Rare-Event Guided Analysis of Infinite-State Chemical Reaction Networks

Mohammad Ahmadi[1], Lukas Buecherl[2], Chris J. Myers[2], Zhen Zhang[3], Chris Winstead[3], and Hao Zheng[1](✉)

[1] University of South Florida, Tampa, FL, USA
haozheng@usf.edu
[2] University of Colorado Boulder, Boulder, CO
[3] Utah State University, Logan, UT

Abstract. Probabilistic Model Checking (PMC) is a valuable tool for automated analysis of systems exhibiting stochastic behavior. However, the effectiveness of PMC algorithms is limited to systems that can be modeled by a finite state-space. Chemical Reaction Networks (CRNs) are commonly used to describe biochemical systems. Since there are usually no upper-bounds on the population of species in a CRN, they can only be modeled as an infinite-state stochastic model. This paper proposes a new approach that can analyze infinite-state CRNs by bounding their state-space. For a property indicating that the probability of the event of interest is less than a certain threshold value, the objective is to generate a bounded range on the population of each species in the CRN such that this bounded CRN already retains sufficient probability to refute the property under investigation. The effectiveness of this approach is demonstrated by analyzing rare-event properties on a number of biochemical systems.

Keywords: Probabilistic Model Checking · Chemical Reaction Networks · Infinite State Systems · Rare Events.

1 Introduction

Stochastic behavior of systems can be modeled using probabilistic formalisms such as Markov chains. Once a system is modeled as a Markov chain, it's behavior can be quantitatively analyzed using *probabilistic model checking* (PMC). PMClis a formal verification technique for verifying quantitative properties of systems that exhibit stochastic behavior[17]. Probabilistic model checkers verify stochastic models against formally specified properties by computing the probability of those properties using numerical methods. For instance, these properties can specify the states in which the given system is failing, enabling PMC to calculate the probability of such failures occurring.

Chemical Reaction Networks (CRNs) are commonly used to describe biochemical systems. A CRN describes the evolution of a biochemical system consisting of a set of chemical species based on a set of chemical reaction rules [3].

Stochastic temporal behavior of a a CRN can be modeled as a Continuous Time Markov Chain (CTMC). An event of the form $\phi : X \xrightarrow{t \leq T} \theta$ is characterized by the population of species X reaching θ molecules within T time units of the initial condition. The probability of the event ϕ on the CTMC model of a CRN is traditionally estimated by statistical approaches like Stochastic Simulation Algorithm (SSA) [10]. However, statistical approaches like SSA are inefficient for estimating the probability of rare-events as the number of simulations required for computing an accurate estimate can grow very large [1]. Since PMC algorithms calculate the probability of properties using numerical methods, the computational cost of analyzing a property is mostly related to the size and structure of the state-space rather than the magnitude of the probability, therefore motivating analysis of rare-events using PMC.

Manually setting upper-bounds on the population of chemical species without in-depth insight into the dynamics of a CRN can lead to mistakenly removing critical behavior from the model. Therefore, there are usually no upper-bounds on the population of species in a CRN and the CTMC models of CRNs have infinite state-spaces. Prominent PMC algorithms require an explicit or symbolic representation of the model's state-space to be stored in the computer memory. As a result, these algorithms are incapable of directly analyzing most CRNs.

This paper presents an approach to analyze infinite-state CRNs' stochastic behavior that applies the concept of *bounded model checking* (BMC) [4]. Fig. 1 shows the overview of the proposed framework. The inputs of the framework are an unbounded CRN $\mathcal{C}$ and a property $P_{\leq p}[\phi : X \xrightarrow{t \leq T} \theta]$. The property $P_{\leq p}[\phi : X \xrightarrow{t \leq T} \theta]$ specifies that the probability of the event ϕ is less than or equal to the threshold value p. Starting with $K = 1$, the framework first generates a set of constraints restricting the range of the values that species' population can take in the CRN's state-space. These constraints are derived such that any state refuting them is guaranteed not to be present on any *witness trace* of length up to K. A witness trace for an event $X \xrightarrow{t \leq T} \theta$ is a finite, time-abstracted, alternating sequence of states and transitions that starts in the initial state of the model and ends in a state where the population of the species X equals to θ molecules. The length of a witness trace is defined as the number of transitions in that trace. After the constraints for bound K are derived, the probabilistic program of the bounded CRN conforming to these constraints is constructed. This probabilistic program represents a finite state-space, and probabilistic model checkers like STORM can be used to calculate the probability of events on this program. If the probability of event ϕ on the bounded CRN is greater than the specified threshold p, the bounded CRN is returned as a counterexample to the property. Otherwise, K is incremented iteratively until a counterexample is formed, or until the allocated computing resources are exhausted. The probability of the event reported at the last iteration before the exhaustion of computing resources is a lower-bound on the actual probability of the event.

This paper first presents a method to derive a bounded range on the population of chemical species such that all the states that could be present on

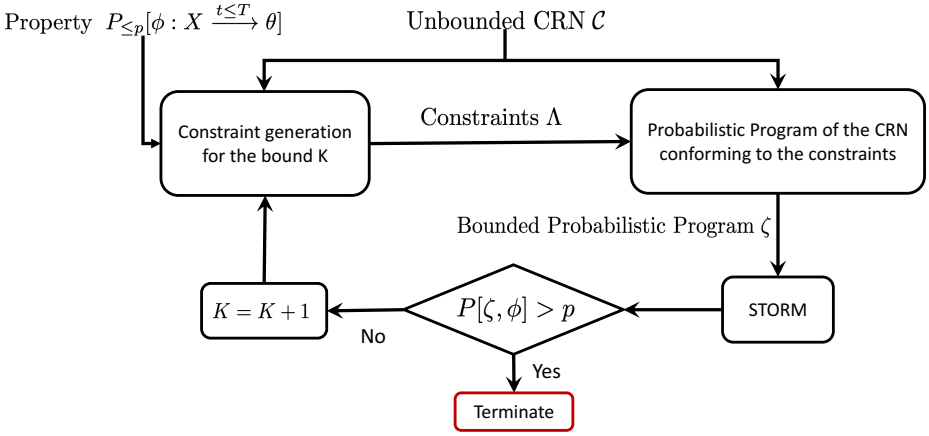

Fig. 1: The flow of the counterexample generation framework.

witness traces of length up to K are included within these ranges. Next, this paper presents a procedure to generate a probabilistic program of a CRN that conforms to the derived bounded ranges without undermining the semantics of the original CRN. The results of applying this approach to four biochemical systems are reported in the Experiments section.

