Compositional Stochastic Model Checking Probabilistic Automata via Assume-guarantee Reasoning

Yang Liu[1,2,*] Rui Li[1]

[1]*School of Information Engineering, Nanjing University of Finance & Economics, Nanjing, Jiangsu 210046, China*
[2]*School of Computing, National University of Singapore, Singapore 117417, Singapore*
[*]*Corresponding author. Email: yliu@nufe.edu.cn*

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

1. INTRODUCTION

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 [4].

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 [5]. There are some works to deal with this problem through bounded probabilistic model checking [6], abstraction refinement [7], compositional verification [8] 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. The compositional verification techniques include assume-guarantee reasoning [9], contract-based methods [10] and invariant-based methods [11]. This paper focuses on assume-guarantee reasoning, which is an automatic method of compositional verification. To account for the relationship between the whole system and its different components, assume-guarantee reasoning gives some rules, which can change the global verification of a system into local verification of individual components.

Theoretically speaking, applying the assume-guarantee reasoning into stochastic model checking is a feasible way to solve the state explosion problem. There is some research work done in this direction [12–15]. We argue that applying the assume-guarantee reasoning into stochastic model checking should solve the following four issues, which is named as AG-SMC problem: (1) How to generate appropriate assumptions. (2) How to check the assume-guarantee triple. (3) How to construct a counterexample. (4) How to verify a stochastic system composed of n ($n \geq 2$) components.

1.1. Related Work

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

1.1.1. Manual interactive assumption generation

On the existing theory of Markov Decision Process (MDP) model of combinatorial analysis [16], Kwiatkowska et al. [17] first gives out assume-guarantee reasoning for verifying probabilistic automaton (PA) model, including asymmetric assumption-guarantee rule (ASYM), circular assumption-guarantee rule (CRIC) and asynchronous assumption-guarantee rule (ASYNC). It solves the AG-SMC problem as follows: (1) It generates the assumptions through the manual interactive method. (2) In the triple of the form $\langle A\rangle_{\geq \mathrm{PA}} M \langle P\rangle_{\geq \mathrm{PG}}$, system model M is a PA, the assumption $\langle A\rangle_{\geq \mathrm{PA}}$ and guarantee $\langle P\rangle_{\geq \mathrm{PG}}$ are probabilistic safety properties, represented by deterministic finite automaton (DFA). When system component M satisfies assumptions A with minimum probability PA, it will be able to satisfy property P with minimum probability PG. Checking the triple can be reduced to multi-objective model checking [18], which is equivalent to a linear programming (LP) problem. (3) It does not involve to construct the counterexamples. (4) It verifies a stochastic system composed of $n \geq 2$ components through multi-component asymmetric assume-guarantee rule (ASYM-N). The core idea of ASYM-N rule is similar to CRIC rule, i.e., the component M_1 satisfies the guarantee $\langle A_1\rangle_{\geq \mathrm{PA}M1}$, then the guarantee $\langle A_1\rangle_{\geq \mathrm{PA}M1}$ as the assumption of the component M_2, let the component M_2 can satisfy the guarantee $\langle A_2\rangle_{\geq \mathrm{PA}M2}$,…, until the component M_n that satisfies the assumption $\langle A_{n-1}\rangle_{\geq \mathrm{PA}Mn\text{-}1}$ can satisfy the guarantee $\langle P\rangle_{\geq \mathrm{PG}}$. If all above-mentioned conditions hold, the entire system model $M_1 \| M_2 \| \cdots \| M_n$ will satisfy the guarantee $\langle P\rangle_{\geq \mathrm{PG}}$.

1.1.2. Automated assumption generation

Bouchekir and Boukala [19], He et al. [20], Komuravelli et al. [21], Feng et al. [22] and [23] 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.

