STAMINA: STochastic Approximate Model-checker for INfinite-state Analysis

Thakur Neupane[1][0000−0002−1870−4079], Chris J. Myers[2][0000−0002−8762−8444], Curtis Madsen[3][0000−0002−0254−0364], Hao Zheng[4][0000−0002−8627−0591], and Zhen Zhang[1][0000−0002−8269−9489]

[1] Utah State University, Logan, UT USA, thakur.neupane@aggiemail.usu.edu, zhen.zhang@usu.edu
[2] University of Utah, Salt Lake City, UT USA, myers@ece.utah.edu
[3] Boston University, Boston, MA USA, ckmadsen@bu.edu
[4] University of South Florida, Tampa, FL USA, haozheng@usf.edu

Abstract. Stochastic model checking is a technique for analyzing systems that possess probabilistic characteristics. However, its scalability is limited as probabilistic models of real-world applications typically have very large or infinite state space. This paper presents a new infinite state CTMC model checker, STAMINA, with improved scalability. It uses a novel state space approximation method to reduce large and possibly infinite state CTMC models to finite state representations that are amenable to existing stochastic model checkers. It is integrated with a new property-guided state expansion approach that improves the analysis accuracy. Demonstration of the tool on several benchmark examples shows promising results in terms of analysis efficiency and accuracy compared with a state-of-the-art CTMC model checker that deploys a similar approximation method.

Keywords: Stochastic model checking · infinite-state · Markov chains

1 Introduction

Stochastic model checking is a formal method that designers and engineers can use to determine the likelihood of *safety* and *liveness* properties. Checking properties using numerical model checking techniques requires enumerating the state space of the system to determine the probability that the system is in any given state at a desired time [18]. Real-world applications often have very large or even infinite state spaces.

Numerous state representation, reduction, and approximation methods have been proposed. Symbolic model checking based on *multi-terminal binary decision diagrams* (MTBDDs) [24] has achieved success in representing large *Markov Decision Process* (MDP) models with a few distinct probabilistic choices at each state, e.g., the shared coin protocol [3]. MTBDDs, however, are often inefficient for models with many different and distinct probability/rate values due to the inefficient representation of solution vectors. *Continuous-time Markov chain* (CTMC) models, whose state transition rate is a function of state variables, generally contain many distinct rate values. As a result, symbolic model checkers can run out of memory while verifying a typical CTMC model

with as few as 73,000 states [24]. State reduction techniques, such as bisimulation minimization [15, 7, 8], abstraction [21, 15, 6, 13], symmetry reduction [17, 5], and partial order reduction [9] have been mainly extended to discrete-time, finite-state probabilistic systems. The three-valued abstraction [15] can reduce large, finite-state CTMCs. It may, however, provide inconclusive verification results due to abstraction.

To the best of our knowledge, only a few tools can analyze infinite-state probabilistic models, namely, STAR [20] and INFAMY [10]. The STAR tool primarily analyzes biochemical reaction networks. It approximates solutions to the *chemical master equation* (CME) using the *method of conditional moments* (MCM) [12] that combines moment-based and state-based representations of probability distributions. This hybrid approach represents species with low concentrations using a discrete stochastic description and numerically integrates a small master equation using the fourth order Runge-Kutta method over a small time interval [2]; and solves a system of conditional moment equations for higher concentration species, conditioned on the low concentration species. This method has been optimized to drop unlikely states and add likely states on-the-fly. STAR relies on a well-structured underlying Markov process with small sensitivity on the transient distribution. Also, it mainly reports state reachability probabilities, instead of checking a given probabilistic property. INFAMY is a truncation-based approach that explores the model's state space up to a certain finite depth k. The truncated state space still grows exponentially with respect to exploration depth. Starting from the initial state, breadth-first state search is performed up to a certain finite depth. The error probability computed during the model checking depends on the depth of state exploration. Therefore, higher exploration depth generally incurs lower error probability.

