# Delayed Continuous-Time Markov Chains for Genetic Regulatory Circuits [⋆]

Călin C. Guet, Ashutosh Gupta, Thomas A. Henzinger, Maria Mateescu, Ali Sezgin

IST Austria, Klosterneuburg, Austria

**Abstract.** Continuous-time Markov chains (CTMC) with their rich theory and efficient simulation algorithms have been successfully used in modeling stochastic processes in diverse areas such as computer science, physics, and biology. However, systems that comprise non-instantaneous events cannot be accurately and efficiently modeled with CTMCs. In this paper we define delayed CTMCs, an extension of CTMCs that allows for the specification of a lower bound on the time interval between an event's initiation and its completion, and we propose an algorithm for the computation of their behavior. Our algorithm effectively decomposes the computation into two stages: a pure CTMC governs event initiations while a deterministic process guarantees lower bounds on event completion times. Furthermore, from the nature of delayed CTMCs, we obtain a parallelized version of our algorithm. We use our formalism to model genetic regulatory circuits (biological systems where delayed events are common) and report on the results of our numerical algorithm as run on a cluster. We compare performance and accuracy of our results with results obtained by using pure CTMCs.

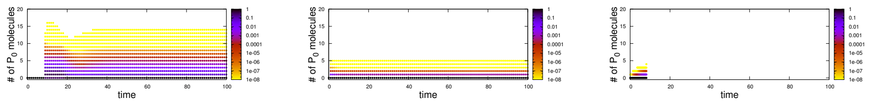
## 1 Introduction

Due to the Brownian motion of molecules inside cells, biological systems are inherently stochastic. The stochastic effects are negligible when all species are present in large numbers, but can be significant when some of the species are present only in low numbers. In particular, when modeling genetic regulatory circuits (GRCs), where different molecules (such as DNA) are present in low numbers, one needs to take stochasticity into account. Indeed, systems biology has been shifting its focus from deterministic models that capture the mean behavior of GRCs to stochastic models that capture their stochastic behavior [9].

One of the most general modes of gene regulation is self-regulation in the form of a *negative feedback* loop [19], where a DNA molecule encodes a repressor protein $R$ with which it interacts according to the following reactions:

$$\mathrm{DNA} \rightarrow \mathrm{DNA} + \mathrm{R}, \quad \mathrm{R} \rightarrow \emptyset, \quad \mathrm{DNA} + \mathrm{R} \rightarrow \mathrm{DNA.R}, \quad \mathrm{DNA.R} \rightarrow \mathrm{DNA} + \mathrm{R},$$

[⋆] This work was supported by the ERC Advanced Investigator grant on Quantitative Reactive Modeling (QUAREM) and by the Swiss National Science Foundation.



(a) Delayed (b) Immediate (c) Cascade

**Fig. 1.** Probability distribution of the levels of repressor protein over time in three different models of the negative feedback loop. (a) an overshoot is observed in the delayed CTMC model; (b) steady state is reached rapidly in the immediate CTMC; (c) the model is intractable in the cascade CTMC and thus we present only the distribution up to time $7.2s$.

that correspond to the production, degradation, binding, and unbinding of protein. Due to biological aspects, the production of proteins behaves as a process with high latency and high throughput. Therefore, if at time $t = 0$ the system contains a single DNA molecule, the production of a large number of proteins is initiated because, before the completion of any of the protein production processes, nothing inhibits the production of proteins (Fig. 1(a)). Consequently, the observed behavior of the system is crucially dependent on the presence of delays, a well known phenomenon in biological systems [5, 9, 11].

A classic modeling formalism for GRCs is offered by continuous-time Markov chains (CTMCs) as proposed by Gillespie [7]. Under the assumptions that the solution is well-stirred, at constant temperature and volume, and that reactions happen instantaneously, the CTMC model considers (i) that the state of the system is given by a population vector whose elements represent the copy number for each molecule type, and (ii) that reactions are triggered at times that follow an exponential distribution, with a rate that depends on the quantum mechanical properties of molecules and on the likelihood of their collision. A popular technique for analyzing this type of models is *probability propagation* [12], which computes the transient probabilities of the Markov chain over time by pushing probability distribution through the state space of the model.

In the classical CTMC model of the negative feedback system each state of the model has three variables that denote the number of DNA (0 or 1), DNA.R (0 or 1) and R (a natural number) molecules. Each state has up to four successors, one for each of the enabled reactions (production, degradation, binding and unbinding of repressor protein). Because reactions happen instantaneously, without any latency, with some strictly positive probability proteins are available at any $t > 0$, and thus can inhibit further production of proteins immediately. The overshooting behavior that is normally present in the negative feedback loop is thus not observed in this *immediate CTMC* model (Fig. 1(b)). In order to overcome this problem, a pure CTMC model needs to use additional auxiliary variables that encode the age of a molecule, and thus produce a delay-like behavior. This increase in the number of variables leads however to a state space explosion that makes even simple models intractable (Fig. 1(c)). We call such a CTMC, extended with auxiliary variables, a *cascade CTMC*.

In this paper, we introduce *delayed CTMCs*, a formalism that we argue to be more natural and more efficient than pure CTMCs when modeling systems that comprise non-instantaneous reactions. In a delayed CTMC with delay $\Delta$, each species $x$ has an associated age $\alpha(x)$ that specifies that $(\alpha(x)-1)\cdot\Delta$ time units must pass between the moment when a reaction that produces $x$ is triggered and the moment when $x$ is available as a reactant of a new reaction.

**Natural.** Delayed CTMCs naturally express non-instantaneous reactions because both the throughput and the latency of a reaction have direct correspondents in the rate of the reaction and the delay of the reaction, respectively. Even though one can try to model both the latency and throughput of reactions in a pure CTMC by adding auxiliary reactions, the determination of the number and parameters of such reactions involve the manipulation of complex functions. Furthermore, one cannot simply approximate the computation of these parameters because it is not clear how such approximations affect the qualitative behavior of the model.

