Utilizing Stochastic Model Checking to Analyze Genetic Circuits

Curtis Madsen*, Chris J. Myers*, Nicholas Roehner*, Chris Winstead† and Zhen Zhang*
*University of Utah, Salt Lake City, UT 84112
Email: curtis.madsen@utah.edu, myers@ece.utah.edu, n.roehner@utah.edu, zhen.zhang@utah.edu
†Utah State University, Logan, UT 84322
Email: winstead@ece.usu.edu

***Abstract*—When designing and analyzing genetic circuits, researchers are often interested in the probability of the system reaching a given state within a certain amount of time. Usually, this involves simulating the system to produce some time series data and analyzing this data to discern the state probabilities. However, as the complexity of models of genetic circuits grow, it becomes more difficult for researchers to reason about the different states by looking only at time series simulation results of the models. To address this problem, this paper employs the use of *stochastic model checking*, a method for determining the likelihood that certain events occur in a system, with *continuous stochastic logic* (CSL) properties to obtain similar results. This goal is accomplished by the introduction of a methodology for converting a *genetic circuit model* (GCM) into a *continuous-time Markov chain* (CTMC). This CTMC is analyzed using transient Markov chain analysis to determine the likelihood that the circuit satisfies a given CSL property in a finite amount of time. This paper illustrates a use of this methodology to determine the likelihood of failure in a genetic toggle switch and compares these results to stochastic simulation-based analysis of this same circuit. Our results show that this method results in a substantial speedup as compared with conventional simulation-based approaches.**

***Keywords*-stochastic model checking; genetic circuits; markov chain analysis; continuous stochastic logic; synthetic biology**

I. Introduction

Recently, biologists and engineers have begun to work together on *synthetic biology* [1], [2]. Synthetic biology is a relatively new area of research that combines biology and engineering with the ultimate goal of designing new, useful biological systems. Although many tools have been developed for synthetic biology [3]–[11], there is still a need for more efficient methods for their modeling, analysis, and design.

Biological systems are typically modeled using chemical reaction networks, and the primary method for analyzing these is to transform them into a set of *ordinary differential equations* (ODEs) using the *law of mass action* [12]. ODEs can be simulated and analyzed using a variety of well known methods [13], [14]. ODE models of reactions, however, assume that there are large amounts of each chemical species allowing them to be represented as continuous variables that react deterministically. Synthetic biological systems, however, are often constructed using DNA to form synthetic *genetic circuits* that often have small, discrete amounts of species which are modified by reactions that occur sporadically. This fact has led researchers to utilize discrete, stochastic methods such as *Gillespie's stochastic simulation algorithm* (SSA) [15] for their analysis. The SSA is an efficient Monte Carlo simulation method that steps over useless time steps (time steps where no reactions occur). The SSA can be improved further using the *next reaction method* [16], *tau leaping* [17], [18], and reaction-based abstractions [19], [20].

As models of genetic circuits become more complex, time-scale separations between species cause researchers to take a hybrid approach to modeling using both discrete and continuous dynamics to represent the circuit. However, this leads to computationally complex models that are extremely difficult to analyze because they require that part of the model be analyzed using deterministic methods while other parts are analyzed using stochastic methods. Therefore, methods to abstract these hybrid models into logical models that can be analyzed using conventional methods are crucial to their analysis. Qualitative logical models of genetic circuits have been considered [21], [22], but these models are incapable of yielding quantitative predictions of behavior. To address this problem, a quantitative logical model can be used to encode the infinite state space of a genetic circuit into a finite number of logical levels for each chemical species [19], [20], [23], [24]. The resulting logical model can then be analyzed using *stochastic model checking* [25]. Stochastic model checking utilizes *Markov chain analysis* to determine the probability that a system has a specified property. As demonstrated in this paper, such an analysis can be extremely useful to quickly compare the robustness of alternative circuit designs and evaluate different design trade-offs.

This paper presents a new methodology for the logical analysis of genetic circuits. In particular, this paper describes a process for translating a molecular biological model, the *genetic circuit model* (GCM), into a *continuous-time Markov chain*. After this translation, it is now possible to reason about this model using stochastic model checking. In particular, our method applies *transient Markov chain analysis* to verify properties of a genetic circuit design specified using *continuous stochastic logic* (CSL). As a case study, this paper presents results for an analysis of a genetic toggle switch [26]. These results demonstrate not only that this method is accurate but also that it is substantially more computationally efficient than conventional stochastic simulation-based approaches.

Section II presents background on genetic circuits and their

978-1-4673-1191-5/12/$31.00 ©2012 IEEE

representation using GCMs. Section III presents our methods including the logical abstraction to translate from a GCM to a *continuous time Markov chain* (CTMC) and the stochastic model checking technique that is used to analyze the resulting Markov chain. Section IV presents our results on a genetic toggle switch, as well as how they compare with stochastic simulation. Lastly, Section V discusses future improvements to the method described in this paper.

II. Background

A genetic circuit is composed of, among other things, *genes*, *operator sites*, and *promoters* on a strand of DNA. An example of a genetic circuit for a toggle switch which was constructed by Gardner et al. [26] is shown in Figure 1(a). Genes are the portion of the DNA that code for *proteins*, the basic building blocks for nearly all molecular machinery within a cell. Proteins can also regulate cellular function by binding to an operator site in order to increase, *activate*, or decrease, *repress*, the associated promoter's affinity. Promoters are the regions on the DNA where *RNA polymerase* (RNAP) binds to start the transcription of a gene to produce *messenger RNA* (mRNA), which in turn is translated by a *ribosome* into a protein. In Figure 1(a), the LacI protein binds to the operator site associated with the P_{trc-2} promoter to repress the production of TetR and *green fluorescent protein* (GFP), a reporter that causes the cell to glow. Similarly, the TetR protein binds to the operator site associated with the $P_{LtetO-1}$ promoter to repress the production of LacI. The molecules IPTG and aTc are known as *chemical inducers*. IPTG can bind with LacI to form a complex C1 which prevents LacI from being able to repress TetR production. Similarly, aTc can bind with TetR to form a complex C2 which prevents TetR from being able to repress LacI production.