1.1 Related Work

Several approximate methods have been proposed for model-checking infinite-state Markov models. These methods report a range of probabilities (an upper-bound and a lower-bound) for the probability of an event instead of a single value. INFAMY [12] is an infinite-state Markov chain model checker that explores the infinite state-space of the model up to a finite depth K. If the probability range for an event on this constructed finite state-space is not tight enough, K is incremented. STAMINA [15,20] works similarly by generating a finite state-state from the model's original infinite state-space, but the exploration of the state-space is based on the time-abstracted reachability probability rather than depth. The framework presented in this paper is similar to these approximate approaches in terms of iteratively expanding the state-space. The expansion of the state-space is however guided by the property under verification. This is essential for rare-event properties, as blindly exploring the state-space based on depth or reachability probability can result in exploring a large number of states before any state of interest is encountered. The proposed framework also constructs a probabilistic program instead of an explicit state-space, enabling the use of different model-checking engines.

Counterexample generation in probabilistic settings has been extensively studied. Han and Katoen [13] previously reduced the problem of finding the minimal witness set refuting a property in a Discrete Time Markov Chain (DTMC) to

computing (hop-constrained) K-Shortest Paths (KSP) in the state-space. They further extended their work with an approximate method to generate counterexamples for CTMC models [14]. However, since traditional KSP algorithms assume finite graphs as input, this approach can not be applied to most CRNs.

The set of witness traces required to form a counterexample can be very large. Therefore, many approaches have been developed that return a *critical subsystem* of the original model as a counterexample, instead of a witness set. A critical subsystem is a subset of a model's states and transitions with enough probability to refute the property under investigation. A subsystem of a model is bounded by the model's size and can provide a more compact representation of the witness set. Aljazzar and Leue [2] proposed a variation of Best First Search called eXtended Best First Search (XBF) over KSP where a subsystem of the original model is returned as a counterexample. XBF expands the state-space on the fly, so it can be applied to systems with infinite state-spaces. However, best first search traversals are not guaranteed to terminate on infinite state-spaces. Wimmer et al. [23] use SMT-solving and MILP (Mixed Integer Linear Programming) to find *minimal* critical subsystem in DTMCs and MDPs. Funke et al. [8] propose an approach to reduce the problem of computing the minimal critical subsystem in DTMCs and MDPs to finding a polytope of Farkas certificates with maximal number of zeros in a vertex. Both of these approaches depend on a finite set of *target states*, i.e. states where the condition of the event is satisfied, and as a result cannot be directly applied to infinite-state systems.

Dehnert et al. [6] propose an approach that returns a *sub-model* of the original PRISM model as the counterexample to the property, as opposed to a critical subsystem. A sub-model is defined to be a subset of the model's original commands. Returning a sub-model as the counterexample is advantageous since it is more comprehensible than a partial state-space, and insight into the dynamic of the erroneous behavior of the model can be obtained possibly without requiring post-processing of counterexamples. Their approach cannot be directly applied to infinite-state systems since a sub-set of the model's command could still represent an infinite state-space.

2 Background

2.1 Chemical Reaction Networks

Here, a formal description of chemical reaction networks is given that is later used for encoding the population constraints as a constraint solving problem.

Let $\mathbb{Z}$, $\mathbb{Z}^+$, and $\mathbb{N}_0$ be the set of integers, the set of positive integers, and the set of non-negative integers, respectively. A CRN consists of a finite set of N chemical species $\mathcal{S} = \{S_1, S_2, ..., S_N\}$ interacting through a finite set of M reaction channels $\mathcal{R} = \{R_1, R_2, ..., R_M\}$. The population of species S_i are represented by variables defined over $\mathbb{N}_0$. A *reaction* $(\rho, \pi, \lambda) \in \mathcal{R}$ defines a rule on the evolution of the system, where $\rho \subseteq \mathcal{S}$ is a set of species called *reactants*, $\pi \subseteq \mathcal{S}$ is a set of species called *products*, and λ, which is a positive real number,

is the coefficient associated with the rate of the reaction. The sets ρ and π can be empty. A reaction R is typically written as

$$c_{ir} \times S_i + c_{jr} \times S_j + \ldots \xrightarrow{\lambda} c_{ip} \times S_i + c_{jp} \times S_j + \ldots$$

where $c_{xr} \in \mathbb{Z}^+$ and $c_{yp} \in \mathbb{Z}^+$ are stoichiometric coefficients indicating the molecule count of reactants and products that are consumed and produced, respectively, when a reaction fires.

The semantics of a CRN $\mathcal{C}$ is generally given in terms of a discrete-state continuous-time stochastic process. A state of a CRN, $\boldsymbol{x}(t)$, is a vector representing the populations of species $\mathcal{S}$ at time $t \geq 0$. A state without time explicitly represented is simply written as $\boldsymbol{x}$. Firing of a reaction causes a state change by changing populations of some species. Let $\boldsymbol{x}[S_i]$ denote the population of species S_i in state $\boldsymbol{x}$ and $\delta(S_i, R_j)$ denote the net effect of reaction R_j's firing on the population of species S_i. A reaction $R = (\rho, \pi, \lambda)$ is *enabled* in state $\boldsymbol{x}$ if $\forall S_i \in \rho, \boldsymbol{x}[S_i] \geq c_{ir}$. The effect of firing a reaction R on the population of a species is defined by the function below.

$$\delta(S_i, R) = \begin{cases} -c_{ir} & \text{if } S_i \in \rho \setminus \pi \\ c_{ip} & \text{if } S_i \in \pi \setminus \rho \\ c_{ip} - c_{ir} & \text{if } S_i \in \pi \cap \rho \\ 0 & \text{if } S_i \notin \pi \cup \rho \end{cases}$$

Firing a reaction R in state $\boldsymbol{x}$ leads to a new state $\boldsymbol{x}'$ such that

$$\forall S_i \in \mathcal{S} : \ \boldsymbol{x}'[S_i] = \boldsymbol{x}[S_i] + \delta(S_i, R).$$

The initial state of a CRN is denoted as $\boldsymbol{x}_0$.

