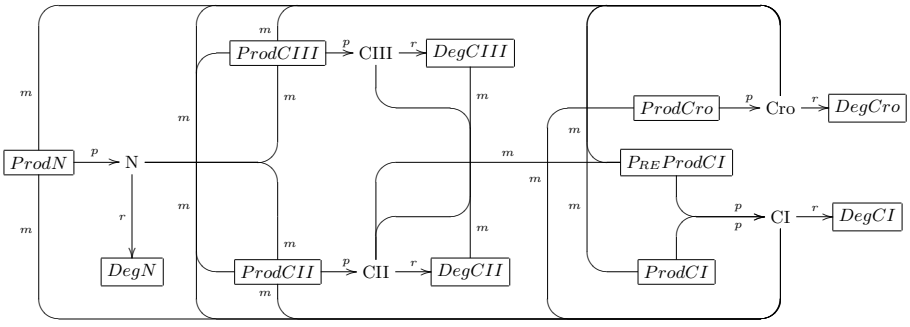

Fig. 10. Structure of the abstracted model of the phage λ developmental decision gene-regulatory pathway

ultimately inhibit and/or activate transcription. The `REB2SAC` tool can also output the abstracted model as SBML to allow it to be visualized or further analyzed using any SBML compliant tool. Finally, `REB2SAC` can output the model in presentation MathML to visualize complex rate laws using an XML/HTML browser.

Both the original model and the abstracted one are simulated for 10,000 runs using the same simulator, an optimized implementation of SSA within `REB2SAC`, on a 3GHz Pentium4 with 1GB of memory to estimate the probability of lysogeny with a reasonable statistical confidence as well as to measure the speed-up gained via abstractions. Each simulation is run for up to one cell cycle while tracking the number of molecules of *CI* and *Cro*. If the number of *CI* molecules exceeds 328 (i.e., 145 *CI* dimers) before the number of *Cro* molecules exceeds 133 (i.e., 55 *Cro* dimers), then the simulation run is said to result in lysogeny [3]. The simulations are run for MOIs ranging from 1 to 50. While the simulation of the original REB model takes 56.5 hours, the abstracted model only takes 9.8 hours, which is a speed-up of more than 5.7 times. Figure 11 shows the probability of lysogeny for MOIs from 0 to 10 for both the original REB model and the abstracted one. The results are nearly the same, yet with a substantial acceleration in runtime.

3 N-ary Transformation

To further improve the analysis time, the reduced REB model can be converted into a SAC model using the *n-ary transformation*. This section first describes the SAC model formally. Next, it describes each of the steps of n-ary transformation in turn. Then, it describes how the resulting SAC model can be analyzed using an iterative Markov chain method. Finally, it presents results using n-ary transformation on the phage λ model.

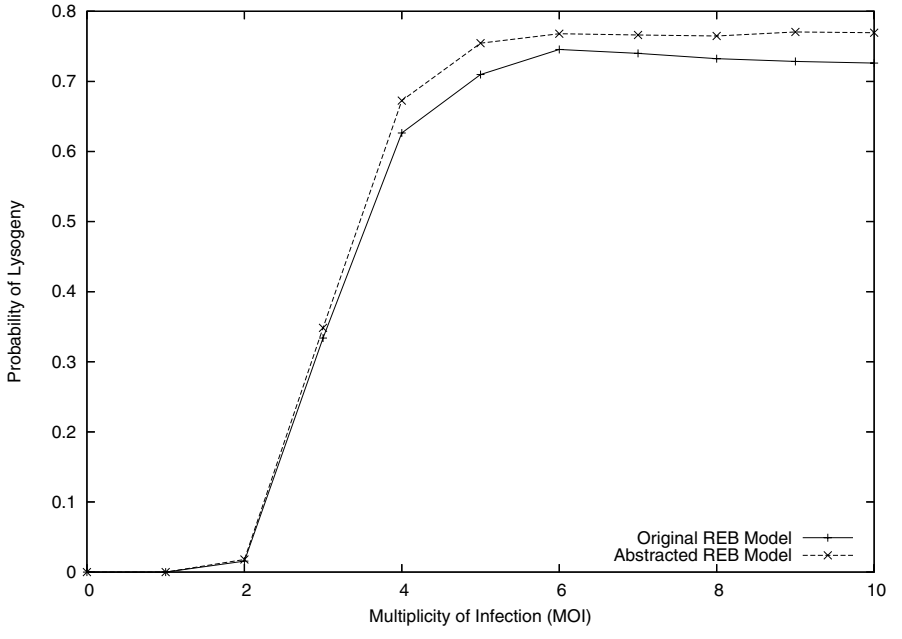

Fig. 11. Comparison of simulation results before and after abstraction where each data point has a margin of error of less than 0.01 with a 95 percent confidence