A logical model that summarizes the behavior of the genetic toggle switch is shown in Figure 1(b). Digital designers should recognize this circuit as a *set-reset* (SR) *latch*, a simple asynchronous state-holding gate. The chemical inducer aTc is the *set* input which is used to set the output, GFP, in the high state. The chemical inducer IPTG is the *reset* input which is used to set the output in the low state. When neither input is applied, the latch retains its previous state. Applying both inputs simultaneously, just like in an electronic SR latch, is illegal as it would make the circuit oscillate erratically. While this diagram indicates a logical behavior, one should keep in mind that the actual behavior of this latch is extremely noisy due to the inherent stochastic nature of genetic circuits described earlier.

When modeling genetic circuits, researchers often use chemical reaction networks specified using the *systems biology markup language* (SBML) [27]. These models are tedious to build because they require the modeler to construct the model at a low level in which each chemical species and chemical reaction must be explicitly defined. In order to help simplify the modeling process for genetic circuits, Nguyen et al. introduced the *genetic circuit model* (GCM) language which allows systems to be specified using only the critical

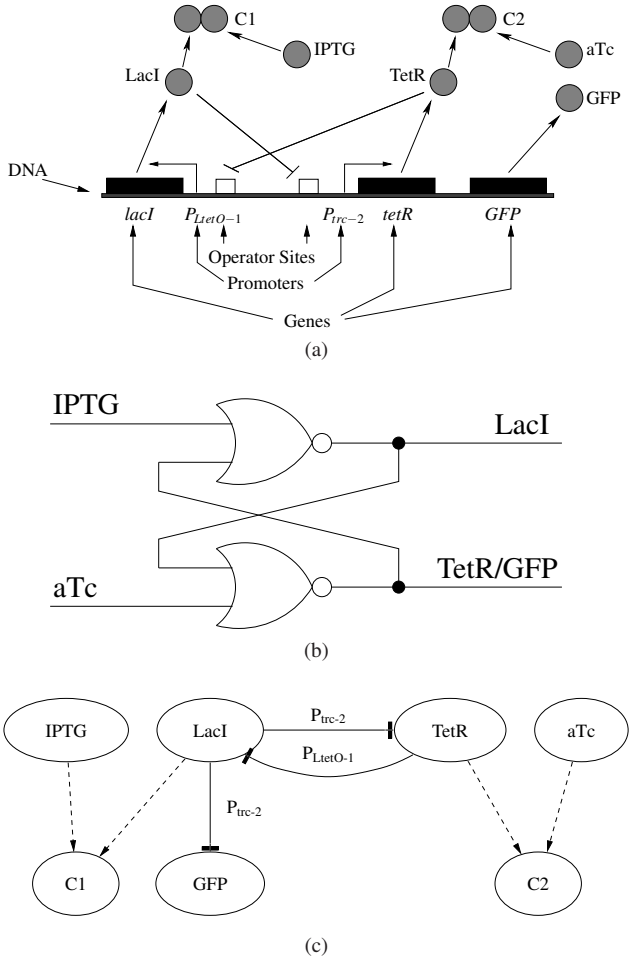


Fig. 1. (a) The genetic toggle switch circuit where LacI and TetR repress each other (denoted by the ⊣ arrows). In this circuit, LacI can be sequestered by IPTG, TetR can be sequestered by aTc, and GFP is the reporter protein causing the cell to glow indicating whether the toggle is in the on or off state. While this figure shows the operator site and promoter as distinct regions of the DNA, their sequences typically overlap. (b) A digital circuit representation of the genetic toggle switch. (c) The GCM for the genetic toggle switch. The ⊣ arrows indicate repression and the dashed arrows indicate that the species come together to form a complex. As a shorthand, the promoters are used as labels on the influences. For example, LacI represses TetR and GFP production by binding to promoter P_{trc-2} while TetR represses LacI production by binding to promoter $P_{LtetO-1}$.

species and relationships between these species [28], [29]. A GCM is a tuple $\langle S, Pr, G, I, C, V_g, A_g \rangle$ where:

- S is a finite set of species (i.e., proteins);
- Pr is a finite set of promoters;
- $G : Pr \mapsto 2^S$ maps promoters to sets of species;
- $I \subseteq S \times Pr \times \{a, r\}$ is a finite set of influences;
- $C \subseteq S \times \mathbb{N} \times S$ is a finite set of complex formations;
- V_g is a finite set of parameter variables used during model generation;
- $A_g \subseteq (V_g \times \mathbb{R})$ is the assignment of the variables with defaults presented in Table I.

TABLE I
GCM PARAMETER LIST

Parameter	Symbol	Value	Units
Initial RNAP count	n_r	30	*molecule*
Degradation rate	k_d	0.0075	$\frac{1}{sec}$
Complex formation equilibrium	K_c	0.05	$\frac{1}{molecule}$
Stoichiometry of binding	n_c	2	*molecule*
Repression binding equilibrium	K_r	0.5	$\frac{1}{molecule}$
Activation binding equilibrium	K_a	0.0033	$\frac{1}{molecule}$
Initial promoter count	n_g	2	*molecule*
RNAP binding equilibrium	K_o	0.033	$\frac{1}{molecule}$
Activated RNAP binding equilibrium	K_{oa}	1	$\frac{1}{molecule}$
Basal production rate	k_b	0.0001	$\frac{1}{sec}$
Open complex production rate	k_o	0.05	$\frac{1}{sec}$
Activated production rate	k_a	0.25	$\frac{1}{sec}$
Stoichiometry of production	n_p	10	$unit-less$

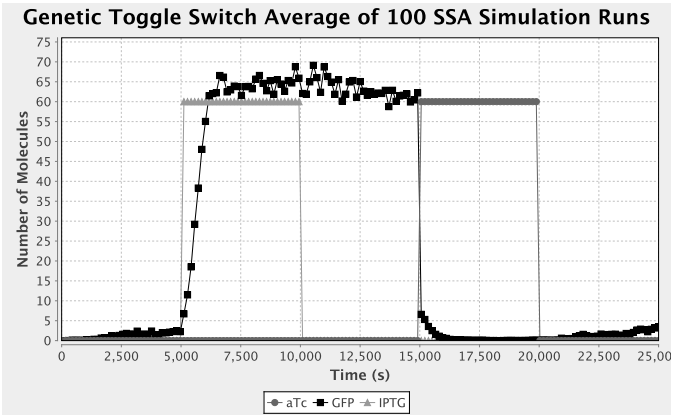
For convenience, the following functions are also defined:

$$\begin{aligned} \text{Pro}(s) &= \{p \in Pr \mid s \in G(p)\} \\ \text{Rep}(p) &= \{s \in S \mid (s,p,r) \in I\} \\ \text{Act}(p) &= \{s \in S \mid (s,p,a) \in I\} \end{aligned}$$

