An Enhanced Interface-Based Probabilistic Compositional Verification Approach

Samir Ouchani, Otmane Ait Mohamed, Mourad Debbabi

An Enhanced Interface-based Probabilistic Compositional Verification Approach

Samir Ouchani[1], Otmane Ait Mohamed[2], and Mourad Debbabi[3]

[1] CESI LINEACT, Aix-en-Provence, France
[2] Hardware Verification Group, Concordia University, Montreal, Canada
[3] CSL Laboratory, Concordia University, Montreal, Canada
souchani@cesi.fr,
otmane.aitmohamed@concordia.camourad.debbabi@concordia.ca

Abstract. In this paper, we aim to advance the state of the art in the verification process of systems, predominantly modeled as Probabilistic Automata (PA). This model accommodates both nondeterministic and probabilistic behaviors. Our primary strategy to address the notorious state space explosion problem inherent in model checking is the adoption of abstraction and compositional verification techniques, culminating in the development of a distributed verification approach centered on the communication interface amongst composed automata. Initially, the abstraction technique refines the system in relation to the requirement under verification and amalgamates states demonstrating comparable behaviors. Not only does it simplify the system, but it also enables a decomposition of global requirements into local ones. This decomposition process facilitates parallel verification and securely allows inference on the global requirement from local results. Moreover, the soundness of our proposed framework has been substantiated, ensuring that it correctly interprets and applies the properties of the system under scrutiny. In the final phase, we leveraged the PRISM model checker to assess the effectiveness of our proposed framework. This evaluation was carried out on three benchmark tests, providing empirical evidence to support the benefits of our approach. Our contribution to the field lies in the novel combination of abstraction and compositional verification techniques in a distributed verification framework, validated through theoretical soundness proofs and practical tests using the PRISM model checker. This result paves the way for more efficient and scalable model-checking processes for Probabilistic Automata.

Keywords: Abstraction; Compositional Verification; Probabilistic Automata; PRISM; PCTL.

1 Introduction

Model checking [1,2] is a well-established formal automatic verification technique designed for finite-state concurrent systems. It examines temporal logic specifications and automata-based formalism on system models. Alongside qualitative

model checking, quantitative verification methods based on probabilistic model checkers [3,4] have recently gained traction. Probabilistic verification enables probabilistic interpretation of a given property's satisfiability in systems that intrinsically exhibit probabilistic behavior. Despite its broad adoption, model checking typically requires substantial memory and processing time, which can be attributed to the potentially exponential growth of the system's state space due to the number of variables and concurrent behaviors. Therefore, reducing the complexity of the verification process is paramount for verifying large-scale systems.

Various techniques have been investigated [1,2,3,5] to address the aforementioned issue in qualitative model checking, which were subsequently extended to the probabilistic case. These solutions broadly aim to enhance the model-checking algorithms by introducing symbolic data structures such as binary decision diagrams, or focusing on model analysis. Predominantly, two classes of solutions are identified in the literature: abstraction and compositional verification. Abstraction provides a compact representation of the global system under verification, while compositional verification bypasses the construction of the whole system parts. Our work delves into both classes.

Abstraction techniques fall into four categories [2]: 1) state merging abstraction, 2) variable abstraction, 3) restriction-based abstraction, and 4) observer automata abstraction. Our proposed framework exploits the first and third categories. Additionally, well-recognized compositional verification techniques [6] include partitioned transition relation, lazy parallel composition, interface processes, and assume-guarantee. We employ the interface processes technique to counteract the state explosion problem in our work.

Contributions. Figure 1 presents an overview of our proposed framework, which accepts a system modeled as a composition of Probabilistic Automata (PA) and a requirement expressed in Probabilistic Computation Tree Logic (PCTL) [4] as input. Initially, the abstraction by restriction disregards the irrelevant propositions concerning a specific requirement. Subsequently, our developed state merging rules aggregate states that depict similar behaviors. To showcase the efficiency of our approach, we employ compositional verification using interface processes to verify local PCTL properties on the resultant PAs separately. Finally, we use the probabilistic model checker PRISM [7] as a probabilistic verification engine.

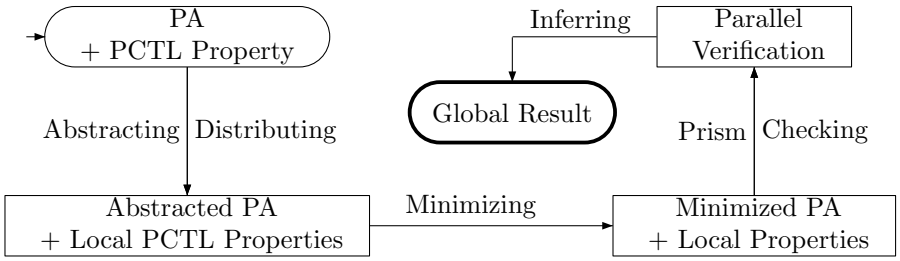

Fig. 1: The Interface-based Probabilistic Compositional Verification.