Based on the learning-based assume-guarantee reasoning (LAGR) technology and the ASYM rule proposed in Segala [16], Feng et al. [22] 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. [22] uses the cases to demonstrate the performance of its method, including the client–server, sensor network and the randomized consensus algorithm. For the AG-CSMC problem, Segala [16] can be specifically described in the following four aspects: (1) Through the L^* learning algorithm, the process of generating an appropriate assumption $\langle A\rangle_{\geq \mathrm{PA}}$ is fully automated, i.e., we need to generate a closed and consistent observation table through membership queries, to generate a conjectured assumption, and then verify the correctness of the assumption through equivalence queries. (2) It checks the assume-guarantee triple through multi-objective model checking [18]. (3) In the whole learning process, Feng et al. [22] adopts the method proposed in Han et al. [24] to generate probabilistic counterexamples for refining the current assumption, i.e., the PRISM [25] is used to obtain the error state nodes in the model, and then the probabilistic counterexamples are constructed by using Eppstein's [26] algorithm. (4) The verification problem of a stochastic system composed of $n \geq 2$ components is not solved.

Feng et al. [23] makes further research based on Feng et al. [22] and uses several large cases to demonstrate the performance of it, including client–server, sensor network, randomized consensus algorithm and Mars Exploration Rovers (MER). For the AG-CSMC problem, compared with Feng et al. [23] and Feng et al. [22], the contribution of Feng et al. [23] is reflected in the better solution of the first sub-problem and the solution of the fourth sub-problem, which will be illustrated in the following two aspects: (1) Feng et al. [23] compares the assumption generation process between the L^* learning algorithm and the NL^* learning algorithm, and finds that NL^* often needs fewer membership and equivalence queries than L^* in large cases. (2) Based on Segala [16], Feng et al. [23] uses the ASYM-N rule to propose a learning framework for compositional stochastic model checking, and uses it to verify the multi-component stochastic system. So far, in the learning-based assumption generation method, four sub-problems of AG-CSMC problem have been solved basically.

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. [20] 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. Through randomized consensus algorithm, wireless LAN protocol, FireWire protocol and randomized dining philosophers, He et al. [20] demonstrates the performance of its method. For the AG-CSMC problem, He et al. [20] can be specifically described in the following four aspects: (1) The weighted assumption can be represented by Multi-terminal Binary Decision Diagrams (MTBDD). Based on the L^* learning algorithm, He et al. [20] proposes an MTBDD learning algorithm to automatically generate

the weighted assumption, which is represented by a k-Deterministic Finite Automaton (k-DFA). MTBDD learning algorithm can make membership queries on binary strings of arbitrary lengths and answer membership queries on valuations over fixed variables by the teacher. (2) Through the weighted extension of the classical simulation relation, He et al. [20] presents a verification method of the assume-guarantee triple containing the weighted assumption. (3) Similarly to Feng et al. [22], He et al. [20] also constructs the necessary probabilistic counterexamples in the learning process through Han et al. [24]. (4) The verification problem of a stochastic system composed of $n \geq 2$ components is not solved.

In Bouchekir and Boukala [19], the method realizes automatic assumption generation through the Symbolic Learning-based Assume-Guarantee Reasoning technology, also known as the Probabilistic Symbolic Compositional Verification (PSCV). The PSCV method provides a sound and complete symbolic assume-guarantee rule to verify whether the MDP model satisfies the Probabilistic Computation Tree Logic (PCTL) property. It is a new approach based on the combination of assume-guarantee reasoning and symbolic model checking techniques. Bouchekir and Boukala [19] uses randomized mutual exclusion, client–server, randomized dining philosophers, randomized self-stabilizing algorithm and Dice to demonstrate the performance of its method. For the AG-CSMC problem, Bouchekir and Boukala [19] can be specifically described in the following four aspects: (1) Appropriate assumptions are automatically generated by symbolic MTBDD learning algorithm, and represented by interval MDP (IMDP), thus ensuring the completeness of symbolic assume-guarantee rule. Moreover, In addition, to reduce the size of the state space, The PSCV method encodes both system components and assumptions implicitly using compact data structures, such as BDD or MTBDD. (2) Bouchekir and Boukala [19] uses the method in He et al. [20] to verify assume-guarantee triple. (3) To refine assumptions, the PSCV method [27] uses the causality method to construct counterexamples, i.e., it uses K^* algorithm [28] in the DiPro tool to construct counterexamples, and applies the algorithms in Debbi and Bourahla [29] to construct the most indicative counterexample. (4) Verification of a stochastic system composed of $n \geq 2$ components is not involved.