3.1 Stochastic Asynchronous Circuit Model

The SAC model is a continuous-time discrete-event system with which one can efficiently analyze the stochastic behavior of each species. So, instead of taking a computationally expensive approach of monitoring every single molecular state, the SAC model keeps track of aggregate levels of molecular counts. This is inspired by switch-like behaviors often seen in genetic regulatory networks.

Definition 2. *A SAC Model is specified using a 3-tuple* $\langle \mathbf{B}, \mathbf{b_0}, \mathbf{C} \rangle$ *where* $\mathbf{B} = \langle B_1, \ldots, B_n \rangle$ *is a vector of Boolean random variables, and thus* $\mathbf{B}(t)$ *represents the system state at time* t*. The initial state* $\mathbf{b_0}$ *contains the values of all the Boolean variables at time* 0*.* $\mathbf{C}$ *is the set of guarded commands that change the values of the Boolean variables. Each guarded command,* c_j*, has a form:*

$$G_j(\mathbf{b}) \xrightarrow{q_j} B_i := v_j$$

where the function $G_j(\mathbf{b}) : \{0,1\}^n \mapsto \{0,1\}$ *is the guard for* c_j *when the system state is* $\mathbf{b}$*,* q_j *is the transition rate for* c_j*, and* $v_j \in \{0,1\}$ *is the value assigned to* B_i *as a result of* c_j*.*

A guard is a conjunction of literals of the form $(B_i = v_j)$. Each guarded command, c_j, is required to change the state of some Boolean variable in $\mathbf{B}$. Therefore, if the state of B_i is changed to v_j by c_j, then the guard must include the term $(B_i = (1 - v_j))$. If the system state is $\mathbf{b}$ at time t (i.e., $\mathbf{B}(t) = \mathbf{b}$), c_j can be executed if its guard is satisfied (i.e., $G_j(\mathbf{b}) = 1$). The result of executing the guarded command in time step τ is that a new state is reached in which $B_i(t + \tau) = v_j$ and for all $k \neq i$, $B_k(t + \tau) = B_k(t)$. The probability that c_j is executed within the next infinitesimal The probability that, given the state is $\mathbf{b}$, c_j is executed within the next infinitesimal time step dt is:

$$P(c_j, dt \mid \mathbf{b}) = G_j(\mathbf{b}) \cdot q_j \cdot dt.$$

Consequently, the probability that no transition is taken within the next time step dt is:

$$1 - (\textstyle\sum_{l=1}^{|C|} G_l(\mathbf{b}) \cdot q_l \cdot dt).$$

A stochastic simulation of a process described using a SAC model begins in the state $\mathbf{b_0}$ at time 0 and selects either a guarded command to execute or no guarded commands to execute in a small time step Δt using the probability functions just defined. If a guarded command is executed at time t_1, then the system moves to a new state $\mathbf{B}(t_1) = \mathbf{b_1}$. It then recalculates all the transition probabilities, and continues until terminated.

This simulation process is inexact and inefficient since Δt is not a true infinitesimal yet for a sufficiently small Δt most simulation steps do not result in a state change. Therefore, the exact SSA which skips over the time steps where no state change occurs can be used instead [31,32] by using the expression $G_j(\mathbf{b}) \cdot q_j$ as the propensity function for the guarded command c_j when the system state is $\mathbf{b}$. In addition to stochastic simulation, a SAC model can be analyzed by constructing a homogeneous continuous-time Markov chain and applying an associated efficient analysis method [33]. This is the approach that is taken in this paper to analyze the phage λ decision circuit.