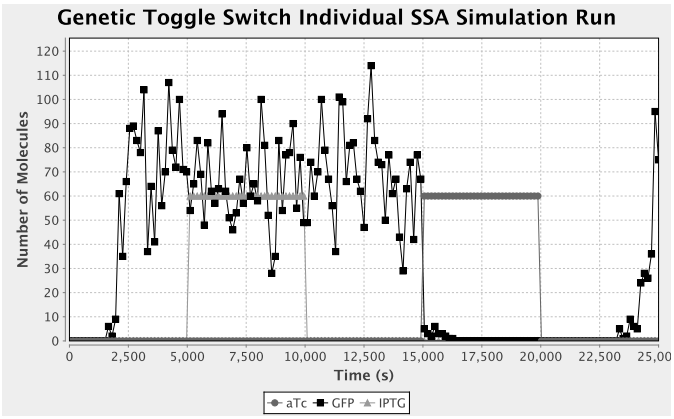
The function $\text{Pro}(s)$ is used to determine the set of promoters that initiate transcription of genes that lead to the production of species s. The functions $\text{Rep}(p)$ and $\text{Act}(p)$ return the species that repress and activate promoter p, respectively. These functions are used in the logical abstraction presented in Section III.

As an example, consider the GCM for the genetic toggle switch which is shown in Figure 1(c). In this model, S is a set of the species { aTc, C1, C2, GFP, IPTG, LacI, TetR }, and Pr is a set of the promoters { P_{trc-2}, $P_{LtetO-1}$ }. G maps the promoter P_{trc-2} to TetR and GFP and the promoter $P_{LtetO-1}$ to LacI. The influences in I indicate that LacI represses P_{trc-2} and TetR represses $P_{LtetO-1}$. C contains the complex formations (IPTG, 1 C1), (LacI, 1, C1), (aTc, 1, C2), and (TetR, 1, C2). The variables, V_g, and their assignments, A_g, are shown in Table I. The function Pro(GFP) returns $\{P_{trc-2}\}$ while $\text{Rep}(P_{trc-2})$ returns { LacI } and $\text{Act}(P_{trc-2})$ returns $\emptyset$.

A GCM can be represented in SBML and automatically translated into a detailed reaction-based model as described in [28], [29]. The resulting SBML model can then be simulated to determine the behavior of the genetic circuit. Figure 2(a) presents the average of performing 100 stochastic simulation runs for 25,000 seconds on a reaction-based model of the genetic toggle switch. These simulations are started in an initial state in which LacI is 60 molecules while TetR, GFP, IPTG, and aTc are all 0 molecules. At time 5,000, 60 molecules of IPTG are provided which causes the state of the toggle switch to change resulting in the production of GFP. At time 10,000, the IPTG is removed, but the toggle switch on average holds the high GFP state. At time 15,000, 60 molecules of aTc are provided which causes the toggle switch to change state again, and GFP degrades away. At time 20,000, the aTc is removed, and once again the toggle switch holds its state. It should be stressed though that this shows the average of 100



(a)



(b)

Fig. 2. These results plot the molecule count of the GFP protein showing how the circuit is set and reset by IPTG and aTc, respectively. IPTG is added at time 5,000s to set the switch and then is removed at time 10,000s. aTc is later added at time 15,000s to reset the switch and then is removed at time 20,000s. (a) The average of 100 stochastic simulation runs of the genetic toggle switch. (b) One individual stochastic simulation run where the genetic toggle switch fails to hold state.

simulations runs. This circuit is noisy meaning that there is a chance that the circuit does not perform ideally and loses its state. For example, Figure 2(b) shows one such simulation run where the circuit switches from the off state to the on state erroneously. One of the goals of this paper is to be able to efficiently determine the probability of erroneous behavior.

III. METHODS

Due to the inherently noisy nature of genetic circuits, any deterministic assertion that is checked would most likely fail to be true. Instead, for these systems, the more interesting question is the probability that a property is true. Determining the likelihood of properties can be accomplished using a technique known as stochastic model checking.

There are two types of stochastic model checking used to compute the likelihood that a property is true: statistical and numerical based techniques [30]. Statistical techniques involve simulating a system a large number of times and terminating whenever a property is shown to be true or false. When all of the simulations are complete, statistics are calculated

on how many simulations satisfied the property in the time allotted versus the number of simulations that failed to do so. One downside of using statistical techniques is that the more rare an event is, the more simulations that need to be run in order to observe it, and this may cause the time that it takes to compute a likelihood to become prohibitively expensive. Numerical methods, on the other hand, attempt to determine these likelihoods in a more direct method. They usually attempt to find the state space of the model and then employ methods such as Markov chain analysis to compute the probability of reaching a state where a given property is satisfied. These methods are often more efficient than statistical techniques; however, they require that the state space be computable. Both statistical and numerical methods have been utilized by many tools such as the probabilistic model checker PRISM [31].