1.1.2.3. Assumption generation based on abstraction-refinement.

The method in Komuravelli et al. [21] is similar to Counterexample Guided Abstraction Refinement (CEGAR) [30]. 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. Komuravelli et al. [21] uses the client–server, MER and wireless sensor network to demonstrate the performance of its method. For the AG-CSMC problem, Komuravelli et al. [21] can be specifically described in the following four aspects: (1) The method can use tree counterexamples from checking one component to refine the abstraction of another component. Then, it uses the abstraction as the assumptions for assume-guarantee reasoning, represented by LPTS. (2) It uses a strong simulation relationship to check the assume-guarantee triple; (3) The process of constructing tree counterexample can be reduced to check the Satisfiability Modulo Theories problem, and then solve it through Yices [31]. (4) It also verifies an n-component stochastic system ($n \geq 2$) by the ASYM-N rule.

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. We thus apply the SYM rule to the compositional verification of PAs and extend the rule to verify an n-component system ($n \geq 2$). Through several large cases, it is shown that our improvements are feasible and efficient.

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. Then, the framework is extended to an n-component system ($n \geq 2$) in Section 4. Section 5 develops a prototype tool for the framework, and compares it with Feng et al. [23] by several large cases. Finally, Section 6 concludes the paper and presents direction for future research.

2. BACKGROUND

2.1. Probabilistic Automata

Probabilistic automata [3, 17, 32, 33] 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.

In the following, Dist(V) is defined as the set of all discrete probability distributions over a set V. η_v is defined as the point distribution on $v \in V$. $\mu_1 \times \mu_2 \in \text{Dist}(V_1 \times V_2)$ is the product distribution of $\mu_1 \in \text{Dist}(V_1)$ and $\mu_2 \in \text{Dist}(V_2)$.

Definition 1. (probabilistic automaton) A probabilistic automaton (PA) is a tuple $M = (V, \bar{v}, \alpha_M, \delta_M, L)$ where V is a set of states, $\bar{v} \in V$ is an initial state, α_M is an alphabet for all the action, $\delta_M \subseteq V \times (\alpha_M \cup \{\tau\}) \times \text{Dist}(V)$ is a probabilistic transition relation. τ is an invisible action, and $L: V \to 2^{AP}$ is a labeling function mapping each state to a set of atomic propositions taken from a set AP.

In any state v of a PA M, we use the transition $v \xrightarrow{\alpha} \mu$ to denote that $(v, \alpha, \mu) \in \delta_M$, where $\alpha \in \alpha_M \cup \{\tau\}$ is an action label. μ is a probability distribution over state v. All transitions are nondeterministic, and it will make a random choice according to the distribution μ. A trace through M is a (finite or infinite) sequence $v_0 \xrightarrow{\alpha_0,\mu_0} v_1 \xrightarrow{\alpha_1,\mu_1} \cdots$ where $v_0 = \bar{v}$, and for each $i \geq 0$, $v_i \xrightarrow{\alpha_i} \mu_i$ is a transition and $\mu_i(v_{i+1}) > 0$. The sequence of actions $\alpha_0, \alpha_1, \ldots$, after removal of any τ, from a trace t is also called a path. An adversary σ is sometimes referred to as scheduler, policy, or strategy, which maps any finite path to a sub-distribution over the available transitions in the last state of the path. This paper focuses on are finite-memory adversaries, which store information about the history in a finite-state automaton (see Baier and Katoen [3] Definition 10.97; pp. 848). We define Trace_M^σ as the set of all traces through M under the control of adversary σ, and Adv_M as the set of all potential adversaries for M. For an adversary, we define a probability space $\Pr_M^\sigma$ on Trace_M^σ, and the probability space can know the probability of the adversary σ.

