Tackling Scalability for Transient Reachability Analysis of Chemical Reaction Networks

Landon Taylor
Electrical and Computer Engineering
Utah State University
Logan, Utah, USA
0000-0002-4071-3625

Zhen Zhang
Electrical and Computer Engineering
Utah State University
Logan, Utah, USA
0000-0002-8269-9489

***Abstract*—Probabilistic notions of correctness are absolutely essential for stochastic systems operating in safety- and mission-critical settings. Among these noisy, stochastic systems are Chemical Reaction Networks (CRNs), which model the interactions within biochemical systems. A CRN induces a Continuous-Time Markov Chain (CTMC), in which a model's state changes based on a transition rate function. Probabilistic Model Checking (PMC) is a crucial tool for making guarantees for CTMCs. However, these systems often involve rare events that can lead to critical failures. Currently, PMC faces state explosion in models with rare events, making quantitative analysis computationally prohibitive. Our work addresses this challenge by focusing on three key objectives: (1) scaling CTMC transient reachability analysis through innovative pre-processing techniques that reduce the explicit state space; (2) extending symbolic verification methods to CTMCs to provide regional probability estimates without the need for exhaustive state enumeration; and (3) developing a provably-correct probabilistic verification tool in Rust. Preliminary results in these objectives show the viability of our approach.**

A. Motivation

Probabilistic systems, including *Chemical Reaction Networks* (CRNs) and other stochastic *Vector Addition Systems* (VASes), often induce *Continuous-Time Markov Chain* (CTMC) models. Formal analysis of CTMCs is a focal point in the study of safety-critical systems, particularly due to their relevance in synthetic biological designs [1]–[3], communication systems [4], dynamic power management [5], autonomous vehicle control [6], and other critical processes [7]. CTMCs can fail in catastrophic ways due to deeply buried design errors. The probability of CTMC time-bounded transient reachability is of significant interest for synthetic biological models, which often pose a formidable challenge to existing tools [1], [2], [8], [9]. Further complicating analysis, safety properties in these models are often extremely unlikely to occur (i.e., *rare events*). Though they are extremely unlikely, rare events are of great interest in CRNs, as they represent undesirable behavior that can lead to severe (i.e., potentially pathological) consequences. Their operating environments are extremely noisy, and irrelevant inputs exponentially increase the challenge of analysis.

The gold standard for evaluating CTMCs is *Probabilistic Model Checking* (PMC), which provides provable guarantees of the probability of reaching an event. Informally, CTMC *transient analysis* can answer the question: "What is the probability that event X occurs within Y seconds?" In a CTMC, PMC relies on a calculation of a model's *explicit* state space. Unfortunately, CMTC transient analysis often suffers from state explosion due to its need to explicitly enumerate the intractible number of states often present in complex models. At present, CTMC transient analysis faces two primary challenges: (1) state explosion makes transient CTMC analysis computationally prohibitive, preventing its use by the industry professionals who need it most; and (2) it is challenging for non-experts to understand the kinetics in a model for decision-making purposes without performing an in-depth analysis. To this end, our work targets state explosion and enhances model intuition for real-world industry professionals with the following objectives, described in the remaining sections:

1) Scale CTMC transient reachability analysis using pre-processing methods.
2) Extend existing symbolic verification approaches to reason about CTMCs.
3) Implement provably-correct CTMC transient reachability analysis tooling in Rust.

B. Objective 1: CTMC Pre-Processing

In many models, symbolic methods are used to reduce the burden of verification. However, CTMC transient analysis requires *explicit* finite state space construction. That is, while some models allow evaluation of their high-level mathematical description, PMC tools must store the *entire* state space explicitly. Thus, a significant amount of pre-processing effort is required to limit the number of states in a model's state space. This process is primarily a manual, labor-intensive process that requires extensive background knowledge of both PMC and the model's subject domain. This can render PMC inaccessible to the domain experts who need it most. Thus, this outcome targets two areas: automatically reducing a state space size through smart variable bounding, and constructing a highly-efficient partial state space from scratch.