Related Work. There is a vast literature dealing with the abstraction and compositional verification of probabilistic systems. For instance, [8] introduced a game-based abstraction approach for systems exhibiting both discrete and continuous events, as well as Markov Decision Processes, as described in [9]. Symmetry reduction has been proposed in [10,11]. The partial order reduction is applied for the probabilistic branching time in [12]. In terms of probabilistic compositional verification, [13,14,15] proposed an assume-guarantee method for verifying the probabilistic safety properties of discrete-time Markov chains. To our knowledge, several probabilistic model checkers, such as PRISM, which supports symmetry reduction, and LiQuor[4], which includes the partial-order reduction, also support abstraction.

Paper Organization. The remainder of this paper is structured as follows. The subsequent section presents the models for our systems and the specifications for their requirements. The compositional reduction approach we propose is detailed in Section 3. The PRISM language is formalized in Section **??**. Experimental results are presented in Section 4, and finally, Section 5 concludes the paper.

2 Probabilistic systems

This section delves into the fundamental characteristics and applications of probabilistic systems, and further examines the methodologies employed in probabilistic modeling. Subsequently, specific focus is dedicated to the precise specification of probabilistic system requirements.

2.1 Probabilistic modeling

In systems featuring both non-determinism and probabilistic choices, Probabilistic Automata (PAs) present an apt formal model for design and representation. For a clearer understanding, let's look at the formal definition of a PA in Definition 1.

Definition 1 (Probabilistic Automata). *A Probabilistic Automaton (PA) is a quintuple* $M = (\overline{s}, S, L, \Sigma, \delta)$ *wherein:*

- $\overline{s}$*, a member of* S*, denotes the initial state,*
- S *is a finite set of states that are reachable from* $\overline{s}$*,*
- $L : S \to 2^{AP}$ *is a labeling function assigning a set of atomic propositions from a set* AP *to each state,*
- Σ *stands for a finite set of actions,*
- $\delta : S \times \Sigma \to Dist(S)$ *is a probabilistic transition function that, for each state* $s \in S$ *and action* $\alpha \in \Sigma$*, assigns a probability distribution* $\mu \in Dist(S)$*. Here,* $Dist(S)$ *symbolizes the set of convex distributions over* S*.*

[4] http://www.i1.informatik.uni-bonn.de/baier/projectpages/LIQUOR/LiQuor

Typically, a system is composed of multiple interacting components. This concept in PAs is manifested through the principle of parallel composition, described in Definition 2.

Definition 2 (Parallel Composition of PAs). *The parallel composition of two PAs:* $M_1 = (\overline{s}_1, S_1, L_1, \Sigma_1, \delta_1)$ *and* $M_2 = (\overline{s}_2, S_2, L_2, \Sigma_2, \delta_2)$*, is represented as a PA* $M = ((\overline{s}_1, \overline{s}_2), S_1 \times S_2, L, \Sigma_1 \cup \Sigma_2, \delta)$ *where:*

- $L(s_1, s_2) = L_1(s_1) \cup L_2(s_2)$,
- *For each pair of probability distributions* $\mu_1 \in \delta_1(s_1, \alpha)$ *and* $\mu_2 \in \delta_2(s_2, \alpha)$*, we define:*

$$\delta((s_1, s_2), \alpha) = \begin{cases} \mu_1 \times \mu_2 & \text{if } \alpha \in \Sigma_1 \cap \Sigma_2, \\ \mu_1 & \text{if } \alpha \in \Sigma_1 \setminus \Sigma_2, \\ \mu_2 & \text{if } \alpha \in \Sigma_2 \setminus \Sigma_1. \end{cases}$$

2.2 Requirements specification

To verify a Probabilistic Automaton (PA), we apply the Probabilistic Computation Tree Logic (PCTL) to define associated specifications. In the BNF grammar provided below, "ap" represents an atomic proposition, k is an element of the natural numbers set $\mathbb{N}$, p is within the interval $[0, 1]$, and $\bowtie$ signifies the set of relational operators $<, \leq, >, \geq$. The symbols "$\wedge$" and "$\neg$" are logical operators denoting conjunction (AND) and negation (NOT), respectively. Temporal logic operators include "X" (next), "$\mathrm{U}^{\leq k}$" (bounded until), and "U" (until). Formally, the PCTL syntax is defined as:

$$\phi \ ::= \ \top \mid ap \mid \phi \wedge \phi \mid \neg\phi \mid \mathrm{P}_{\bowtie p}[\psi], \tag{1}$$

where

$$\psi \ ::= \mathrm{X}\phi \mid \phi \mathrm{U}^{\leq k} \phi \mid \phi \mathrm{U} \phi. \tag{2}$$

We can also derive additional operators:

- True logic: $\top \equiv \neg\bot$.
- Disjunction: $\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$.
- Implication: $\phi_1 \rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$.
- Future operator: $F\phi \equiv \ \top \ U \ \phi$ or $F^{\leq k}\phi \equiv \ \top \ U^{\leq k} \ \phi$.
- Globally operator: $G\phi \equiv \ \neg(F\neg\phi)$ or $G^{\leq k}\phi \equiv \ \neg(F^{\leq k}\neg\phi)$.
- Probability of always ϕ: $\mathrm{P}_{\geq p}[G\phi] = \mathrm{P}_{\leq 1-p}[F\neg\phi]$.

