Compositional Stochastic Model Checking Probabilistic Automata via Assume-guarantee Reasoning(Article-Title)

Yanqing Liu1,2,\* Rui Li1,Jiashu Chen2(Author-Name)

1School of Information Engineering, Nanjing University of Finance & Economics, Nanjing, Jiangsu 210046, China(Affiliation)

2School of Computing, National University of Singapore, Singapore 117417, Singapore(Affiliation)

\*Corresponding author. Email: yliu@nufe.edu.cn(Corresponding)

**ABSTRACT**

Stochastic model checking is the extension and generalization of the classical model checking. Compared with classical model checking, stochastic model checking faces more severe state explosion problem, because it combines classical model checking algorithms and numerical methods for calculating probabilities. For dealing with this, we first apply symmetric assume-guarantee rule symmetric (SYM) for two-component systems and symmetric assume-guarantee rule for n-component systems into stochastic model checking in this paper, and propose a compositional stochastic model checking framework of probabilistic automata based on the NL\* algorithm. It optimizes the existed compositional stochastic model checking process to draw a conclusion quickly, in cases the system model does not satisfy the quantitative properties. We implement the framework based on the PRISM tool, and several large cases are used to demonstrate the performance of it.(Abstract)

**Keywords:** Stochastic model checking, assume-guarantee reasoning, symmetric assume-guarantee rule, learning algorithm, probabilistic automata(Keywords)

# 1. INTRODUCTION(Heading1)

Formal verification can reveal the unexposed defects in a safety-critical system. As a prominent formal verification technique, model checking is an automatic and complete verification technique of finite state systems against correctness properties, which was pioneered respectively by Clarke and Emerson [1] and by Queille and Sifakis [2] in the early 1980’s. Whereas model checking techniques focus on the absolute correctness of systems, in practice such rigid notions are hard, or even impossible, to ensure. Instead, many systems exhibit stochastic aspects [3] which are essential for among others: modeling unreliable and unpredictable system behavior (message garbling or loss), model-based performance evaluation (i.e., estimating system performance and dependability) and randomized algorithms (leader election or consensus algorithms). Automatic formal verification of stochastic systems by model checking is called stochastic model checking or probabilistic model checking.(Para)

Stochastic model checking algorithms rely on a combination of model checking techniques for classical model checking and numerical methods for calculating probabilities. So, stochastic model checking faces more severe state explosion problem, compared with classical model checking. There are some works to deal with this problem through bounded probabilistic model checking, abstraction refinement, compositional verification and so on. The crucial notion of compositional verification is “divide and conquer”. It can decompose the whole system into separate components and conquer each component separately. (Para)

## 1.1. Related Work(Heading 2)

According to the generation type of assumptions, we divided the existed work into two categories.

### 1.1.1. Manual interactive assumption generation(Heading3)

On the existing theory of Markov Decision Process (MDP) model of combinatorial analysis, Kwiatkowska et al.

### 1.1.2. Automated assumption generation

Bouchekir and Boukala, He et al., Komuravelli et al., Feng et al. and are the automated assumption generation methods for solving the AG-SMC problem. They can be divided into the following three kinds further.

#### 1.1.2.1. Learning-based assumption generation(Heading4)

Based on the learning-based assume-guarantee reasoning (LAGR) technology and the ASYM rule proposed in Segala, Feng et al. proposes L\*-based learning framework for PA model, which can be used to verify whether the given PA model satisfies the probabilistic safety property. Feng et al.

#### 1.1.2.2. Symbolic learning-based assumption generation

One deficiency of learning-based assumption generation method is that the learning framework is sound but incomplete. Based on ASYM rule, He et al. proposes an assume-guarantee rule containing weighted assumption for the first time, and provides a sound and complete learning framework, which can verify whether the probabilistic safety properties are satisfied on the MDP model.

#### 1.1.2.3. Assumption generation based on abstraction-refinement

The method in Komuravelli et al. is similar to Counterexample Guided Abstraction Refinement (CEGAR). It uses the Assume-Guarantee Abstraction Refinement technology to propose an assume-guarantee compositional verification framework for Labeled Probabilistic Transition Systems (LPTSes), which can verify whether the given LPTS model satisfies the safe-PCTL property.