Definition 2. (Parallel composition of PAs) If $M_1 = (V_1, \bar{v}_1, \alpha_{M_1}, \delta_{M_1}, L_1)$ and $M_2 = (V_2, \bar{v}_2, \alpha_{M_2}, \delta_{M_2}, L_2)$ are PAs, then their parallel composition is denoted as $M_1 \| M_2$. It is given by the $\text{PA}\left(V_1 \times V_2, (\bar{v}_1, \bar{v}_2), \alpha_{M_1} \cup \alpha_{M_2}, \delta_{M_1 \| M_2}, L\right)$

where $\delta_{M1\|M2}$ is defined such that $(v_1, v_2) \xrightarrow{\alpha} \mu_1 \times \mu_2$ if and only if one of the following holds:

$$v_1 \xrightarrow{\alpha} \mu_1, v_2 \xrightarrow{\alpha} \mu_2 \text{ and } \alpha \in \alpha_{M_1} \cap \alpha_{M_2} \quad (1)$$

$$v_1 \xrightarrow{\alpha} \mu_1, \mu_2 = \eta_{v_2} \text{ and } \alpha \in (\alpha_{M_1} \setminus \alpha_{M_2}) \cup \{\tau\} \quad (2)$$

$$v_2 \xrightarrow{\alpha} \mu_2, \mu_1 = \eta_{v_1} \text{ and } \alpha \in (\alpha_{M_2} \setminus \alpha_{M_1}) \cup \{\tau\} \quad (3)$$

and

$$L(v_1, v_2) = L_1(v_1) \cup L_2(v_2) \quad (4)$$

Definition 3. (Alphabet extension of PA) For any $\text{PA}\, M = (V, \bar{v}, \alpha_M, \delta_M, L)$ and set of actions y, we extend the alphabet of M to y, denoted $M[y]$, as follows: $M[y] = (V, \bar{v}, \alpha_M \cup y, \delta_{M[y]}, L)$ where $\delta_{M[y]}$ is a probabilistic transition relation on $M[y]$, and $\delta_{M[y]} = \delta_M \cup \{(v, \alpha, \eta_v) | v \in V \wedge \alpha \in y \setminus \alpha_M\}$.

For any state $v = (v_1, v_2)$ of $M_1 \| M_2$, the projection of v on M_i, denoted by $v\restriction_{Mi}$. Then, we extend it to distributions on the state space $V_1 \times V_2$ of $M_1 \| M_2$. For each trace t on $M_1 \| M_2$, the projection of t on M_i, denoted by $t\restriction_{Mi}$, i.e., the trace can be acquired from M_i by projecting each state of t onto M_i and removing all the actions not in the alphabet α_{Mi}.

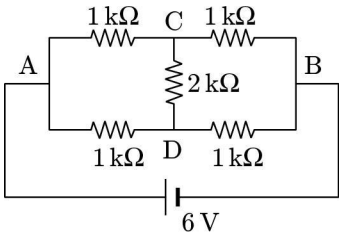

Figure 1 (a) Probabilistic automata M_1 (b) probabilistic automata M_1 and (c) DFA P^{err} for the safety property P

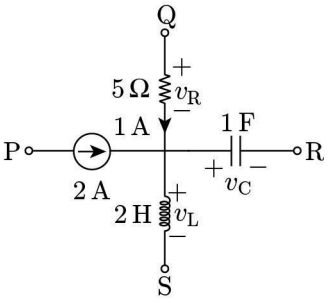

Figure 2 Assumptions $A_{M_1}^{\text{err}}$, $A_{M_2}^{\text{err}}$ for M_1, M_2

Example 1. Figure 1 shows two PAs M_1 and M_2. The switch of a device M_2 is controlled by a controller M_1. Once the emergence of the detect signal, M_1 can send a warn signal before the shutdown signal, but the attempt may be not successful with probability 0.2. M_1 issues the shutdown signal directly, this will lead to the occurrence of a mistake in the device M_2 with probability 0.1(i.e., M_2 will not shut down correctly). The DFA P^{err} indicates that action fail never occurs. We need to verify whether $M_1 \| M_2 \vDash \langle P \rangle_{\geq 0.98}$ holds.