Stochastic temporal behavior of a CRN $\mathcal{C}$ can be modeled as a countably infinite continuous-time Markov chain (CTMC) $\mathcal{M} = (\boldsymbol{\Omega}, \boldsymbol{x}_0, \boldsymbol{R})$ where $\boldsymbol{x}_0$ is the initial state of $\mathcal{C}$, $\boldsymbol{\Omega}$ the set of reachable states from $\boldsymbol{x}_0$ via a valid sequence of reaction firings, and $\boldsymbol{R}$ the set of transitions. Each transition in $\boldsymbol{R}$ is given by $\boldsymbol{x} \xrightarrow{r} \boldsymbol{x}'$ if there is a reaction $R = (\rho, \pi, \lambda)$ and a state $\boldsymbol{x}$ such that firing R in state $\boldsymbol{x}$ leads to a new state $\boldsymbol{x}'$. The transition rate r is defined by a propensity function as follows.

$$r = \lambda \times \prod_{\forall S_i \in \rho} \binom{\boldsymbol{x}[S_i]}{c_{ir}} \tag{1}$$

Note that the set of transitions $\boldsymbol{R}$ can also be expressed as a transition rate matrix as in the traditional CTMC definition. As an example, the simple *single species production and degradation* CRN [16] is considered.

$$R_1 : \ S_1 \xrightarrow{\lambda_1} S_1 + S_2, \qquad R_2 : \ S_2 \xrightarrow{\lambda_2} \emptyset \tag{2}$$

where the populations of species S_1 and S_2 in the initial state are 1 and 40, respectively, and the reaction rate constants of reactions R_1 and R_2 are $\lambda_1 = 1.0$ and $\lambda_2 = 0.025$, respectively. For this model the δ function is defines as:

$$\delta(S_1, R_1) = 0 \quad \delta(S_2, R_1) = 1 \quad \delta(S_1, R_2) = 0 \quad \delta(S_2, R_2) = -1$$

2.2 Probabilistic Guarded Command Language

This paper considers a simplified version of the PRISM[18]'s guarded command language that only considers *flattened* CTMC models comprising only one module, without any synchronizations between the commands.

Definition 1 (Probabilistic Program, Command). *A* probabilistic program *is a tuple* $\zeta = (Var, v_{init}, \Gamma)$ *where* Var *is a set of integer variables,* v_{init} *is the initial variable valuation and* Γ *is a finite set of commands.*

Each command $\gamma \in \Gamma$ *is of the form* $\gamma = g \rightarrow r : u$ *where* g*, referred to as the* guard *of the command, is a logical expression over* Var *encoding the enabling conditions of the command,* r *is the rate by which* update u *takes place and the update* u *is a function that takes the current variable valuations in* Var *and returns a new valuation for each of those variables.*

Any CRN can be modeled by a probabilistic program by the following procedure. First, for each species S_i in the CRN, an integer variable $\boldsymbol{S_i}$ is added to the set of variables Var that represents the population of species S_i at any given state. The initial value for $\boldsymbol{S_i}$ is set to be the initial population of species S_i. Next, for each reaction R_i in the CRN, a command γ_i is added to the program's set of commands Γ. The guard of this added command, g_i, encodes the enabling conditions of the corresponding reaction, that is the value assigned to each variable $v_i \in Var$ in a state must be at least equal to the amount of S_i consumed in the firing of the reaction. The update of this added command, u_i is the effect that the corresponding reaction's firing would have on the variables in Var. As an example the single species production and degradation CRN can be modeled as a probabilistic program $\zeta = (Var, v_{init}, \Gamma)$ where $Var = \{int\ S_1, int\ S_2\}$ is the set of integer variables tracking the population of each species , $v_{init} = (S_1 = 1, S_2 = 40)$ is the initial population assigned to each species and $\Gamma = \{\gamma_1, \gamma_2\}$ is a set of commands modeling the reactions of the CRN as follows:

$$\gamma_1 = (S_1 > 0) \rightarrow (S_1 * S_2 * \lambda_1) : (S_1' = S_1\ \&\ S_2' = S_2 + 1)$$
$$\gamma_2 = (S_2 > 0) \rightarrow (S_2 * \lambda_2) : (S_1' = S_1\ \&\ S_2' = S_2 - 1)$$

with S_1' and S_2' being the new assignments to the variables S_1 and S_2 after the command fires.

3 Methods

3.1 Generating K-Bounded Constraints

Generating a bounded probabilistic program for a CRN by simply bounding the the species' populations to the range $[0, K]$ and increasing K iteratively results in a partial state space that likely includes states that do not appear on any witness traces. Such states do not contribute to the probability of the event, but instead cause the rapid growth of the state space, increasing the time

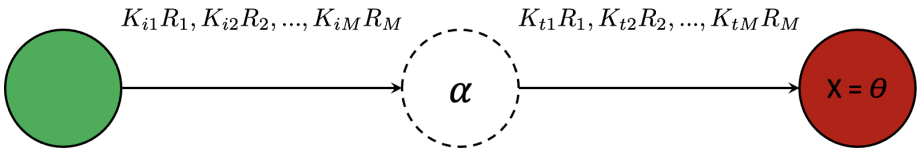

Fig. 2: General form of a witness trace starting in the initial state of the CRN (green) and ending in a state satisfying the (time-abstracted) condition of the event $X \xrightarrow{t \leq T} \theta$ (red) that passes through an arbitrary state α.

required for checking the probability of the event on the generated probabilistic program. Therefore, identifying and excluding these states makes the framework more efficient. The objective is to generate the tightest range of values that the population of each species can take without excluding any states that appear on a witness trace of length less than or equal to K. In this section, a method for generating such K-bounded constraints is described.