**Efficient.** Delayed CTMC have two important performance advantages with respect to cascade CTMCs. First, by decoupling the waiting times of molecules in the process of being produced from the dynamics of the currently available molecules, we reduce the size of the state space on which to run a pure CTMC computation. Second, since no probability can flow between states with different number of waiting molecules (molecules that are not yet "mature"), our probability transition matrix accepts a simple partitioning, which is efficiently parallelizable.

The algorithm that we propose for the computation of the probability propagation of the delayed CTMC consists of alternating rounds of pure CTMC computation and aging steps. Each pure CTMC computation can be parallelized due to the absence of interactions between subspaces of the model, and thus we are able to solve delayed CTMC models that have large state spaces. For example, we solve the negative feedback example for parameters that generate a state space of up to 112 million states. The result of these experiments (see Figure 1) show that the delayed CTMC model of the negative feedback system indeed matches the experimental evidence of an initial overshoot in the production of protein [9], while the pure CTMC models do not.

Reaction delays have already been embedded in stochastic simulation tools for GRCs [18], but we are not aware of any work that embeds such delays in a probability propagation algorithm. The probability propagation problem relates to stochastic simulations as verification relates to testing. Due to the advantages of probability propagation algorithms over simulation based approaches (when computing the transient probabilities of a system) [3], it is valuable to provide such algorithms. As probability propagation of stochastic systems is analogous to solving reachability problems in the non-stochastic setting, we use techniques such as discretization of a continuous domain and on-the-fly state space exploration, which are standard techniques for verification problems.

**Related work.** There has been a wide interest in defining formalisms for genetic regulatory circuits [2, 7, 15, 17]. In particular, using delays within stochastic simulations has recently drawn interest (see [18] for a review of these models). Delayed CTMCs differ from these models in that they discretize the delay time as opposed to having continuous delays.

There has also been much work on the transient analysis of pure CTMCs coming especially from the field of probabilistic verification [8, 10, 13, 14], but none of these methods consider reaction delays.

Recent efforts [1, 20] have parallelized the transient analysis of CTMCs by applying parallel algorithms for sparse matrix multiplication. Since the data dependencies flow across the entire state space, they have achieved limited speed-ups. Our work is orthogonal to these approaches. Due to the nature of delayed CTMCs, the probability transition matrix of the model is amenable to being partitioned into disconnected sub-matrices, and thus we obtain a highly parallelizable algorithm. This can lead to large speed-ups (hundreds of times for our examples). The techniques presented in these related works can be used for further parallelization of our algorithm by using them in the computation of each disconnected part of the matrix.

Delayed CTMCs are a subclass of generalized Markov processes (GMPs). Our extension of CTMCs is limited as compared to GMPs, in that our single extension is in capturing the property of latent reactions. However, we are able to find efficient probability propagation algorithms for delayed CTMCs.

## 2 Delayed CTMC

For a set $A$, let $f|_A$ denote the function obtained by restricting the domain of $f$ to $A \cap Dom(f)$. $Pow(A)$ denotes the powerset of $A$.

**Continuous-time Markov chain (CTMC).** A probability distribution $\rho$ over a countable set $A$ is a mapping from $A$ to $[0, 1]$ such that $\sum_{a\in A} \rho(a) = 1$. The set of all probability distributions over $A$ is denoted by $\mathcal{P}^A$. The *support* $F_\rho$ of a probability distribution $\rho$ over $A$ is the subset of $A$ containing exactly those elements of $A$ mapped to a non-zero value. Formally, $F_\rho = \{a \in A \mid \rho(a) \neq 0\}$.

A CTMC $M$ is a tuple $(S, \Lambda)$, where $S$ is the set of *states*, and $\Lambda : S \times S \mapsto \mathbb{R}$ is the *transition rate matrix*. We require that for all $s, s' \in S$, $\Lambda(s, s') \geq 0$ iff $s \neq s'$, and $\sum_{s'\in S} \Lambda(s, s') = 0$.

The *behavior* of $M = (S, \Lambda)$ is a mapping $\mathbf{p}_M$ from $\mathbb{R}$ to a probability distribution over $S$ satisfying the *Kolmogorov differential equation*

$$\frac{d}{dt}\,\mathbf{p}_M(t) = \mathbf{p}_M(t) \cdot \Lambda$$

If the value of $\mathbf{p}_M(0)$ is known, the above differential equation has the unique solution

$$\mathbf{p}_M(t) = \mathbf{p}_M(0) \cdot e^{\Lambda t}$$

In the case where $|S| < \infty$, the series expansion for $e^{\Lambda t}$ yields $\sum_{i=0}^{\infty} (\Lambda t)^i/i!$ for which analytic solutions can be derived only for special cases. In general, finding the probability distribution $\mathbf{p}_M$ as a symbolic function of time ($t$) is not possible.

We will let $M(\rho, t)$, the behavior of $M$ initiated at $\rho$, denote the value of $\mathbf{p}_M(t)$ with $\mathbf{p}_M(0) = \rho$.

**Aging boundary, configurations.** An *aging boundary* is a pair $(X, \alpha)$, where $X$ is a finite set of *variables*, $\alpha : X \mapsto \mathbb{N}$ is an *age* function. The *expansion* of the aging boundary $(X, \alpha)$, is the set $[X, \alpha] = \{(x, a) \,|\, x \in X, 0 \leq a \leq \alpha(x)\}$, the elements of which are called *aged variables*. For an expansion $[X, \alpha]$, we define the sets of *immediate*, *new* and *waiting* variables as $[X, \alpha]^i = \cup_{x \in X}\{(x, 0)\}$, $[X, \alpha]^n = \cup_{x \in X}\{(x, \alpha(x))\}$, and $[X, \alpha]^w = \{(x, a) \,|\, x \in X, 0 < a < \alpha(x)\}$, respectively.