Variable Bounding. For CTMC transient analysis, infinite-state models must be converted to finite-state models. Without a good intuition about a model, a user often resorts to bounding variables by unreasonably large over-approximations of their feasible values. This quickly leads to state explosion,

as the state space grows exponentially as variable bound ranges expand. Our preliminary work uses *Bounded Model Checking* (BMC) to find precise variable bounds. BMC is an efficient, probability-agnostic method of exploring a model's state space to find states of interest—in our case, evidence for variable bounds. This method has proven effective, generating variable bounds for complex models within minutes. To further improve the efficiency of variable bounding, we will use more advanced algorithms to generate bounds. These methods include *Property-Directed Reachability* (PDR), which iteratively constructs symbolic over-approximations of reachable states to efficiently generate proofs of state reachability.

Partial State Space Construction Our preliminary work constructs a partial state space for transient CTMC analysis [10], [11]. A main challenge in partial state space construction is the *efficiency* of the state space. That is, because the partial state space does not encapsulate *all* behavior, it is desirable to include regions that eventually lead to an interesting behavior with a reasonably high probability. The problem of creating a perfectly efficient state space is as hard as the original CTMC transient analyis, so state space construction must rely on heuristics. For example, RAGTIMER [11] constructs a collection of useful states by first analyzing a probability-agnostic model, and STAMINA [12], a PMC tool produced by our research group, eliminates states that are unlikely to be reached within a time bound.

We propose enhancing partial state space construction through a number of heuristics. First, we use Reinforcement Learning (RL) methods to automatically generate a state space, rewarding states that have a higher probability of reaching a target. Preliminary results with RL are promising, and fine-tuning is required to see the full benefit of this approach. Further, we propose developing a selection of heuristics optimized for problem domains of interest, including CRNs.

C. Objective 2: Symbolic Approaches for CTMCs

Many approaches have extended existing qualitative and symbolic verification methods to *Discrete-Time* Markov Chains and Markov Decision Processes. For example, PrIC3 extends qualitative PDR into the quantitative domain for Markov Decision Processes [13], and Caesar extends program proving to the discrete-time probabilistic domain [14]. However, due to the challenges of CTMC transient reachability analysis, symbolic verification methods have not been extended to CTMCs.

While the gold standard for CTMC transient analysis is the precise probability found through PMC, the need to enumerate an explicit state space makes this approach impossible for many models. In this case, we propose it is valuable to provide probability bounds or an estimated probability.

PDR's scalability has caused it to become increasingly popular, and efforts have recently been made to improve its efficiency and broaden its use cases [13]. We propose extending PDR to reason about CTMC models, particularly by providing regional probability estimates. While traditional PDR guarantees a strict inability to leave a given region, we propose extending it to reason about the *probability* of leaving or entering a region. While this method does not provide a precise probability calculation for transient CTMC analysis, we believe it will be highly efficient at providing a reasonable probability estimate. Further, it will be able to provide accurate upper and lower bounds for the transient reachability probability. While there is no silver bullet for CTMC transient analysis, this approach can provide *some* information for models that are currently *impossible* to analyze due to their computational complexity. We propose that this approach can provide a reasonable tradeoff: a fast runtime at the expense of probabilistic precision.

D. Objective 3: Provably-Correct Tooling in Rust

Verification by automated methods makes a critical assumption: the automated method is implemented correctly, so the verification result is trustworthy. During verification, an engineer moves the burden of trust from their own system to the verification tool. If a verification tool has flaws, it may lead to the introduction of bugs into safety-critical systems. While it is nearly impossible to prove that every aspect of a system from high-level code to microchip is correct, it is imperative to provide reasonable assurance of correctness.

Rust, a memory-safe programming language, is gaining traction for its *borrow checker*, which runs at compile time and adds no overhead to the program binary. Rust is able to guarantee memory safety and other desirable properties without slowing program execution, making it an extremely attractive language for safety-critical tool development. Further, many proof tools extend Rust's correctness by introducing deductive program proving techniques [15]. Essentially, these tools allow a developer to verify that their code behaves exactly as specified using pre- and post-conditions, loop invariants, and other formal techniques. Because these tools use static verification methods, they provide trust without additional overhead to the compiled program.

We are developing a production-ready probabilistic verification tool suite using provably-correct Rust. This tool suite will include verified implementations of our existing CTMC analysis tools: RAGTIMER [11], STAMINA [12], and Wayfarer (a vector-based state space construction tool). We are working with visualization experts to design a user-friendly interface for users without a background in verification.

