Cycle and Commute: Rare-Event Probability Verification for Chemical Reaction Networks

Landon Taylor
Utah State University
Logan, Utah, USA
landon.jeffrey.taylor@usu.edu

Bryant Israelsen
Utah State University
Logan, Utah, USA
bryant.israelsen@usu.edu

Zhen Zhang
Utah State University
Logan, Utah, USA
zhen.zhang@usu.edu

***Abstract*—In synthetic biological systems, rare events can cause undesirable behavior leading to pathological effects. Due to their low observability, rare events are challenging to analyze using existing stochastic simulation methods. Chemical Reaction Networks (CRNs) are a general-purpose formal language for modeling chemical kinetics. This paper presents a fully automated approach to efficiently construct a large number of concurrent traces by expanding a sample of known traces. These traces constitute a partial state space containing only traces leading to a rare event of interest. This state space is then used to compute a lower bound for the rare event's probability. We propose a novel approach for the analysis of highly concurrent CRNs, including a CRN reaction independence analysis and an algorithm that exploits CRN concurrency to rapidly enumerate parallel traces. We then present a novel algorithm to add cycles to a partial state space to further increase the rare event's probability lower bound to its actual value. The resulting prototype tool, RAGTIMER, demonstrates improvement over stochastic simulation and probabilistic model checking.**

***Index Terms*—concurrency, rare events, chemical reaction networks**

I. Introduction

Chemical Reaction Networks (CRNs) are a general-purpose language for modeling chemical kinetics in genetic regulatory networks [1], molecular programming [2], and biochemical reaction systems [3]. Probabilistic behavior is inherent to many systems modeled in CRNs. For example, gene and protein expressions include reactions that occur simultaneously with distinct probabilities. Further, noisy biological systems can easily introduce unexpected and erroneous behavior. In these systems, rare events are often highly relevant, as they can represent infrequent but undesirable behavior that may lead to pathological consequences. Obtaining reliability guarantees is thus essential for CRNs. Existing formal verification techniques, such as *probabilistic model checking* (PMC), can provide provable guarantees to quantify a rare event's probability in CRNs. In practice, it is often necessary to generate a large number of traces to guarantee an accurate lower bound for a rare-event probability, as a single trace or a small number of traces often yields an insufficient estimate. Existing PMC tools are often unable to enumerate large or infinite state spaces to gather traces and verify a rare event's probability [4]. The computation of a rare event's probability can easily become intractable in this case.

This paper presents a *fully* automated approach to exploit a CRN model's concurrency to rapidly expand a small sample of traces into a partial state space that *only* includes traces leading to a rare event of interest. This partial state space guarantees a lower bound for the rare-event probability. We first propose an independence relation analysis for CRN reactions. It enables a novel parallel trace discovery algorithm that effectively expands a small number of traces. Additionally, we present a novel algorithm to detect and add productive cycles to explored states. Together, the constructed partial state space is used to compute the rare event's probability lower bound.

In benchmarking tests, a prototype implementation, *Cycle & Commute* expansion of the *Random Assume Guarantee Testing Induced Model Executions for Reachability* (RAGTIMER) tool [5], demonstrates encouraging results for several challenging CRN models. We believe that this unique combination of parallel trace discovery and cycle addition has not been proposed elsewhere and is a fully automated, effective, and user-friendly alternative to existing rare-event simulation approaches for the analysis of CRN rare-event properties.

II. Motivating Example

The *modified yeast polarization* model [6] was modified from the pheromone-induced G-protein cycle in Saccharomyces cerevisia [7] with a constant ligand population that keeps it away from reaching equilibrium [8], as follows:

$$\begin{aligned}
&\mathcal{R}_1: \emptyset \xrightarrow{0.0038} \mathrm{R}, && \mathcal{R}_2: \mathrm{R} \xrightarrow{4.00\times 10^{-4}} \emptyset,\\
&\mathcal{R}_3: \mathrm{L}+\mathrm{R} \xrightarrow{0.042} \mathrm{RL}+\mathrm{L}, && \mathcal{R}_4: \mathrm{RL} \xrightarrow{0.010} \mathrm{R},\\
&\mathcal{R}_5: \mathrm{RL}+\mathrm{G} \xrightarrow{0.011} \mathrm{G_a}+\mathrm{G_{bg}}, && \mathcal{R}_6: \mathrm{G_a} \xrightarrow{0.100} \mathrm{G_d},\\
&\mathcal{R}_7: \mathrm{G_d}+\mathrm{G_{bg}} \xrightarrow{1.05\times 10^{3}} \mathrm{G}, && \mathcal{R}_8: \emptyset \xrightarrow{3.21} \mathrm{RL}.
\end{aligned}$$

This CRN has eight chemical reactions interacting with the species vector $[R, L, RL, G, G_a, G_{bg}, G_d]$. All reaction propensities are in molecules per second. The initial state $s_0 = [50, 2, 0, 50, 0, 0, 0]$ represents the corresponding molecule count. This model incurs a large state space due to its highly concurrent nature, e.g., $\mathcal{R}_1$ and $\mathcal{R}_8$ are both independent of all other reactions. Also, by inspection, one can see that at least 100 reactions must execute to reach a state where $G_{bg} = 50$. As discussed in Section VIII, this model challenges several cutting-edge probabilistic model checking tools.

III. Preliminaries

1) Chemical Reaction Networks (CRNs): A CRN is a tuple $\mathcal{M}$ composed of m chemical species $\mathfrak{X} = \{\mathcal{X}_0, \ldots, \mathcal{X}_{m-1}\}$, n reactions $\mathfrak{R} = \{\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}\}$, an initial state $s_0 : \mathfrak{X}^m \to \mathbb{Z}_{\geqslant 0}$, and a vector of all species' initial molecule count, where $m, n \in \mathbb{Z}_{\geqslant 0}$ and $m, n < \infty$. A CRN is represented as a *Vector Addition System* (VAS) as follows, adapted from [9]. A reaction tuple $\mathcal{R}_i = \langle \mathbf{rv}_i, \mathbf{pv}_i, k_i, \theta_i \rangle$ includes the following: a reactant vector $\mathbf{rv}_i \in \mathbb{Z}^m_{\geqslant 0}$ representing the stoichiometry of reactants, a product vector $\mathbf{pv}_i \in \mathbb{Z}^m_{\geqslant 0}$ representing the stoichiometry of products, a reaction rate coefficient $k_i \in \mathbb{R}^+$, and a *propensity function* $\theta_i : \mathbb{Z}^m_{\geqslant 0} \to \mathbb{R}^+$ representing the probability that $\mathcal{R}_i$ occurs in a state. The *state change vector*, $\lambda_i = \mathbf{pv}_i - \mathbf{rv}_i$, represents the molecule count update for each species involved in $\mathcal{R}_i$. In this work, all CRN models follow the *Stochastic Chemical Kinetic* (SCK) assumption, which requires that each reaction $\mathcal{R}_i$ occurs nearly instantaneously, practically limiting elements of λ_i to the values of $0, \pm 1, \pm 2$ and at most three reactants in one reaction [1].