A *configuration* $c$ over an expansion $[X, \alpha]$ is a total function from $[X, \alpha]$ to $\mathbb{N}$. We will also write $((x, a), n) \in c$ whenever $c(x, a) = n$. A *sub-configuration* $c_F$ of $c$ relative to $F \subseteq [X, \alpha]$ is the restriction of $c$ to $F$; formally, $c_F(x)$ is defined to be equal to $c(x)$ iff $x \in F$. For any $F \subseteq [X, \alpha]$, $\mathcal{C}^F$ denotes the set of all sub-configurations over $F$. For any configuration $c \in \mathcal{C}^{[X,\alpha]}$, let $c_i$, $c_n$ and $c_w$ denote the sub-configurations of $c$ relative to $[X, \alpha]^i$, $[X, \alpha]^n$ and $[X, \alpha]^w$, respectively. Intuitively, in the context of gene expression, an aged variable will represent a molecule with a time stamp denoting the delay until the molecule is produced, and configurations will represent a a collection of molecules with time stamps. In what follows, we will be exclusively working on CTMCs having configurations as states, and thus will use the two terms, states and configurations, interchangeably.

**Delayed CTMCs.** Let $(X, \alpha)$ be an aging boundary. A CTMC $M = (\mathcal{C}^{[X,\alpha]}, \Lambda)$ is *$\alpha$-safe*, if $\Lambda(c, c') \neq 0$ implies that $c'_w = c_w$ and also, $c'(x, \alpha(x)) < c(x, \alpha(x))$ implies $\alpha(x) = 0$. Intuitively, $\alpha$-safe means that the waiting variables cannot change and only the value of immediate variables can decrease.

**Definition 1 (Delayed CTMC).** *A* delayed CTMC *$D$ is a tuple $(X, \alpha, \Lambda, \Delta)$, where $(X, \alpha)$ is an aging boundary, $M_D = (\mathcal{C}^{[X,\alpha]}, \Lambda)$ is $\alpha$-safe, and $\Delta \in \mathbb{R}^+$ is the* delay*.*

A *behavior* of a delayed CTMC $D = (X, \alpha, \Lambda, \Delta)$ is a finite sequence $\rho_0\rho_1\rho_2 \ldots \rho_N$ of probability distributions over $\mathcal{C}^{[X,\alpha]}$ that satisfies

$$\rho_{i+1} = \mathsf{Tick}(M_D(\rho_i, \Delta)) \qquad \text{for } 0 \leq i < N \tag{1}$$

The definition of $\mathsf{Tick}$ is given as

$$\mathsf{Tick}(\rho)(c') \overset{def}{=} \sum_{c^{+1}=c'} \rho(c), \quad c^{+1}(x, a) = \begin{cases} c(x, a+1) & \text{, if } 0 < a < \alpha(x) \\ c(x, a+1) + c(x, a) & \text{, if } a = 0 \\ 0 & \text{, if } a = \alpha(x) \end{cases}$$

Intuitively, a *behavior* of a delayed CTMC $D$ is an alternating sequence of running the CTMC $M_D$ for $\Delta$ units of time, *deterministically* decrementing the age of each variable in every configuration (i.e. propagating probability from $c$ to $c^{+1}$), and computing the new probability distribution ($\mathsf{Tick}$).

A *continuous behavior* of $D = (X, \alpha, \Lambda, \Delta)$ is given as

$$\mathbf{b}_D(t) \overset{def}{=} M_D(\rho_q, z)$$

where $t = q \cdot \Delta + z$ for some $0 \leq z < \Delta$, and $\rho_0 \ldots \rho_q$ is a behavior of $D$.

## 3 Genetic Regulatory Circuits

In this section, we will first give a simple formalism for defining genetic regulatory circuits. We will then provide three different semantics for genetic regulatory circuits using delayed CTMCs.

### 3.1 Specifying Genetic Regulatory Circuits

A genetic regulatory circuit (GRC) $G$ is a tuple $(X, \alpha, R)$, where

- $(X, \alpha)$ is an aging boundary,
- $R$ is a set of *reactions*.

Each reaction $r \in R$ is a tuple $(i_r, \tau, o_r)$, where $i_r$ and $o_r$, the *reactant* and *production* list, respectively, are mappings from $X$ to $\mathbb{N}$, and $\tau$, the *reaction rate*, is a positive real-valued number. For reactions, we will use the more familiar notation

$$a_1 x_1 + \ldots + a_n x_n \xrightarrow{\tau} b_1 x_1 + \ldots + b_n x_n$$

where $a_j = i_r(j)$, and $b_j = o_r(j)$. Intuitively, each reaction represents a chemical reaction, and variables of $X$ represent the molecular species that take part in at least one reaction of the system. Each reaction defines the necessary number of each molecular species that enables a chemical reaction, the base rate of the reaction, and the number of produced molecular species as a result of this chemical reaction.

Overall, a reaction can be seen as a *difference vector* $\mathbf{r} = [r_1 \; r_2 \; \ldots \; r_n]$, where $r_i = o_r(x_i) - i_r(x_i)$ is the net change in the number of molecule $x_i$. We will assume that the difference vector of each $r$ is unique. In writing down a reaction, we will leave out the molecular species that are mapped to 0.

### 3.2 Dynamics in terms of Delayed CTMCs

In this section, we will give the semantics of a GRC $G = (X, \alpha, R)$ in terms of a delayed CTMC. A computation framework for $G$ is parameterized over delay values. For the following, we fix a delay value, $\Delta$.

A reaction $r \in R$ is *enabled* in a configuration $c$, if for all $x_i \in X$, $c(x_i, 0) \geq i_r(x_i)$ holds. In other words, reaction $r$ is enabled in $c$ if the number of reactants that $r$ requires is at most as high as the number of immediately available reactants in $c$. Let $En(c) \subseteq R$ denote the set of reactions enabled in $c$.

For configurations $c, c'$, and reaction $r \in R$, we say that $c$ can go to $c'$ by firing $r$, written $c \xrightarrow{r} c'$, if $r \in En(c)$, and there exists a configuration $\widehat{c}$ such that

- $c$ and $\widehat{c}$ are the same except for all $x_i \in X$, $\widehat{c}(x_i, 0) = c(x_i, 0) - i_r(x_i)$, and
- $c'$ and $\widehat{c}$ are the same except for all $x_i \in X$, $c'(x_i, \alpha(x_i)) = \widehat{c}(x_i, \alpha(x_i)) + o_r(x_i)$.