This paper presents a new infinite-state stochastic model checker, *STochastic Approximate Model-checker for INfinite-state Analysis* (STAMINA). Our tool also takes a truncation-based approach. In particular, it maintains a probability estimate of each path being explored in the state space, and when the currently explored path probability drops below a specified threshold, it halts exploration of this path. All transitions exiting this state are redirected to an absorbing state. After all paths have been explored or truncated, transient Markov chain analysis is applied to determine the probability of a transient property of interest specified using *Continuous Stochastic Logic* (CSL) [4]. The calculated probability forms a lower bound on the probability, while the upper bound also includes the probability of the absorbing state. The actual probability of the CSL property is guaranteed to be within this range. An initial version of our tool and preliminary results are reported in [23]. Since that paper, our tool has been tightly integrated within the PRISM model checker [19] to improve performance, and we have also developed a new property-guided state expansion technique to expand the state space to tighten the reported probability range incrementally. This paper reports our results, which show significant improvement on both efficiency and verification accuracy over several non-trivial case studies from various application domains.

2 STAMINA

Figure 1 presents the architecture of STAMINA. Based on a user-specified probability threshold $\varkappa$ (kappa), it first constructs a finite-state CTMC model $\mathcal{C}\rfloor_{\varkappa}$ from the original

infinite-state CTMC model $\mathcal{C}$ using the state space approximation method presented in Section 2.1. $\mathcal{C}\rfloor_{\varkappa}$ is then checked using the PRISM explicit-state model checker against a given CSL property $\mathsf{P}_{\sim p}(\phi)$, where $\sim \in \{<, >, \leqslant, \geqslant\}$ and $p \in [0, 1]$ (for cases where it is desired that a predicate be true within a certain probability bound) or $\mathsf{P}_{=?}(\phi)$ (for cases where it is desired that the exact probability of the predicate being true be calculated). Lower- and upper-bound probabilities that ϕ holds, namely, P_{min} and P_{max}, are then obtained, and their difference, i.e., $(P_{max} - P_{min})$, is the probability accumulated in the absorbing state $\mathbf{x}_{abs}$ which abstracts all the states not included in the current state space. If $p \in [P_{min}, P_{max}]$, it is not known whether $\mathsf{P}_{\sim p}(\phi)$ holds. If exact probability is of interest and the probability range is larger than the user-defined precision ϵ, i.e., $(P_{max} - P_{min}) > \epsilon$, then the method does not give a meaningful result.

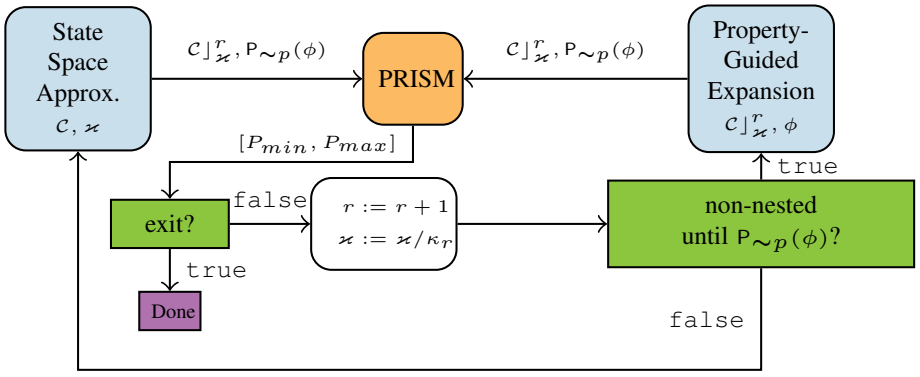

Fig. 1: Architecture of STAMINA.