2) CRN Semantics: The underlying model of a CRN is a *Continuous-time Markov Chain* (CTMC), where state updates occur in discrete amounts and the probability of state change is a function of time. Formally, a CTMC is a tuple $\mathcal{C} = \langle \mathbf{S}, s_0, \mathbf{R}, \mathbf{L} \rangle$ where $\mathbf{S}$ is a finite state set called the *state space*; $s_0 \in \mathbf{S}$ is the initial state; $\mathbf{R} : \mathbf{S} \times \mathbf{S} \to \mathbb{R}_{\geqslant 0}$ is the transition rate matrix; and $\mathbf{L} : \mathbf{S} \to 2^{AP}$ is a state labeling function with atomic proposition set AP. A reaction $\mathcal{R}_i$ is *enabled* in state s if its propensity function $\theta_i(s)$ evaluates to a positive value. The propensity function is the product of k_i and the number of possible combinations of reactant molecules: $\theta_i(s) = k_i \prod_{\mathcal{X}_j \in \mathsf{Reactant}_i}(s[j])$. $\mathsf{Reactant}_i \subseteq \mathfrak{X}$ is the set of reactants for $\mathcal{R}_i$: $\mathsf{Reactant}_i = \{\mathcal{X}_\alpha \mid \mathbf{rv}_i[\alpha] > 0, \forall 0 \leqslant \alpha < m\}$. The propensity function is the transition rate $\mathbf{R}(s, s')$ from state s to s' in the CTMC $\mathcal{C}$ induced by a CRN. The probability that reaction $\mathcal{R}_i$ is selected to occur out of many reactions is $p(s, s') = \frac{\mathbf{R}(s,s')}{E(s)}$, where the *exit rate* $E(s) = \sum_{s' \in post(s)} \mathbf{R}(s, s')$ is the sum of all enabled reaction rates in s. A CTMC has a non-zero probability of staying in a state. The probability of exiting a state s in time interval $[0, t]$ is $1 - e^{-E(s) \cdot t}$, where $t \in \mathbb{R}_{\geqslant 0}$ represents real time. For example, $\mathcal{R}_3$ is executed from the given initial state in the motivating example to reach $s_1 = [49, 2, 1, 50, 0, 0, 0]$. In this state, $\mathbf{rv}_5$ and $\mathbf{pv}_5$ for $\mathcal{R}_5$ are $[0, 0, 1, 1, 0, 0, 0]$ and $[0, 0, 0, 0, 1, 1, 0]$, respectively; k_5 is 0.011; and $\theta_5(s_1) = k_5(s_1[2])(s_1[3]) = 0.011 \cdot 1 \cdot 50 = 0.55 > 0$, indicating that $\mathcal{R}_5$ is enabled in s_1. The state change vector λ_5 is $[0, 0, -1, -1, 1, 1, 0]$. Additional enabled reactions and their propensities at this state are $\mathcal{R}_1$ (0.0038), $\mathcal{R}_2$ (0.0196), $\mathcal{R}_3$ (4.116), $\mathcal{R}_4$ (0.01), and $\mathcal{R}_8$ (3.21). The exit rate $E(s_1)$ is 7.9094 and the probability that $\mathcal{R}_5$ executes is $0.55/7.9094 \approx 0.0695$.

3) Time-bounded Reachability Property and Target States: In *Continuous Stochastic Logic* (CSL) [10], [11], the non-nested time-bounded transient reachability probability is specified as $\mathsf{P}_{=?}(\Diamond^{[0,T]}\ \Psi)$. It represents the probability of reaching rare-event Ψ-states within a time bound of T. In this work, a *target* Ψ is an equality condition on exactly one species and is not satisfied in s_0. Formally, let condition Ψ be $\mathcal{X}_\Psi = C_\Psi$, where $C_\Psi \in \mathbb{Z}_{\geqslant 0}$ and $s_0(\mathcal{X}_\Psi) \neq C_\Psi$. A state s_i is a *target state* s_Ψ if and only if $s_i \models \Psi$. This work provides a guaranteed lower bound on the solution to $\mathsf{P}_{=?}(\Diamond^{[0,T]}\ \Psi)$.

4) Model Execution: Denote an execution of reaction $\mathcal{R}_i$ from state s_k as $s'_k = s_k + \lambda_i$. Denote "reaction $\mathcal{R}_i$ is enabled to execute at state s_k" as $\forall 0 \leqslant \alpha < m,\ s_k[\alpha] + \lambda_i[\alpha] \geqslant 0$. Let a *run* Ξ indicate a sequence of reactions. Reactions $\mathcal{R}_i$ and $\mathcal{R}_j$ are *adjacent* if $\mathcal{R}_j$ immediately follows $\mathcal{R}_i$ in Ξ. Run Ξ is a *valid run* from a state s_i (i.e., Valid(Ξ) holds for s_i) if no reaction in Ξ is disabled when Ξ's execution begins at state s_i. A *trace* ρ indicates a valid run starting with s_0 and terminating at a target state s_Ψ. A *seed trace* is a trace used as an input for the methods presented in this paper. Note that CRNs are often provided without upper bounds on the species count, which creates an infinite-state CTMC. However, because this work explores only finite traces from s_0 to Ψ-states, the partial state space constructed from these traces is finite.

In this work, seed traces are generated using the trace generation feature in RAGTIMER. RAGTIMER uses compositional testing with assume-guarantee reasoning to rapidly generate many shortest traces.

IV. Related Work

A CRN can be represented as a Vector Addition System (VAS) [9], sometimes described as a Petri net [12]. Reachability analysis, cycle detection, and other properties make VAS a convenient formalism to represent a CRN [13]–[16].

Rare-event properties often found in CRNs pose a challenge to modern stochastic simulation and probabilistic verification methods due to their extremely low observability. The effectiveness of the *weighted Stochastic Simulation Algorithm* (wSSA) [17] heavily relies on a user-specified probability biasing scheme to favor reactions leading to a rare event. Extensions of wSSA (e.g., [18]–[20]) have substantially improved its efficiency. As an alternative to wSSA, the *weighted ensemble* (WE) technique [21], [22] has been used to sample CRN rare events [23], [24]. Existing *statistical model checking* (SMC) techniques (e.g., [25], [26]) integrate rare-event methods. *Importance sampling* [27], [28] weighs the rare-event probability to bias simulation in order to increase the likelihood of encountering rare events of interest. It then compensates for the loss to yield an unbiased probability. In *importance splitting* [29]–[31], an importance function, potentially constructed manually, is used to reward or terminate simulation traces to divide a model's state space into contiguous levels ordered by increasing likelihood of reaching a rare event [32], [33]. Authors of [34] present an automated importance function derivation technique and recently re-implemented the extended RESTART with the *prolonged retrials* importance technique [35], [36] in the SMC engine `modes` [34], [37], available in the MODEST TOOLSET [38].

The proposed method is *fully* automated and does not require expert knowledge of the CRN model. It is less computationally intensive than other rare-event analysis methods, as it neither requires rare-event biasing computations nor wastes computational effort pursuing runs that do not lead to a rare event. Lastly, it yields a probability lower-bound with provable guarantees instead of a probability estimate.

V. CRN Independence and Commutability

CRNs are intrinsically highly concurrent. Consider the motivating example in Section II. Reactions $\mathcal{R}_1$ and $\mathcal{R}_8$ are *always* enabled, regardless of the current state of the CRN. By leveraging properties of the VAS representation of a CRN, we present a novel analysis of the independence relation among CRN reactions, enabling effective state space exploration.

A. Independence Relation for CRN Reactions

The study of action independence can be traced back to the work of Lipton [39] and Mazurkiewicz [40] on commuting concurrent actions. Mazurkiewicz traces are equivalent classes of action sequences. Action independence has also been the foundation of partial order reduction techniques (e.g. [41]–[43]) for verifying concurrent system correctness. We propose an independence relation specific to reactions in a CRN.

Definition 1 (Independence of CRN Reactions): Two adjacent reactions $\mathcal{R}_i$ and $\mathcal{R}_j$ (defined in Section III-4) are *independent* and *enabled* at state s_k if and only if:

1) $\mathcal{R}_i$ and $\mathcal{R}_j$ can execute in either order from s_k: $(s_k + \lambda_i) + \lambda_j = (s_k + \lambda_j) + \lambda_i$.
2) $\mathcal{R}_j$ is enabled after $\mathcal{R}_i$ executes at s_k: $\forall 0 \leqslant \alpha < m,\ (s_k + \lambda_i)[\alpha] + \lambda_j[\alpha] \geqslant 0$.
3) $\mathcal{R}_i$ is enabled after $\mathcal{R}_j$ executes at s_k: $\forall 0 \leqslant \alpha < m,\ (s_k + \lambda_j)[\alpha] + \lambda_i[\alpha] \geqslant 0$.