To describe a satisfaction relation of a PCTL formula in a given state "s", we introduce the concept of a class of adversaries (Adv) [4]. An adversary is employed to resolve the non-deterministic choices in a PA, meaning a PCTL formula should be satisfied under all possible adversaries. Thus, the satisfaction relation ($\models_{Adv}$) of a PCTL formula is defined as follows, where "s" is a state and "π" is a sequence of states, or path:

- $s \models_{Adv} \top$ is always satisfied.
- $s \models_{Adv} a \Leftrightarrow\ a \in\ L(s)$, where $L(s)$ is the labeling function of state s.
- $s \models_{Adv} \phi_1 \wedge \phi_2 \Leftrightarrow\ s \models_{Adv} \phi_1$ and $s \models_{Adv} \phi_2$, and $s \models_{Adv} \neg\phi \Leftrightarrow\ s \not\models_{Adv} \phi$.
- $\pi \models_{Adv} X\phi \Leftrightarrow\ \pi(1) \models_{Adv} \phi$, where $\pi(1)$ is the second state in path π.
- $\pi \models_{Adv} \phi_1 U^{\leq k} \phi_2 \Leftrightarrow \exists i \geq k, \forall j < i : \pi(i) \models_{Adv} \phi_2$ and $\pi(j) \models_{Adv} \phi_1$.
- $\pi \models_{Adv} \phi_1\ U\phi_2 \Leftrightarrow\ \exists\ k \geq 0 : \pi \models_{Adv} \phi_1 U^{\leq k} \phi_2$.
- $s \models_{Adv} \mathrm{P}_{\bowtie p}[\psi] \Leftrightarrow\ P_{s_{Adv}}(\{\pi | \pi_{Adv} \models \psi\}) \bowtie p$, where $P_{s_{Adv}}$ represents the probability measure over all paths π starting from state s under adversary Adv.

3 Interface-based Verification Approach

This section provides a comprehensive overview of our methodology for verifying a PCTL property in a probabilistic system. The explanation is structured into two key subsections. The first subsection, i.e., Section 3.3, introduces a minimization algorithm that aims at reducing a Probabilistic Automaton (PA) while considering a PCTL property. Suppose S is a probabilistic system composed of n Probabilistic Automata (PAs), represented as $\mathrm{M}_{j:1\leq j\leq n}$. The system S can be denoted by the expression $\mathrm{M}_1 \parallel_{i_1} \cdots \parallel_{i_{n-1}} \mathrm{M}_n$, where "$\parallel_i$" signifies a composition operation involving synchronization between PAs through a particular interface i. Crucial properties associated with the composition operator "$\parallel_i$" are commutativity and associativity.

3.1 Minimization Phase

In an attempt to manage the complexity of model composition, we propose a strategy to reduce the behavior of a model utilizing a minimization operator. This operator is defined in Definition 3 and represented by "$\downarrow$". We define $M' = M \downarrow_{ap}$ as the operation of minimizing the behavior of M to obtain M' with respect to the atomic proposition ap. We also use $\mathcal{L}(ap)$ to denote the set of PCTL formulas that include $ap \in AP$.

Definition 3 (Model Minimization). *The process of minimizing a model M to $M \downarrow_{ap}$ should abide by the following rules:*

1. *Rule 1:* $\forall ap \notin AP_{s_i} : M(s_{i-1} \rightarrow s_i \rightarrow s_{i+1}) \downarrow_{ap} = M(s_{i-1} \rightarrow s_{i+1})$.
2. *Rule 2:* $\forall ap \notin AP_{s_i} : M(s_{i-1} \rightarrow_{p_1} s_i \rightarrow_{p_2} s_{i+1}) \downarrow_{ap} = M(s_{i-1} \rightarrow_{p_1 \times p_2} s_{i+1})$.
3. *Rule 3:* $\forall ap \notin AP_{s_i} \cap AP_{s'_i} : M(s_{i-1} \rightarrow_p s_i \rightarrow_{p_1} s_{i+1}; s_i \rightarrow_{1-p_1} s'_{i+1}; s_{i-1} \rightarrow_{1-p} s'_i \rightarrow_{p_2} s_{i+1}; s'_i \rightarrow_{1-p_2} s'_{i+1}) \downarrow_{ap} = M(s_{i-1} \rightarrow_{p_1 \times p \times \frac{1}{p+p'} + p_2 \times \frac{p'}{p+p'}} s_{i+1}; s_{i-1} \rightarrow_{1-p_1 \times \frac{p}{p+p'} + p_2 \times \frac{p'}{p+p'}} s'_{i+1})$.

In this paper, we describe a relation between two Probabilistic Automata (PAs), denoted as M and M' and symbolized as $M \mathcal{R} M'$. This relationship is characterized by the concept of weak simulation, which incorporates the idea of observable actions interspersed between invisible actions. We denote the latter

using the symbol "↓". Formally, the probabilistic weak simulation relation introduces the concept of observable action "a", which is preceded and followed by sequences of invisible steps. This weak transition is notated as $s \stackrel{a}{\Longrightarrow} P$, where P is the distribution over states reached from state s through a series of combined steps.