For an inconclusive verification result from the previous step, STAMINA applies a property-guided approach, described in Section 2.2, to further expand $\mathcal{C}\rfloor_{\varkappa}$, provided $\mathsf{P}_{\sim p}(\phi)$ is a non-nested "until" formula; otherwise, it uses the previous method to expand the state space. Note that $\varkappa$ also drops by the reduction factor κ_r to enable states that were previously ignored due to a low probability estimate to be included in the current state expansion. The expanded CTMC model $\mathcal{C}\rfloor_{\varkappa}$ is then checked to obtain a new probability bound $[P_{min}, P_{max}]$. This iterative process repeats until one of the following conditions holds: (1) the target probability p falls outside the probability bound $[P_{min}, P_{max}]$, (2) the probability bound is sufficiently small, i.e, $(P_{max} - P_{min}) < \epsilon$, or (3) a maximal number of iterations N has been reached ($r \geqslant N$).

2.1 State Space Approximation

The state space approximation method [23] truncates the state space based on a user-specified reachability threshold $\varkappa$. During state exploration, the reachability-value function, $\hat{\kappa} : \mathbf{X} \to \mathbb{R}^+$, estimates the probability of reaching a state on-the-fly, and is compared against $\varkappa$ to determine whether the state search should terminate. Only states with a higher reachability-value than the reachability threshold are explored further.

Figure 2 illustrates the standard *breadth first search* (BFS) state exploration for reachability threshold $\varkappa = 0.25$. It starts from the initial state whose reachability-value i.e., $\hat{\kappa}(\mathbf{x}_0)$, is initialized to 1.0 as shown in Figure 2a. In the first step, two new states $\mathbf{x}_1$ and $\mathbf{x}_4$ are generated and their reachability-values are 0.8 and 0.2, respectively, as shown in Figure 2b. The reachability-value in $\mathbf{x}_0$ is distributed to its successor states, based on the probability of outgoing transitions from $\mathbf{x}_0$ to its successor state. For the next step, only state $\mathbf{x}_1$ is scheduled for exploration because $\hat{\kappa}(\mathbf{x}_1) \geq \varkappa$. Note that the transition from $\mathbf{x}_4$ to $\mathbf{x}_0$ is executed because $\mathbf{x}_0$ is already in the explored set. Expanding $\mathbf{x}_1$ leads to two new states, namely $\mathbf{x}_2$ and $\mathbf{x}_5$ as shown in Figure 2c, from which only $\mathbf{x}_5$ is scheduled for further exploration. This leads to the generation of $\mathbf{x}_6$ and $\mathbf{x}_9$ shown in Figure 2d. State exploration terminates after Figure 2e since both newly generated states have reachability-values less than 0.25. States $\mathbf{x}_2$, $\mathbf{x}_4$, $\mathbf{x}_6$ and $\mathbf{x}_9$ are marked as terminal states. During state exploration, the reachability-value update is performed every time a new incoming path is added to a state because a new incoming path can add its contribution to the state, potentially bringing the reachability-value above $\varkappa$, which in turn changes a terminal state to be non-terminal. When the truncated CTMC model $\mathcal{C}\rfloor_{\varkappa}$ is analyzed, it introduces some error in the probability value of the property under verification, because of leakage the probability (i.e., cumulative path probabilities of reaching states not included in the explored state space) during the CTMC analysis. To account for probability loss, an abstract absorbing state $\mathbf{x}_{abs}$ is created as the sole successor state for all terminal states on each truncated path. Figure 2e shows the addition of the absorbing state.