If $\mathcal{R}_i$ and $\mathcal{R}_j$ are not independent, they are *dependent*.

Because a VAS representation of a CRN reaction involves only vector addition, condition (1) is true in every state for which both conditions (2) and (3) hold. That is, because vector addition is commutative and associative, firing a series of enabled reactions from a designated state in any order *always* results in the same final state. Conditions (2) and (3) thus become sufficient and necessary conditions for the independence of CRN reactions.

B. Commutability of Reactions

Conditions (2) and (3) described above enable reaction independence (and thus commutability) to be further categorized. We propose three classes to represent commutability between adjacent reactions: trivially, semi-trivially, and conditionally commutable pairs. Adjacent reactions $\mathcal{R}_i$ and $\mathcal{R}_j$ are a *trivially commutable pair* iff $\forall\ 0 \leqslant \alpha < m,\ \lambda_i[\alpha], \lambda_j[\alpha] \in \mathbb{Z}_{\geqslant 0}$. That is, $\mathcal{R}_i$ and $\mathcal{R}_j$ are trivially commutable at *all states* if they require no reactants to produce their products. $\mathcal{R}_i$ and $\mathcal{R}_j$ are a *semi-trivially commutable pair* iff $\forall\ 0 \leqslant \alpha < m,\ \mathbf{rv}_i[\alpha] = 0 \vee \mathbf{rv}_j[\alpha] = 0$. That is, $\mathcal{R}_i$ and $\mathcal{R}_j$ are semi-trivially commutable at s_k if they share no reactants and are both enabled at s_k. Intuitively, reactions $\mathcal{R}_2$ and $\mathcal{R}_4$ in the motivating example are semi-trivially independent because they share no reactants. If a state s_k provides sufficient R to enable $\mathcal{R}_2$ and sufficient RL to enable $\mathcal{R}_4$, then it is always the case that $\mathcal{R}_2$ and $\mathcal{R}_4$ are enabled to execute in any order from s_k. If $\mathcal{R}_i$ and $\mathcal{R}_j$ are neither trivially nor semi-trivially commutable, they are *conditionally commutable*.

Trivially and semi-trivially commutable pairs do not require explicitly checking conditions (2) or (3), enabling state exploration to bypass the need to simulate adjacent reactions to determine commutability. In trivially commutable pairs, λ_i and λ_j contain only non-negative integers, so it is always the case that $\lambda_i + \lambda_j$ contains only non-negative integers. In semi-trivially commutable pairs, each element of $\lambda_i + \lambda_j$ contains at least the lowest negative value in reactants of either λ_i or λ_j, because $\mathcal{R}_i$ and $\mathcal{R}_j$ do not share reactants. Thus, one reaction in a semi-trivially commutable pair cannot disable the other, so if Equation 1 holds, conditions (2) and (3) must also hold. Checking Equation 1 removes the need to simulate semi-trivially commutable reactions directly, conserving effort while exploring the state space. Conditionally commutable pairs require conditions (2) and (3) to be checked explicitly.

$$\forall 0 \leqslant \alpha < m,\ (s_k[\alpha] + \lambda_i[\alpha] \geqslant 0 \wedge s_k[\alpha] + \lambda_j[\alpha] \geqslant 0) \quad (1)$$

C. Sequences of Conditionally Commutable Reactions

Given a run consisting of a sequence of κ (potentially repeating) reactions $\Xi = \mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{\kappa-1}$, it may be desirable to check that firing a sequence of reactions Ξ from s_0 produces a valid run (i.e. each reaction in Ξ is enabled when Ξ is executed in order). If Ξ contains conditionally commutable pairs of reactions, Equation 2 checks that Ξ is a valid run from a state s_x. In the motivating example, a valid run from the initial state s_0 is $\mathcal{R}_8$, $\mathcal{R}_5$; while an invalid run from s_0 is $\mathcal{R}_5$, $\mathcal{R}_8$ because $\mathcal{R}_5$ is not enabled from the initial state, but firing $\mathcal{R}_8$ enables the execution of $\mathcal{R}_5$.

$$\text{Valid}(\Xi) := (\forall j \leqslant \kappa, 0 \leqslant i < m,\ s_x + \sum_{\alpha=0}^{j} \lambda_\alpha[i] \geqslant 0) \quad (2)$$

VI. Parallel Traces via Commutation

Exploring the inherent concurrency in CRN models helps to discover traces contributing to a rare event's probability. These traces may be obtained by various methods. Results presented in this paper use traces generated by the prototype tool RAGTIMER. CRN models often contain a large volume of parallel traces (i.e., traces that differ by a small number of reactions or arrive at the same state while passing through alternative intermediate states).

To obtain a lower bound for the rare event's probability, we desire to accumulate probability from a large number of traces to a rare event as efficiently as possible. We suggest parallel traces are an efficient way to accumulate probability for rare events. Algorithm 1 finds parallel traces using Equations 1 and 2 to discover pairs of independent, commutable reactions. For example, interrupting a seed trace from the motivating example by firing $\mathcal{R}_1$ at a random state forms a nearly-identical trace and increases the overall probability lower bound relative to the seed trace alone.

Figure 1 illustrates this principle on a small toy example. In this example, the seed trace (blue) contains reactions $\mathcal{R}_0$, $\mathcal{R}_1$, and $\mathcal{R}_2$. By interrupting this seed trace with a universally-enabled reaction $\mathcal{R}_a$, it is possible to obtain many parallel traces and increase the lower-bound of the probability of reaching a target state. This particular example shows two unique target states with four additional traces, so the probability of reaching a target is increased compared to the probability of the seed trace alone. In some models, parallel traces arrive at the same target state as the seed trace via an alternative reaction sequence. Having two target states, as is the case in Figure 1, is allowed but not required for parallel trace exploration; only target state s'_Ψ is required.

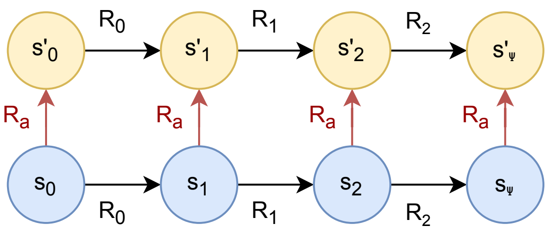


Fig. 1: Parallel trace construction via transition commutation.

A. Trace Commutation Algorithm

Algorithm 1 details the procedure for exploring parallel traces contained in the "Traces" set, which contains seed traces generated by RAGTIMER or a user's method of choice. As the main procedure, BUILDTRACES builds a partial state space for each seed trace in "Traces", then calls the recursive function COMMUTE on each trace, which recursively explores traces parallel to each seed trace and builds a partial state space as it explores. Using commutability conditions presented in Section V, Algorithm 1 attempts to find commutable reactions along the entire length of a seed trace. To efficiently explore traces leading to a rare event, it attempts to commute reactions that are enabled from *every* state along the seed trace.

In Figure 1, for instance, line 2 of Algorithm 1 selects the seed trace $\mathcal{R}_0$, $\mathcal{R}_1$, $\mathcal{R}_2$ as ρ. In lines 3 and 4, it builds a partial state space for the seed trace, then discovers $\mathcal{R}_a \in E$. In COMMUTE, it executes the prefix (line 11), which is initially empty but is extended during recursion to list the sequence of commuted reactions to fire before firing the reactions from the seed trace. In lines 12 and 13, states along the parallel trace ρ' (shown in yellow on the top of Figure 1) are discovered. In lines 14 and 15, ρ' is built from the commuted transition $\mathcal{R}_a$. Finally, in line 16, the function recursively attempts to commute transitions along the parallel trace ρ', which now includes prefix $\mathcal{R}_a$. This recursive process is shown in Figure 2. The seed trace, shown in blue with s_0, leads to the discovery of three parallel traces shown in yellow. These traces, with s_b, s_c, and s_d, are then recursively analyzed. For instance, it may lead to the discovery of two more parallel traces, shown in green with states s_e and s_f.