Definition 4 (Probabilistic Weak Simulation). *A probabilistic weak simulation between two Probabilistic Automata (PAs) M_1 and M_2 is a relation $\mathcal{R} \subseteq S_1 \times S_2$ if, and only if:*

1. *Each initial state of M_1 is related to at least one initial state of M_2, and*
2. *For each pair of states (s_1, s_2) such that $s_1 \mathcal{R} s_2$, and for each transition $s_1 \stackrel{a}{\rightarrow} \mu_1$ in M_1, there exists a weak combined transition $s_2 \stackrel{a}{\Longrightarrow} \mu_2$ in M_2 such that $\mu_1 \sqsubseteq_{\mathcal{R}} \mu_2$.*

In the above definition, $\sqsubseteq_{\mathcal{R}}$ is the lifting of relation $\mathcal{R}$ to probability distributions, achieved by employing a weight function as introduced by Segala [16]. This weight function, denoted by $\triangle$, associates each state in M_1 with a set of states in M_2, each assigned a particular probability. The formal definition of the weight function is provided below.

Definition 5 (Weight Function). *A function $\triangle : S \times S' \rightarrow [0, 1]$ is a weight function for the two distributions $\mu_1, \mu_2 \in Dist(S)$ with respect to $\mathcal{R} \subseteq S \times S'$, if and only if:*

1. *If $\triangle(s_1, s_2) > 0$ then $(s_1, s_2) \in \mathcal{R}$,*
2. *For all $s_1 \in S$: $\sum_{s_2 \in S} \triangle(s_1, s_2) = \mu_1(s_1)$, and*
3. *For all $s_2 \in S'$: $\sum_{s_1 \in S} \triangle(s_1, s_2) = \mu_2(s_2)$.*

Below, Proposition 1 establishes that the relation between two Probabilistic Automata, M and M', is a probabilistic weak simulation.

Proposition 1 (Minimization Relation). *Given two Probabilistic Automata, M and M', we denote the probabilistic weak simulation of M by M' as $M \precsim_w M'$, where M' is the result of a reduction process applied to M which abstracts away invisible actions, symbolized by $M' = M \downarrow_{ap}$.*

Proof. The proof is provided in the appendix section. □

Property 1 (Minimization Preservation). Given a property ϕ expressed in the PCTL language $\mathcal{L}(ap)$, and a set of atomic propositions ap that are relevant in the context of M, if the minimized version of M, denoted by $M \downarrow_{ap}$, satisfies ϕ, then the original PA M must also satisfy ϕ. Mathematically, this is expressed as $\forall \phi \in \mathcal{L}(ap), ap \in AP_M : \mathrm{M} \downarrow_{ap} \models \phi \Rightarrow \mathrm{M} \models \phi$.

Proof. The proof is given in the appendix. □

Proposition 2 ($\precsim_w$ Composition). *The probabilistic weak simulation relation preserves the composition of PAs: if M_1 is weakly simulated by M_1', then the composition of M_1 with another PA M_2 is weakly simulated by the composition of M_1' with the same M_2. Mathematically, this is expressed as $M_1 \precsim_w M_1' \Rightarrow M_1 \parallel_i \mathrm{M}_2 \precsim_w M_1' \parallel_i \mathrm{M}_2$.*

Proof. The proof is in the appendix. □

Definition 6 (Minimization Rule). *Given a composed PA $\mathrm{M}_1 \parallel_i \mathrm{M}_2$, the minimization rule prescribes the following transformations:*

- *Construct M_2' as the minimized version of M_2 with respect to the shared interface i, formally $M_2' \equiv M_2 \downarrow_{ap(i)}$. Here, M_2' represents the behavior of M_2 that is observable by M_1.*
- *For any property ϕ expressed in the language $\mathcal{L}(ap(i) \cup AP_{M_1})$, if $\mathrm{M}_1 \parallel_i \mathrm{M'}_2$ satisfies ϕ, then $\mathrm{M}_1 \parallel_i \mathrm{M}_2$ also satisfies ϕ. Mathematically, $\forall \phi \in \mathcal{L}(ap(i) \cup AP_{M_1}) : \mathrm{M}_1 \parallel_i \mathrm{M'}_2 \models \phi \Rightarrow \mathrm{M}_1 \parallel_i \mathrm{M}_2 \models \phi$.*

Theorem 1 (Soundness). *The minimization rule, as specified in Definition 6, is sound.*

Proof. The proof of the soundness is given in the appendix. □

3.2 The Composition Phase

In this section, we detail the process of decomposing a global property into local ones. Definition 7 introduces the decomposition operator "$\natural$", which enables compositional verification by substituting the propositions of a PA with propositions related to its interfaces. This operator utilizes the substitution notation $Q[z/y]$ from the π-calculus, which denotes that in the structure Q, the term z replaces the term y.

Definition 7 (PCTL Property Decomposition). *Let ϕ be a PCTL property to be verified on $\mathrm{M}_1 \parallel_i \mathrm{M}_2$. The decomposition of ϕ into ϕ_1 and ϕ_2 is denoted by $\phi = \phi_1 \natural_i \phi_2$, where:*

1. *$AP(\phi) = (AP(\phi_1) \cup AP(\phi_2)) \backslash \{ap(i)\}$, with $ap(i)$ representing the atomic propositions related to interface i,*
2. *$AP(\phi_1)$ is a subset of AP_{M_1},*
3. *$AP(\phi_2)$ is a subset of AP_{M_2},*
4. *$\phi_1 = \phi([\![ap(i)/AP_{M_2}]\!])$,*
5. *$\phi_2 = \phi([\![ap(i)/AP_{M_1}]\!])$.*

Property 2 demonstrates that the decomposition operator "$\natural$" exhibits commutative and associative properties.