Suppose that the event $X \xrightarrow{t \leq T} \theta$ is considered for a CRN comprised of N chemical species $\mathcal{S} = \{S_1, S_2, ..., S_N\}$ interacting through M reactions $\mathcal{R} = \{R_1, ..., R_M\}$. Fig. 2 shows the format of a general witness trace starting in the initial state and ending in the target state that passes through an arbitrary state α. State α is reached by firing reaction R_1 $\boldsymbol{K_{i1}}$ times, reaction R_2 $\boldsymbol{K_{i2}}$ times, ... Note that all $\boldsymbol{K_{iv}}$ values could be zero, mapping α to the initial state of the model. In order to form a witness, from state α, reactions $R_1, R_2, \ldots$ are fired $\boldsymbol{K_{t1}}$, $\boldsymbol{K_{t2}}$, ... times respectively. Note that all $\boldsymbol{K_{tv}}$ values could also be zero, mapping α to the target state. These variables are defined as:

$\boldsymbol{K_{iv}}$: # of times reaction R_v is fired to reach state α from the initial state
$\boldsymbol{K_{tv}}$: # of times reaction R_v is fired to reach the target state from state α

In order to form a witness trace with length less than or equal to K, the number of reactions taken to reach the state α and subsequently the target state must be less than or equal to K. This condition is encoded in Formula 3.

$$\mathcal{E}_1 \ : \ \sum_{v=1}^{M} \boldsymbol{K_{iv}} + \boldsymbol{K_{tv}} \leq K \tag{3}$$

The condition that the number of reaction firings is always non-negative is encoded in Formula 4.

$$\mathcal{E}_2 \ : \ \forall v \in 1, ..., M \ : \ \boldsymbol{K_{iv}}, \boldsymbol{K_{tv}} \geq 0 \tag{4}$$

Also, for each species S_i, the total number of molecules of S_i that are consumed along the trace must be no more than the population of species S_i in the initial state plus the total number of the S_i molecules that are produced along the

trace. This condition is encoded in Formula 5.

$$\mathcal{E}_3 \ : \ \forall S_i \in \mathcal{S} \ : \ \sum_{v=1}^{M}[(\boldsymbol{K_{iv}}+\boldsymbol{K_{tv}})\times c_{vr}] \leq (\boldsymbol{x}_0[S_i]+\sum_{v=1}^{M}[(\boldsymbol{K_{iv}}+\boldsymbol{K_{tv}})\times c_{vp}]) \quad (5)$$

In order to form a witness, the population of species X must reach θ from its value in the initial state via up to K firings of some reactions. This restriction is encoded in Formula 6 by enforcing the net effects of the reactions' firings to take the model from the initial state to a target state.

$$\mathcal{E}_4 \ : \ \boldsymbol{x}_0[X] + \sum_{v=1}^{M}[(\boldsymbol{K_{iv}} + \boldsymbol{K_{tv}}) \times \delta(X, R_v)] = \theta \quad (6)$$

This encoding represents all possible firings of reactions that have the net effect of moving the initial state to a target state. The following formula encodes the relation between reaction firings and the population of a species S_j in any state $\boldsymbol{\alpha}$ present on a witness trace

$$\mathcal{E}_5 \ : \ \boldsymbol{\alpha_{S_j}} = \boldsymbol{x}_0[S_j] + \sum_{v=1}^{M} \boldsymbol{K_{iv}} \times \delta(S_j, R_v) \quad (7)$$

where $\boldsymbol{\alpha_{S_j}}$ is an auxiliary variable representing the population of species S_j after $\boldsymbol{K_{iv}}$ firings of each reaction R_v from the initial state.

The conjunction of the above encodings, $\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \wedge \mathcal{E}_4 \wedge \mathcal{E}_5$, includes all possible witnesses of lengths up to K. It also carries information on the populations of species present on any state on these witnesses. Therefore, these encodings can be solved by a constraint solver and queried to obtain the witness state invariants as indicated at the beginning of this section. The invariants are formulated as

$$\forall 1 \leq j \leq N : l_{S_j} \leq \boldsymbol{\alpha_{S_j}} \wedge \boldsymbol{\alpha_{S_j}} \leq u_{S_j} \quad (8)$$

where $l_{S_j}, u_{S_j} \in \mathbb{Z}$ are are lower and upper bounds of the populations of species S_j on any state on any witness traces.

In the following, a method to find the upper bound u_{S_j} is explained first. Initially, the upper-bound on the population of species S_j is set to be the population of S_j in the initial state, *i.e.* $u_{S_j} = \boldsymbol{x_0}[S_j]$. Then the following formula is solved by a constraint solver.

$$\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \wedge \mathcal{E}_4 \wedge \mathcal{E}_5 \wedge (\boldsymbol{\alpha_{S_j}} > u_{S_j}) \quad (9)$$

If Formula 9 is *sat*, it indicates that the formula allows a value larger than u_{S_j} assigned to $\boldsymbol{\alpha_{S_j}}$, therefore u_{S_j} is not a true upper bound of $\boldsymbol{\alpha_{S_j}}$. In this case, Formula 9 is updated with u_{S_j} changed to the value returned for $\boldsymbol{\alpha_{S_j}}$, and is solved again. This process of querying the solver and updating the upper-bound continues until the solver returns *unsat*. When the solver returns *unsat*, it indicates that Formula 9 does not allow any value larger than u_{S_j} assigned to $\boldsymbol{\alpha_{S_j}}$, which implies that u_{S_j} at the termination of the solver with *unsat* is an

upper bound for $\boldsymbol{\alpha_{S_j}}$. Finding the lower bound is similar by repetitively solving the following formula

$$\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \wedge \mathcal{E}_4 \wedge \mathcal{E}_5 \wedge (\boldsymbol{\alpha_{S_j}} < l_{S_j}) \tag{10}$$

until it becomes *unsat*. The initial lower-bound on the population of species S_j is set to be the population of S_j in the initial state, *i.e.* $l_{S_j} = \boldsymbol{x_0}[S_j]$. The above process is repeated for every species in the CRN to find its invariants, and the conjunction of these invariants is referred to as Λ in the format as specified in Formula 8.

The theorem below shows that the K-bounded constraints generated by solving the described encodings is correct in terms of preserving the witness traces for CRNs.

Theorem 1. *Given a CRN $\mathcal{C}$ and an event ϕ, for every witness trace w present in $\mathcal{C}$, there exists K such that w is included in the state space formed using constraints Λ, where Λ are the constraints found for bound K.*

Proof of Theorem 1 is given in the Appendix A.