Algorithm 1 Commuting universally enabled transitions

```
Require: $\mathcal{M} = \langle \mathfrak{X}, \mathfrak{R}, s_0 \rangle$, $\Psi$, Traces.
 1: procedure BUILDTRACES
 2:     for Trace $\rho$ in Traces do
 3:         Build the state space for states along $\rho$
 4:         $E \leftarrow$ enabled reactions along $\rho$
 5:         COMMUTE($\emptyset$, $\rho$, $E$)
 6:     BUILDCYCLES      ▷ Defined formally in Algorithm 2
 7:     Clean up the model to save time and memory
 8:     Export explicit state-transition matrices
 9: procedure COMMUTE(Prefix, $\rho$, Enabled)
10:     for $\mathcal{R}_a \in$ Enabled do
11:         Execute Prefix.
12:         Fire $\mathcal{R}_a$ from each state in $\rho$ to find $\rho'$
13:         $E' \leftarrow$ enabled reactions along $\rho'$.
14:         Execute $\mathcal{R}_a$ from $s_0$ of $\rho$ to find $s'_0$ in $\rho'$.
15:         Execute reactions in $\rho$ from $s'_0$ to construct $\rho'$.
16:         COMMUTE((Prefix.append($\mathcal{R}_a$)), $\rho'$, $E'$)
```

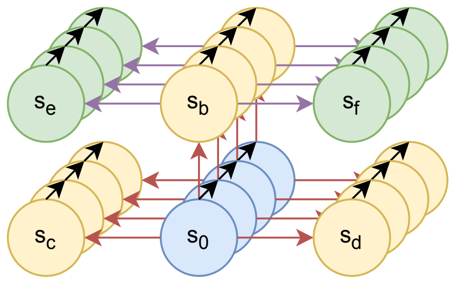


Fig. 2: Recursively-commuted partial state space.

B. Termination Conditions on Algorithm 1

Algorithm 1 does not necessarily terminate. Thus, we propose two methods for determining when to terminate commutation recursion:

1) The user can specify a maximum recursion depth. This is a naive approach, but it guarantees termination and gives flexibility to advanced users; or
2) The algorithm can terminate based on the time bound T from the model's CSL property $\mathsf{P}_{=?}(\Diamond^{[0,T]}\ \Psi)$. The mean state residence time for s_i is provided by $\mathrm{MRT}(s_i) = 1/E(s_i)$. The sum of mean state residence times along a trace provides the average duration for that trace. If the average trace duration exceeds or approaches the property time bound, it is likely not worth exploring further as traces will become increasingly unlikely. This approach requires less understanding of the model and algorithm, so it provides less flexibility but a more streamlined user experience.

Algorithm 1 is implemented as an extension of the RAGTIMER tool. In this implementation, a user can specify if they prefer to terminate by recursion depth or by analyzing average trace durations. Approach (1) terminates trivially. Termination of approach (2) is justified because each reaction adds a positive amount of time to the total average trace duration. The algorithm will thus either explore the entire available state space or the average duration will eventually increase to meet the termination threshold. It is our experience that a very small time or depth bound is sufficient to obtain a significant probability boost in the seed traces.

C. Exporting Explicit Models

After exploring parallel traces, each unexplored enabled reaction at any state is replaced by an absorbing reaction (i.e., a new reaction transitioning to an absorbing state) with an equivalent probability. Formally, let $En(s_i)$ represent the set of all reactions enabled in state s_i. Let $Disc(s_i)$ represent the set of all reactions added to the explicit state space, such that $Disc(s_i) \subseteq En(s_i)$. Let $Undisc(s_i)$ represent the set reactions in $En(s_i)$ but not included in the explicit state space, such that $Undisc(s_i) = En(s_i) - Disc(s_i)$.

Let $A(s_i)$, defined in Equation 3, indicate the sum of transition rates directed from state s_i to an abstract absorbing state. The absorbing state preserves probability correctness by consuming any probability that would have been directed to an unexplored portion of the state space. To obtain a probabilistic lower-bound, it is assumed that the absorbing state does not satisfy Ψ. An explicit state space can then be exported for model checking in a tool such as PRISM [44] or Storm [45].

$$A(s_i) = \sum_{\mathcal{R}_j \in Undisc(s_i)} \theta(s_j) \tag{3}$$

D. Lower-Bound Probability Guarantee

Because the presented method explicitly enumerates traces, the probability obtained by performing probabilistic model checking on the explicit state graph is guaranteed to be a lower bound. The seed trace is known to reach a target state, so finding parallel traces through commutation also produces traces leading to the same target state. This method is efficient because every state and reaction (except the absorbing state) is guaranteed to contribute to a rare event's probability.

VII. Cycles for Probability Recapture

CRN models often contain cyclic behavior. Including cycles in a state space is an effective way to increase the total number of explored traces without greatly increasing the total number of states explored. In many models, the exploration of cycles can increase the probability lower bound by redirecting some of the probability that would otherwise be redirected to an absorbing state (see Section VI-C) to a target state.

While a number of cycle exploration methods have been explored (in [13], for instance), we found a simple combinatorial analysis of reactions sufficient to efficiently generate a large number of cycles for the purposes of this work. This approach involves testing multisets of reactions up to a user-specified bound and selecting multisets of reactions such that the sum of state change vectors corresponding to reactions within each multiset is equal to the zero vector.

Formally, a cycle c_i is a κ-multiset containing κ reactions such that the sum of all reaction state change vectors in c_i is the zero vector. Let "CycleList" be a set of known cycles. Algorithm 2 presents an approach to augmenting the probability lower bound by adding cycles into a partial state space. Let $\omega(c_i)$ represent a permutation of reactions in c_i, with $\Omega(c_i)$ defined as the set of all possible $\omega(c_i)$. Define $\min(\omega(c_i))$ as a vector of length m (i.e., a vector with one element per species) such that $\forall 0 \leq \alpha < m$, $\min(\omega(c_i))[\alpha] = \min_{\mathcal{R}_j \in \omega(c_i)} \sum_{k=0}^{j} \lambda_k[\alpha]$. In Line 4 of Algorithm 2, this is achieved via a vector copy operation. Intuitively, the minimal value of species α in $\omega(c_i)$ is either non-negative, indicating species α is never consumed, or it is negative. If $\min(\omega(c_i))[\alpha] = -\gamma$, at some point, $\omega(c_i)$ has consumed and has not replenished γ molecules of species l. In Algorithm 2, $\min(\omega(c_i))$ determines which states are candidates for the addition of $\omega(c_i)$. If a state s_i does not provide enough of a given reactant to execute $\omega(c_i)$, i.e. if Valid($\omega(c_i)$) does not hold at state s_i, cycle $\omega(c_i)$ cannot be added to s_i. By finding the minimal value for a molecule count during a cycle, it becomes unnecessary to simulate a cycle from every state to determine if it is possible to add the cycle to the state. This saves computational effort while enabling cycles to be added to every allowable state. Maximum cycle *lengths* are specified by users, and *all* allowable cycles up to the user-specified length are added to the state space.

In the motivating example, executing $\mathcal{R}_2$ followed by $\mathcal{R}_1$ constitutes a permutation $\omega(c)$ of the cycle with length two, i.e., $c = \{\mathcal{R}_1, \mathcal{R}_2\}$. Because this cycle causes a degradation of R followed by a generation of R, $\min(\omega(c))[0] = -1$. That is, $\omega(c)$ can only be added to state s_i if $s_i[0] \geqslant 1$.