Informally, to move from configuration $c$ to $c'$ via reaction $r$, $c$ must have at least as many *immediate* molecules as required by the reactant list of $r$, and $c'$ is obtained by removing all immediate molecules consumed by $r$ and adding all the *new* molecules produced by $r$.

**Delayed semantics.** For $G = (X, \alpha, R)$, we define the delayed CTMC $D_G = (X, \alpha, \Lambda, \Delta)$. We only need to give the definition of $\Lambda$.

$G$ induces the transition rate matrix $\Lambda$ defined as $\Lambda(c, c') = \mathsf{Fire}(c, r)$ only when $c \xrightarrow{r} c'$ holds. $\mathsf{Fire}(c, r)$ is given as

$$\mathsf{Fire}(c, r) = \prod_{x_i \in X} \tau_r \binom{c(x_i, 0)}{i_r(x_i)}$$

where $\binom{n}{r} = n!/(n-r)!$ represents the choose operator. $\Lambda(c, c')$ is well-defined because there can be at most one reaction that can satisfy $c \xrightarrow{r} c'$ since we assumed that the difference vector of each reaction is unique. Observe also that no changes to waiting variables can happen in any transition with non-zero rate, and only the number of immediate variables can decrease. Hence, $M_{D_G}$ as defined is $\alpha$-safe.

**Immediate semantics.** Given a GRC $G = (X, \alpha, R)$, we define the *immediate* version of $G$, written $G^{\downarrow}$ as the GRC $(X, \alpha', R)$, where $\alpha'(x) = 0$, for all $x \in X$. Intuitively, $G^{\downarrow}$ ignores all the delays, and treats the reactions as instantaneously generating their products. Note that, a delayed CTMC with an age function assigning 0 to all the variables is a pure CTMC. The immediate semantics for $G$ are given by the behavior of the (delayed) CTMC constructed for $G^{\downarrow}$.

**Cascade semantics.** Given a GRC $G = (X, \alpha, R)$, we define the *cascade* version of $G$, written as $G^*$, as the GRC $(X', \alpha', R')$, where $X' = [X, \alpha]$, $\alpha'(x) = 0$, for all $x \in X'$, and $R' = R_{trig} \cup R_{age}$, where

- $R_{trig}$ is the set of reactions of $R$ re-written in a way that all the reactants have age 0, and all the products have their maximum age. Formally, for each reaction $r = \sum_i a_i x_i \xrightarrow{\tau} \sum_i b_i x_i \in R$, we define $\tilde{r} = \sum_i a_i (x_i, 0) \xrightarrow{\tau} \sum_i b_i (x_i, \alpha(x_i))$ and let $R_{trig} = \{\tilde{r} \mid r \in R\}$.
- $R_{age}$ is the representation of delays in terms of a sequence of fictitious events intended to count down the necessary number of stages. Formally, $R_{age} = \{(x, a) \xrightarrow{\Delta^{-1}} (x, a-1) \mid a > 0, (x, a) \in X'\}$.

Cascade semantics for $G$ are given by the behavior of the (delayed) CTMC constructed for $G^*$.

**Remark.** A protein generated by a GRC has two dynamic properties: the rate of its production and the delay with which it is produced, both of which are intuitively captured by delayed semantics. The immediate semantics keeps the rate intact at the expense of the production delay which is reduced to 0. On the other hand, the cascade semantics approximates the delay, by a chain of events with average delay $\Delta$, while increasing the overall variance of the production time. In Section 4.1, we show that a close approximation of the mean and the variance of a delayed CTMC by cascade semantics causes a blow-up of the state space of the former by $O(\alpha(x)^2\tau_r)$, where $r$ is a reaction producing protein $x$.

## 4 Comparison of Different Semantics for GRCs

In this section, we will do a comparative analysis of the delayed CTMC model. First, we will derive the probability distribution expression for the three different semantics we have given in the previous section. We will show that delayed CTMCs are more succinct than cascade CTMCs. We will then give two examples which demonstrate the qualitative differences between the immediate, cascade and delayed semantics for the same GRC.

### 4.1 Probability Distributions for Delayed Reactions

Let $G = (X, \alpha, R)$ be a GRC, and let $x \in X$ be such that $\alpha(x) = k$, for some $k > 0$. We would like to analyze the probability distribution of the time of producing $x$ due to a reaction $r \in R$ with $o_r(x) > 0$ in the three different semantics we have given in the previous section. We use $Pr(t_p(x) \leq T)$ to denote the probability of producing $x$ at most $T$ time units after the reaction $r$ took place.

For immediate semantics, the cumulative distribution is simply a step function, switching from 0 to 1 at time 0 since the initiation and completion times of a reaction in immediate semantics are equal. In other words, we have $Pr(t_p(x) \leq T) = 1$, for all $T \geq 0$.

For cascade semantics, we have $k$ intermediate reactions, each with an identical exponential distribution. This means that the probability density function for producing $x$ at exactly $t$ units of time is given by the $(k-1)$ convolution of the individual probability density functions. This is known as the Erlang distribution, and has the closed form $f_{k,\Delta}(t) = \frac{t^{k-1}}{(k-1)!\Delta^k} e^{-t/\Delta}$. The mean and the variance of this distribution are given as $k\Delta$ and $k\Delta^2$, respectively. This implies that as $k$ increases, both the mean and the variance of the distribution increase, a fact which we show has an important consequence in terms of model size.