E. Conclusion

CMTC transient analysis is a challenging problem. To that end, we are improving scalability using pre-processing methods, extending existing symbolic verification approaches to reason about CTMCs, and implementing provably-correct CTMC transient reachability analysis tooling in Rust. These objectives can greatly benefit not only the formal verification community, but domain experts who need to prove the correctness of their systems. Improving the accessibility of CTMC analysis can enable the development of safer systems and prevent design errors, greatly reducing the potential for harm from probabilistic systems.

REFERENCES

[1] 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, Oct. 2021.

[2] M. D. Friedenberg, A. Lita, M. R. Gilbert, M. Larion, and O. Celiku, "Probabilistic model checking of cancer metabolism," *Scientific Reports*, vol. 12, no. 1, p. 18870, 2022. [Online]. Available: https://doi.org/10.1038/s41598-022-21846-5

[3] C. Madsen, C. Myers, N. Roehner, C. Winstead, and Z. Zhang, *Efficient Analysis Methods in Synthetic Biology*. New York, NY: Springer New York, 2015, pp. 217–257. [Online]. Available: https://doi.org/10.1007/978-1-4939-1878-2-11

[4] C. Daws, M. Kwiatkowska, and G. Norman, "Automatic verification of the IEEE 1394 root contention protocol with KRONOS and PRISM," *International Journal on Software Tools for Technology Transfer*, vol. 5, no. 2, pp. 221–236, Mar. 2004. [Online]. Available: https://doi.org/10.1007/s10009-003-0118-5

[5] A. Sesic, S. Dautovic, and V. Malbasa, "Dynamic Power Management of a System With a Two-Priority Request Queue Using Probabilistic-Model Checking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 2, pp. 403–407, Feb. 2008. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4378215

[6] I. Cizelj, X. C. D. Ding, M. Lahijanian, A. Pinto, and C. Belta, "Probabilistically Safe Vehicle Control in a Hostile Environment," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 11 803–11 808, Jan. 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016455126

[7] M. Volk, S. Junges, and J.-P. Katoen, "Fast Dynamic Fault Tree Analysis by Model Checking Techniques," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 1, pp. 370–379, Jan. 2018.

[8] M. Cotner, J. Zhan, and Z. Zhang, "A computational metabolic model for engineered production of resveratrol in Escherichia coli," *ACS Synthetic Biology*, vol. 10, no. 8, pp. 1992–2001, 2021, pMID: 34237218. [Online]. Available: https://doi.org/10.1021/acssynbio.1c00163

[9] "Fluentverification/CaseStudiesStochasticModelChecking," fluentverification, Jun. 2024. [Online]. Available: https://github.com/fluentverification/CaseStudies-StochasticModelChecking

[10] 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., vol. 13872. Cham: Springer Nature Switzerland, May 2023, pp. 83–102.

[11] L. Taylor, B. Israelsen, and Z. Zhang, "Cycle and Commute: Rare-Event Probability Verification for Chemical Reaction Networks," in *Formal Methods in Computer-Aided Design*. TU Wien Academic Press, Oct. 2023, pp. 284–293.

[12] J. Jeppson, M. Volk, B. Israelsen, R. Roberts, A. Williams, L. Buecherl, C. J. Myers, H. Zheng, C. Winstead, and Z. Zhang, "STAMINA in C++: Modernizing an Infinite-State Probabilistic Model Checker," in *Quantitative Evaluation of Systems*, N. Jansen and M. Tribastone, Eds. Cham: Springer Nature Switzerland, 2023, pp. 101–109.

[13] K. Batz, S. Junges, B. L. Kaminski, J.-P. Katoen, C. Matheja, and P. Schröer, "PrIC3: Property Directed Reachability for MDPs," in *Computer Aided Verification*, S. K. Lahiri and C. Wang, Eds., vol. 12225. Springer International Publishing, 2020, pp. 512–538.

[14] P. Schröer, K. Batz, B. L. Kaminski, J.-P. Katoen, and C. Matheja, "A Deductive Verification Infrastructure for Probabilistic Programs," *Proceedings of the ACM on Programming Languages*, vol. 7, no. OOPSLA2, pp. 2052–2082, Oct. 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3622870

[15] A. L. Blanc and P. Lam, "Surveying the Rust Verification Landscape," Oct. 2024, arXiv:2410.01981 [cs] version: 1. [Online]. Available: http://arxiv.org/abs/2410.01981