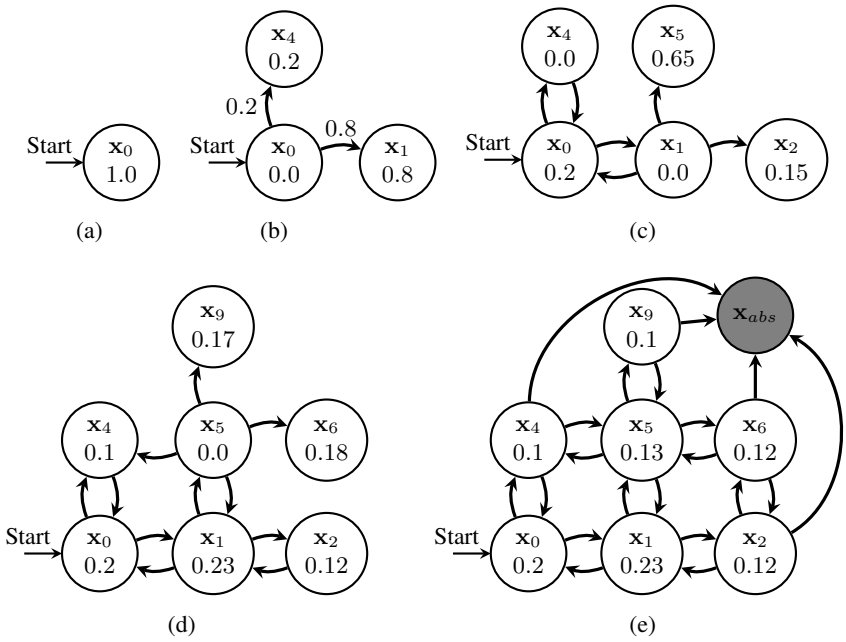

Fig. 2: State space approximation.

2.2 Property Based State Space Exploration

This paper introduces a property-guided state expansion method, in order to efficiently obtain a tightened probability bound. Since all non-nested CSL path formulas ϕ (except those containing the "next" operator) derive from the "until" formula, $\Phi\ \mathcal{U}^I\ \Psi$, construction of the set of terminal states for further expansion boils down to eliminating states that are known to satisfy or dissatisfy $\Phi\ \mathcal{U}\ \Psi$. Given a state graph, a path starting from the initial state can never satisfy $\Phi\ \mathcal{U}\ \Psi$, if it includes a state satisfying $\neg\Phi \wedge \neg\Psi$. Also, if a path includes a state satisfying Ψ, satisfiability of $\Phi\ \mathcal{U}\ \Psi$ can be determined without further expanding this path beyond the first Ψ-state. Our property-guided state space expansion method identifies the path prefixes, from which satisfiability of $\Phi\ \mathcal{U}\ \Psi$ can be determined, and shortens them by making the last state of each prefix absorbing based on the satisfiability of $(\neg\Phi \vee \Psi)$. Only the non-absorbing states whose path probability is greater than the state probability estimate threshold $\varkappa$ are expanded further. For detailed algorithms of STAMINA, readers are encouraged to read [22].

3 Results

This section presents results on the following case studies to illustrate the accuracy and efficiency of STAMINA: a genetic toggle switch [21, 23]; the following examples from the PRISM benchmark suite [16]: grid world robot, cyclic server polling system, and tandem queuing network; and the Jackson queuing network from INFAMY case studies [1]. All case studies are evaluated on STAMINA and INFAMY, except the genetic toggle switch [5]. Experiments are performed on a 3.2 GHz AMD Debian Linux PC with six cores and 64 GB of RAM. For all experiments, the maximal number of iterations N is set to 10, and the reduction factor κ_r is set to 1000. All experiments terminate due to $(P_{max} - P_{min}) < \epsilon$, where $\epsilon = 10^{-3}$, before they reach N. STAMINA is freely available at: https://github.com/formal-verification-research/stamina.

We compare the runtime, state size, and verification results between STAMINA and INFAMY using the same precision $\epsilon = 10^{-3}$. For all tables in this section, column $\varkappa$ reports the probability estimate threshold used to terminate state generation in STAMINA. The state space size is listed in column $|\mathcal{G}|(K)$, where K indicates one thousand states. Column $T(C/A)$ reports the state space construction (C) and analysis (A) time in seconds. For STAMINA, the total construction and analysis time is the cumulation of runtime for all $\varkappa$ values for a model configuration. Columns P_{min} and P_{max} list the lower and upper probability bounds for the property under verification, and column P lists the single probability value (within the precision ϵ) reported by INFAMY. We select the best runtime reported by three configurations of INFAMY. The improvement in state size (column $|\mathcal{G}|(X)$) and runtime (column $T(\%)$) are represented by the ratio of state count generated by INFAMY to that of STAMINA (higher is better) and percentage improvement in runtime (higher is better), respectively.