Stochastic model checking can be utilized to check a CSL property [25], [32] specified using the following grammar:[1]

$$\begin{aligned} \mathit{Prop} &::= \mathtt{P}_{\sim p}\{\Psi\,\mathtt{U}^{\mathrm{I}}\,\Psi\} \mid \mathtt{S}_{\sim p}\{\Psi\} \\ \Psi &::= \mathtt{true} \mid \Psi \wedge \Psi \mid \neg\Psi \mid v_i \geq c_i \mid v_i > c_i \mid v_i = c_i \end{aligned}$$

where $\sim$ represents an element from the set $\{<, \leq, =, >, \geq\}$, p is a probability in the range $[0,1]$, I is a non-negative interval of the form $[t,t']$, v_i is a variable, and c_i is a constant. Ψ represents a state formula that must be true in a given state. The formula $\mathtt{P}_{\sim p}\{\Psi_1\,\mathtt{U}^{\mathrm{I}}\,\Psi_2\}$ represents the probability that an execution of the system satisfies the until formula $\Psi_1\,\mathtt{U}^{\mathrm{I}}\,\Psi_2$ which means that Ψ_1 must remain true until Ψ_2 becomes true in the interval I within the probability bound $\sim p$. The formula $\mathtt{S}_{\sim p}\{\Psi\}$ represents the probability that once the system reaches its steady state, it is in a state where Ψ is satisfied within the probability bound $\sim p$. To compute the truth of these formulas requires solving the more general problem of what is the probability of the formula. Therefore, this paper focuses on determining the actual probability in which $\sim p$ is replaced in the notation with $=?$. As a shorthand, Ψ can also contain `false`, $\vee$, $<$, and $\leq$ which are easily derived. The formulas are also allowed to contain the eventually operator, $\Diamond$, and the globally true operator, $\Box$, defined as follows:

$$\begin{aligned} \mathtt{P}_{\sim p}\{\Diamond^{\mathrm{I}}\Psi\} &\equiv \mathtt{P}_{\sim p}\{\mathtt{true}\,\mathtt{U}^{\mathrm{I}}\,\Psi\} \\ \mathtt{P}_{\sim p}\{\Box^{\mathrm{I}}\Psi\} &\equiv \mathtt{P}_{\tilde{\sim} 1-p}\{\Diamond^{\mathrm{I}}\neg\Psi\} \end{aligned}$$

The eventually operator is essentially used as a shorthand for describing an until property where the left-hand side of the formula is true. This means that the eventually formula $\mathtt{P}_{\sim p}\{\Diamond^{\mathrm{I}}\Psi\}$ would simply require that Ψ becomes true in the interval I within the probability bound $\sim p$. The globally true formula $\mathtt{P}_{\sim p}\{\Box^{\mathrm{I}}\Psi\}$ requires that Ψ remains true during the interval I within the probability bound $\sim p$. This formula builds off of the eventually operator by requiring that $\neg\Psi$ does not eventually become true during the interval I within the probability bound $\tilde{\sim} 1-p$ where $\tilde{\sim}$ flips the direction of the inequality, if one is used.

[1]Our tool currently does not allow nesting of transient and steady-state properties.

Figure 3 presents an algorithm that determines the probability within an error bound, ε, of a given transient CSL property, Φ, on a GCM model, N. We also have a similar algorithm for checking steady-state properties, but it is omitted due to space constraints. Additionally, this algorithm requires a set of levels, L, that includes an ordered list of threshold levels, L_s, for each species $s \in S$ in the GCM. Each level, $l_{s,i}$ represents a critical threshold in the amount of the species s. It is assumed that $l_{s,0}$ is always 0, and $l_{s,i-1} < l_{s,i}$ for all $i > 0$.

Input: GCM G; Levels L; CSL property Φ; Error-bound ε

1: $T = \text{computeCTMC}(G, L, \Phi)$
2: $t = \text{determineTimeLimit}(\Phi)$
3: Find infinitesimal generator matrix, Q^X, of T
4: Compute $\Gamma = \max_i |q^X_{ii}|$ where q^X_{ii} is diagonal entry of Q^X
5: Find stochastic transition probability matrix, $P = I + \frac{1}{\Gamma}Q^X$
6: Set $K = 0$, $\xi = 1$, $\sigma = 1$, and $\eta = \frac{1-\varepsilon}{e^{-\Gamma t}}$
7: **while** $\sigma < \eta$ **do**
8: Compute $K = K + 1$, $\xi = \xi \times \frac{\Gamma t}{K}$, and $\sigma = \sigma + \xi$
9: **end while**
10: Set $\pi(0)$ so initial state has probability 1 and all others 0
11: Set $\pi = \pi(0)$ and $y = \pi(0)$
12: **for** $k = 1$ to K **do**
13: Compute $y = yP \times \frac{\Gamma t}{k}$ and $\pi = \pi + y$
14: **end for**
15: Compute $\pi(t) = e^{-\Gamma t}\pi$
16: **return** $\sum$ of all states in $\pi(t)$ that satisfy Φ

Fig. 3. Method to check that a GCM satisfies a transient CSL property.

The first step of the algorithm converts the GCM into a *continuous-time Markov chain* (CTMC) (line 1). Details of this conversion are given below. Next, the algorithm determines the amount of time necessary for the analysis, t, which is essentially the maximum value of the interval in the transient property (line 2). Next, the infinitesimal generator matrix is derived from the CTMC by assigning the negation of the sum of the transition rates out of each state to the diagonal entries of the matrix (line 3). After that, the absolute value of the largest diagonal entry is selected as Γ (line 4) and the discretized stochastic probability matrix, P, is computed (line 5). The remainder of the algorithm analyzes the CTMC using a *transient Markov chain analysis* method known as *uniformization* [33]. The algorithm first needs to know the number of terms in its summation, K, which it determines iteratively (lines 6-9). The algorithm then initializes the initial state's probability to 1 and all other states' probabilities to 0 (line 10) and proceeds by iteratively performing vector-matrix multiplications and vector additions to simulate the evolution of the system's likelihood of being in any state (lines 11-14). After this is complete, the uniformization algorithm normalizes the final probabilities in π (line 15). Finally, after applying this method, each state is now annotated with the probability of being in that state at the specified time. At this point, the final probability for the CSL property is determined by summing over all states in which the right-hand side of the until formula

is satisfied (line 16).

The critical step in this algorithm is the conversion of a GCM into a CTMC. This conversion process begins by creating a sparse matrix where each entry, $p_{i,j}$, represents the rate of moving from state i to state j. Next, a state is created with an encoding of the initial values of the species in the GCM. The conversion then performs a depth first search by changing one species encoding at a time to a higher or lower encoding if they exist in the level set L. Each valid change found this way is pushed onto a stack. The algorithm then pops an encoding off the stack and checks to see if a transition rate for moving from the current state to the new state exists in the matrix. If it does, the algorithm stops exploring this path and pops the next change off the stack to explore further. Otherwise, the transition rate is calculated using the equations below and is added to the matrix. During exploration, the algorithm also checks to see if the new encoding either satisfies or fails the supplied CSL property. Since this method accepts only transient CSL properties, this means that the encoding either does not satisfy the left hand side of an until formula or satisfies the right hand side of the formula. If this is the case, the state is marked as absorbing and no further exploration is done from that state.