It is important to highlight that the conjunction of formulas $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4$ and $\mathcal{E}_5$ could encode invalid traces. This is due to the fact that $\mathcal{E}_3$, enforcing that the total number of consumed molecules must be no more than the total number of available molecules, only considers the total effect of reaction firings, and does not enforce the enabling conditions of each individual reaction firing. This means that solving these encodings might result in an assignment to the variables that can only be realized by taking invalid reactions, i.e. taking reactions where there are not enough reactant molecules necessary for their firing. Nevertheless, this does not affect the correctness of the framework, as it is already shown that any witness trace in the original CRN model will eventually be included by constraints generated with some K. This implies that although the upper-bounds (lower-bounds) found for a given K are indeed true upper-bounds (lower-bounds) on the population of species among all witnesses of length no more than K, they might not be the optimal upper-bounds (lower-bounds).

3.2 Probabilistic Program Conforming to K-Bounded Constraints

Section 2.2 explains how a CRN containing N species and M reactions can be modeled as a probabilistic program with N variables and M commands. This section describes how a probabilistic program for a CRN can be modified to incorporate the K-bounded constraints while maintaining the semantics of the original CRN. Suppose the CRN $\mathcal{C}$ with the set of species $\mathcal{S}$ and the set of reactions $\mathcal{R}$ is modeled by the probabilistic program $\zeta = (Var, v_{init}, \Gamma)$ where $v_{S_i} \in Var$ denotes the variable tracking the population of species $S_i \in \mathcal{S}$. The K-bounded constraints Λ put upper-bound and lower-bound restrictions on the population of each species $S_i \in \mathcal{S}$. First, the set Var in ζ is modified to include the bounded range generated for the population of each species.

$$\forall v_{S_i} \in Var : int\ v_{S_i} \Rrightarrow int[l_{S_i}, u_{S_i}]\ v_{S_i} \tag{11}$$

where l_{S_i} and u_{S_i} are the lower-bound and upper-bound values generated for the species S_i respectively and $\Rrightarrow$ represents the transformation of a variable to its modified form.

The probabilistic program generated following this step might undermine the semantics of the original CRN. In any given state, if the update of an enabled reaction takes the system into a state where the constraints are invalidated, that reaction is going to be disregarded. Disregarding such reactions would increase the probability of the remaining outgoing reactions, and therefore disturbs the probability distribution of witness traces. Model checking a probabilistic program with modified probability distribution will result in an unreliable lower-bound. In order to maintain the stochastic semantics of the original CRN, the disregarded reactions must be identified and be directed to a *sink* state. The sink state is defined as a state where 1) The guard of all of the commands is disabled and 2) The condition of the event is not satisfied in this state.

Assume that for a command $\gamma_j = (g_j) \rightarrow (r_j : u_j)$, the variable valuation after the update u_j is given by $\boldsymbol{Var}_{u_j}$. In order to identify disregarded reactions and direct them to the sink state, for each command $\gamma_j = (g_j) \rightarrow (r_j : u_j)$ in the original CRN, a new command γ_j^{sink} is added to the program with the following structure:

$$\gamma_j^{sink} = (g_j \wedge (\boldsymbol{Var}_{u_j}) \not\models \Lambda) \rightarrow (r_j : sink) \tag{12}$$

All the variables in the probabilistic program generated following these two steps are bounded, resulting in a finite state space. The generated probabilistic program also maintains the stochastic semantics of the original CRN. Therefore, the probability of an event on this program can be calculated by a model checker and this probability is indeed a lower-bound for the probability of the event on the original CRN.

4 Experiments

This section reports the results obtained by applying the proposed framework to four CRNs. These CRNs were previously studied in the context of rare-event simulation [16,9,5,1]. Rare-events are known to be problematic for stochastic simulation as the number of simulations required to obtain an estimate with a reasonable accuracy can grow very large. The authors in [1] argued that the variance reduction techniques such as importance sampling that were previously proposed to improve the performance of rare-event simulations are not robust and require extensive parameter tuning. The reliance on precise parameter tuning makes these techniques far from being automated and motivates the analysis of rare-events with probabilistic model checking.

The framework is implemented in Python with Z3 [19] as the underlying constraint solver and STORM [7] as the back-end probabilistic model-checker. The framework is compared to the STAMINA-STORM model checker introduced in [15]. STAMINA-STORM also produces a finite representation of an infinite-state model and uses STORM as the back-end model checker. Unlike the proposed framework in this paper, STAMINA-STORM generates an explicit

finite state-space, as opposed to a probabilistic program. Starting from the initial state of the model, STAMINA-STORM explores the model's state-space and accepts all the states with time-abstracted reachability probability above a given threshold κ. It then calculates a probability range, a lower-bound *and* and an upper-bound, for the probability of the event on the constructed finite state-space. If the probability range is tighter than a specified value, the algorithm terminates. If not, κ is reduced and the state-space exploration is continued. STAMINA-STORM is selected for comparison as it also permits model checking infinite-state CRNs and uses STORM as the back-end model checker. Since the proposed framework in this paper only returns a lower-bound for the probability of an event, the results reported for STAMINA-STORM in this section only includes the lower-bound probability that was reported upon termination. The results reported for STAMINA-STORM are obtained by setting the initial value for κ and the reduction factor of κ to their default values, which are 1.0, 1.25 respectively. For each experiment, the specified probability window for STAMINA-STORM is initially set to its default value which is 10^{-3}. If the reported lower bound upon termination is smaller than the required threshold, the probability window is reduced by a factor of 10 and the experiment is run again.

For the experiments reported in this section, events of the form $X \xrightarrow{t \leq T} \theta$ are considered. In each experiment, a set of increasing probability thresholds are considered based on the probability estimate of the event. The performance of the framework is evaluated based on the time it takes to return a lower-bound probability greater than the given threshold, proving that the event has a probability at least greater than this threshold. The experiments were performed on a 3.6 GHz Intel machine with 32GBs of memory running Ubuntu 22.04. Any experiment that took more than 1800 seconds was terminated and is marked with *TO* in the results tables.