Genetic toggle switch. The genetic toggle switch circuit model has two inputs, aTc and IPTG. It can be set to the OFF state by supplying it with aTc and can be set to the ON state by supplying it with IPTG [21]. Two important properties for a toggle

[5] INFAMY generates arithmetic errors on the genetic toggle switch model.

switch circuit are the response time and the failure rate. The first experiments set IPTG to 100 to measure the toggle switch's response time. It should be noted that the input value of 100 molecules of IPTG is chosen to ensure that the circuit switches to the ON state. The later experiments initialize IPTG to 0 to compute the failure rate, i.e., the probability that the circuit changes state erroneously within a cell cycle of $2,100$ seconds (an approximation of the cell cycle in *E. coli* [25]). Initially, LacI is set to 60 and TetR is set to 0 for both experiments. The CSL property used for both experiments, $\mathsf{P}_{=?}\ [true\ \mathcal{U}^{\leqslant 2100}\ (TetR > 40 \wedge LacI < 20)]$, describes the probability of the circuit switching to the ON state within a cell cycle of $2,100$ seconds. The ON state is defined as LacI below 20 and TetR above 40 molecules.

Table 1: Verification results for genetic toggle switch.

$IPTG$	STAMINA					
	$\varkappa$	$\lvert\mathcal{G}\rvert$	$T(C/A)$	P_{min}	P_{max}	Remark
100	10^{-3}	$1,127$	$0.15/0.67$	0.000000	0.999671	Property Guided
	10^{-6}	$4,461$	$0.43/2.84$	0.966947	0.992908	
	10^{-9}	$7,163$	$0.43/5.25$	0.991738	0.991797	
100	10^{-6}	$5,171$	$0.17/1.90$	0.977942	0.992850	Property
	10^{-9}	$8,908$	$0.18/3.74$	0.991739	0.991797	Agnostic
0	10^{-3}	182	$0.05/0.07$	0.000000	0.697500	Property Guided
	10^{-6}	$2,438$	$0.16/1.08$	0.008814	0.060424	
	10^{-9}	$4,284$	$0.09/2.12$	0.013097	0.013609	
0	10^{-6}	$2,446$	$0.16/1.05$	0.009169	0.060420	Property
	10^{-9}	$4,820$	$0.13/2.13$	0.013097	0.013609	Agnostic

The property-agnostic state space is generated with the probability estimate threshold $\varkappa = 10^{-3}$. Table 1 shows large probability bounds: $[0, 0.999671]$ for IPTG $= 100$ and $[0, 0.6975]$ for IPTG $= 0$. It is obvious that they are significantly inaccurate w.r.t. the precision ϵ of 10^{-3}. The $\varkappa$ is then reduced to 10^{-6} and state generation switches to the property-guided state expansion mode, where the CSL property is used to guide state exploration, based on the previous state graph. Each state expansion step reduces the $\varkappa$ value by a factor of $\kappa_r = 1000$. To measure the effectiveness of the property-guided state expansion approach, we compare state graphs generated with and without the property-guided state expansion, as indicated by the "property agnostic" and "property guided" rows in the table. Property-guided state expansion reduces the size of the state space without losing the analysis precision for the same value of $\varkappa$. Specifically, the state expansion approach reduces the state space by almost 20% for the response rate experiment.