3.2 Reaction Splitization

The n-ary transformation requires the REB model to satisfy the property that all reactions should have either one reactant *or* one product, but not both. This is often the case after applying the abstractions described earlier as it is for the phage λ model. If this property does not hold, however, it can be made to hold using *reaction splitization*. One form of reaction splitization is called *single reactant single product reaction splitization*, which splits an irreversible reaction with a single reactant and a single product into an irreversible reaction with no reactant and a single product and an irreversible reaction with a single reactant and no product. In order to illustrate this transformation, consider the reaction shown in Figure 12(a) that converts species s_1 into species s_2 with a rate law

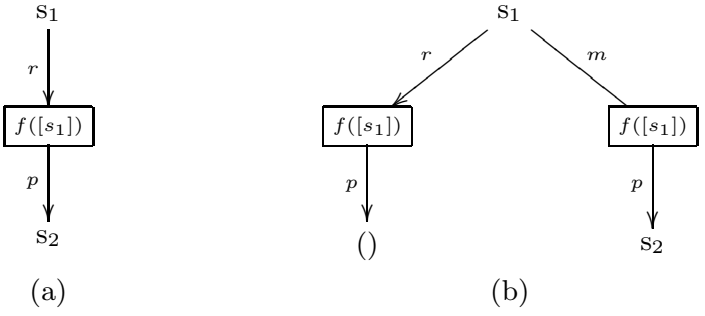

Fig. 12. Reaction splitization: (a) original reaction and (b) split-up reactions

$f([s_1])$. After splitization, this is transformed into the two reactions shown in Figure 12(b). This includes a degradation reaction for s_1 and a production reaction for s_2, with the same rate law. In addition, there is also *multiple reactants reaction splitization* to split a reaction with multiple reactants into multiple reactions with a single reactant, and *multiple products reaction splitization* that splits a reaction with multiple products into multiple reactions with a single product.

3.3 Boolean Variable Generation

Let X be a random variable representing the state of species s. Our method partitions the states of X into an ordered set $\mathbf{A} : (A_0, A_1, \ldots, A_n)$ such that, $\forall i.\ A_i = [\theta_i, \theta_{i+1})$ where $\theta_0 = 0$, and $\theta_{n+1} = \infty$. We call $A_0, \ldots, A_n$ *critical intervals*, and $\theta_0, \ldots, \theta_n$ *critical levels*. Depending on the nature of the application, these critical levels can be either specified by the user and taken to be model inputs—such as might be the case when our system is utilized by an expert already familiar with the *in situ* behavior of the underlying regulatory network—or estimated automatically from the kinetic rate laws as described next.

In order to identify the critical levels of species s, our method first automatically finds all reactions with kinetic rate laws that include a denominator term of the form $K[s]^n$. For each such reaction, one critical level of s is generated with the form $\sqrt[n]{a/(K - aK)}$ where a is an amplifier in the range $[0.5, 1.0)$ selected by the user. Figure 13(a) shows two reactions which have kinetic rate laws containing *CII* terms. Assuming that a equals 0.5, these two reactions imply the following four critical levels:

$$0, \frac{k_9}{k_8}, \frac{k_{-2}}{k_2 \cdot RNAP_0}, \text{ and } \frac{k_{-3}}{k_3}.$$

These levels come from the fact that θ_0 is by definition 0, the denominator of the left reaction rate law in Figure 13(a) has the term $k_8/k_9[CII]$, and the denominator of the right reaction rate law has two terms of this form, $k_2/k_{-2}[CII]RNAP_0$ and $k_3/k_{-3}[CII]$.

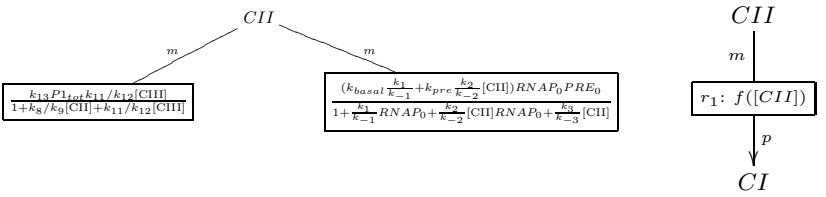

Fig. 13. (a) Critical level identification. (b) Production of *CI* with activator *CII*.

If there are $n+1$ critical levels, $\theta_0, \ldots, \theta_n$, for s, our method creates n Boolean variables $B_1 \ldots B_n$ with initial values $b_1 \ldots b_n$ in which $b_i = 1$ if $[s]_0 \geq \theta_i$ and 0 otherwise. Y is a discrete random variable that denotes the state of critical levels that species s is in. So, the relationship between Y and $B_1, \ldots, B_n$ is: $Y(t) = i$ iff $(\forall j \in [1, i].\ B_j(t) = 1) \wedge (\forall j \in [i+1, n].\ B_j(t) = 0)$.