The experiments in this section analyze rare-event properties on 4 biochemical models:

- **Enzymatic Futile Cycle** is a CRN consisting of 6 species reacting through 6 reaction channels. The property of interest for this model is $P_{\leq p}[S_5 \xrightarrow{t \leq 100} 25]$. Kuwahara et al. [16] estimate the probability of the event $S_5 \xrightarrow{t \leq 100} 25$ to be 1.7×10^{-7}.
- **Motility Regulation** is a CRN of a gene regulatory network consisting of 9 species reacting through 12 reaction channels. The property of interest for this model is $P_{\leq p}[CodY \xrightarrow{t \leq 10} 20]$. In [1] the probability of the event $CodY \xrightarrow{t \leq 10} 20$ is estimated to be 2.16×10^{-7} by running 10^7 *weighted SSA* [16] simulations.
- **Yeast Polarization** is a CRN consisting of 7 species and 8 reactions. The property of interest for this model is $P_{\leq p}[G_{bg} \xrightarrow{t \leq 20} 50]$. Roh et al. [22] report the probability of the event to be $1.23 \times 10^{-6} \pm 0.05 \times 10^{-6}$ at two standard-errors using weighted SSA.
- **Genetic Circuit0x8E** [21] is a genetic circuit consisting of 18 species and 15 reactions. The property of interest for this model is $P_{\leq p}[S_8 \xrightarrow{t \leq 1000} 90]$.

The 99% confidence interval for the probability of the event in the property is estimated to be $8.2 \times 10^{-5} \pm 2.33 \times 10^{-5}$ by running 10^6 SSA simulations.

The full description of the CRNs can be found in the GitHub repository accessible at this **link**[4].

Discussion. Table 1 reports the results obtained by running the proposed framework and STAMINA-STORM on the selected models. In each experiment, the probability estimate reported by stochastic simulation is used for selecting a set of increasing probability thresholds. Assuming the probability estimate of the event ϕ is $\hat{p}$, a counterexample for the property $P_{\leq p}[\phi]$ can be formed for all $p < \hat{p}$.

For enzymatic futile cycle, the first counterexample returned by the framework has probability 1.54×10^{-7} which has the same order of magnitude as the probability estimate reported by stochastic simulation. Upon termination, STAMINA-STORM reports the lower-bound probability of 1.73×10^{-7}.

For motility regulation, both STAMINA-STORM and the proposed framework can generate counterexamples for probability thresholds up to 10^{-7}, which has the same order of magnitude as the estimate reported by stochastic simulation. The proposed framework generates probabilistic programs that represent smaller state-spaces and terminates faster, especially for smaller probability thresholds.

In the yeast polarization experiment, the proposed framework can generate a counterexample for threshold 10^{-6}, which has the same order of magnitude as the estimate reported by stochastic simulation. STAMINA-STORM does not terminate before the 1800 seconds timeout on this model. In fact, STAMINA-STORM does not terminate on this CRN even if the the timeout is extended to 24 hours. This CRN has a large disparity between the rate of the reactions. Models showing *stiffness*, i.e. those with a large disparity between the rates of transitions, are known to cause issues for probabilistic model checking algorithms by increasing the computational complexity of getting an accurate result. This signifies the importance of selecting a small subset of the original CRN's state-space that contains most of the witness traces. STAMINA-STORM only considers the reachability probability of states when expanding the state-space. This could result in selecting a large state-space, containing a large number of non-witness traces or longer witness traces. A large, stiff state-space will significantly increases the time required by the back-end model checker to calculate the probability of the event.

In the experiment on Circuit 0x8E CRN, STAMINA-STORM does not terminate before the 1800 seconds timeout. The proposed framework can form counterexamples for thresholds up to 10^{-7}, but fails to return counterexamples for thresholds greater than this value before timeout. The large growth in the size of the state-space represented by the probabilistic program will result in larger amount of time required by the model-checker to calculate the probability

[4] https://github.com/fluentverification/bmc_counterexample/blob/8fdbf4e710b57b13a587a2bb0843f137c9dc74ae/CRNs/Readme.md

of the event on the program. This highlights a limitation of this approach. If the probability of the event is not concentrated in a relatively small portion of the CRN's state space, forming counterexamples for larger probability thresholds require model checking probabilistic programs that represent larger state-spaces, which can be computationally expensive.

Constraint generation step of the framework finished quickly in all experiments, and the majority of the time was spent by the model checker to calculate the probability of the event on the generated probabilistic program. For circuit0x8E, where constraint generation was slowest among all experiments, the total amount of time spent for generating constraints was less than 2 seconds.

Although the estimates reported by stochastic simulation were used as baselines to evaluate the performance of the method, these estimates were obtained by running the simulations with highly tuned importance sampling [11] parameters. As argued in [1], importance sampling parameters could be highly sensitive and tuning them requires deep insight into the dynamics of the CRN. The proposed method in this paper is completely automated, eliminating the need for knowledge about such dynamics.

Computing an upper-bound on the probability of the event. An additional limitation of the current approach is that it only generates a lower-bound on the probability of events. An upper-bound could be formed by selecting a portions of the state space that have high reachability probability in general, but do not contain witnesses to the event. An upper-bound for an event of the form $F^{t\leq T}X = \theta$ (population of species X will eventually reaches θ within T time units) can be found by forming a lower-bound on the event $G^{t\leq T}X \neq \theta$ (population of species X will not reach the value θ in any state within T time units). The upper-bound to the probability of $F^{t\leq T}X = \theta$ can be then computed as $1 - Pr(G^{t\leq T}X \neq \theta)$. The encodings proposed in this work focus on finding traces that eventually reach the target state, and do not check for dynamic behaviors such as the population of a species remaining below/above a certain value *along* a witness trace. As a result, using these encodings to find an upper-bound on the probability of the event will result in iterative unfolding of the state space to depth K, including states that both satisfy and dissatisfy the property $G^{t\leq T}X \neq \theta$. Therefore, the lower-bound found for the event $G^{t\leq T}X \neq \theta$ is not optimal (close to 0), resulting in a weak upper-bound for the event $F^{t\leq T}X = \theta$ (close to 1). Improving the upper-bound probability will require encodings that account for the dynamic behavior of the model, and is left for future work.