Robot World. This case study considers a robot moving in an n-by-n grid and a janitor moving in a larger grid Kn-by-Kn, where the constant K is used to significantly scale up the state space. The robot starts from the bottom left corner to reach the top right corner. The janitor moves around randomly. Either the robot or janitor can occupy one grid location at any given time. The robot also randomly communicates with the base station. The property of interest is the probability that the robot reaches the

top right corner within 100 time units while periodically communicating with the base station, encoded as $\mathsf{P}_{=?}$ [$(\mathsf{P}_{\geqslant 0.5}$ [$true\ \mathcal{U}^{\leqslant 7}\ communicate$]) $\mathcal{U}^{\leqslant 100}\ goal$].

Table 2 provides a comparison of results for $K = 1024, 64$ and $n = 64, 32$. For smaller grid size i.e, 32-by-32, the robot can reach the goal with a high probability of 97.56%. Where as for a larger value of $n = 64$ and $K = 64$, the robot is not able to reach the goal with considerable probability. STAMINA generates precise results that are similar to INFAMY, while exploring less than half of states with shorter runtime.

Table 2: Comparison between STAMINA and INFAMY.

Model	Params	STAMINA				INFAMY			Improvement	
		$\lvert\mathcal{G}\rvert$ (K)	T (C/A)	P_{min}	P_{max}	$\lvert\mathcal{G}\rvert$ (K)	T (C/A)	P	$\lvert\mathcal{G}\rvert$ (X)	T $(\%)$
Robot (n/K)	32/ 64	696	41/ 279	0.975	0.975	1,591	492/ 18	0.975	2.3	37.3
	32/ 1024	696	41/ 258	0.975	0.975	1,591	501/ 18	0.975	2.3	42.4
	64/ 64	2,273	135/ 669	1.46e-4	1.68e-4	5,088	1,625/ 53	1.5e-4	2.2	52.1
	64/ 1024	2,273	132/ 621	1.46e-4	1.68e-4	5,088	1,625/ 53	1.5e-4	2.2	55.2
Jackson (N/λ)	4/ 5	201	22/ 51	0.865	0.865	635	109/ 5	0.865	3.2	36.1
	5/ 5	2,539	990/ 996	0.819	0.819	7,029	1668/ 108	0.819	2.8	−11.8
Polling (N)	12	19	3/ 21	1.0	1.0	74	1/ 2	1.0	3.9	−732.2
	16	57	18/ 70	1.0	1.0	1,573	5/ 54	1.0	27.6	−48.2
	20	113	30/ 77	1.0	1.0	31,457	151/ 1347	1.0	278.4	92.9
Tandem (c)	2047	33	1/ 41	0.498	0.498	2,392	3/ 38	0.498	72.5	−1.4
	4095	66	1/ 141	0.499	0.499	9,216	11/ 265	0.499	139.6	48.7

Jackson Queuing Network. A Jackson queuing network consists of N interconnected nodes (queues) with infinite queue capacity. Initially, all queues are considered empty. Each station is connected to a single server which distributes the arrived jobs to different stations. Customers arrive as a Poisson stream with intensity λ for N queues. The model is taken from [14, 11]. We compute the probability that, within 10 time units, the first queue has more that 3 jobs and the second queue has more than 5 jobs, given by $\mathsf{P}_{=?}$ [$true\ \mathcal{U}^{\leqslant 10}$ $(jobs\_1 \geqslant 4\ \wedge\ jobs\_2 \geqslant 6)$].

Table 2 summarizes the results for this model. STAMINA uses roughly equal time to construct and analyze the model for $N = 5$, whereas INFAMY takes significantly longer to construct the state space, making it slower in overall runtime. For $N = 4$, STAMINA is faster in generating verification results In both configurations, STAMINA only explores approximately one third of the states explored by INFAMY.

Cyclic Server Polling System. This case study is based on a cyclic server attending N stations. We consider the probability that station one is polled within 10 time units, $\mathsf{P}_{=?}\ [\ true\ \mathcal{U}^{\leqslant 10}\ station1\_polled\ ]$. Table 2 summarizes the verification results for $N = 12, 16, 20$. The probability of station one being polled within 10 seconds is 1.0 for all configurations. Similar to previous case studies, STAMINA explores significantly smaller state space. The advantage of STAMINA in terms of runtime starts to manifest as the size of model (and hence the state space size) grows.