Algorithm 2 Adding cycles to a partial state space

```
Require: $\mathcal{M} = \langle \mathfrak{X}, \mathfrak{R}, s_0 \rangle$, CycleList.
1: procedure BUILDCYCLES
2:     for Cycle $c_i$ in CycleList do
3:         for Cycle permutation $\omega(c_i)$ in $\Omega(c_i)$ do
4:             $\min(\omega(c_i)) \leftarrow \min_{\mathcal{R}_j \in \omega(c_i)} \sum_{k=0}^{j} \lambda_k$
5:             for State $s_x$ in discovered state space do
6:                 for $\alpha \in [0, m)$ do
7:                     if $s_x[\alpha] + \min(\omega(c_i))[\alpha] \geq 0$ then
8:                         Add $\omega(c_i)$ to $s_x$
```

Adding even one cycle to a trace can increase the probability of that trace. Because Algorithm 1 produces an explicit state space, the task of evaluating the overall probability impact of cycles is given to a probabilistic model checker. This enables only a few additional states in the explicit state space to influence the probability of the model by providing a larger number of traces. For example, Figure 3 shows the partial state space from Figure 1. An arbitrary cycle (represented by three green states) is enabled to be executed from five states, so rather than direct reactions from those states to an absorbing state, the cycles redirect part of the probability back into the trace leading to the target rare-event state. Note that in this example, the cycle is not added to state s_1. In a realistic model, this happens when s_1 does not provide sufficient reactants to enable the full execution of the cycle.

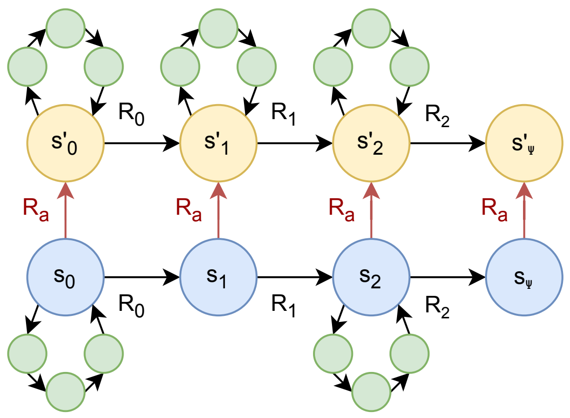


Fig. 3: Cycles added to five states in a partial state space.

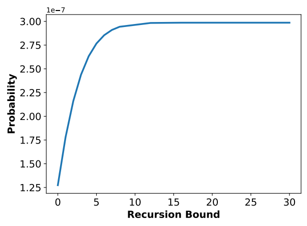
It is occasionally the case that cycles add states and transitions (and thus computation time and memory for probabilistic model checking) to a state space without contributing significantly to the rare event's probability bound. This is largely the case when one or more states along a cycle has a large absorbing rate relative to its other outgoing rates (i.e., when it is more likely that starting a cycle will lead to an absorbing state than return to the original trace). If enough of the probability that flows into the cycle does not flow back toward the target state, the cycle is not sufficiently valuable and need not be added. Given a desirable threshold $\mathbb{T}$ (in our tests, a high threshold of around 0.98) of the ratio $A(s_i)/E(s_i)$, Lines 7-8 of Algorithm 2 may be adapted to include a cycle addition benefit heuristic as shown in Equation 4. The ratio $A(s_i)/E(s_i)$ intuitively represents how much probability is directed to an absorbing state versus into a trace.

$$\textbf{if } s_x[y] + min(\omega(c_i))[y] \geq 0 \wedge A(s_i)/E(s_i) < \mathbb{T} \quad \textbf{then } \text{Add } \omega(c_i) \text{ to } s_x \qquad (4)$$

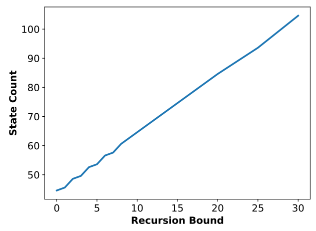
VIII. Results and Discussion

The parallel trace exploration and cycle addition methods presented in this paper are implemented as part of a prototype tool, RAGTIMER, which interfaces with the PRISM API [44]. Prototype versions of RAGTIMER and its Cycle & Commute expansion are freely available[1]. This tool quickly generates many seed traces. The benchmarking results presented in this paper were obtained on an AMD Ryzen Threadripper 12-Core 3.5 GHz Processor and 132 GB of RAM, running Ubuntu 22.04 LTS. We allocated one CPU and 16 GB of RAM to test our approach on all four challenging case studies and compared our method's results to those of other probabilistic verification tools. In each case study, "Default Cycle & Commute" indicates that the default settings for RAGTIMER are used. The default settings include generating 100 shortest traces, using a fixed recursion bound of 20 (i.e., limit the number of calls to the COMMUTE function in Algorithm 1 to 20), and adding cycles of two reactions after state space construction. These default settings give users an acceptable result for most models, while "Optimized Cycle & Commute" indicates custom settings for each model.

1) Single Species Production-Degradation Model: The model describes a production-degradation interaction between two species [17]: $\mathcal{R}_1 : S_1 \xrightarrow{1.0} S_1 + S_2, \mathcal{R}_2 : S_2 \xrightarrow{0.025} \emptyset$. The initial state for the species vector $[S_1, S_2]$ is $s_0 = [1, 40]$, while the desired CSL property is $\mathsf{P}_{=?}(\Diamond^{[0,100]} S_2 = 80)$. Figure 4a shows an increase in the lower bound of the model's probability as the parallel trace exploration recursion bound increases during exploration of a *single* seed trace. The probability bound asymptotically approaches the actual rare-event probability, which is 3.0631×10^{-7} [17]. Figure 4b shows that while the probability increases exponentially, the number of states increases linearly. Thus, we argue this method explores a productive part of the state space. Partial state space exploration required less than 4 seconds at any recursion depth.



(a)



(b)

Fig. 4: Single Species Production-Degradation Model.

It is interesting to compare the methods presented in this paper to simple trace generation, which RAGTIMER is already capable of. In our benchmarks, RAGTIMER generated a single seed trace for this model with a probability of 1.03×10^{-16} in 13.26 seconds. It then generated 43 additional traces, increasing the state space's probability to 1.34×10^{-16} in 31.35 seconds. By expanding the seed trace using methods presented in this paper, however, RAGTIMER achieved a probability bound of 2.99×10^{-7} in 21.21 seconds, including the duration of trace generation, commuting, and model checking. This is a reasonable lower bound to the true probability of the model (3.0631×10^{-7}). Table I summarizes these results. In all result tables, "Default Commuting Options" indicates that the default options implemented in RAGTIMER are selected; "Optimized Commuting Options" indicates that configurations were modified to produce an improved result. In this model, the default settings produced the best probability bound.

2) Enzymatic Futile Cycle Model: A futile cycle interaction is modeled in this CRN with six species reacting through six reactions [17]:

$$\begin{aligned}
\mathcal{R}_1 &: S_1 + S_2 \xrightarrow{1.0} S_3, & \mathcal{R}_2 &: S_3 \xrightarrow{1.0} S_1 + S_2, \\
\mathcal{R}_3 &: S_3 \xrightarrow{0.1} S_1 + S_5, & \mathcal{R}_4 &: S_4 + S_5 \xrightarrow{1.0} S_6, \\
\mathcal{R}_5 &: S_6 \xrightarrow{1.0} S_4 + S_5, & \mathcal{R}_6 &: S_6 \xrightarrow{0.1} S_4 + S_2.
\end{aligned}$$

[1]RAGTIMER v0.0 (trace generation) and v0.1 (Cycle & Commute) are available as releases at https://github.com/fluentverification/ragtimer/tags.

TABLE I: Single-Species Production-Degradation Model.

