Software Engineering Group
Department of Computer Science
Nanjing University
http://seg.nju.edu.cn

Technical Report No. NJU-SEG-2016-IJ-004

2016-IJ-004

A Game-based Approach PCTL* Stochastic Model Checking with Evidence

Yang Liu, Xuandong Li, Yan Ma

Journal of Computer Science and Technology 2016

Liu Y, Li XD, Ma Y. A game-based approach for PCTL* stochastic model checking with evidence. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 31(1): 198–216 Jan. 2016. DOI 10.1007/s11390-016-1621-y

A Game-Based Approach for PCTL* Stochastic Model Checking with Evidence

Yang Liu [1,2], *Member, CCF, ACM*, Xuan-Dong Li [1], *Fellow, CCF*, and Yan Ma [3,*]

[1] *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210046, China*

[2] *Department of Computer Science, School of Computing, National University of Singapore, Singapore 117417, Singapore*

[3] *College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics Nanjing 210016, China*

E-mail: yangliu@seg.nju.edu.cn; lxd@nju.edu.cn; mayan1616@163.com

Received July 10, 2014; revised June 19, 2015.

Abstract Stochastic model checking is a recent extension and generalization of the classical model checking, which focuses on quantitatively checking the temporal property of a system model. PCTL* is one of the important quantitative property specification languages, which is strictly more expressive than either PCTL (probabilistic computation tree logic) or LTL (linear temporal logic) with probability bounds. At present, PCTL* stochastic model checking algorithm is very complicated, and cannot provide any relevant explanation of why a formula does or does not hold in a given model. For dealing with this problem, an intuitive and succinct approach for PCTL* stochastic model checking with evidence is put forward in this paper, which includes: presenting the game semantics for PCTL* in release-PNF (release-positive normal form), defining the PCTL* stochastic model checking game, using strategy solving in game to achieve the PCTL* stochastic model checking, and refining winning strategy as the evidence to certify stochastic model checking result. The soundness and the completeness of game-based PCTL* stochastic model checking are proved, and its complexity matches the known lower and upper bounds. The game-based PCTL* stochastic model checking algorithm is implemented in a visual prototype tool, and its feasibility is demonstrated by an illustrative example.

Keywords PCTL*, stochastic model checking, game semantics, strategy, evidence

1 Introduction

Model checking[1-3] is an automatic and complete method for deciding whether the system model meets the property specification, which converts checking system into a decidable problem by means of restricting the system model as a finite state model and specification as the propositional temporal logic. Model checking emphasizes on the absolute guarantee of correctness. However, computer systems are becoming large and complex, and some of them are accompanied with the stochastic behavioral characteristics. The reasons for stochastic behavior characteristics[4] exhibited in systems can be classified as: 1) the system contains randomness itself, such as using the probabilistic algorithm or the randomized algorithm; 2) the running environment of the system is complex, which results in the occurrence of random failures, such as failing to invoke some components in the system or the loss of message; 3) for performance evaluation and analysis, the random variable is factitiously employed to capture performance index. It is named stochastic model checking or probabilistic model checking[5-6] to quantitatively analyze the complex system with stochastic behaviors

Regular Paper

The work was supported by the National Natural Science Foundation of China under Grant Nos. 61303022 and 61472179, the China Postdoctoral Science Foundation under Grant No. 2013M531328, the Natural Science Foundation of Shandong Province of China under Grant No. ZR2012FQ013, the Project of Shandong Province Higher Educational Science and Technology Program under Grant No. J13LN10, and the Science and Technology Program of Taian of China under Grant No. 201330629. This work was partially supported by Jiangsu Collaborative Innovation Center of Novel Software Technology and Industrialization of China.

*Corresponding Author

by model checking. In recent decade, stochastic model checking has aroused widespread concern in formal verification community and made a great progress. Its representative groups include Software Modeling and Verification Group at RWTH Aachen University①, PRISM Group at the University of Oxford②, Dependable Systems and Software group at Saarland University③, Algebraic and Logic Foundations of Computer Science group at TU Dresden④ and so on. Now, stochastic model checking has been applied to the correctness analysis of probabilistic program[7], system performance analysis[8], communication protocols reliability analysis[9], the quality optimization of service flow[10], and even the systems biology[11].

Stochastic model checking is the extension and generalization of classical model checking. In the setting of stochastic model checking, the models commonly used are DTMC (discrete time Markov chain), MDP (Markov decision process), CTMC (continuous time Markov chain) and so on; the quantitative property specifications commonly used are PCTL (probabilistic computation tree logic)[12], LTL (linear temporal logic)[13] with probability bounds, PCTL*[12], CSL (continuous stochastic logic)[14-15] and so on. Thereinto, PCTL* can be seen as the extension of PCTL and LTL with probability bounds, and it is strictly more expressive than anyone of them. At present, the PCTL* stochastic model checking is obtained by the appropriate combination of recursive descent procedure (as for PCTL stochastic model checking) and LTL with probability bounds stochastic model checking. As shown in Fig.1, the main procedures of PCTL* stochastic model checking are as follows: 1) like PCTL, bottom-up traverse the parse tree of PCTL* formula; 2) for PCTL* state formula $\mathrm{P}_{\sim p}(\Psi)$, translate Ψ into LTL formula Ψ' by replacing each maximal state-subformula of Ψ with a fresh atomic proposition; 3) transform Ψ' to DRA (deterministic Rabin automaton) by ω automaton; 4) compute the product of stochastic system model M and DRA, and compute accepting BSCC (bottom strongly connected component) of product by graph analysis; 5) get the probability p_{BSCC} to reach an accepting BSCC by solving linear equation system; 6) decide whether M satisfies $\mathrm{P}_{\sim p}(\Psi)$ by comparing p_{BSCC} and $\sim p$.