Tandem Queuing Network. A tandem queuing network is the simplest interconnected queuing network of two finite capacity (c) queues with one server each [19]. Customers join the first queue and enter the second queue immediately after completing the service. This paper considers the probability that the first queue becomes full in 0.25 time units, depicted by the CSL property $\mathsf{P}_{=?}\ [\ true\, \mathcal{U}^{\leqslant 0.25}\ queue1\_full\ ]$.

As seen in Table 2, there is almost fifty percent probability that the first queue is full in 0.25 seconds irrespective of the queue capacity. As in the polling server, STAMINA explores significantly smaller state space. The runtime is similar for model with smaller queue capacity ($c = 2047$). But the runtime improves as the queue capacity is increased.

4 Conclusions

This paper presents an infinite-state stochastic model checker, STAMINA, that uses path probability estimates to generate states with high probability and truncate unlikely states based on a specified threshold. Initial state construction is property agnostic, and the state space is used for stochastic model checking of a given CSL property. The calculated probability forms a lower and upper bound on the probability for the CSL property, which is guaranteed to include the actual probability. Next, if finer precision of the probability bound is required, it uses a property-guided state expansion technique to explore states to tighten the reported probability range incrementally. Implementation of STAMINA is built on top of the PRISM model checker with tight integration to its API. Performance and accuracy evaluation is performed on case studies taken from various application domains, and shows significant improvement over the state-of-art infinite-state stochastic model checker INFAMY. For future work, we plan to investigate methods to determine the reduction factor on-the-fly based on the probability bound. Another direction is to investigate heuristics to further improve the property-guided state expansion, as well as, techniques to dynamically remove unlikely states.

Acknowledgment

Chris Myers is supported by the National Science Foundation under CCF-1748200. 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 NSF.