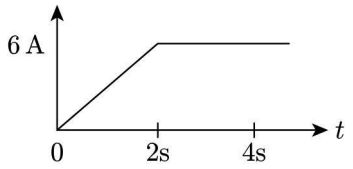

Figure 3 NL*-based learning framework for the rule SYM

On the contrary, we need to check whether it is a spurious counterexample, let the conjectured assumption becomes stronger than necessary. If the spurious counterexample exists, the conjectured assumption must be refined once again. When the conjectured assumption is updated, the framework will return a lower and an upper bound on the minimum probability of safety property P holding. This measure means that it can provide some valuable information to the user, even if the framework could not produce an accurate judgment. More details are described in the following sections.

property P. **Figure 6** Assumptions $A_{M_1}^{\text{err}}$, $A_{M_2}^{\text{err}}$, $A_{M_3}^{\text{err}}$ for M_1, M_2, M_3

Through premise $n + 1$, we can find a spurious counterexample trace cex(0.2, ⟨shutdown⟩) in $\left\langle A_{M_1}^{\text{err}} \right\rangle_{\geq 0.1}$ and cex(1, ⟨shutdown⟩) in $\left\langle A_{M_2}^{\text{err}} \right\rangle_{\geq 0}$, but corresponding spurious counterexample trace in $\left\langle A_{M_3}^{\text{err}} \right\rangle_{\geq 0.2}$ is nonexistent (since action fail exists). So prefixes of all infinite traces in $\left\langle A_{M_1}^{\text{err}} \right\rangle_{\geq 0.1} \| \left\langle A_{M_2}^{\text{err}} \right\rangle_{\geq 0} \| \left\langle A_{M_3}^{\text{err}} \right\rangle_{\geq 0.2}$ can be accepted by $\mathrm{L}\left(\left\langle A_{M_1}^{\text{err}} \right\rangle_{\geq 0.1} \| \left\langle A_{M_2}^{\text{err}} \right\rangle_{\geq 0} \| \left\langle A_{M_3}^{\text{err}} \right\rangle_{\geq 0.2} \right)$ and we can think $M_1 \| M_2 \| M_3 \vDash \langle P \rangle_{\geq 0.98}$ holds.

Table 1 Sensor network experimental results

Case study [sensor network]	Sensor numbers	Component sizes		SYM		ASYM [23]	
		$\|M_1\|$	$\|M_2\|$	***MQ***	***Time(s)***	***MQ***	***Time(s)***
	1	72	32	16	1.5	25	2.7
	2	1184	32	16	1.8	25	2.9
	3	10662	32	16	2.4	25	3.9

The second case is the client–server model studied from Pasareanu et al. [42]. Feng et al. [23] injects (probabilistic) failures into one or more of the N clients and changes the model into a stochastic system. In client–server model, each client can send requests for reservations to use a common resource, the server can grant or deny a client's request, and the model must satisfy the mutual exclusion property (i.e., conflict in using resources between clients) with certain minimum probability. Through the SYM rule, we make the server as a component M1 and the composition of N clients as the other component M2. The verified property is ⟨P⟩≥0.9.We use the method of Feng et al. [23] to inject (nonprobabilistic and probabilistic) failures into the server respectively. Table 2 shows experimental results for the client–server.We first present a sound SYM for compositional stochastic model checking. Then, we propose a learning framework for compositional stochastic model checking PAs with rule SYM, based on the optimization of LAGR techniques. Our optimization can terminate the learning process in advance, if a counterexample appears in any membership and equivalence query. We also extend the framework to support the assume-guarantee rule SYM-N which can be used for reasoning about a stochastic system composed of n ≥ 2 components: M1 ‖ M2 ‖ ⋯ ‖ Mn. Experimental results show that our method can improve the efficiency of the original learning framework [23]. Similar to Feng et al. [22] and Kwiatkowska et al. [33], it can return the tightest bounds for the safety property as a reference as well.

In the future, we intend to develop our learning framework to produce richer classes of probabilistic assumption (for example weighted automata as assumptions [39]) and extend it to deal with more expressive types of probabilistic models.