Property 2. For $\mathrm{M}_1 \parallel_i \mathrm{M}_2$, the decomposition operator $\natural_i$ is both commutative and associative:

1. $\phi_1 \natural_i \phi_2 \equiv \phi_2 \natural_i \phi_1$.
2. $\phi_1 \natural_{i_1} (\phi_2 \natural_{i_2} \phi_3) \equiv (\phi_1 \natural_{i_1} \phi_2) \natural_{i_2} \phi_3$.

Proof. The proof approach for Property 2 follows a style analogous to that of Property **??**, and is based on Definition 7. □

Proposition 3. *The decomposition of the PCTL property ϕ using the decomposition operator $\natural$ for the parallel composition* $\mathrm{M}_1 \parallel_i \mathrm{M}_2$ *is sound.*

Proof. The proof of Proposition 4 is in the appendix. □

The key advantage of Proposition 4 is its ability to derive the deduction rule for the interface operator, as outlined in Theorem 2.

Theorem 2 (Compositional Reduction - CR). *Let ϕ be a PCTL property to be verified on M, such that: $M = M_1 \parallel_i M_2$ and $\phi = \phi_1 \natural_i \phi_2$. The following deduction rule applies:*

$$\frac{M_1 \parallel_i M_2 \downarrow_{ap(i)} \models \phi_1 \qquad M_2 \models \phi_2 \qquad \phi = \phi_1 \natural_i \phi_2}{M_1 \parallel_i M_2 \models \phi}$$

Property 3 (CR-Symmetry). The compositional reduction rule is symmetric.

$$\frac{M_1 \parallel_i M_2 \downarrow_{ap(i)} \models \phi_1 \qquad M_2 \models \phi_2 \qquad \phi = \phi_1 \natural_i \phi_2}{M_1 \parallel_i M_2 \models \phi}$$

Proof. The proof follows the same structure as the proof of Theorem 2. □

Proposition 4. *The decomposition of a PCTL property ϕ by the decomposition operator $\natural$ for* $\mathrm{M}_1 \parallel_i \mathrm{M}_2$ *is sound.*

Proof. The proof of Proposition 4 is in the appendix. □

3.3 The Verifition Phase

Our approach fundamentally relies on the PRISM model checker to ensure the soundness of the probabilistic system under scrutiny, which is modelled as a composition of Probabilistic Automata (PAs). In this section, we delve into defining the syntax and semantics inherent in the PRISM programming language. PRISM has a versatile architecture that can accommodate a range of probabilistic models. These include Discrete-Time Markov Chains (DTMCs), Continuous-Time Markov Chains (CTMCs), Markov Decision Processes (MDPs), Probabilistic Timed Automata (PTAs), and Probabilistic Automata (PAs). It's noteworthy to mention that within the PRISM framework, PAs are referred to as MDPs. However, in the process of formalizing the approach for this study, our primary focus will be directed towards PAs. The choice of focusing on PAs is due to their ability to model a wide array of complex, probabilistic behaviors that are crucial for our study.

A probabilistic system S described by a PRISM program P comprises a set of n modules ($n > 0$). Each module's state is defined by an evaluation of a set of finite-ranging local variables. The system's global state is the union of evaluations of the local variables V_l and the global variables V_g, denoted as $V = V_g \cup V_l$. The behavior of each module is defined by a set of guarded commands.

Each command dictates the primary behavior changes of P. A command is of the form: [a] g $\rightarrow p_1 : u_1 + ... + p_m : u_m$ or [a] g $\rightarrow u$. This implies that, for the action 'a', if the guard 'g' holds, then an update 'u_i' is enabled with a probability 'p_i'. Guards are propositional formulas based on expressions' comparisons. An update 'u_i' is a conjunction of assignments to variables: $(v'_j = val_j) \& \cdots (v'_k = val_k)$, where v_i are variables and val_i are values evaluated via expressions, ensuring type consistency.

Definition 8 (PRISM Command). *A PRISM command c is a tuple $c = (a, g, u)$ where:*

- *a is an action label,*
- *g is a predicate over V,*
- *$u = \{(p_i, u_i) | m > 1, i \leq m, \sum_{i=1, p_i > 0}^{m} p_i = 1$ and $u_i = \{(v, eval(v))\}\}$ where $eval : V \rightarrow \mathbb{N} \cup \{true, false\}$ assigns an integer or a boolean value to each variable $v \in V$. If $p_i = 1$, we omit the probability.*

A module, which describes the behavior of a system's sub-part, can be considered as a set of commands. Formally, it is defined as follows:

Definition 9 (PRISM Module). *A PRISM module M is a tuple $M = (var, init, cmd)$ where:*

- *var is a finite set of local variables for the module,*
- *$init$ are the initial values of var,*
- *$cmd = \{c_i : 0 \leq i \leq m\}$ is a set of commands defining the module's behavior.*

A system S, comprising n parts, can be described by a PRISM program P containing n modules. The system components are combined using a Communicating Sequential Processes (CSP) expression.