References

1. https://depend.cs.uni-saarland.de/tools/infamy/casestudies/
2. Andreychenko, A., Mikeev, L., Spieler, D., Wolf, V.: Parameter identification for markov models of biochemical reactions. In: Gopalakrishnan, G., Qadeer, S. (eds.) Computer Aided Verification. pp. 83–98. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
3. Aspnes, J., Herlihy, M.: Fast randomized consensus using shared memory. J. Algorithms **11**(3), 441–461 (Sep 1990)
4. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.: Model-checking continuous-time markov chains. ACM Trans. Comput. Logic **1**(1), 162–170 (Jul 2000)
5. Donaldson, A.F., Miller, A.: Symmetry reduction for probabilistic model checking using generic representatives. In: Proceedings of the 4th International Conference on Automated Technology for Verification and Analysis. pp. 9–23. ATVA'06, Springer-Verlag, Berlin, Heidelberg (2006)
6. Fecher, H., Leucker, M., Wolf, V.: Don't know in probabilistic systems. In: Proceedings of the 13th International Conference on Model Checking Software. pp. 71–88. SPIN'06, Springer-Verlag, Berlin, Heidelberg (2006)
7. Fisler, K., Vardi, M.Y.: Bisimulation and model checking. In: Proceedings of the 10th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods. pp. 338–341. CHARME '99, Springer-Verlag, London, UK, UK (1999)
8. Fisler, K., Vardi, M.Y.: Bisimulation minimization and symbolic model checking. Form. Methods Syst. Des. **21**(1), 39–78 (Jul 2002)
9. Groesser, M., Baier, C.: Partial order reduction for markov decision processes: A survey. In: Proceedings of the 4th International Conference on Formal Methods for Components and Objects. pp. 408–427. FMCO'05, Springer-Verlag, Berlin, Heidelberg (2006)
10. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: Infamy: An infinite-state markov model checker. In: Proceedings of the 21st International Conference on Computer Aided Verification. pp. 641–647. CAV '09, Springer-Verlag, Berlin, Heidelberg (2009)
11. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: INFAMY: An infinite-state markov model checker. In: CAV. pp. 641–647 (2009)
12. Hasenauer, J., Wolf, V., Kazeroonian, A., Theis, F.J.: Method of conditional moments (mcm) for the chemical master equation. Journal of Mathematical Biology **69**(3), 687–735 (Sep 2014). https://doi.org/10.1007/s00285-013-0711-5, https://doi.org/10.1007/s00285-013-0711-5
13. Hermanns, H., Wachter, B., Zhang, L.: Probabilistic CEGAR. In: Proceedings of the 20th International Conference on Computer Aided Verification. pp. 162–175. CAV '08, Springer-Verlag, Berlin, Heidelberg (2008)
14. Jackson, J.: Networks of Waiting Lines. Operations Research **5**, 518–521 (1957)
15. Katoen, J.P., Klink, D., Leucker, M., Wolf, V.: Three-valued abstraction for continuous-time markov chains. In: Proceedings of the 19th International Conference on Computer Aided Verification. pp. 311–324. CAV'07, Springer-Verlag, Berlin, Heidelberg (2007)
16. Kwiatkowsa, M., Norman, G., Parker, D.: The prism benchmark suite. In: Quantitative Evaluation of Systems, International Conference on(QEST). vol. 00, pp. 203–204 (09 2012). https://doi.org/10.1109/QEST.2012.14, doi.ieeecomputersociety.org/10.1109/QEST.2012.14
17. Kwiatkowska, M., Norman, G., Parker, D.: Symmetry reduction for probabilistic model checking. In: Proceedings of the 18th International Conference on Computer Aided Verification. pp. 234–248. CAV'06, Springer-Verlag, Berlin, Heidelberg (2006)
18. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic Model Checking, pp. 220–270. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

19. Kwiatkowska, M., Norman, G., Parker, D.: Prism 4.0: Verification of probabilistic real-time systems. In: Proceedings of the 23rd International Conference on Computer Aided Verification. pp. 585–591. CAV'11, Springer-Verlag, Berlin, Heidelberg (2011)
20. Lapin, M., Mikeev, L., Wolf, V.: Shave: Stochastic hybrid analysis of markov population models. In: Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control. pp. 311–312. HSCC '11, ACM, New York, NY, USA (2011)
21. Madsen, C., Zhang, Z., Roehner, N., Winstead, C., Myers, C.: Stochastic model checking of genetic circuits. J. Emerg. Technol. Comput. Syst. **11**(3), 23:1–23:21 (Dec 2014). https://doi.org/10.1145/2644817, http://doi.acm.org/10.1145/2644817
22. Neupane, T.: STAMINA: STochastic Approximate Model-checker for INfinite-state Analysis. Master's thesis, Utah State University (May 2019)
23. Neupane, T., Zhang, Z., Madsen, C., Zheng, H., Myers, C.J.: Approximation Techniques for Stochastic Analysis of Biological Systems. In: Pietro Lió, P.Z. (ed.) Automated Reasoning for Systems Biology and Medicine, Computational Biology, vol. 30, chap. 12, p. 480. Springer International Publishing, 1 edn. (Sep 2019). https://doi.org/10.1007/978-3-030-17297-8_12, https://doi.org/10.1007/978-3-030-17297-8_12
24. Parker, D.: Implementation of Symbolic Model Checking for Probabilistic Systems. Ph.D. thesis, University of Birmingham (2002)
25. Zheng, H., Ho, P.Y., Jiang, M., Tang, B., Liu, W., Li, D., Yu, X., Kleckner, N.E., Amir, A., Liu, C.: Interrogating the escherichia coli cell cycle by cell dimension perturbations. Proceedings of the National Academy of Sciences **113**(52), 15000–15005 (2016)