For delayed semantics, we know that for $x$ at time $n\Delta$ to have, for the first time, age 0 which makes it immediate (and produced), the reaction $r$ producing $x$ must have occurred during the (half-open) interval $((n-k)\Delta, (n-k+1)\Delta]$. Since this means that the production time of $x$ cannot be greater than $k\Delta$ and cannot be less than $(k-1)\Delta$, the probability density function of the production time of $x$ due to $r$ is non-zero only in the interval $[(k-1)\Delta, k\Delta)$. Let us denote this interval with $p^+(x, r)$. Let $\delta$ range over the real numbers in the interval $p^+(x, r)$. Then, the probability of $x$ being produced by $r$ in $(k-1)\Delta + \delta$ units of time is equal to the probability of $r$ taking place at $\Delta - \delta$ units of time given that $r$ takes place in the interval $(0, \Delta]$. As the calculation of the transition rate matrix $\Lambda$ has shown, the probability of reaction $r$ firing depends on configurations; the base rate $\tau_r$ defines a lower bound on the actual rate of $r$. Since the lower the actual rate the higher the variation is, we are going to compute the distribution for the base rate. Then, the probability expression for $p^+(x, r)$ becomes

$$Pr(t_p(x) \leq (k-1)\Delta + \delta) = 1 - \frac{1 - e^{-\tau_r(\Delta - \delta)}}{1 - e^{-\tau_r \Delta}}, \quad \delta \in [0, \Delta]$$

This expression shows that with increasing values of reaction rate $\tau_r$, the probability of the production of $x$ taking time close to $\alpha(x)$ also increases. This is expected since as the rate of the reaction $r$ producing $x$ gets higher, the probability of $r$ taking place close to the beginning of the interval in which it is known to happen also gets higher. In other words, it is possible to generate a probability distribution for the production time of $x$ such that

$$Pr(\alpha(x) - \delta \leq t_p(x) \leq \alpha(x)) = 1 - \varepsilon$$

for arbitrary $\delta$ and $\varepsilon$, which we consider further in the rest of this subsection.

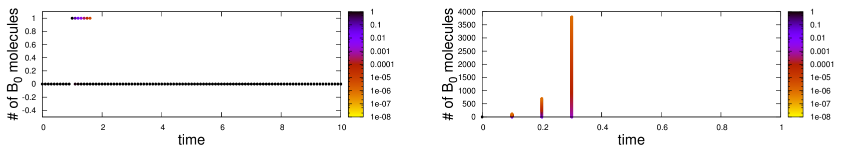
**Quasi-periodicity of delayed CTMCs.** Previously, we have given three alternative semantics for GRCs. In giving cascade semantics, the intuition was to replace each *deterministic* aging step of the delayed CTMC with an intermediate fictitious aging reaction. For each intermediate reaction, we have chosen the rates to be the inverse of $\Delta$ so that the collective cascading behavior has a mean delay equal to the age of the produced element. As we shall see in the next section, this conversion leads to different qualitative behaviors.

We will now compare delayed CTMC to what we call cascade CTMCs, a generalization of the cascade semantics, and show that preserving a property called quasi-periodicity requires a blow-up in the state space while converting a delayed CTMC into a cascade CTMC.

Let $\mathbf{b}$ be a continuous behavior of a delayed CTMC $D$. Recall that $\mathbf{b}$ is a mapping from $\mathbb{R}$, representing time, to a probability distribution over some $\mathcal{C}^{[X,\alpha]}$. We will call $\mathbf{b}$ *quasi-periodic* with $(\varepsilon, \delta, p, l)$ at configuration $c$ if for all $t \leq l$, $\mathbf{b}(t)(c) \geq 1 - \varepsilon$ implies that there exists a number $k \in \mathbb{N}$ such that $t \in [kp, kp + \delta]$, and if $t \notin [kp, kp + \delta]$ for any $k < l$, then $\mathbf{b}(t)(c) \leq \varepsilon$. Intuitively, if the behavior $\mathbf{b}$ is quasi-periodic with $(\varepsilon, \delta, p, l)$ at $c$, then the probability of visiting $c$ is almost 1 ($\varepsilon$ is the error margin) only at multiples of the period $p$ within $\delta$ time units. In all other times, the probability of being in $c$ is almost 0 (less than $\varepsilon$). The parameter $l$ gives the period of valid behavior; nothing is required of $\mathbf{b}$ for times exceeding $l$. Typically, we will want to maximize $l$, the length of the behavior displaying the desired behavior, and minimize $\varepsilon$ and $\delta$, the uncertainty in periodic behavior. Because periodicity is crucial during biological processes such as embryo development [16] or circadian clocks [5], quasi-periodicity defines an important subclass of behaviors.

For a set $S$, totally ordered by $\prec$, and elements $s, s' \in S$, let $s' = s + 1$ hold only when $s'$ is the least element greater than $s$ in $S$ according to $\prec$. Let $s_{min}$ denote the minimal element in the totally ordered set $S$. If $S$ is finite, then for the maximal element $s_{max}$ of $S$, we let $s_{min} = s_{max} + 1$.

A CTMC $M = (S, \Lambda)$ is called a *cascade CTMC* if $S$ is totally ordered, and $\Lambda(s, s') > 0$ iff $s' = s + 1$. A cascade CTMC is $\lambda$-*homogenenous* if $\Lambda(s, s') > 0$ iff


(a) Delayed Semantics (b) Immediate Semantics

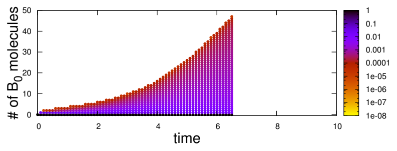

(c) Cascade Semantics

**Fig. 2.** Behaviors of the GRC `ConvDiv`. Due to intractability of the immediate and cascade models, (b) depicts the probability distribution up to time $0.3s$ and (c) represents the probability distribution up to time $6s$.

$\Lambda(s, s') = \lambda$ or $s = s_{min}$. In other words, a cascade CTMC is $\lambda$-homogeneous if all the state transitions have the same rate $\lambda$ with the possible exception of the transition out of the minimal state.