Definition 10 (PRISM System). *A PRISM system is a tuple $P = (var, exp, M_1, \ldots, M_n, sys)$ where:*

- *$var = V_G \bigcup_{i=1}^{n} V_{li}$ is a finite set comprising the union of global and local variables,*
- *exp is a global logic expression,*
- *$M_1, \ldots, M_n$ is a finite set of modules,*
- *sys is a CSP algebraic expression defining the combination of the models.*

4 Experimental Results

We implement our proposed framework on the Probabilistic Broadcast Protocol (PBP), the Randomized Dining Philosophers (RDP), and Leader Election Protocol (LEP) benchmarks[5]. To compare the results of our approach, we verify PCTL properties on Probabilistic Automata (PA) models with and without our method applied. We measure and compare the number of states (#S), the number of transitions (#T), the time needed for the model construction (T_c in seconds), and the time needed for the model verification (T_v in seconds). We undertake this comparison with the aim of presenting empirical results that validate the efficiency and effectiveness of our approach.

For the PBP benchmark, we aim to measure "the minimum probability that the message sent by the base node 0 is successfully received by node j", where j ranges from 1 to 8. This property can be expressed in PCTL as follows:

$$P_{\min} =? \left[G \left((\text{active}_0 \wedge \neg\text{send}_0) \Rightarrow F \left(\neg\text{active}_j \wedge \neg\text{send}_j\right)\right)\right] \tag{3}$$

Following the PBP model and the decomposition rule (Definition 7), we decompose Property 3 into two properties, 4 and 5. Property 4 concerns the atomic propositions of node 0 and the propositions related to the interaction between nodes 0 and j. As per Definition 7, the propositions of Property 5 only belong to node j's propositions.

$$P_{\min} =? \left[G \left((\text{active}_0 \wedge \neg\text{send}_0) \Rightarrow F \left(\text{active}_j \wedge \neg\text{send}_j\right)\right)\right] \tag{4}$$

$$P_{\min} =? \left[G \left((\text{active}_j \wedge \neg\text{send}_j) \Rightarrow F \left(\neg\text{active}_j \wedge \neg\text{send}_j\right)\right)\right] \tag{5}$$

Table 1 shows the verification costs for these properties.

j	1	2	3	4	5	6	7	8
Res(3)	0.246	0.0615	0.246	0.104	0.035	0.0615	0.035	0.0148
Res(4)	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Res(5)	0.246	0.0615	0.246	0.104	0.035	0.0615	0.035	0.0148

Table 1: Verification Cost for PBP Benchmark

For the RDP benchmark, we aim to verify the proposition "if a philosopher is hungry, then eventually some philosopher eats". Here, "hungry" refers to any action prior to the "eat" action. This property can be expressed in PCTL as follows:

$$\text{"hungry"} \Rightarrow P \geq 1[\text{true } U \text{ "eat"}] \tag{6}$$

[5] `http://www.prismmodelchecker.org/benchmarks`

This benchmark does not incorporate interfaces, so we confine our experiments to the minimization algorithm. The verification costs of Property 6 are shown in Table 2, where $\#Phil$ denotes the number of philosophers.

	PRISM					CR-Approach				
$\#Phil$	$\#S$	$\#T$	T_c	T_v	Res	$\#S$	$\#T$	T_c	T_v	Res
3	956	3696	0.027	0.379	T	596	2286	0.025	0.135	T
4	9440	48656	0.127	8.357	T	3964	20360	0.000905	4.396	T
5	93068	599600	0.358	124.707	T	36788	233208	0.006481	49.448	T
6	1288574	9879228	1.011	week	T	460336	3475264	0.56	35289.578	T

Table 2: Verification Cost for RDP Benchmark

For the LEP benchmark, we evaluate "the minimum probability that two different processes do not have the same source". For two different processes P_i and P_j, the PCTL property for verification is expressed as follows:

$$P_{\min} =? [(P_i \neq P_j) \Rightarrow (\text{source}(P_i) \neq \text{source}(P_j))] \tag{7}$$

Table 3 presents the verification cost of Property 7, where $\#Proc$ is the number of processes. Similar to the RDP benchmark, LEP does not incorporate common interfaces.

	PRISM					CR-Approach				
$\#Proc$	$\#S$	$\#T$	T_c	T_v	Res	$\#S$	$\#T$	T_c	T_v	Res
3	27	108	0.023	0.005	1	8	24	0.0	0.005	1
4	81	432	0.028	0.034	1	16	64	0.020	0.031	1
5	243	1620	0.113	0.249	1	32	160	0.036	0.041	1
6	729	5832	0.154	0.893	1	64	384	0.044	0.43	1
7	6561	69984	0.327	9.994	1	128	896	0.047	0.49	1

Table 3: Verification Cost For LEP Benchmark

The above results demonstrate that our verification framework preserves the verification of PCTL properties while significantly reducing the verification size and time.

5 Conclusion

In this paper, we presented a verification framework aimed at enhancing the scalability of probabilistic model-checking. In particular, we targeted systems modeled as probabilistic automata, which accommodate both nondeterminism

and probabilistic choice behaviors. Our proposed framework harnesses probabilistic abstraction to disregard and amalgamate behaviors deemed irrelevant to a specified PCTL property. In addition, we introduced a probabilistic compositional verification mechanism to optimize model-checking efficiency. This mechanism operates by decomposing a global PCTL property into local properties corresponding to the interfaces between PAs. We established the soundness of our algorithms by elucidating the relationship between the abstract and concrete PAs, demonstrating that this relationship upholds the satisfaction of PCTL properties. Considering our reliance on PRISM, we proposed dedicated syntax and semantics for PAs specified within this context. Finally, we substantiated the effectiveness of our approach by applying it to a benchmark.