Method	Probability	Runtime (s)
Generate 1 Trace	$\geqslant 1.03 \times 10^{-16}$	13.26
Generate 44 Traces	$\geqslant 1.34 \times 10^{-16}$	31.35
Default Cycle & Commute	$\geqslant 2.99 \times 10^{-7}$	23.21
Optimized Cycle & Commute	$\geqslant 2.99 \times 10^{-7}$	23.21

The initial molecule count for species vector $[S_1, S_2, S_3, S_4, S_5, S_6]$ forms the initial state: $s_0 = [1, 50, 0, 1, 50, 0]$ and the rare-event property of interest is $\mathsf{P}_{=?}(\Diamond^{[0,100]} S_5 = 25)$. Figure 5 shows the probability and state count for this model's partial state space as the recursion depth increases for a *single* seed trace. The probability bound sharply increases while the state space grows linearly, illustrating that for this model, states that are considered in the partial state space contribute significantly to the probability bound. State space construction required less than four seconds for all recursion depths for this model. RAGTIMER generated one shortest seed trace for this

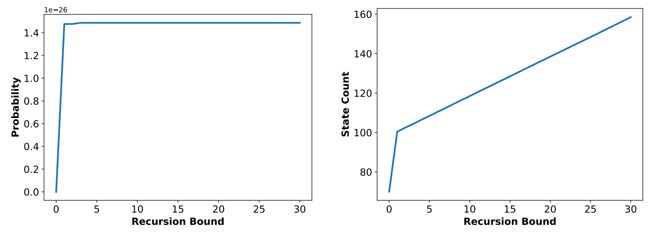


Fig. 5: Enzymatic Futile Cycle Model.

model with a probability of 1.73×10^{-78} in 15.52 seconds. Generating 99 more traces increased the probability bound to 2.71×10^{-64} in 41.1 seconds. By expanding 100 seed traces using methods presented in this paper, RAGTIMER achieved a probability bound of 4.32×10^{-18} in 31.69 seconds, including time for trace generation, commuting, and model checking. This is an improvement of 60 orders of magnitude that requires less runtime than generating a small sample of additional traces. Table II summarizes these results. This model's results appear to be influenced by the quality of the seed traces used to generate the partial state space. Its optimized settings involve asking RAGTIMER to generate a set of short but unique seed traces. After generating 42 unique traces, it explored a recursion depth of 10 and added all possible cycles of length two to the state space. Further, cycle

TABLE II: Enzymatic Futile Cycle Model.

Method	Probability	Runtime (s)
Generate 1 Trace	$\geqslant 1.73 \times 10^{-78}$	15.52
Generate 100 Traces	$\geqslant 2.71 \times 10^{-64}$	41.10
Default Cycle & Commute	$\geqslant 1.45 \times 10^{-26}$	27.92
Optimized Cycle & Commute	$\geqslant 4.32 \times 10^{-18}$	31.69

addition boosts the probability of this model without requiring significant additional time. We ran 36 tests to account for this method's stochastic nature and found that adding only cycles of length two to the model as described in Section VII increased the average discovered lower probability bound by two orders of magnitude (from 2.28×10^{-21} to 4.92×10^{-19}) while increasing the average total runtime by less than one second (from 25.6 to 26.4 seconds).

3) Modified Yeast Polarization Model: The rare event of interest for our motivating example is the rapid build-up of G_{bg}. This is described by the probability of the molecule count of G_{bg} increasing from 0 to 50 within 20 seconds: $\mathsf{P}_{=?}(\Diamond^{[0,20]} G_{bg} = 50)$. When this model is simulated using the standard *stochastic simulation algorithm* (SSA) implemented in the PRISM probabilistic model checking tool, the total probability for over $500,000$ traces is rounded to 0 due to floating-point precision limitations, indicating that SSA alone produced a probability lower than 4.9×10^{-324}. However, when two seed traces are expanded, the probability is found to be greater than 5.8×10^{-72} after a recursion depth of 6 and with a state space consisting of about 3500 states, produced in less than 10 seconds, as can be seen in Figure 6. Because of this model's infinite state space, the state count

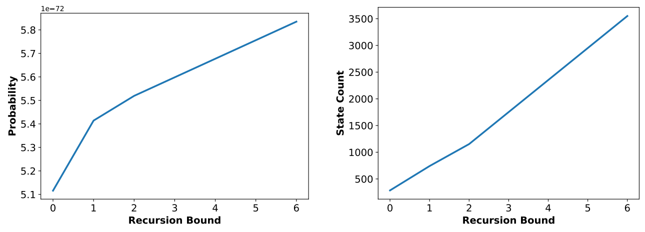


Fig. 6: Modified Yeast Polarization Model.

and probability both appear to increase nearly-linearly with recursion depth. Therefore, the states in this model's partial state space contribute significantly to its probability bound.

RAGTIMER generated 100 seed traces for which PRISM is unable to compute a nonzero probability (due to floating-point constraints). By expanding these seed traces using methods presented in this paper, however, RAGTIMER achieved a probability of 5.26×10^{-26} in 125.64 seconds, including trace generation, commuting, and model checking time. These results are summarized in Table III. Optimized settings for this model are identical to default settings, but the optimal test used a set of 100 higher-probability seed traces due to the stochastic nature of RAGTIMER trace generation.

TABLE III: Modified Yeast Polarization Model.

Method	Probability	Runtime (s)
Generate 1 Trace	$\geqslant 0.0$	23.34
Generate 100 Traces	$\geqslant 0.0$	66.78
Default Cycle & Commute	$\geqslant 1.01 \times 10^{-32}$	167.11
Optimized Cycle & Commute	$\geqslant 5.26 \times 10^{-26}$	125.64

4) Simplified motility regulation model: This model consists of nine species reacting through twelve reactions and represents the genetic mechanism which regulates flagella

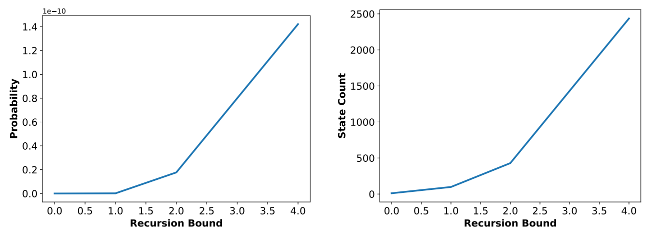


Fig. 7: Simplified Motility Regulation Model.

formation in *Bacillus subtilis* [46]:

$$\begin{aligned}
\mathcal{R}_1 &: \text{codY} \xrightarrow{0.1} \text{codY} + \text{CodY},\\
\mathcal{R}_2 &: \text{CodY} \xrightarrow{0.0002} \emptyset,\\
\mathcal{R}_3 &: \text{flache} \xrightarrow{1.0} \text{flache} + \text{SigD},\\
\mathcal{R}_4 &: \text{SigD} \xrightarrow{0.0002} \emptyset,\\
\mathcal{R}_5 &: \text{SigD\_hag} \xrightarrow{1.0} \text{SigD} + \text{hag} + \text{Hag},\\
\mathcal{R}_6 &: \text{Hag} \xrightarrow{0.0002} \emptyset,\\
\mathcal{R}_7 &: \text{SigD} + \text{hag} \xrightarrow{0.01} \text{SigD\_hag},\\
\mathcal{R}_8 &: \text{SigD\_hag} \xrightarrow{0.1} \text{SigD} + \text{hag},\\
\mathcal{R}_9 &: \text{CodY} + \text{flache} \xrightarrow{0.02} \text{CodY\_flache},\\
\mathcal{R}_{10} &: \text{CodY\_flache} \xrightarrow{0.1} \text{CodY} + \text{flache},\\
\mathcal{R}_{11} &: \text{CodY} + \text{hag} \xrightarrow{0.01} \text{CodY\_hag},\\
\mathcal{R}_{12} &: \text{CodY\_hag} \xrightarrow{0.1} \text{CodY} + \text{hag}.
\end{aligned}$$

The initial molecule count for species vector [codY, flache, SigD_hag, CodY, CodY_flache, hag,CodY_hag, SigD, Hag] forms the initial state $s_0 = [1, 1, 1, 10, 1, 1, 1, 10, 10]$. The rare event property is $\mathsf{P}_{=?}(\Diamond^{[0,10]}\,\text{CodY} = 20)$. Figure 7 shows that while a single seed trace's probability is found to be zero, expanding a single seed trace quickly increases the probability bound to 1.45×10^{-10}. Because the probability and state count growth both appear to grow exponentially relative to the recursion bound, it suggests the states explored by this method contribute efficiently to the rare event probability.