The transition rates between the states are determined using the formulas below:

$$\begin{aligned} \text{production}(s,l,l') &= \frac{\sum_{p \in Pro(s)} n_p \cdot \text{rate}(p)}{(l'-l)} \\ \text{degradation}(s,l,l') &= \frac{k_d l'}{(l'-l)} \end{aligned}$$

When the state transition increases the level of species s from l to l', then the production formula is used, and when the state transition decreases the level of s from l' to l, then the degradation formula is used. The rate for production is computed by determining the rate of production for each promoter p which produces species s using the rate(p) function defined below. This rate is then multiplied by n_p, the number of proteins produced per transcript, to convert this rate into the rate for a single protein production. The rate of degradation is computed as $k_d l'$ where k_d is the degradation rate parameter and l' is the starting level for species s before degradation. In both cases, these rates must be normalized by the difference in the level before and after the state change. This is because the rates are for the production or degradation of a single molecule of s while the state change only occurs after $l'-l$ molecules are produced or degraded.

Using a *quasi-steady-state approximation*, the function rate(p), shown below, can be derived which returns the rate of production initiated from promoter p.

$$\text{rate}(p) = \begin{cases} \dfrac{n_p k_o n_g K_o n_r}{1 + K_o n_r + \sum_{s_r \in \text{Rep}(p)} (K_r v_{s_r})^{n_c}} & \text{if } |\text{Act}(p)| = 0 \\ \dfrac{n_p k_b n_g K_o n_r + \sum_{s_a \in \text{Act}(p)} n_p k_a n_g K_{oa} n_r (K_a v_{s_a})^{n_c}}{1 + K_o n_r + \sum_{s_r \in \text{Rep}(p)} (K_r v_{s_r})^{n_c} + \sum_{s_a \in \text{Act}(p)} K_{oa} n_r (K_a v_{s_a})^{n_c}} & \text{otherwise} \end{cases}$$

The derivation of this function is a bit involved, so this paper just presents an informal overview of the process. The rate function is made up of constants which can be found in Table I and variables for the repressing species, v_{s_r}, and activating species, v_{s_a}, for this promoter. This function breaks this down into two cases. The first case is for a promoter that does not have any species which are activating it. In this case, it is assumed that the promoter is *constitutive* which simply means that it can initiate transcription at a significant rate without the aid of another activating species. Assuming that there are no repressor molecules present, this rate is approximately $n_p k_o n_g$ where n_p is the number of proteins produced per mRNA produced, k_o is the transcription rate for a constitutive promoter, and n_g is the number of copies of the gene. However, this rate is reduced as the number of repressor molecules, v_{s_r}, increases. The second case is for a promoter that must be activated for significant transcription. Assuming that there are no activator or repressor molecules present, the rate of production of this promoter is $n_p k_b n_g$ where k_b is a low *basal* rate of production which is typically much smaller than k_o. In this case, as the number of activator molecules, v_{s_a}, increases so does the rate of production from this promoter. Like the first case, this production can also potentially be inhibited, if there exists species which can repress this promoter. Lastly, if there are complex formation reactions between repressing species and chemical inducers such as that between LacI and IPTG in Figure 1(c), we apply a complex formation abstraction that uses both the quasi-steady-state approximation and the law of mass conservation. This abstraction replaces the variable v_{s_r} in the rate function with the expression $\frac{v_{sr_{total}}}{1+K_c v_i}$, where $v_{s_{r_{total}}}$ is the variable for the total repressing species (free and in complex), K_c is the complex formation equilibrium constant, and v_i is the variable for the chemical inducer. Consequently, as the amount of chemical inducer increases, the effective amount of repressing species decreases and production from the promoter increases. For more details about derivation of rates using the quasi-steady-state approximation, please see, for example, [34].

The conversion process coupled with the corresponding rate functions have been carefully constructed such that the CTMC generated gives a reasonable approximation of the behavior of the GCM. This translation procedure allows the user to efficiently trade-off between accuracy and analysis time. Namely, the more levels used in the level set, the more accurate the model becomes. Of course, using more levels also increases analysis time, so the user should select the minimal number of levels necessary to perform the desired analysis. While we have had very good results when making simple, intuitive choices of these levels, further research on determining better levels automatically is of interest.

An example of using this method to analyze the GCM in Figure 1(c) is presented in Figure 4. Here, the levels are selected at 0, 30, and 60 for LacI and TetR and the initial value is 60 for LacI and 0 for TetR. For simplicity, in this model, we have assumed that IPTG, aTc, C1, and C2 are 0 (and remain

0) while GFP is not shown as it follows TetR's value. The property being considered is the probability of LacI going to 0 within 100 seconds which is represented by the following CSL property:

$$\mathtt{P}_{=?}\{\Diamond^{[0,100]}\,\mathrm{LacI}=0\}$$

There are nine states in the resulting CTMC; however, since states *S*6, *S*7, and *S*8 satisfy the property, outgoing transitions from these states are pruned. The sum of the probability of reaching these states represents the probability of satisfying the property, which is about 5.9 percent.

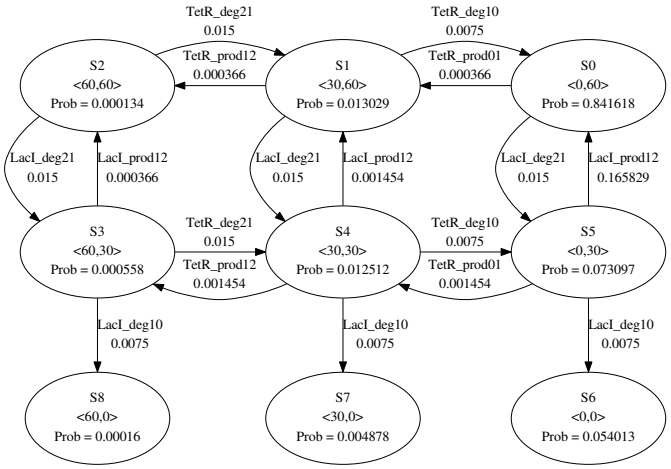


Fig. 4. The CTMC shown graphically annotated with probabilities after applying stochastic model checking with the CSL property, $\mathtt{P}_{=?}\{\Diamond^{[0,100]}\,\mathrm{LacI}=0\}$, to the GCM in Figure 1(c) with levels selected at 0, 30, and 60 for LacI and TetR and an initial value of 60 for LacI and 0 for TetR. Note that states *S*6, *S*7, and *S*8 are absorbing since they satisfy the CSL property.