3.4 Guarded Command Generation

The guard for a reaction is derived from the Boolean variables for the species used in that reaction. Suppose species *CII* is an activator in the reaction r_1 for the production of *CI* shown in Figure 13(b). Also, suppose that three critical levels are used for *CII* which are $(0, \theta_1^{CII}, \theta_2^{CII})$, and the critical levels for *CI* are $(0, \theta_1^{CI}, \theta_2^{CI})$, respectively. Since r_1 is a production reaction for species CI, the legal moves that Y_{CI} (the random variable for the levels of CI) can take are only two: $0 \rightarrow 1$ and $1 \rightarrow 2$. The guarded commands for r_1 are below:

$$B_2^{CI} = 0 \wedge B_1^{CI} = 0 \wedge B_2^{CII} = 0 \wedge B_1^{CII} = 0 \xrightarrow{q_1} B_1^{CI} := 1$$
$$B_2^{CI} = 0 \wedge B_1^{CI} = 0 \wedge B_2^{CII} = 0 \wedge B_1^{CII} = 1 \xrightarrow{q_2} B_1^{CI} := 1$$
$$B_2^{CI} = 0 \wedge B_1^{CI} = 0 \wedge B_2^{CII} = 1 \wedge B_1^{CII} = 1 \xrightarrow{q_3} B_1^{CI} := 1$$
$$B_2^{CI} = 0 \wedge B_1^{CI} = 1 \wedge B_2^{CII} = 0 \wedge B_1^{CII} = 0 \xrightarrow{q_4} B_2^{CI} := 1$$
$$B_2^{CI} = 0 \wedge B_1^{CI} = 1 \wedge B_2^{CII} = 0 \wedge B_1^{CII} = 1 \xrightarrow{q_5} B_2^{CI} := 1$$
$$B_2^{CI} = 0 \wedge B_1^{CI} = 1 \wedge B_2^{CII} = 1 \wedge B_1^{CII} = 1 \xrightarrow{q_6} B_2^{CI} := 1$$

3.5 Transition Rate Generation

The final step to generate a SAC model is to assign a transition rate, q_i, to each guarded command. The random variable, X, which represents the state of s can be approximately expressed in terms of the Boolean variables by defining a discrete random variable, X', as follows:

$$X'(t) = (\theta_n - \theta_{n-1})B_n(t) + \cdots + (\theta_2 - \theta_1)B_2(t) + (\theta_1 - \theta_0)B_1(t),$$

and then approximating evolution of X by the average of X'. Taking the derivative with respect to the mean of $B_i(t)$ results in:

$$\frac{\partial X(t)}{\partial \langle B_i(t) \rangle} \approx (\theta_i - \theta_{i-1}).$$

Using this approximation, the time derivative of the mean of B_i is:

$$\frac{d\langle B_i(t) \rangle}{dt} = \frac{\partial \langle B_i(t) \rangle}{\partial X(t)} \frac{dX(t)}{dt} \approx \frac{1}{\theta_i - \theta_{i-1}} \frac{dX(t)}{dt}.$$

Notice $\langle B_i(t) \rangle$ is a continuous variable in the range $[0, 1]$. By letting $\langle B_i(t) \rangle$ be the probability that $B_i = 1$ at t, our method finds the transition rate functions for B_i to move from 0 to 1 and from 1 to 0 from the rate laws of reactions that change the value of $[s]$. The transition rate function of a guarded command changing the value of B_i, which is generated from reaction r is:

$$f = \frac{E \cdot \mathbf{K}(r)}{\theta_i - \theta_{i-1}} \qquad \text{where } E = \begin{cases} \mathbf{E}(s, r) & \text{if } s \text{ is a reactant of } r \\ \mathbf{E}(r, s) & \text{if } s \text{ is a product of } r \end{cases}$$