Similarly to the Modified Yeast Reaction Model, generating 100 seed traces produced a low probability bound that was rounded to zero. By expanding the seed trace using methods presented in this paper, however, RAGTIMER achieved a probability of 1.42×10^{-9} in 34.67 seconds, including trace generation, commuting, and model checking time. Results from this model are summarized in Table IV. Due to this model's complexity, its default recursion depth is 2. Increasing the recursion depth to 10 results in the optimized probability.

5) *Comparison to `modes` rare-event simulation engine:* The `modes` statistical model checking tool in the MODEST TOOLSET was able to compute rare-event probabilities efficiently for the presented case studies, and the reported probabilities closely match those reported in [17] and [23]. However, `modes` requires a compositional importance function for rare-event simulation, which limits the use of global variables shared between multiple components. While manual modifications to the model's importance function can be made to circumvent this, it requires user intervention and an in-depth understanding of the CRN model and MODEST language.

TABLE IV: Simplified Motility Regulation Model.

Method	Probability	Runtime (s)
Generate 1 Trace	$\geqslant 0.0$	13.39
Generate 100 Traces	$\geqslant 0.0$	17.16
Default Cycle & Commute	$\geqslant 1.77 \times 10^{-11}$	28.05
Optimized Cycle & Commute	$\geqslant 1.42 \times 10^{-9}$	34.67

6) *Comparison to probabilistic model checking tools:* We attempted to verify the modified yeast polarization model's rare event property with all species' molecule counts bounded by the reasonably large range of $[0, 150]$ in the probabilistic model checker Storm with the SYLVAN library [47]. Although Storm completed symbolic state space construction quickly, it failed to complete the CTMC analysis of the model within 30 days due to the task of converting a symbolic state space to a sparse matrix representation for time-bounded transient analysis. In another test, the state-truncation probabilistic model checker STAMINA [48] produced a probability bound of $[1.64\times 10^{-6}, 23.01\times 10^{-6}]$ on the same model after 2 days.

7) *Discussion:* We claim that our method can compete effectively against these tools because it requires no in-depth understanding of a CRN model, a formal modeling language, or a verification tool. Rather, it requires only a single trace (obtainable from the implemented functionality in RAGTIMER or another user-selected method). We argue that our method is effective because increasing the recursion bound and number of cycles in our benchmarks reliably provides an improved probability bound, demonstrating this method explores an effective region of a state space. This method can also be used to gain insights to guide model synthesis, as it can report information about which reactions and cycles cause a rare event to be more likely. We firmly believe that this model analysis is a useful tool to guide design decisions and reveal design flaws, and that in many cases, it is more useful to a user than a probability report alone.

IX. CONCLUSION

This paper presents a fully-automated approach to expand a small sample of traces and build a partial state space containing only states and transitions leading to a rare event of interest in a CRN model. We propose CRN-specific independence conditions accompanied by an algorithmic method to effectively discover parallel traces that are guaranteed to reach the rare event of interest. This increases the lower bound for the rare event's probability. Adding cycles to a partial state space further increases the rare-event probability lower bound. The promising results from the prototype tool RAGTIMER demonstrate it as an effective and user-friendly method for CRN model analysis. Future work may include further investigation of the properties of cycles in CRN explicit state spaces, integration with other existing probabilistic model checking tools, and improvement on seed trace generation.

Acknowledgment: We thank Arnd Hartmanns (U. Twente) for helping with MODEST TOOLSET; Chris Winstead (Utah State U.), Chris Myers (U. Colorado Boulder), and Hao Zheng (U. South Florida) for their feedback. This work was supported by the National Science Foundation under Grant No. 1856733. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] C. J. Myers, *Engineering Genetic Circuits*, 1st ed., ser. Chapman & Hall/CRC Mathematical and Computational Biology. Chapman & Hall/CRC, July 2009.

[2] D. Soloveichik, G. Seelig, and E. Winfree, "Dna as a universal substrate for chemical kinetics," *Proceedings of the National Academy of Sciences*, vol. 107, no. 12, pp. 5393–5398, 2010. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.0909380107

[3] V. Chellaboina, S. P. Bhat, W. M. Haddad, and D. S. Bernstein, "Modeling and analysis of mass-action kinetics," *IEEE Control Systems Magazine*, vol. 29, no. 4, pp. 60–78, 2009.

[4] L. Buecherl, R. Roberts, P. Fontanarrosa, P. J. Thomas, J. Mante, Z. Zhang, and C. J. Myers, "Stochastic hazard analysis of genetic circuits in iBioSim and STAMINA," *ACS Synthetic Biology*, vol. 10, no. 10, pp. 2532–2540, 2021, pMID: 34606710. [Online]. Available: https://doi.org/10.1021/acssynbio.1c00159

[5] B. Israelsen, L. Taylor, and Z. Zhang, "Efficient trace generation for rare-event analysis in chemical reaction networks," in *Model Checking Software*, G. Caltais and C. Schilling, Eds. Cham: Springer Nature Switzerland, 2023, pp. 83–102.

[6] B. J. Daigle, M. K. Roh, D. T. Gillespie, and L. R. Petzold, "Automated estimation of rare event probabilities in biochemical systems," *The Journal of Chemical Physics*, vol. 134, no. 4, p. 044110, Jan. 2011.

[7] B. Drawert, M. J. Lawson, L. Petzold, and M. Khammash, "The diffusive finite state projection algorithm for efficient simulation of the stochastic reaction-diffusion master equation," *The Journal of Chemical Physics*, vol. 132, no. 7, p. 074101, 2010. [Online]. Available: https://doi.org/10.1063/1.3310809

[8] M. K. Roh, D. T. Gillespie, and L. R. Petzold, "State-dependent biasing method for importance sampling in the weighted stochastic simulation algorithm," *The Journal of Chemical Physics*, vol. 133, no. 17, p. 174106, Nov. 2010.

[9] M. Češka and J. Křetínský, "Semi-quantitative abstraction and analysis of chemical reaction networks," in *Computer Aided Verification*, I. Dillig and S. Tasiran, Eds. Cham: Springer International Publishing, 2019, pp. 475–496.

[10] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton, "Model-checking continuous-time Markov chains," *ACM Transactions on Computational Logic*, vol. 1, no. 1, pp. 162–170, Jul. 2000.

[11] M. Kwiatkowska, G. Norman, and D. Parker, *Stochastic Model Checking*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 220–270.

[12] I. Koch, "Petri Nets – A Mathematical Formalism to Analyze Chemical Reaction Networks," *Molecular Informatics*, vol. 29, no. 12, pp. 838–843, 2010.

[13] J. Leroux, "Polynomial Vector Addition Systems With States," in *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018), July 9-13, 2018, Prague, Czech Republic*, ser. LIPIcs, I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella, Eds., vol. 107. Prague, Czech Republic: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Jul. 2018, pp. 134:1–134:13. [Online]. Available: https://hal.science/hal-01711089

[14] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, "Modelling with Generalized Stochastic Petri Nets," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 2, p. 2, Aug. 1998.