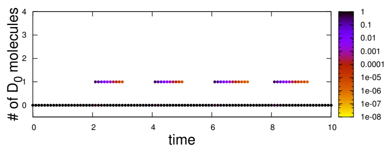
**Theorem 1.** *There exists a class of delayed CTMCs $M_i$ that are quasi-periodic such that there is no corresponding class of $\lambda$-homogeneous cascade CTMCs $M'_i$ with $|M'_i| = O(|M_i|)$.*

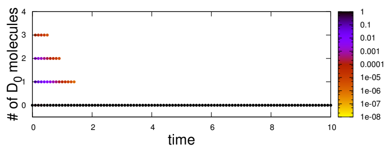
## 4.2 Examples Demonstrating Qualitatively Different Behavior

The first example GRC, `ConvDiv`, is given as

$$\begin{array}{l} A \xrightarrow{l} B + C \quad B + C \xrightarrow{vh} 2B + C \quad B \xrightarrow{l} \emptyset \\ \alpha: \quad A \mapsto 0,\ B \mapsto 1,\ C \mapsto 2 \end{array}$$

The symbols $l$, $h$, $vh$, represent low, high and very high rates, respectively. At $t = 0$ the model contains a single molecule, of type $A$. After the first reaction produces one $B$ and one $C$, observe that the number of $B$ molecules will increase only if the second reaction fires, which requires for both $B$ and $C$ to be present in the system. In immediate semantics, the expected behavior is divergence: the number of $B$ molecules should increase without bound. However, when the delay values are taken into account, we see that $B$ and $C$ molecules with different ages are unlikely to be present in the system simultaneously. Thus, a stable behavior should be observed for delayed semantics. Since cascade semantics still allow for non-zero probability of producing $B$ and $C$, albeit at a lower probability, divergence should also be observed for cascade semantics. The computed behaviors given in Figure 2 are in accordance with our explanations.


(a) Delayed Semantics


(b) Immediate Semantics

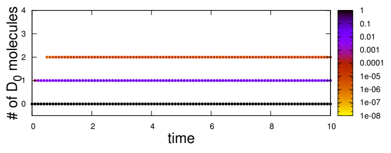

(c) Cascade Semantics

**Fig. 3.** The behaviors of the GRC `Periodic`.

The second example GRC, `Periodic`, is given as

$$\begin{array}{l} A \xrightarrow{h} B + A \quad B \xrightarrow{vh} \emptyset \quad 3B \xrightarrow{vh} C \quad C \xrightarrow{vh} \emptyset \\ \alpha: \quad A \mapsto 0,\, B \mapsto 1,\, C \mapsto 0 \end{array}$$

We observe that the production rate of $B$ from the first reaction is slower than the degradation rate of $B$, which means that in immediate semantics, it is very unlikely to have 3 $B$'s at any time, which in turn implies that $C$ is not likely to be produced at all. However, in delayed semantics, $A$ will keep producing $B$ during $\Delta$ time units, which are likely to be more than 3 in the beginning of the next step. This increases the probability of producing $C$ considerably. In fact, $C$ must be exhibiting quasi-periodic behavior. The computed behaviors are given in Figure 3. As expected from the arguments of the previous section, cascade semantics, even though does not stabilize at $C = 0$ like the immediate semantics, still can not exhibit a discernible separation between the times where $C = 0$ and the times where $C > 0$.

**Remark.** The examples of this section illustrate the impact of incorporating delay into models. As for representing a given biological system in a GRC, some of the encoding issues pertain to a natural extension of the syntax. For instance, having different production delays for the same molecular species in different reactions or allowing more than one reaction with the same difference vector are simple extensions to our formalism. Another issue is that the (delayed) molecular species are produced at exact multiples of $\Delta$; this can be modified by using additional reactions. For instance, a reaction $A \xrightarrow{k} B$ with $\alpha(B) = 1$ can be replaced with two reactions $A \xrightarrow{k} ZB$ and $ZB \xrightarrow{k_z} B$ with $\alpha(ZB) = 1$, $\alpha(B) = 0$, and where $ZB$ is a fictitious molecular species. Then, adjusting the value of $k_z$ will define a distribution for the production of the molecule $B$.

```
function ComputeBehavior
input
  $D = (X, \alpha, \Lambda, \Delta)$ : delayed CTMC
  $\rho_0$ : initial probability distribution
  $n$ : time count
  $W$ : number of workers
  $C^1, \ldots, C^W \subseteq \mathcal{C}^{[X,\alpha]}$
assume
  $C^1 \uplus \cdots \uplus C^W = \mathcal{C}^{[X,\alpha]}$
  $\forall c, c' \in \mathcal{C}^{[X,\alpha]} \exists i. \ (c_w = c'_w \Rightarrow \{c, c'\} \subseteq C^i)$
output
  $\rho_1, \ldots, \rho_n$ : probability distributions
begin
  for each $i \in 1..W$ do
    $\rho_0^i := \rho_0|_{C^i}$
  done
  LaunchAndWait(Worker, $W$)
  for each $k \in 1..n$ do
    $\rho_k := \rho_k^1 \cup \cdots \cup \rho_k^W$
  done
end
```

```
function Worker
input (local)
  $i$ : worker index
locals
  $k, j$ : $\mathbb{N}$
  $\rho$ : probability distribution
begin
  $k := 0$
  while $k < n$ do
    $\rho :=$ RunCTMC($\rho_k^i, M_D, \Delta$)
    for each $c \in Dom(\rho)$ do
      $j$ such that $c^{+1} \in C^j$
      atomic{
        $\rho_{k+1}^j(c^{+1}) := \rho_{k+1}^j(c^{+1}) + \rho(c)$
      }
    done
    SyncWorkers()
    $k := k + 1$
  done
  TerminationSignal()
end
```

**Fig. 4.** A parallel algorithm for computing the behavior of a delayed CTMC. The call of LaunchAndWait starts $W$ processes each running the function Worker and waits for the execution of TerminationSignal call in all worker processes. SyncWorkers synchronizes all worker processes. We assume that all variables are global (including the input parameters of ComputeBehavior) except for the explicitly declared locals.

## 5 Behavior Computation of Delayed CTMC

In this section we present an algorithm for computing the behavior of a delayed CTMC given an initial state of the model. Since this behavior cannot be computed analytically, we propagate the probability distribution over time using Equation (1).

The configuration space of the delayed CTMC can be divided into subspaces such that, between two consecutive tick instants, probability is not propagated from one subspace to another. Let $c$ and $c'$ be two configurations with different values of their waiting variables, i.e. $c_w \neq c'_w$. Due to the $\alpha$-safe property of the delayed CTMC, there can be no propagation of probability from $c$ to $c'$ between two consecutive tick instants. Therefore, the behaviors corresponding to each subspace can be computed independently for this time period, which has length $\Delta$. For this computation we can use any pure CTMC behavior computation algorithm. Furthermore, these independent computations can be executed in parallel. In the case of pure CTMC, a similar parallelization is not possible because the behavior dependencies flow through the full configuration space.