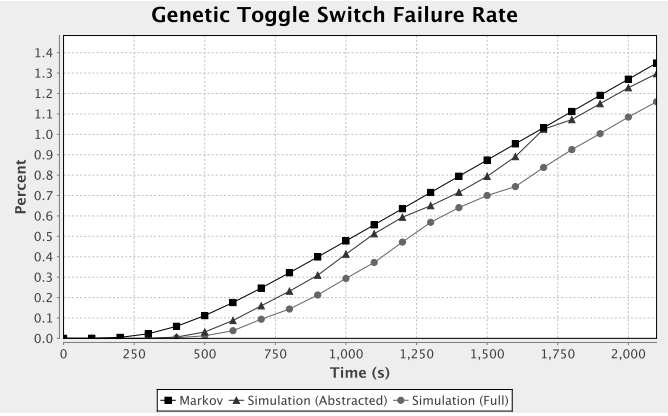
IV. Results

This section presents the application of our methodology to the analysis of the genetic toggle switch that has been the running example in the previous sections. A useful experiment for this circuit is to determine the probability that it changes state erroneously within a cell cycle (2,100 seconds) which occurs if some spurious production of the low signal inhibits the high signal enough to allow it to degrade away and switch state. For this experiment, the toggle switch is initialized to a starting state where LacI is set to a high state of 60 molecules and TetR is set to a low state of 0 molecules. In order to test whether or not it changes state, the following CSL property is checked:

$$\mathtt{P}_{=?}\{\Diamond^{[0,2100]}\,\mathrm{LacI}<20\wedge\mathrm{TetR}>40\}$$

This property makes states absorbing in which LacI has dropped below 20 (the low state) and TetR has risen above 40 (the high state). For this analysis, the 9 levels are selected for LacI uniformly distributed between 0 and 80, and 11 levels are selected for TetR uniformly distributed between 0 and 50, which produces a CTMC with 99 states. Levels are selected to ensure that one of the levels captures the initial amount for each species and that the levels span over the possible values for each species going slightly above and below the property bounds. It should be emphasized that this is a very simple and straightforward choice for the levels.

Figure 5 shows a comparison of results found using 32,000 simulation runs both with and without reaction-based abstraction [19] and applying transient Markov chain analysis. This figure shows that the transient Markov chain analysis tracks the simulation results fairly closely. However, as shown in Table II, the transient Markov chain analysis method greatly outperforms the simulation based methods.



(a)

Fig. 5. Time course plot showing the probability of the genetic toggle switch changing state erroneously. This plot compares the results of using simulation both with and without reaction-based abstraction and analysis of the CTMC using Markov chain analysis with the same CSL property, $\mathtt{P}_{=?}\{\Diamond^{[0,2100]}\,\mathrm{LacI}<20\wedge\mathrm{TetR}>40\}$. The simulation results use 32,000 simulation runs in order to achieve a relative error bound of 10 percent assuming a probability of failure of 1.2 percent.

TABLE II
Genetic Toggle Switch Analysis Run-time Comparison

	Failure Rate	Response Rate
Simulation w/o Abstraction	43 minutes	3 hours 12 minutes
Simulation w/ Abstraction	3 minutes 15 seconds	1 minute
Markov Chain Analysis	1 second	0.5 seconds

The choice of 32,000 simulation runs is chosen in order to achieve 95 percent confidence that the result is within 10 percent assuming the true failure rate is 1.2 percent, the approximate value after 2100 seconds. This value is determined using the equation below:

$$d \quad = \quad 1.96\times\sqrt{\frac{1-p}{p\times n}} \tag{1}$$

where d is the relative error bound, p is the predicted probability, and n is the number of simulation runs [35], [36]. It should be noted that for earlier time points where the failure rate is lower, the error increases. For example, at 1000 seconds, the full simulation predicts a probability of failure of 0.3 percent, but we are only 95 percent confident that this result is within 20 percent of the true value.

The next experiment is to determine the response time of the circuit when switching from the off state to the on state,

and these results are presented in Figure 6. This analysis uses the same CSL property but a slightly different initial condition. As before, LacI is set to 60 and TetR is set to 0, but IPTG is set to 100 representing that it has just been added to set the toggle switch to the high state. For this experiment, 14 levels for LacI are selected uniformly distributed between 0 and 130, since individual simulation results show it reaching a much higher value than in the last experiment. For TetR, only 5 levels are used uniformly selected between 0 to 60 because less resolution is required for catch its change from a low to high state. This results in a CTMC of 70 states. Again, transient Markov chain analysis tracks the simulation results fairly closely ending up with a final probability of 98.7 percent while the simulation of the full model results in 98.9 percent. Also like the previous example, the transient Markov chain analysis method outperforms the simulation-based approaches as shown by the run-times in Table II.