## 1.2. Our Contribution

This paper presents some improvements based on the probabilistic assume-guarantee framework proposed in Feng et al. [23]. On one hand, our optimization is to verify each membership and equivalence query, to seek a counterexample, which can prove the property is not satisfied. If the counterexample is not spurious, the generation of the assumptions will stop, and the verification process will also terminate immediately. On the other hand, a potential shortage of the ASYM displays that the sole assumption *A* about *M*1 is present, but the additional assumption about *M*2 is nonexistent.

## 1.3. Paper Structure

The rest of the paper is organized as follows. Section 2 introduces the preliminaries used in this paper, which include PAs, model checking and the NL\* algorithm. Section 3 presents a compositional stochastic model checking framework based on the SYM rule and optimizes the learning framework.

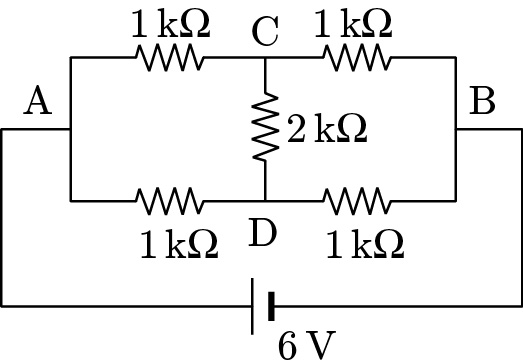
# 2. BACKGROUND

## 2.1. Probabilistic Automata

Probabilistic automata can model both probabilistic and nondeterministic behavior of systems, which is a slight generalization of MDPs. The verification algorithms for MDPs can be adapted for PAs.

 (1)

Figure 1 shows two PAs M1 and M2.

****

**Figure 1** (a) Probabilistic automata *M*1 (b) probabilistic automata *M*1 and (c) DFA *P*err for the safety property P(Fig caption)

**Table 1** Client–server experimental results(Table title)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Case study [consensus]** | **[*N R K*]** | **Component sizes** | | **SYM** | **ASYM [23]** |
| |*M*1| | |*M*2| | **Time (s)** | **Time (s)** |
|  | 2 3 20 | 3217 | 389 | 12.1 | 11.6 |
| 2 4 4 | 431649 | 571 | 82.2 | 80.7 |
| 3 3 20 | 38193 | 8837 | 355.8 | 350.2 |

To consider the case where the model satisfies the properties, the last case is randomized consensus algorithm from Feng et al. [23] without modification.(Table foot)

In Table 1, the component sizes of the *M*1 and *M*2 are denoted as |*M*1| and |*M*2|, and the performance is measured by the total number of Membership Queries (MQ) and runtimes (Time). Note that Time includes counterexample construction, NFA translation and the learning process. Moreover, for the accuracy of the results, we select the counterexamples in the same order as Feng et al. in each equivalence query. Note that Feng et al. has included comparisons with non-compositional verification, so this paper only compares with Feng et al.

**ACKNOWLEDGMENT(Acknowledgment)**

This work was supported by National Natural Science Foundation of China (61303022), Natural Science Major Project of Jiangsu Higher Education Institutions (17KJA520002), and Nanjing Scientific & Technological Innovation Project for Outstanding Overseas Returnees.

REFERENCES(Reference title)

[1] M.O. Rabin, D.S. Scott, Finite automata and their decision problems, IBM J. Res. Dev. 3(2) (1959) 114–125. DOI: https://doi.org/10.1147/rd.32.0114

[2] C. Baier, J-P. Katoen, Principles of Model Checking, MIT Press, 2008.

[3] H. Barringer, R. Kuiper, A. Pnueli, Now you may compose temporal logic specifications, in: Proceedings of the Sixteenth Annual ACM Symposium on the Theory of Computing (STOC), ACM, 1984, pp. 51–63. DOI: <https://doi.org/10.1145/800057.808665(Reference> item)