To consider the case where the model satisfies the properties, the last case is randomized consensus algorithm from Feng et al. [23] without modification. The algorithm models N distributed processes trying to reach consensus and uses, in each round, a shared coin protocol parameterized by K. The verified property is $\langle P \rangle_{\geq 0.97504}$, and 0.97504 is the minimum probability of consensus being reached within R rounds. Through the SYM rule, the system is decomposed into two PA components: M_1 for the coin protocol and M_2 for the interleaving of N processes.

In Tables 1 and 2, 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. [23] in each equivalence query. Note that Feng et al. [23] has included comparisons with non-compositional verification, so this paper only compares with Feng et al. [23].

Table 2 Client–server experimental results

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. The algorithm models N distributed processes trying to reach consensus and uses, in each round, a shared coin protocol parameterized by K. The verified property is $\langle P \rangle_{\geq 0.97504}$, and 0.97504 is the minimum probability of consensus being reached within R rounds. Through the SYM rule, the system is decomposed into two PA components: M_1 for the coin protocol and M_2 for the interleaving of N processes.

In Tables 1 and 2, 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. [23] in each equivalence query. Note that Feng et al. [23] has included comparisons with non-compositional verification, so this paper only compares with Feng et al. [23].

As shown in Tables 1 and 2, the experiment results show that our framework is more efficient than Feng et al. [23]. Obviously, we can observe that, for all cases, runtimes and the number of the membership queries in our framework are less than Feng et al. [23]. Moreover, the runtimes need less in our framework, when the model has a large scale. A larger size model may have less runtimes and the number of membership queries than a smaller model. However, this is not proportion with the model size. The efficiency of our framework depends only on the time of a counterexample (indicate that the probabilistic safety property is violated) appears in conjectured assumptions. The earlier a counterexample appears, the more efficient our framework performs.

In Table 3, the component sizes of the M_1 and M_2 is also denoted as $|M_1|$ and $|M_2|$. The performance is measured only by total runtimes (Time), because both methods have the same amount of MQ if the model satisfies the properties. Because of the cost of early detection, we can find that our methods need to spend more time than Feng et al. [23] and cost grows with the model size. But compared with acquirement of optimization in Tables 1 and 2, the cost is acceptable in Table 3.

2. CONCLUSION

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

[1] E.M. Clarke, E.A. Emerson, Design and synthesis of synchronization skeletons using branching time temporal logic, in: D. Kozen (Eds.), Workshop on Logics of Programs, Lecture Notes in Computer Science, vol. 131, Springer, Berlin, Heidelberg, 1981, pp. 52–71. DOI: https://doi.org/10.1007/BFb0025774

[2] J.P. Queille, J. Sifakis, Specification and verification of concurrent systems in CESAR, in: M. Dezani-Ciancaglini and U. Montanari (Eds.), Proceedings of the 5th International Symposium on Programming, Lecture Notes in Computer Science, vol. 137, Springer, Berlin, Heidelberg, 1982, pp. 337–351. DOI: https://doi.org/10.1007/3-540-11494-7_22

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

[4] M. Kwiatkowska, G. Norman, D. Parker, Stochastic model checking, in: M. Bernardo, J. Hillston (Eds.), Proceedings of the Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM), Springer, Berlin, Heidelberg, 2007, pp. 220–270. DOI: https://doi.org/10.1007/978-3-540-72522-0_6

[5] V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, Automated verification techniques for probabilistic systems, in: M. Bernardo, V. Issarny (Eds.), Proceedings of the Formal Methods for Eternal Networked Software Systems (SFM), Springer, Berlin, Heidelberg, 2011, pp. 53–113. DOI: https://doi.org/10.1007/978-3-642-21455-4_3

[6] G.D. Penna, B. Intrigila, I. Melatti, E. Tronci, M.V. Zilli, Bounded probabilistic model checking with the muralpha verifier, in: A.J. Hu, A.K. Martin (Eds.), Proceedings of the Formal Methods in Computer-Aided Design, Springer, Berlin, Heidelberg, 2004, pp. 214–229. DOI: https://doi.org/10.1007/978-3-540-30494-4_16