[15] D. Angeli, P. De Leenheer, and E. D. Sontag, "A Petri net approach to the study of persistence in chemical reaction networks," Dec. 2007.

[16] W. Czerwiński, S. Lasota, R. Lazić, J. Leroux, and F. Mazowiecki, "Reachability in fixed dimension vector addition systems with states," May 2020.

[17] H. Kuwahara and I. Mura, "An efficient and exact stochastic simulation method to analyze rare events in biochemical systems," *The Journal of Chemical Physics*, vol. 129, no. 16, p. 165101, Oct. 2008.

[18] C. Jegourel, A. Legay, and S. Sedwards, "Cross-entropy optimisation of importance sampling parameters for statistical model checking," in *Proceedings of the 24th international conference on Computer Aided Verification*, ser. CAV'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 327–342.

[19] M. Roh, B. J. J. Daigle, D. T. Gillespie, and L. R. Petzold, "State-dependent doubly weighted stochastic simulation algorithm for automatic characterization of stochastic biochemical rare events," in *Journal of Chemical Physics*, vol. 135. American Institute of Physics, 2011.

[20] M. K. Roh and B. J. Daigle, "Sparse++: improved event-based stochastic parameter search," *BMC Systems Biology*, vol. 10, no. 1, p. 109, 2016. [Online]. Available: https://doi.org/10.1186/s12918-016-0367-z

[21] B. W. Zhang, D. Jasnow, and D. M. Zuckerman, "Efficient and verified simulation of a path ensemble for conformational change in a united-residue model of calmodulin," *Proceedings of the National Academy of Sciences*, vol. 104, no. 46, pp. 18 043–18 048, 2007. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.0706349104

[22] J. L. Adelman and M. Grabe, "Simulating rare events using a weighted ensemble-based string method," *The Journal of Chemical Physics*, vol. 138, no. 4, p. 044105, 2013. [Online]. Available: https://doi.org/10.1063/1.4773892

[23] R. M. Donovan, A. J. Sedgewick, J. R. Faeder, and D. M. Zuckerman, "Efficient stochastic simulation of chemical kinetics networks using a weighted ensemble of trajectories," *The Journal of Chemical Physics*, vol. 139, no. 11, p. 115105, Sep. 2013.

[24] D. M. Zuckerman and L. T. Chong, "Weighted ensemble simulation: Review of methodology, applications, and software." *Annu Rev Biophys*, vol. 46, pp. 43–57, May 2017.

[25] M. Okamoto, "Some inequalities relating to the partial sum of binomial probabilities," *Annals of the Institute of Statistical Mathematics*, vol. 10, no. 1, pp. 29–35, 1959. [Online]. Available: https://doi.org/10.1007/BF02883985

[26] A. Wald, "Sequential tests of statistical hypotheses," *The Annals of Mathematical Statistics*, vol. 16, no. 2, pp. 117–186, 1945. [Online]. Available: http://www.jstor.org/stable/2235829

[27] H. Kahn, "Random sampling (monte carlo) techniques in neutron attenuation problems–I." *Nucleonics*, vol. 6, no. 5, p. 27; passim, May 1950.

[28] H. Kahn and A. W. Marshall, "Methods of reducing sample size in monte carlo computations," *Journal of the Operations Research Society of America*, vol. 1, no. 5, pp. 263–278, 1953. [Online]. Available: https://doi.org/10.1287/opre.1.5.263

[29] H. Kahn and T. E. Harris, "Estimation of particle transmission by random sampling," *National Bureau of Standards applied mathematics series*, vol. 12, pp. 27–30, 1951.

[30] M. N. Rosenbluth and A. W. Rosenbluth, "Monte carlo calculation of the average extension of molecular chains," *The Journal of Chemical Physics*, vol. 23, no. 2, pp. 356–359, 1955. [Online]. Available: https://doi.org/10.1063/1.1741967

[31] M. Villen-Altamirano, J. Villen-Altamirano *et al.*, "Restart: a method for accelerating rare event simulations," *Queueing, Performance and Control in ATM (ITC-13)*, pp. 71–76, 1991.

[32] P. L'Ecuyer, F. Le Gland, P. Lezaud, and B. Tuffin, "Splitting Techniques," in *Rare Event Simulation Using Monte Carlo Methods*. John Wiley & Sons, Ltd, 2009, ch. 3, pp. 39–61.

[33] M. Villén-Altamirano and J. Villén-Altamirano, *The Rare Event Simulation Method RESTART: Efficiency Analysis and Guidelines for Its Application*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 509–547.

[34] C. E. Budde, P. R. D'Argenio, and A. Hartmanns, "Automated compositional importance splitting," *Science of Computer Programming*, vol. 174, pp. 90–108, Apr. 2019.

[35] J. Villén-Altamirano, "Restart vs splitting: A comparative study," *Performance Evaluation*, vol. 121-122, pp. 38–47, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166531616300839

[36] ——, "An improved variant of the rare event simulation method restart using prolonged retrials," *Operations Research Perspectives*, vol. 6, pp. 1–9, 2019. [Online]. Available: http://hdl.handle.net/10419/246387

[37] C. E. Budde and A. Hartmanns, "Replicating RESTART with prolonged retrials: An experimental report," in *Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings, Part II*, ser.

Lecture Notes in Computer Science, J. F. Groote and K. G. Larsen, Eds., vol. 12652. Springer, 2021, pp. 373–380. [Online]. Available: https://doi.org/10.1007/978-3-030-72013-1_21

[38] A. Hartmanns and H. Hermanns, "The Modest Toolset: An integrated environment for quantitative modelling and verification," in *TACAS*, ser. LNCS, E. Ábrahám and K. Havelund, Eds., vol. 8413. Springer, 2014, pp. 593–598.

[39] R. J. Lipton, "Reduction: A method of proving properties of parallel programs," *Commun. ACM*, vol. 18, no. 12, pp. 717–721, dec 1975. [Online]. Available: https://doi.org/10.1145/361227.361234

[40] A. Mazurkiewicz, "Trace theory," in *Petri Nets: Applications and Relationships to Other Models of Concurrency*, W. Brauer, W. Reisig, and G. Rozenberg, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 278–324.

[41] D. Peled, "All from one, one for all: on model checking using representatives," in *Computer Aided Verification*, C. Courcoubetis, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 409–423.

[42] P. Godefroid, "Using partial orders to improve automatic verification methods," in *Computer-Aided Verification*, E. M. Clarke and R. P. Kurshan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 176–185.

[43] A. Valmari, "Stubborn sets for reduced state space generation," in *Advances in Petri Nets 1990*, G. Rozenberg, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 491–515.

[44] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," in *Proceedings of the 23rd International Conference on Computer Aided Verification*, ser. CAV'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 585–591.

[45] C. Hensel, S. Junges, J.-P. Katoen, T. Quatmann, and M. Volk, "The probabilistic model checker Storm," *International Journal on Software Tools for Technology Transfer*, vol. 24, no. 4, pp. 589–610, Aug. 2022.

[46] D. B. Kearns and R. Losick, "Cell population heterogeneity during growth of bacillus subtilis." *Genes & development*, vol. 19 24, pp. 3083–94, 2005.

[47] T. Dijk and J. Pol, "Sylvan: Multi-core framework for decision diagrams," *Int. J. Softw. Tools Technol. Transf.*, vol. 19, no. 6, pp. 675–696, nov 2017. [Online]. Available: https://doi.org/10.1007/s10009-016-0433-2

[48] R. Roberts, T. Neupane, L. Buecherl, C. J. Myers, and Z. Zhang, "STAMINA 2.0: Improving scalability of infinite-state stochastic model checking," in *Verification, Model Checking, and Abstract Interpretation*, B. Finkbeiner and T. Wies, Eds. Cham: Springer International Publishing, 2022, pp. 319–331.