Finally, our method must evaluate the transition rate functions with appropriate values. Consider reaction r with $\mathbf{K}(r)$ containing X. Given that the corresponding Y is i, our method uses θ_i as the value of X. For example, the transition rates of the guarded commands in Figure 13(b) are derived from $\mathbf{K}(r_1)$. Since the derived transition rate function is $f([CII])/(\theta_i^{CI} - \theta_{i-1}^{CI})$, the transition rates for the guarded commands for reaction r_1 are:

$$\begin{aligned} q_1 &= f(0)/\theta_1^{CI} \\ q_2 &= f(\theta_1^{CII})/\theta_1^{CI} \\ q_3 &= f(\theta_2^{CII})/\theta_1^{CI} \\ q_4 &= f(0)/(\theta_2^{CI} - \theta_1^{CI}) \\ q_5 &= f(\theta_1^{CII})/(\theta_2^{CI} - \theta_1^{CI}) \\ q_6 &= f(\theta_2^{CII})/(\theta_2^{CI} - \theta_1^{CI}) \end{aligned}$$

3.6 Markov Analysis

A SAC model can be efficiently analyzed using Markov chain analysis within the `ATACS` tool [20]. Beginning in the initial state, $\mathbf{b_0}$, a transition can occur to a new state, $\mathbf{b_1}$, by executing a guarded command which has its guard satisfied in $\mathbf{b_0}$. A depth-first-search can be used to generate a state graph containing all states reachable from $\mathbf{b_0}$. If there exists a state transition from state $\mathbf{b_i}$ to $\mathbf{b_j}$ due to the execution of the guarded command c_k, then this state transition can be annotated with its transition rate, q_k. The result of this annotation is that the state graph is now a continuous-time Markov chain. This can be analyzed by first converting it into its embedded Markov chain by normalizing each transition rate by the sum of all the rates leaving the state resulting in a transition probability. Finally, this embedded Markov chain can be analyzed to determine the stationary probability distribution using an efficient iterative method [33].

3.7 N-ary Transformation Results

The n-ary transformation is able to automatically convert our reduced REB model for the phage λ decision circuit into a SAC model. However, since the species *CI* and *Cro* influence many reactions, our automated analysis finds that 10 critical levels are needed for species *CI* and 10 are needed for species *Cro*. This is too many critical levels for the Markov chain analyzer within `ATACS`. Fortunately, many of these critical levels are very close together and can be combined with little loss in accuracy. Therefore, while we decided to use eight Boolean variables for species *CI* and three for *CII*, we only used one Boolean variable for each of the species *Cro*, *N*, and *CIII*.

We analyzed the SAC model using Markov chain analysis. The probability of lysogeny is calculated by summing the probability of states that reach the highest level of *CI*. We compare our results with both experimental data and previous simulations performed by Arkin et al. on a complete master equation model. The experimental results are from Kourilsky [30]. Since it was not practical to measure the number of phages that infect any given cell, Kourilsky measured the fraction of cells that commit to lysogeny versus *average phage input* (API) (i.e., the proportion of phages to *E. coli* within the population). Kourilsky performed experiments for both "starved" *E. coli* and those in a "well-fed" environment. He found that the fraction that commits to lysogeny increases with increasing API, and that this fraction increases by more than an order of magnitude in a starved environment over a well-fed environment.

To map simulated MOI data onto API data, Arkin et al. used a Poisson distribution of the phage infections over the populations:

$$P(M, A) = \frac{A^M}{M!} e^{-A}$$
$$F_{\text{lysogens}}(A) = \sum_M P(M, A) \cdot F(M)$$

where M is the MOI, A is the API, and $F(M)$ is the probability of lysogeny determined by Markov analysis. We also used this method to map our MOI data. The results are shown in Figure 14. The individual points represent experimental measurements while the lines represent simulation results. Both Arkin et al.'s simulation and our SAC model results track the starved data points reasonably well. Our SAC model results, however, are found in less than 7 minutes of computation time on a 3GHz Pentium4 with 1GB of memory. While modern computer technology and algorithmic improvements would greatly improve the simulation time of Arkin et al.'s model, these results would still take several hours to generate on a similar computer to ours. Another notable benefit of our SAC method is that it can also produce simulation results for the well-fed case in about 7 minutes. These results could likely not be generated even today using Arkin's master equation simulation method, since the number of simulation runs necessary is inversely proportional to the probability of lysogeny (i.e., about two orders of magnitude greater in the well-fed case than in the starved one).