[7] E. Clarke, O. Grumberg, S. Jha, et al., Counterexample-guided abstraction refinement, in: E.A. Emerson, A.P. Sistla (Eds.), Computer Aided Verification, Springer, Berlin, Heidelberg, 2000, pp. 154–169. DOI: https://doi.org/10.1007/10722167_15

[8] 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

[9] A. Pnueli, In transition from global to modular temporal reasoning about programs, in: K.R. Apt (Ed.), Logics and Models of Concurrent Systems, Springer, Berlin, Heidelberg, 1984, pp. 123–144. DOI: https://doi.org/10.1007/978-3-642-82453-1_5

[10] B. Meyer, Applying "Design by Contract", Computer 25(10) (1992) 40–51. DOI: https://doi.org/10.1109/2.161279

[11] S. Bensalem, M. Bogza, A. Legay, T.H. Nguyen, J. Sifakis, R. Yan, Incremental component-based construction and verification using invariants, in: Proceedings of the Conference on Formal Methods in Computer Aided Design (FMCAD), IEEE Press, Piscataway, NJ, 2010, pp. 257–256.

[12] H. Barringer, C.S. Pasareanu, D. Giannakopolou, Proof rules for automated compositional verification through learning, in Proc. of the 2nd International Workshop on Specification and Verification of Component Based Systems, 2003.

[13] M.G. Bobaru, C.S. Pasareanu, D. Giannakopoulou, Automated assume-guarantee reasoning by abstraction refinement, in: A. Gupta, S. Malik (Eds.), Proceedings of the Computer Aided Verification, Springer, Berlin, Heidelberg, 2008, pp. 135–148. DOI: https://doi.org/10.1007/978-3-540-70545-1_14

[14] J.M. Cobleigh, D. Giannakopoulou, C.S. Păsăreanu, Learning assumptions for compositional verification, in: H. Garavel, J. Hatcliff (Eds.), Proceedings of the 9th Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Lecture Notes in Computer Science, vol. 2619, Springer, Berlin, Heidelberg, 2003, pp. 331–346. DOI: https://doi.org/10.1007/3-540-36577-X_24

[15] O. Grumberg, D.E. Long, Model checking and modular verification, ACM Trans. Program. Lang. Syst. 16(3) (1994) 843–871. DOI: https://doi.org/10.1145/177492.177725

[16] R. Segala, Modeling and verification of randomized distributed real-time systems, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, 1995 (Also appears as Technical Report MIT/LCS/TR-676).

[17] M. Kwiatkowska, G. Norman, D. Parker, H. Qu, Assume-guarantee verification for probabilistic systems, in: J. Esparza, R. Majumdar (Eds.), Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Springer, Berlin, Heidelberg, 2010, pp. 23–37. DOI: https://doi.org/10.1007/978-3-642-12002-2_3

[18] K. Etessami, M. Kwiatkowska, M. Vardi, M. Yannakakis, Multi-objective model checking of Markov decision processes, in: O. Grumberg and M. Huth (Eds.), Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Springer, Berlin, Heidelberg, 2007, pp. 50–65. DOI: https://doi.org/10.1007/978-3-540-71209-1_6

[19] R. Bouchekir, M.C. Boukala, Learning-based symbolic assume-guarantee reasoning for Markov decision process by using interval Markov process, Innov. Syst. Softw. Eng. 14(3) (2018) 229–244. DOI: https://doi.org/10.1007/s11334-018-0316-7

[20] F. He, X. Gao, M. Wang, B-Y. Wang, L. Zhang, Learning weighted assumptions for compositional verification of markov decision processes, ACM Trans. Softw. Eng. Meth. 25(3) (2016) 21. DOI: https://doi.org/10.1145/2907943

[21] A. Komuravelli, C.S. Păsăreanu, E.M. Clarke. Assume-guarantee abstraction refinement for probabilistic systems, in: P. Madhusudan, S.A. Seshia (Eds.), Proceedings of the International Conference on Computer Aided Verification, Springer, Berlin, Heidelberg, 2012, pp. 310–326. DOI: https://doi.org/10.1007/978-3-642-31424-7_25