Acknowledgements The authors are supported by the National Science Foundation under Grant Nos. 1900542, 1856733, and 1856740. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

Additional Information All Python scripts used in this work are accessible at `https://github.com/fluentverification/bmc_counterexample/tree/QEST2024`.

Table 1: The results for running the proposed method and STAMINA-STORM on four CRNs. For each CRN, a set of increasing probability thresholds are considered. For each threshold, the table reports the time, the size of the partial state-space, and the probability of the event on this partial state-space at the first iteration where the probability of the event is greater than the provided threshold. The time is measured in seconds. The size of the state-space is defined as the sum of the number of states and transitions. For the proposed method in this paper, the size of the state-space is reported by STORM after it builds the state-space from the generated probabilistic program. *TO* indicates timeout.

Enzymatic Futile Cycle						
	This Method			**STAMINA-STORM**		
Thresh.	**Prob.**	**|State-Space|**	**Time**	**Prob.**	**|State-Space|**	**Time**
1×10^{-10}	1.54×10^{-7}	400	0.70	1.73×10^{-7}	782	1.44
Motility Regulation						
	This Method			**STAMINA-STORM**		
Thresh.	**Prob.**	**|State-Space|**	**Time**	**Prob.**	**|State-Space|**	**Time**
1×10^{-10}	1.65×10^{-10}	242	0.27	2.41×10^{-7}	$13,492,874$	595.01
1×10^{-9}	5.00×10^{-9}	$57,269$	0.53	2.41×10^{-7}	$13,492,874$	595.01
1×10^{-8}	1.80×10^{-8}	$122,549$	0.74	2.41×10^{-7}	$13,492,874$	595.01
1×10^{-7}	1.05×10^{-7}	$1,354,996$	7.61	2.41×10^{-7}	$13,492,874$	595.01
Yeast Polarization						
	This Method			**STAMINA-STORM**		
Thresh.	**Prob.**	**|State-Space|**	**Time**	**Prob.**	**|State-Space|**	**Time**
1×10^{-15}	4.26×10^{-15}	$1,022,702$	2.80	–	–	*TO*
1×10^{-10}	1.66×10^{-6}	$2,243,533$	1723.56	–	–	*TO*
Genetic Circtuit 0x8E						
	This Method			**STAMINA-STORM**		
Thresh.	**Prob.**	**|State-Space|**	**Time**	**Prob.**	**|State-Space|**	**Time**
1×10^{-10}	2.31×10^{-10}	$4,778,902$	122.41	–	–	*TO*
1×10^{-9}	1.61×10^{-9}	$8,036,816$	252.23	–	–	*TO*
1×10^{-8}	1.61×10^{-8}	$15,658,892$	607.03	–	–	*TO*
1×10^{-7}	1.59×10^{-7}	$32,858,704$	1574.96	–	–	*TO*
1×10^{-6}	–	–	*TO*	–	–	*TO*
1×10^{-5}	–	–	*TO*	–	–	*TO*

5 Appendix A

Lemma 1 *Let Λ be the constraints found for $\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \wedge \mathcal{E}_4 \wedge \mathcal{E}_5$ where $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$, $\mathcal{E}_4$ and $\mathcal{E}_5$ are defined in (3), (4), (5),(6), (7), respectively. The following property holds.*

$$\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \wedge \mathcal{E}_4 \wedge \mathcal{E}_5 \models \Lambda$$

Proof Let $l_{S_j} \leq \boldsymbol{\alpha}_{S_j}$ and $\boldsymbol{\alpha}_{S_j} \leq u_{S_j}$ be the constraints of Λ on species S_j. The solver terminates with unsat when

$$\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \wedge \mathcal{E}_4 \wedge \mathcal{E}_5 \wedge (\boldsymbol{\alpha}_{\boldsymbol{S_j}} > u_{S_j}) \text{ is false.}$$

It is equivalent to that

$$\neg(\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \wedge \mathcal{E}_4) \wedge \mathcal{E}_5 \vee \neg(\boldsymbol{\alpha}_{\boldsymbol{S_j}} > u_{S_j}) \text{ holds.}$$

Again, it is equivalent to that

$$\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \wedge \mathcal{E}_4 \wedge \mathcal{E}_5 \implies \boldsymbol{\alpha}_{\boldsymbol{S_j}} \leq u_{S_j} \text{ holds.}$$

It can be proved similarly that

$$\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \wedge \mathcal{E}_4 \wedge \mathcal{E}_5 \implies l_{S_j} \leq \boldsymbol{\alpha}_{\boldsymbol{S_j}} \text{ holds.}$$

Since the above proofs hold for constraints on any species, we have proven that

$$\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \wedge \mathcal{E}_4 \wedge \mathcal{E}_5 \models \Lambda.$$

■

Theorem 1 *Given a CRN $\mathcal{C}$ and an event ϕ, for every witness trace w present in $\mathcal{C}$, there exists K such that w is included in the state space formed using constraints Λ, where Λ are the constraints found for bound K.*

Proof Let w be a witness trace with length K in $\mathcal{C}$. The sum of reaction firings on w is K, thus $\mathcal{E}_1$ as defined in (3) holds for w and K. The number of times each reaction is fired on w is larger than or equal to 0, thus $\mathcal{E}_2$ as defined in (4) also holds for w. Since w is a valid witness in $\mathcal{C}$, for every species, the total number of consumed molecules along w must be no more than the total number of available molecules, thus $\mathcal{E}_3$ as defined in (5) holds. Since w starts from the initial state $\boldsymbol{x}_0$ and ends in the target state via a set of reaction firings as included in w, $\mathcal{E}_4$ as defined in (6) holds as well. Let $\boldsymbol{\alpha}$ be a state on w. Then, for each reaction R_v, $\boldsymbol{K_{iv}}$ can be found by counting the number of firings of R_v on the witness segment $\boldsymbol{x_0}$ to $\boldsymbol{\alpha}$. Since $\boldsymbol{\alpha}_{\boldsymbol{S_j}}$ in $\mathcal{E}_4$ is a free variable, For $\mathcal{E}_4$ to hold, $\boldsymbol{\alpha}_{\boldsymbol{S_j}}$ is set to be equal to $\boldsymbol{\alpha}[S_j]$, the population of S_j in $\boldsymbol{\alpha}$.