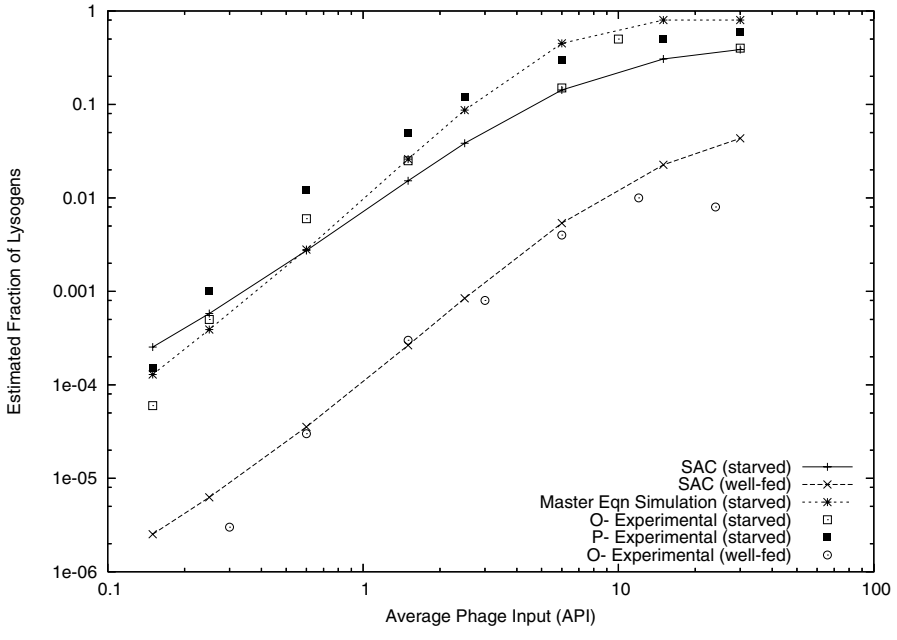

Fig. 14. Comparison of SAC results to experimental data

4 Conclusions

This paper presents a general methodology for systematically and automatically abstracting the complexities of large-scale biochemical reaction-based networks (REBs) to a reduced stochastic asynchronous circuit (SAC) representation. It significantly facilitates efficient non-deterministic analysis of such systems by substantially reducing the problem dimensionality in both reaction and molecular state spaces, thus potentially allowing for both simulation time acceleration and computability gains while facilitating a high-level view of the network. Furthermore, since our approach allows for multiple levels of abstraction, it is broadly applicable to a wide range of biological systems and their representations—from classical differential equation models to fully discrete and stochastic bio-molecular pathways—including the genetic regulatory networks upon which we have chosen to focus in this work.

As a case study, we have applied our method to the phage λ developmental decision pathway. The preliminary results are promising. Among other things, we are able to: (1) ascertain the internal self-consistency of our approach by successfully cross-validating each abstraction level output against the results of the full underlying discrete-stochastic model simulations; and (2) accurately estimate the biologically relevant (observable) pathway selection probabilities, which typically require substantial numbers of hours of computation time via

the original REB representation, yet could be computed in only minutes using our SAC approach.

Future work includes the development of more abstraction methods, refinement of the critical level assignment algorithm, and integration of a more scalable Markov chain analyzer. We are also working on a tighter integration with other tools for modeling and analysis of (bio)chemical networks such as BioSPICE [23]. Finally, we are applying our abstraction methodology to efficient analysis of other systems—such as the *E. coli* Fim mechanism and *B. Subtilis* stress response network—that may benefit from our automated abstraction methodology due to, among others, their inherently stochastic behavior *in situ*.

Acknowledgments. The authors would like to thank David Dill (Stanford University), Satoru Miyano (University of Tokyo), Hiroaki Kitano (Sony Corporation), and Arkin's research group (University of California, Berkeley) for numerous helpful discussions.