In Figure 4 we illustrate our parallel algorithm ComputeBehavior that computes the behavior of a delayed CTMC $D = (X, \alpha, \Lambda, \Delta)$ starting from initial

```
function SPACEID($s$)
begin
  $i := 1, idx := 0$
  for each $(x,a) \in Dom(c_w)$ do    // $(x,a)$ with larger $a$ is chosen first
    $idx := idx + 4^i s_w((x,a))$
    $i := i + 1$
  done
  return $(i\%W) + 1$
end
```

**Fig. 5.** In our implementation, SPACEID is used to divide the state space in partitions for the worker processes.

distribution $\rho_0$. The algorithm computes the behavior of $D$ until $n$ time steps, i.e., $\rho_1, \dots, \rho_n$. COMPUTEBEHAVIOR uses $W$ number of worker processes to compute the probability distributions. Each of the $W$ works is assigned a subspace of $\mathcal{C}^{[X,\alpha]}$, as decided by the input partitions $C^1, \dots, C^W$. These partitions must ensure that if two configurations have equal values of waiting variables then both configurations are assigned to the same worker.

COMPUTEBEHAVIOR divides $\rho_0$ into the sub-distributions $\rho_0^1, \dots, \rho_0^W$ according to the input partitions. Then, it launches $W$ number of workers who operate using these initial sub-distributions. Workers operate in synchronized rounds from 0 to $n-1$. At the $k$-th round, they compute the probability sub-distributions of the $k+1$-th time step $\rho_{k+1}^1, \dots \rho_{k+1}^W$. The $i$-th worker first runs a standard CTMC behavior computation algorithm RUNCTMC on $\rho_k^i$ that propagates the probability distribution until $\Delta$ time and the final result of the propagation is stored in $\rho$. Then, the inner loop of the worker applies Tick operation on $\rho$. For each configuration $c$ in $Dom(\rho)$, Tick adds $\rho(c)$ to the probability of $c^{+1}$ in the appropriate sub-distribution decided by the configuration space partitions. Note that a configuration $c$ may be the successor of many configurations, i.e., there may exist two configurations $c_1$ and $c_2$ such that $c_1^{+1} = c_2^{+1} = c$. and multiple workers may access $\rho_{k+1}^j(c)$, where $j$ is such that $c \in C^j$. Therefore, we require this update operation to be atomic. After the Tick operation, workers move to next round synchronously. After all workers terminate their jobs, COMPUTEBEHAVIOR aggregates the sub-distributions into full distributions for each time step and produces the final result.

## 6 Implementation and Results

**Implementation.** We extended the SABRE-toolkit [4], a tool for probability propagation of pure CTMCs, to solve delayed CTMCs. We implemented COMPUTEBEHAVIOR as a multi-process system with an inter-process communication implemented using MPI [6]. We use an implementation of the fast adaptive uniformization method [13] for RUNCTMC. Furthermore, we use function SPACEID, shown in the Figure 5, to define the partitions on the space:

| **Semantics** | **Example** | **Figure** | **Time Horizon** | **Run Time** | **Avg. Space** | **Qualitative Behavior** |
|---|---|---|---|---|---|---|
| Delayed CTMC | `ConvDiv` | 2(a) | $10s$ | $23s$ | 695 | converge |
| | `Periodic` | 3(a) | $10s$ | $< 1s$ | 13 | periodic |
| | `Feedback` | 1(a) | $7.2s$ | $23m$ | $9 \times 10^5$ | overshoot |
| | `Feedback` | 1(a) | $100s$ | 7.65h$\times$200* | $3.2 \times 10^7$ | overshoot |
| Cascade CTMC | `ConvDiv` | 2(c) | $6s$ | $311m$ | 119344 | diverge |
| | `Periodic` | 3(c) | $10s$ | $3s$ | 351 | uniform |
| | `Feedback` | 1(c) | $7.2s$ | $21h$ | $5.00 \times 10^6$ | *intractable* |
| Immediate CTMC | `ConvDiv` | 2(b) | $0.3s$ | $40m$ | 4007 | diverge |
| | `Periodic` | 3(b) | $10s$ | $1s$ | 26 | decay |
| | `Feedback` | 1(b) | $100s$ | $9s$ | 96 | fast stable |

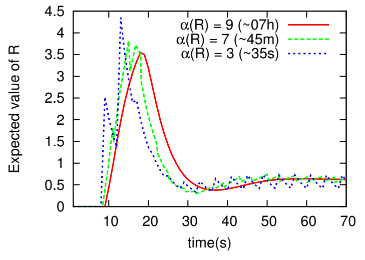
**Fig. 6.** Performance results for computing the behaviors corresponding to the examples presented earlier in this paper. We computed the behaviors until the time horizons within the run times. The avg. space column shows the configuration space with significant probabilities during the computation. In the `Feedback` example we use the following reaction rates: production = 1, binding = 1, unbinding=0.1, degradation=0.2. We assume that R remains latent 9 seconds after its production. We use multiple workers only for the `Feedback` example (*run time × number of workers). The last column provides an intuitive description of the observed qualitative behavior.

$C^i = \{c \in \mathcal{C}^{[X,\alpha]} | \textsc{spaceId}(c) = i\}$. In our examples, this policy leads to fairly balanced partitions of configurations among processes.

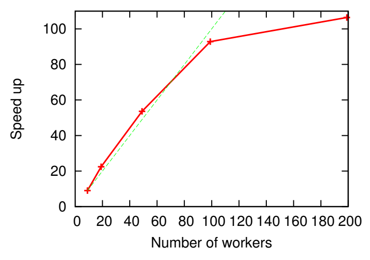
**Experiments.** In Figure 6 we present performance results for the behavior computation of the three discussed examples under the three semantics that we have introduced. We observe that delayed CTMCs offer an efficient modeling framework of interesting behaviors such as overshooting, convergence and periodicity.