As shown above, $\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3 \wedge \mathcal{E}_4 \wedge \mathcal{E}_5$ holds for w, and $\boldsymbol{\alpha}_{\boldsymbol{S_j}} = \boldsymbol{\alpha}[S_j]$ holds for any state $\boldsymbol{\alpha}$ on w. By Lemma 1, constraint Λ also holds. Since both $\boldsymbol{\alpha}_{\boldsymbol{S_j}} = \boldsymbol{\alpha}[S_j]$ and $l_{S_j} \leq \boldsymbol{\alpha}_{\boldsymbol{S_j}} \wedge \boldsymbol{\alpha}_{\boldsymbol{S_j}} \leq u_{S_j}$ hold, $l_{S_j} \leq \boldsymbol{\alpha}[\boldsymbol{S_j}] \wedge \boldsymbol{\alpha}[\boldsymbol{S_j}] \leq u_{S_j}$ holds as well. Since this is true for any species in any state $\boldsymbol{\alpha}$ on w, this shows that $\forall S_j : \boldsymbol{\alpha}[\boldsymbol{S_j}] \models \Lambda$. This implies that state $\boldsymbol{\alpha}$ is also accepted by the constraints Λ. Therefore, for every transition $\alpha_1 \to \alpha_2$ in w both α_1 and α_2 are accepted by the constraints Λ, resulting in w to be found in its entirety. ■

References

1. Ahmadi, M., Thomas, P.J., Buecherl, L., Winstead, C., Myers, C.J., Zheng, H.: A comparison of weighted stochastic simulation methods for the analysis of genetic circuits. ACS Synthetic Biology (2022)
2. Aljazzar, H., Leue, S.: Directed explicit state-space search in the generation of counterexamples for stochastic model checking. IEEE Transactions on Software Engineering **36**(1), 37–60 (2009)
3. Chellaboina, V., Bhat, S.P., Haddad, W.M., Bernstein, D.S.: Modeling and analysis of mass-action kinetics. IEEE Control Systems Magazine **29**(4), 60–78 (2009)
4. Clarke, E., Biere, A., Raimi, R., Zhu, Y.: Bounded model checking using satisfiability solving. Formal methods in system design **19**(1), 7–34 (2001)
5. Daigle Jr, B.J., Roh, M.K., Gillespie, D.T., Petzold, L.R.: Automated estimation of rare event probabilities in biochemical systems. The Journal of chemical physics **134**(4), 01B628 (2011)
6. Dehnert, C., Jansen, N., Wimmer, R., Ábrahám, E., Katoen, J.P.: Fast debugging of prism models. In: International Symposium on Automated Technology for Verification and Analysis. pp. 146–162. Springer (2014)
7. Dehnert, C., Junges, S., Katoen, J.P., Volk, M.: A storm is coming: A modern probabilistic model checker. In: International Conference on Computer Aided Verification. pp. 592–600. Springer (2017)
8. Funke, F., Jantsch, S., Baier, C.: Farkas certificates and minimal witnesses for probabilistic reachability constraints. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 324–345. Springer (2020)
9. Gillespie, C.S., Golightly, A.: Guided proposals for efficient weighted stochastic simulation. The Journal of chemical physics **150**(22), 224103 (2019)
10. Gillespie, D.T.: A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. Journal of Computational Physics **22**(4), 403–434 (1976). https://doi.org/https://doi.org/10.1016/0021-9991(76)90041-3, `https://www.sciencedirect.com/science/article/pii/0021999176900413`
11. Glynn, P.W., Iglehart, D.L.: Importance sampling for stochastic simulations. Management science **35**(11), 1367–1392 (1989)
12. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: Infamy: An infinite-state markov model checker. In: Bouajjani, A., Maler, O. (eds.) Computer Aided Verification. pp. 641–647. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
13. Han, T., Katoen, J.P.: Counterexamples in probabilistic model checking. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 72–86. Springer (2007)
14. Han, T., Katoen, J.P.: Providing evidence of likely being on time: Counterexample generation for ctmc model checking. In: International Symposium on Automated Technology for Verification and Analysis. pp. 331–346. Springer (2007)
15. Jeppson, J., Volk, M., Israelsen, B., Roberts, R., Williams, A., Buecherl, L., Myers, C.J., Zheng, H., Winstead, C., Zhang, Z.: Stamina in c++: Modernizing an infinite-state probabilistic model checker. In: Jansen, N., Tribastone, M. (eds.) Quantitative Evaluation of Systems. pp. 101–109. Springer Nature Switzerland, Cham (2023)
16. Kuwahara, H., Mura, I.: An efficient and exact stochastic simulation method to analyze rare events in biochemical systems. The Journal of chemical physics **129**(16), 10B619 (2008)

17. Kwiatkowska, M., Norman, G., Parker, D.: Advances and challenges of probabilistic model checking. In: 2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton). pp. 1691–1698. IEEE (2010)
18. Kwiatkowska, M., Norman, G., Parker, D.: Prism 4.0: Verification of probabilistic real-time systems. In: International conference on computer aided verification. pp. 585–591. Springer (2011)
19. Moura, L.d., Bjørner, N.: Z3: An efficient smt solver. In: International conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer (2008)
20. Neupane, T., Myers, C.J., Madsen, C., Zheng, H., Zhang, Z.: Stamina: Stochastic approximate model-checker for infinite-state analysis. In: International Conference on Computer Aided Verification. pp. 540–549. Springer (2019)
21. Nielsen, A.A.K., Der, B.S., Shin, J., Vaidyanathan, P., Paralanov, V., Strychalski, E.A., Ross, D., Densmore, D., Voigt, C.A.: Genetic circuit design automation. Science **352**(6281) (2016). https://doi.org/10.1126/science.aac7341, `https://www.sciencemag.org/lookup/doi/10.1126/science.aac7341`
22. Roh, M.K., Gillespie, D.T., Petzold, L.R.: State-dependent biasing method for importance sampling in the weighted stochastic simulation algorithm. The Journal of chemical physics **133**(17), 174106 (2010)
23. Wimmer, R., Jansen, N., Ábrahám, E., Becker, B., Katoen, J.P.: Minimal critical subsystems for discrete-time markov models. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 299–314. Springer (2012)