References

1. Jong, H.D.: Modeling and simulation of genetic regulatory systems: A literature review. J. Comp. Biol. **9**(1) (2002) 67–103
2. Baldi, P., Hatfield, G.W.: DNA Microarrays and Gene Expression. Cambridge University Press (2002)
3. Arkin, A., Ross, J., McAdams, H.: Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells. Genetics **149** (1998) 1633–1648
4. Elowitz, M.B., Levine, A.J., Siggia, E.D., Swain, P.S.: Stochastic gene expression in a single cell. Science **297** (2002) 1183–1186
5. Rao, C.V., Wolf, D.M., Arkin, A.P.: Control, exploitation and tolerance of intracellular noise. Nature **420** (2002) 231–238
6. Samoilov, M., Plyasunov, S., Arkin, A.P.: Stochastic amplification and signaling in enzymatic futile cycles through noise-induced bistability with oscillations. Proceedings of the National Academy of Sciences US **102**(7) (2005) 2310–5
7. Raser, J.M., O'Shea, E.K.: Control of stochasticity in eukaryotic gene expression. Science **304** (2004) 1811–1814
8. Kierzek, A.M., Zaim, J., Zielenkiewicz, P.: The effect of transcription and translation initiation frequencies on the stochastic fluctuations in prokaryotic gene expression. J. Biol. Chem **276** (2001) 8165
9. Gillespie, D.T.: A rigorous derivation of the chemical master equation. Physica A **188** (1992) 404–425
10. Gillespie, D.T.: A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. Journal of Computational Physics **22** (1976) 403–434
11. Turner, T.E., Schnell, S., Burrage, K.: Stochastic approaches for modelling in vivo reactions. Computational Biology **28** (2004)
12. Gibson, M., Bruck, J.: Efficient exact stochastic simulation of chemical systems with many species and many channels. J. Phys. Chem. **A 104** (2000) 1876–1889

13. Gibson, M., Bruck, J.: An efficient algorithm for generating trajectories of stochastic gene regulation reactions. Technical report, California Institute of Technology (1998)
14. Gillespie, D.T.: Approximate accelerated stochastic simulation of chemically reacting systems. Journal of Chemical Physics **115**(4) (2001) 1716–1733
15. Rathinam, M., Cao, Y., Petzold, L., Gillespie, D.: Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. Journal of Chemical Physics **119** (2003) p12784–94
16. Gillespie, D., Petzold, L.: Improved leap-size selection for accelerated stochastic simulation. Journal of Chemical Physics **119** (2003)
17. Cao, Y., Gillespie, D., Petzold, L.: Avoiding negative populations in explicit tau leaping. Journal of Chemical Physics **123** (2005)
18. Rao, C.V., Arkin, A.P.: Stochastic chemical kinetics and the quasi-steady-state assumption: Application to the gillespie algorithm. J. Phys. Chem. **118**(11) (2003)
19. Schnell, S., Maini, P.K.: A century of enzyme kinetics: Reliability of the k_m and v_{max} estimates. Comments on Theoretical Biology **8** (2003) 169–187
20. Myers, C.J., Belluomini, W., Killpack, K., Mercer, E., Peskin, E., Zheng, H.: Timed circuits: A new paradigm for high-speed design. (2001) 335–340
21. Berry, R.S., Rice, S.A., Ross, J.: Physical Chemistry (2nd Edition). Oxford University Press, New York (2000)
22. Systems Biology Workbench Development Group. (`http://www.sbw-sbml.org/`)
23. BioSPICE. (`http://www.biospice.org/`)
24. Schnell, S., Mendoza, C.: Enzyme kinetics of multiple alternative substrates. Journal of Mathematical Chemistry **27** (2000) 155–170
25. Ackers, G.K., Johnson, A.D., Shea, M.A.: Quantitative model for gene regulation by λ phage repressor. Proc. Natl. Acad. Sci. USA **79** (1982) 1129–1133
26. Santillán, M., Mackey, M.C.: Why the lysogenic state of phase λ is stable: A mathematical modeling approch. Biophysical Jounal **86** (2004)
27. Dacol, D., Rabitz, H.: Sensitivity analysis of stochastic kinetic models. J. Math. Phys. **25** (1984)
28. Gunawan, R., Cao, Y., Petzold, L., Doyle, F.J.: Sensitivity analysis of discrete stochastic systems. Biophysical Journal **88** (2005) 2530–2540
29. REB2SAC. (`http://www.async.ece.utah.edu/tools/`)
30. Kourilsky, P.: Lysogenization by bacteriophage lambda: I. multiple infection and the lysogenic response. Mol. Gen. Genet. **122** (1973) 183–195
31. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. **81**(25) (1977) 2340–2361
32. Gillespie, D.T.: Markov Processes An Introduction for Physical Scientists. Academic Press, Inc. (1992)
33. Stewart, W.J.: Introduction to the Numerical Solution of Markov Chains. Princeton University Press (1994)