Looking forward, we aim to extend our approach along several trajectories. First, we intend to integrate our algorithms within the PRISM model checker. Second, we plan to expand our proposed abstraction to address other formalisms such as probabilistic timed and priced automata, stochastic Petri nets, and SysML activity diagrams. Furthermore, we aim to investigate other abstraction approaches, specifically data abstraction targeting system features like time and data. Lastly, we propose to explore strategies to reduce property within the model.

References

1. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press (1999)
2. Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, Ph.: Systems and Software Verification. Springer (2001)
3. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press (may 2008)
4. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: Automated Verification Techniques for Probabilistic Systems. In: Formal Methods for Eternal Net. Soft. Sys. (SFM'11). LNCS, Springer (2011) 53–113
5. Xin-feng, Z., Jian-dong, W., Xin-feng, Z., Bin, L., Jun-wu, Z., Jun, W.: Methods to tackle state explosion problem in model checking. In: Proceedings of the 3rd int. conf. on IITA, NJ, USA, IEEE Press (2009) 329–331
6. Berezin, S., Campos, S., Clarke, E.M.: Compositional reasoning in model checking. In: In Proc. COMPOS. 81–102
7. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of Probabilistic Real-Time Systems. In: CAV. LNCS, Springer (2011) 585–591
8. Hahn, E.M., Norman, G., Parker, D., Wachter, B., Zhang, L.: Game-based Abstraction and Controller Synthesis for Probabilistic Hybrid Systems. In: Proc. of the 2011 Eighth Int. Conf. on Quant.e Eval. of SysT. QEST '11 (2011) 69–78
9. Kattenbelt, M., Kwiatkowska, M., Norman, G., Parker, D.: A game-based abstraction-refinement framework for markov decision processes. Form. Methods Syst. Des. **36**(3) (September 2010) 246–280
10. Donaldson, A., Miller, A., Parker, D.: Language-level symmetry reduction for probabilistic model checking. In: Quantitative Evaluation of Systems, 2009. QEST '09. Sixth International Conference on the. (sept. 2009) 289 –298
11. Donaldson, A., Miller, A.: Symmetry Reduction for Probabilistic Model Checking Using Generic Representatives. In: Automated Technology for Verification and Analysis. Volume 4218 of Lec. Not. in Comp. Sc. (2006) 9–23

12. Baier, C., D'Argenio, P., Groesser, M.: Partial order reduction for probabilistic branching time. Electron. Notes Theor. Comput. Sci. **153**(2) (May 2006) 97–116
13. Feng, L., Han, T., Kwiatkowska, M., Parker, D.: Learning-based compositional verification for synchronous probabilistic systems. In: Proc. of the 9th int. conf. on Aut. tech. for verif. and analy. ATVA'11, Springer-Verlag (2011) 511–521
14. Feng, L., Kwiatkowska, M., Parker, D.: Automated learning of probabilistic assumptions for compositional reasoning. In: Proc. of the 14th int. conf. on Fund. approaches to software engineering. FASE'11/ETAPS'11 (2011) 2–17
15. Feng, L., Kwiatkowska, M., Parker, D.: Compositional verification of probabilistic systems using learning. In: Proceedings of the 2010 Seventh Int. Conf. on the Quant. Eval. of Sys. QEST '10, IEEE Computer Society (2010) 133–142
16. Segala, R.: A compositional trace-based semantics for probabilistic automata. In Lee, I., Smolka, S., eds.: CONCUR '95: Concurrency Theory. Volume 962 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (1995) 234–248

A Appendix

Proposition 5 (Minimization Relation). *Given two Probabilistic Automata, M and M', we denote the probabilistic weak simulation of M by M' as $M \precsim_w M'$, where M' is the result of a reduction process applied to M which abstracts away invisible actions, symbolized by $M' = M \downarrow_{ap}$.*

Proof. The proof of Proposition 1 centers around the establishment of a relationship or correspondence between the states of M and M'. This correspondence takes the form of a mapping realized through the development of a weight function.

Step 1: Construction of the Weight Function: The weight function is a critical tool used to build a bridge between each state of the original Probabilistic Automaton M and the states in the minimized automaton M'. This function, denoted as $\triangle$, effectively measures how much each state in M contributes to each state in M' by associating states in M to those in M' according to certain probability values. This function is the centerpiece of our proof.

Step 2: Application of the Minimization Rules: To minimize M into M', we apply a set of rules (as outlined in Definition 3). The minimization rules serve to reduce the complexity of M by abstracting away certain transitions. This process is denoted as $M' = M \downarrow_{ap}$, which signifies that M' is obtained from M by abstracting away invisible actions.

Step 3: Verification of the Weight Function: After applying the minimization rules, it is crucial to confirm that the weight function we initially set up still accurately maps the states of M to their corresponding states in M'. To do this, we ensure that for each state s in M, the sum of weights associated with s equals to the corresponding distribution in M'. This step verifies that the behavior of M and M' is indeed the same, despite the reduction in transitions.