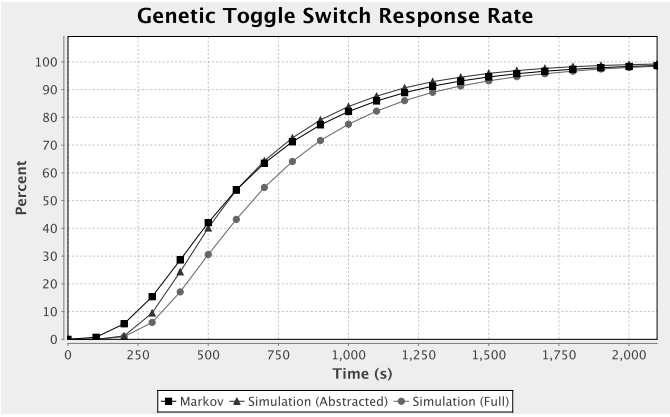
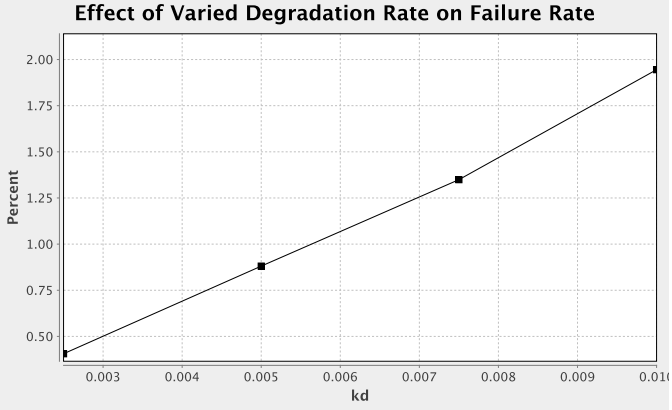


Fig. 6. Time course plot showing the probability of the genetic toggle switch changing state correctly in response to an input change. Like Figure 5, this plot compares the results of using simulation both with and without reaction-based abstraction and analysis of the CTMC using Markov chain analysis with the same CSL property, $\mathrm{P}_{=?}\{\Diamond^{[0,2100]}\ \mathrm{LacI} < 20 \wedge \mathrm{TetR} > 40\}$, but with a different initial value of IPTG.

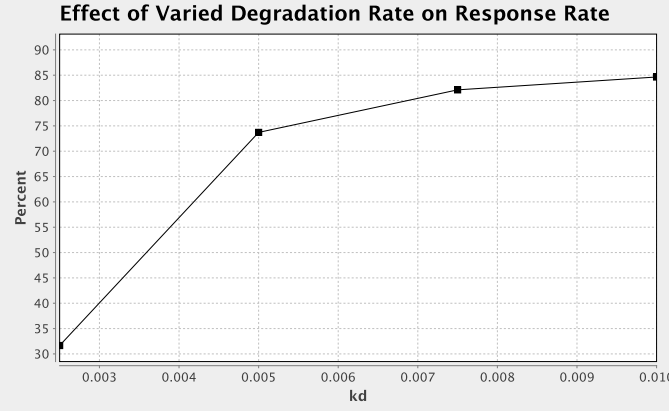
With this analysis method, the design space can be efficiently explored. For example, a genetic designer may consider the effect of parameter variation on robustness and performance. One important parameter for the genetic toggle switch is the degradation rate, k_d, and the results of varying this parameter are shown in Figure 7. These results indicate that tuning the degradation rate has a significant effect. If it is too high, the circuit is less robust, but if it is too low, it responds too slowly.

V. Discussion

Utilizing stochastic model checking, synthetic biologists can explore the effect of varying parameters in their genetic circuits more rapidly than using traditional methods allowing them to rapidly explore design trade-offs in an effort to make their circuits more responsive to inputs and more robust to failures. The methods discussed here have been implemented in a tool called `iBioSim` [8] which is freely available at



(a)



(b)

Fig. 7. (a) Plot depicting the probability of the genetic toggle switch changing state erroneously within 2100 seconds for different values of k_d. (b) Plot depicting the probability of the genetic toggle switch changing state correctly within 1000 seconds in response to input change for different values of k_d.

`http://www.async.ece.utah.edu/iBioSim/`. This tool includes a schematic capture tool for constructing GCM representations of designs, automated model construction and abstraction, and a variety of simulation and visualization methods. While we believe that this tool and the methods described in this paper represent an excellent first step towards a fully functional *genetic design automation* (GDA) tool, there is still significant work that needs to be done.

One area of improvement would be in a better method for choosing the levels used in the generation of the CTMC. Currently, a user performs a small number of simulation runs in order to get an idea for the range of values of interest, and then selects a number of levels in which to divide this range uniformly. While adding more levels improves accuracy, it does so at the cost of a larger state space which makes the CTMC analysis less efficient. While the resulting CTMC can likely still be analyzed using simulation methods, a better approach would be to make a better level assignment. For example, using a non-uniform choice of levels may be more efficient, since it may result in more accuracy at a lower state space size. Using a non-uniform choice of levels though would likely require an automated selection to assist in this

choice. We have done some preliminary investigations which are promising that analyze the rate equations of the original GCM in order to determine the levels to use.

In addition to this work, we are also developing a variant of the SSA called the *incremental stochastic simulation algorithm* (iSSA) [37], [38]. The idea behind this algorithm is to perform stochastic simulations in small time increments, compute statistics at the end of each increment, and determine a new starting state for each run at the beginning of the next increment with these statistics. This allows users to perform many simulation runs to observe the "typical" behavior of their systems instead of simply averaging several SSA runs together which can often hide interesting behaviors due to a washing out effect. Ideally, iSSA and stochastic model checking could be used in concert to improve the genetic circuit design process. For example, the typical behavior of the system could be determined using iSSA, then the probability of this typical behavior being observed could be checked using a CSL property and stochastic model checking.

ACKNOWLEDGMENT

This work is supported by the National Science Foundation under Grants CCF-0916105 and CCF-0916042.

REFERENCES

[1] D. Endy, "Foundations for engineering biology," *Nature*, vol. 438, pp. 449–453, 2005.

[2] A. Arkin, "Setting the standard in synthetic biology," *Nature Biotech.*, vol. 26, pp. 771–774, 2008.

[3] J. Beal and J. Bachrach, "Cells are plausible targets for high-level spatial languages," *Proceedings of the 2008 Second IEEE International*, pp. 284–291, 2008.

[4] L. Bilitchenko, A. Liu, S. Cheung, E. Weeding, B. Xia, M. Leguia, J. C. Anderson, and D. Densmore, "Eugene - a domain specific language for specifying and constraining synthetic biological parts, devices, and systems," *PLoS ONE*, vol. 6, no. 4, 2011.

[5] Y. Cai, M. L. Wilson, and J. Peccoud, "GenoCAD for iGEM: a grammatical approach to the design of standard-compliant constructs," *Nucleic Acids Res.*, vol. 38, no. 8, pp. 2637–44, 2010.