[22] L. Feng, M. Kwiatkowska, D. Parker, Compositional verification of probabilistic systems using learning, in: Proceedings of the Seventh International Conference on the Quantitative Evaluation of Systems, IEEE Press, Williamsburg, VA, USA, 2010, pp. 133–142. DOI: https://doi.org/10.1109/QEST.2010.24

[23] L. Feng, M. Kwiatkowska, D. Parker, Automated learning of probabilistic assumptions for compositional reasoning, in: D. Giannakopoulou, F. Orejas (Eds.), Proceedings of the Fundamental Approaches to Software Engineering (FASE), Springer, Berlin, Heidelberg, 2011, pp. 2–17. DOI: https://doi.org/10.1007/978-3-642-19811-3_2

[24] T. Han, J.P. Katoen, D. Berteun, Counterexample generation in

probabilistic model checking, IEEE Trans. Softw. Eng. 35(2) (2009) 241–257. DOI: https://doi.org/10.1109/TSE.2009.5

[25] A. Hinton, M. Kwiatkowska, G. Norman, D. Parker, PRISM: a tool for automatic verification of probabilistic systems, in: H. Hermanns, J. Palsberg, Proceedings of the Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Springer, Berlin, Heidelberg, 2006, pp. 441–444. DOI: https://doi.org/10.1007/11691372_29

[26] D. Eppstein, Finding the k shortest paths, SIAM J. Comput. 28(2) (1998) 652–673. DOI: https://doi.org/10.1137/S0097539795290477

[27] H. Debbi, A. Debbi, M. Bourahla, Debugging of probabilistic systems using structural equation modelling, Int. J. Critic. Comput. Based Syst. 6(4) (2017) 250–274. DOI: https://doi.org/10.1504/IJCCBS.2016.081805

[28] H. Aljazzar, S. Leue, K*: a heuristic search algorithm for finding the k shortest paths. Artif. Intell. 175(18) (2011) 2129–2154. DOI: https://doi.org/10.1016/j.artint.2011.07.003

[29] H. Debbi, M. Bourahla, Generating diagnoses for probabilistic model checking using causality, Comput. Inform. Technol. 21(1) (2013) 13–22. DOI: https://doi.org/10.2498/cit.1002115

[30] H. Hermanns, B. Wachter, L. Zhang, Probabilistic CEGAR, in: A. Gupta, S. Malik (Eds.), Proceedings of the Computer Aided Verification (CAV), Springer, Berlin, Heidelberg, 2008, pp. 162–175. DOI: https://doi.org/10.1007/978-3-540-70545-1_16

[31] B. Dutertre, L. de Moura, The Yices SMT Solver, Technical Report, SRI International, 2006.

[32] M.O. Rabin, Probabilistic automata, Inform. Control. 6(3) (1963) 230–245. DOI: https://doi.org/10.1016/S0019-9958(63)90290-0

[33] M. Kwiatkowska, G. Norman, D. Parker, H. Qu, Assume guarantee verification for probabilistic systems, in: J. Esparza, R. Majumdar (Eds.), Proceedings of the Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Springer, Berlin, Heidelberg, 2010, pp. 23–37. DOI: https://doi.org/10.1007/978-3-642-12002-2_3

[34] B. Bollig, P. Habermehl, C. Kern, M. Leucker, Angluin-style learning of NFA*, in: Boutilier and Craig (Eds.), Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI), AAAI Press, Pasadena, CA, USA, 2009, pp. 1004–1009.

[35] F. Denis, A. Lemay, A. Terlutte, Residual finite state automata, Fund. Inform. 51(4) (2002) 339–368.

[36] F. Denis, A. Lemay, A. Terlutte, Learning regular languages using RFSAs, Theor. Comput. Sci. 313(2) (2004) 267–294. DOI: https://doi.org/10.1016/j.tcs.2003.11.008

[37] L. de Alfaro, Formal Verification of Probabilistic Systems, Ph.D. Thesis, Stanford University, 1997.

[38] 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