Through this thorough process, we demonstrate that the weight function can successfully map the states of M and M', thereby validating the weak simulation relation, and hence proving the proposition. □

Property 4 (Minimization Preservation). Given a property ϕ expressed in the PCTL language $\mathcal{L}(ap)$, and a set of atomic propositions ap that are relevant in the context of M, if the minimized version of M, denoted by $M \downarrow_{ap}$, satisfies ϕ, then the original PA M must also satisfy ϕ. Mathematically, this is expressed as $\forall \phi \in \mathcal{L}(ap), ap \in AP_M : \mathrm{M} \downarrow_{ap} \models \phi \Rightarrow \mathrm{M} \models \phi$.

Proof. The proof for the preservation of PCTL properties through minimization can be obtained through an inductive argument based on the structure of the PCTL properties. The inductive base case can be the satisfaction of ϕ in $M \downarrow_{ap}$, and the inductive step can assume the property holds for a given structure and then prove it for the next level of structure complexity. □

Proposition 6 ($\precsim_w$ Composition). *The probabilistic weak simulation relation preserves the composition of PAs: if M_1 is weakly simulated by M_1', then the composition of M_1 with another PA M_2 is weakly simulated by the composition of M_1' with the same M_2. Mathematically, this is expressed as $M_1 \precsim_w M_1' \Rightarrow M_1 \parallel_i \mathrm{M}_2 \precsim_w M_1' \parallel_i \mathrm{M}_2$.*

Proof. The proof of Proposition 2 relies on the previously established Probabilistic Weak Simulation Relation (Proposition 1) and the specific details of the PA composition (Definition 2). More specifically, we must show that each transition in the composed $M_1 \parallel_i \mathrm{M}_2$ can be simulated by a corresponding transition in $M_1' \parallel_i \mathrm{M}_2$, which in turn relies on the definition of weak simulation. □

Theorem 3 (Soundness). *The minimization rule, as specified in Definition 6, is sound.*

Proof. The soundness of the minimization rule can be proven by combining the results of Proposition 2 and Property 1. In detail, Property 1 ensures that minimization preserves the PCTL properties, and Proposition 2 guarantees that the weak simulation preserves the composition. Hence, the combination of these two results ensures that the minimization process, which involves both reducing M_2 to M_2' and composing M_1 with M_2', does not affect the satisfaction of ϕ by the composed system, confirming the soundness of the minimization rule. □

Proposition 7. *The decomposition of the PCTL property ϕ using the decomposition operator $\natural$ for the parallel composition $\mathrm{M}_1 \parallel_i \mathrm{M}_2$ is sound.*

Proof. The proof of Proposition 4 is built upon Definition 2 and Definition 7. It proceeds by using structural induction on the structure of PCTL formulas.

As an example, consider the "until" operator "U". Let $\phi = \alpha_1 U \alpha_2$ where $\alpha_1 \in AP_{M_1}$ and $\alpha_2 \in AP_{M_2}$. From here, we can deduce the following:

1. $\phi_1 = \alpha_1 U \alpha(i)$ and $\phi_2 = \alpha(i) U \alpha_2$, as per Definition 7.
2. From the hypothesis, it is assumed that $M_1 \parallel_i \mathrm{M}_2 \models \alpha_1 U \alpha(i)$ and $M_1 \parallel_i \mathrm{M}_2 \models \alpha(i) U \alpha_2$.
3. According to the semantics of PCTL, we have $M_1 \parallel_i \mathrm{M}_2 \models (\alpha(i) U \alpha_2) \wedge (\alpha_1 U \alpha(i))$.

4. Hence, we obtain $M_1 \parallel_i \mathrm{M}_2 \models \phi_1 U \phi_2$ based on PCTL semantics.
5. Therefore, it follows that $M_1 \parallel_i \mathrm{M}_2 \models \phi_1 \natural_i \phi_2$, which is exactly what Proposition 4 claims.

By adopting a similar proof strategy for the rest of the PCTL operators, we can assert that Proposition 4 is valid. □

Proposition 8. *The decomposition of a PCTL property ϕ by the decomposition operator $\natural$ for $\mathrm{M}_1 \parallel_i \mathrm{M}_2$ is sound.*

Proof. The proof of Proposition 4 relies on Definition 2 and Definition 7, and it proceeds by a structural induction on the PCTL structure. For each $\alpha_1 \in AP_{M_1}$, $\alpha_2 \in AP_{M_2}$, and i in the interface, we consider the until operator "U" and let $\phi = \alpha_1 U \alpha_2$. We can perform the following steps:

1. Define $\phi_1 = \alpha_1 U \alpha(i)$ and $\phi_2 = \alpha(i) U \alpha_2$ based on Definition 7.
2. Assume that $M_1 \parallel_i \mathrm{M}_2 \models \phi_1$ and $M_1 \parallel_i \mathrm{M}_2 \models \phi_2$.
3. From PCTL semantics, it follows that if both ϕ_1 and ϕ_2 hold, then their conjunction also holds: $M_1 \parallel_i \mathrm{M}_2 \models (\phi_1) \wedge (\phi_2)$.
4. Furthermore, again by PCTL semantics, if the conjunction of two properties holds, then their until operation also holds: $M_1 \parallel_i \mathrm{M}_2 \models \phi_1 U \phi_2$.
5. Finally, by Proposition 4, we can express the until operation as a decomposition operation: $M_1 \parallel_i \mathrm{M}_2 \models \phi_1 \natural_i \phi_2$.

By repeating the above style of proof for the remaining operators in the PCTL structure, we find that Proposition 4 holds. □