[6] D. Chandran, F. T. Bergmann, and H. M. Sauro, "TinkerCell: modular CAD tool for synthetic biology," *J. Biol. Eng.*, vol. 3, no. 19, 2009.

[7] M. A. Marchisio and J. Stelling, "Computational design of synthetic gene circuits with composable parts," *Bioinform.*, vol. 24, no. 17, pp. 1903–10, 2008.

[8] C. J. Myers, N. Barker, K. Jones, H. Kuwahara, C. Madsen, and N.-P. D. Nguyen, "iBioSim: a tool for the analysis and design of genetic circuits," *Bioinform.*, vol. 25, no. 21, pp. 2848–2849, 2009.

[9] M. A. Marchisio and J. Stelling, "Automatic design of digital synthetic gene circuits," *PLoS Comput Biol*, vol. 7, no. 2, Feb. 2011.

[10] M. Pedersen and A. Phillips, "Towards programming languages for genetic engineering of living cells," *J. R. Soc. Interface*, vol. 6, no. (Suppl. 4), pp. S437–S450, 2009.

[11] B. Yordanov, J. Tumova, C. Belta, I. Cerna, and J. Barnat, "Formal analysis of piecewise affine systems through formula-guided refinement," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5899–5904.

[12] P. Waage and C. M. Guldberg, "Studies concerning affinity," *Forhandlinger: Videnskabs - Selskabet i Christinia*, vol. 35, 1864.

[13] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The art of Scientic Computing, 2nd ed.* Cambridge University Press, 1992.

[14] S. H. Strogatz, *Nonlinear Dynamics And Chaos: With Applications To Physics, Biology, Chemistry And Engineering*. Westview Press, 1994.

[15] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *J. Phys. Chem.*, vol. 81, no. 25, pp. 2340–2361, 1977.

[16] M. Gibson and J. Bruck, "Efficient exact stochastic simulation of chemical systems with many species and many channels," *J. Phys. Chem.*, vol. A 104, pp. 1876–1889, 2000.

[17] Y. Cao, D. T. Gillespie, and L. R. Petzold, "Efficient step size selection for the tau-leaping simulation method," *J. Chem. Phys.*, vol. 124, 2006.

[18] D. T. Gillespie and L. R. Petzold, "Tau leaping," *J. Chem. Phys.*, vol. 119, pp. 8229–8234, 2003.

[19] H. Kuwahara, C. Myers, N. Barker, M. Samoilov, and A. Arkin, "Automated abstraction methodology for genetic regulatory networks," *Trans. Comp. Syst. Biol.*, vol. VI, pp. 150–175, 2006.

[20] H. Kuwahara, "Model abstraction and temporal behavior analysis of genetic regulatory networks," Ph.D. dissertation, U. of Utah, 2007.

[21] D. Thieffry and R. Thomas, "Dynamical behaviour of biological networks: II. Immunity control in bacteriophage lamabda," *Bull. Math. Biol.*, vol. 57, no. 2, pp. 277–297, 1995.

[22] R. Thomas, "Regulatory networks seen as asynchronous automata: A logical description," *Journal of Theoretical Biology*, vol. 153, pp. 1–23, 1991.

[23] T. Henzinger, M. Mateescu, and V. Wolf, "Sliding window abstraction for infinite markov chains," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, A. Bouajjani and O. Maler, Eds. Springer Berlin / Heidelberg, 2009, vol. 5643, pp. 337–352.

[24] F. Ciocchetta, A. Degasperi, J. Hillston, and M. Calder, "Some investigations concerning the CTMC and the ODE model derived from Bio-PEPA," *Electronic Notes in Theoretical Computer Science*, vol. 229, no. 1, pp. 145 – 163, 2009, proceedings of the Second Workshop From Biology to Concurrency and Back (FBTC 2008).

[25] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic model checking," in *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, ser. LNCS (Tutorial Volume), M. Bernardo and J. Hillston, Eds., vol. 4486. Springer, 2007, pp. 220–270.

[26] T. S. Gardner, C. R. Cantor, and J. J. Collins, "Construction of a genetic toggle switch in *escherichia coli*," *Nature*, vol. 403, pp. 339–342, 2000.

[27] M. Hucka *et al.*, "The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models," *Bioinform.*, vol. 19, no. 4, pp. 524–531, 2003.

[28] N. Nguyen, "Design and analysis of genetic circuits," Master's thesis, U. of Utah, 2008.

[29] N. Nguyen, C. Myers, H. Kuwahara, C. Winstead, and J. Keener, "Design and analysis of a robust genetic Muller C-element," *Journal of Theoretical Biology*, vol. 264, no. 2, pp. 174 – 187, 2010.

[30] H. Younes, M. Kwiatkowska, G. Norman, and D. Parker, "Numerical vs. statistical probabilistic model checking," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 8, pp. 216–228, 2006, 10.1007/s10009-005-0187-8.

[31] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: A tool for automatic verification of probabilistic systems," in *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, ser. LNCS, H. Hermanns and J. Palsberg, Eds., vol. 3920. Springer, 2006, pp. 441–444.

[32] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton, "Model-checking continuous-time markov chains," *ACM Trans. Comput. Logic*, vol. 1, pp. 162–170, July 2000.

[33] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*. 41 William Street, Princeton, NJ, 08540: Priceton University Press, 1994.

[34] C. J. Myers, *Engineering Genetic Circuits*. Chapman and Hall/CRC, 2009.

[35] H. Kuwahara and I. Mura, "An efficient and exact stochastic simulation method to analyze rare events in biochemical systems," *The Journal of Chemical Physics*, vol. 129, no. 16, 2008.

[36] D. T. Gillespie, M. Roh, and L. R. Petzold, "Refining the weighted stochastic simulation algorithm," *J Chem Phys*, vol. 130, no. 17, 2009.

[37] C. Winstead, C. Madsen, and C. J. Myers, "iSSA: An incremental stochastic simulation algorithm for genetic circuits," in *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2010, pp. 553–556.

[38] H. Kuwahara, C. Madsen, I. Mura, C. Myers, A. Tejada, and C. Winstead, "Efficient stochastic simulation to analyze targeted properties of biological systems," in *Stochastic Control*, C. Myers, Ed. Sciyo, 2010, pp. 505–532.