We applied our implementation of ComputeBehavior on `Feedback` with delayed CTMC semantics using 200 workers. We were able to compute the behaviour until 100 seconds in 7.65 hours. ComputeBehavior with a single worker was able to compute the behaviour of `Feedback` until 7.2 seconds in 23 minutes, and for the same time horizon the behavior computation of `Feedback` under cascade semantics using sequential RunCTMC took 21 hours to complete. Since the cascade CTMCs cannot be similarly parallelized, they suffer from a state space blowup and the negative impact on the performance cannot be avoided. In the case of immediate CTMC semantics, even if computing the behavior is relatively faster (except for `ConvDiv`, which has diverging behavior under the immediate CTMC semantics) the observed behavior does not correspond to the expectations of the model.

We also ran ComputeBehavior on `Feedback` for different values of $\varDelta$. Since the reaction rates in the real time remain the same, $\alpha(R)$ changes with varying $\varDelta$. In Figure 7(a), we plot expected values of $R$ at different times for three values of $\alpha(R)$. We observe that with increasing precision, i.e. smaller $\varDelta$ and higher $\alpha(R)$, the computed behaviors are converging to a limit behavior, but the running



(a)



(b)

**Fig. 7.** (a) Expected numbers of molecules of protein $R$ with varying $\alpha(R)$ in `Feedback`. We show the run time for computing each behavior in the legends. (b) Speed up vs. number of workers (negative feedback with $\alpha(R) = 5$).

time of the computation increases rapidly, which is due to the the increase in number of aged variables causing an exponential blowup in configuration space. In Figure 7(b), we show the speed of computing the behaviour of `Feedback` with $\alpha(R) = 5$ for different number of workers. We observe that up to 100 workers the performance improves linearly, and there is no significant speedup after 100 workers. This is because each worker, when there are many of them, may not have significant computations per round, and communication and synchronization costs become the dominating factor.

## 7 Conclusion

Much like the introduction of time by time automata into a frame which was capable of representing ordering patterns without the ability to quantify these orderings more directly and accurately, we extended the widely used CTMC formalism by augmenting it with a time component in order to capture the behavior of biological systems containing reactions of relatively different durations, e.g. DNA transcription versus molecule bindings. We argue that our formalism achieves a more natural way to model such systems than CTMC (possibly extended with auxiliary reactions). We show that our approach also provides an efficient way of parallelizing the behaviour computation of the model.

As a continuation of this work, we are currently developing synthetic biology experiments meant to validate our predictions for the behavior of the `Periodic` example introduced in this paper.

## References

1. D. Bosnacki, S. Edelkamp, D. Sulewski, and A. Wijs. Parallel probabilistic model checking on general purpose graphics processors. *STTT*, 13(1):21–35, 2011.
2. F. Ciocchetta and J. Hillston. Bio-pepa: A framework for the modelling and analysis of biological systems. *Theor. Comput. Sci.*, 410(33-34):3065–3084, 2009.

3. F. Didier, T. A. Henzinger, M. Mateescu, and V. Wolf. Approximation of event probabilities in noisy cellular processes. In *Proc. of CMSB*, volume 5688 of *LNBI*, page 173, 2009.
4. F. Didier, T. A. Henzinger, M. Mateescu, and V. Wolf. Sabre: A tool for stochastic analysis of biochemical reaction networks. In *QEST*, pages 193–194, 2010.
5. H. A. Duong, M. S. Robles, D. Knutti, and C. J. Weitz. A Molecular Mechanism for Circadian Clock Negative Feedback. *Science*, 332(6036):1436–1439, June 2011.
6. E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.
7. D. T. Gillespie. A general method for numerically simulating the time evolution of coupled chemical reactions. *J. Comput. Phys.*, 22:403–434, 1976.
8. E. M. Hahn, H. Hermanns, B. Wachter, and L. Zhang. Infamy: An infinite-state markov model checker. In A. Bouajjani and O. Maler, editors, *CAV*, volume 5643 of *Lecture Notes in Computer Science*, pages 641–647. Springer, 2009.
9. R. I. Joh and J. S. Weitz. To lyse or not to lyse: Transient-mediated stochastic fate determination in cells infected by bacteriophages. *PLoS Comput Biol*, 7(3):e1002006, 03 2011.
10. M. Z. Kwiatkowska, G. Norman, and D. Parker. Prism 2.0: A tool for probabilistic model checking. In *QEST*, pages 322–323, 2004.
11. R. Maithreye, R. R. Sarkar, V. K. Parnaik, and S. Sinha. Delay-induced transient increase and heterogeneity in gene expression in negatively auto-regulated gene circuits. *PLoS ONE*, 3(8):e2972, 08 2008.
12. M. Mateescu. *Propagation Models for Biochemical Reaction Networks*. Phd thesis, EPFL, Switzerland, 2011.
13. M. Mateescu, V. Wolf, F. Didier, and T. A. Henzinger. Fast adaptive uniformisation of the chemical master equation. *IET SYSTEMS BIOLOGY*, 4(6):441–452, NOV 2010. 3rd q-bio Conference on Cellular Information Processing, St John Coll, Santa Fe, NM, AUG 05-09, 2009.
14. B. Munsky and M. Khammash. The finite state projection algorithm for the solution of the chemical master equation. *J. Chem. Phys.*, 124:044144, 2006.
15. C. Myers. *Engineering genetic circuits*. Chapman and Hall/CRC mathematical & computational biology series. CRC Press, 2009.
16. A. Oswald and A. Oates. Control of endogenous gene expression timing by introns. *Genome Biology*, (3), 2011.
17. A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In *Pacific Symposium on Biocomputing*, pages 459–470, 2001.
18. A. S. and Ribeiro. Stochastic and delayed stochastic models of gene expression and regulation. *Mathematical Biosciences*, 223(1):1 – 11, 2010.
19. M. A. Savageau. Comparison of classical and autogenous systems of regulation in inducible operons. *Nature*, 1974.
20. J. Zhang, M. Sosonkina, L. T. Watson, and Y. Cao. Parallel solution of the chemical master equation. In *SpringSim*, 2009.