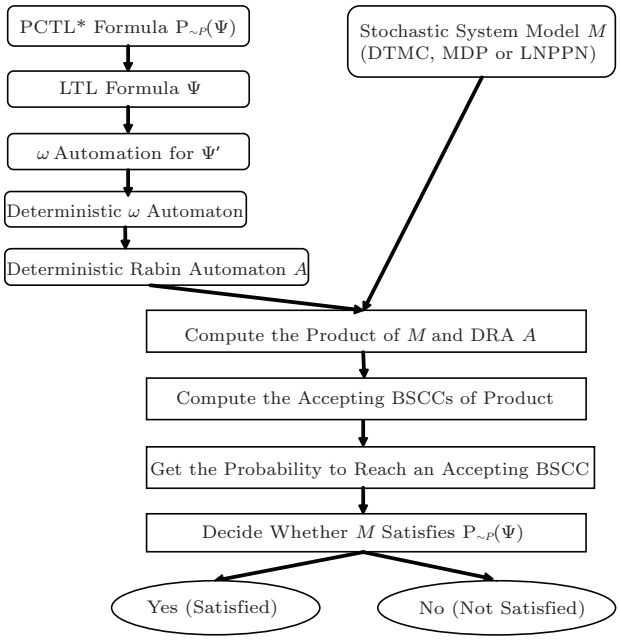


Fig.1. Traditional PCTL* stochastic model checking.

The above algorithm is based on Tarski's denotational semantics (truth as a predicate) at essence. The deficiencies of the algorithm are: 1) the algorithm process is complicated; 2) the returned result is "yes" or "no", without other further information for supporting result. Therefore, the result returned by traditional PCTL* stochastic model checking algorithm is obscure for the specifier.

In this paper, as shown in Fig.2, using nondeterministic probabilistic Petri net (NPPN)[16] with label as the stochastic system model, we put forward a complete and rigorous PCTL* stochastic model checking algorithm based on Hintikka's operational semantics[17] (truth as winning strategy in two-player game, i.e., game semantics), which can easily provide evidence for verification or refutation on the premise of keeping the same complexity with traditional PCTL* stochastic model checking algorithm. The evidence can be used for helping the specifier to understand why the property is true or false in the model. To this end, the rest of the paper is organized as follows. In next section, we define the label-extended NPPN and the logical semantics PCTL* about it. Section 3 presents the game semantics for PCTL* in release-PNF (positive normal

①http://moves.rwth-aachen.de/, Sept. 2015.
②http://www.prismmodelchecker.org/, Sept. 2015.
③http://depend.cs.uni-sb.de/index.php?id=156, Sept. 2015.
④http://www.inf.tu-dresden.de/index.php?node_id=1438&ln=en, Sept. 2015.

formal). Based on this, Section 4 uses the strategy solving in game to implement the PCTL* stochastic model checking algorithm, and Section 5 describes how to automatically construct the evidence for verification and refutation in PCTL* stochastic model checking. In Section 6, we illustrate the validity and feasibility of the techniques from Section 3 to Section 5 by an illustrative example in our prototype tool. We discuss the related work in Section 7. Section 8 concludes the paper and points out the future work in this direction.

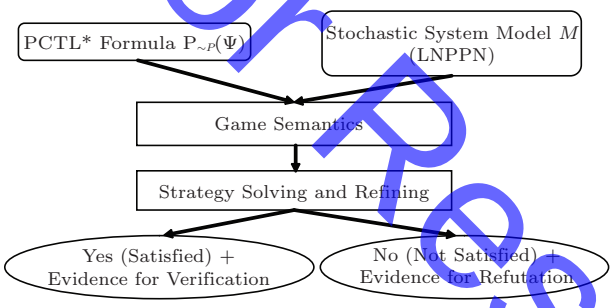


Fig.2. Game-based PCTL* stochastic model checking.

2 Extending NPPN with Labeling Function

NPPN[16] is a high-level formal model for modeling the nondeterministic and probabilistic behavior characteristics of systems, which is a natural development of the original Petri net by introducing probability measurement theory under the guidance of general net theory[18]. In this section, we extend the NPPN with labeling function and define its logical semantics.

2.1 Definition and Measurement for NPPN with Label

Definition 1 (NPPN with Label (LNPPN)). *The LNPPN can be defined as a 7-tuple* $M = (S, T; F, f; C; AP, L)$, *where: 1)* $T = (Transition, Prt)$, *Transition denotes the transition act,* $Prt \in [0, 1]$ *denotes the success probability of the transition, and* T *is the act transition* (AT) *with the probability equaling* 1, *or the probabilistic act transition* (AT, Prt) *with an act satisfying a certain probability distribution, or the pure probability transition* (PT) *without any act; if* $Prt = 0$, *it means that the transition is invalid; 2)* $S \cap T = \emptyset$, $S \cup T \neq \emptyset$, $F \subseteq S \times T \cup T \times S$, *which is the flow relation of net, and* $N = (S, AT, F)$ *is the pure net, where* AT *is the act transition with probability equaling* 1*; 3)* $f = f_T \cup f_S \cup f_{S\times T} \cup f_{T\times S}$, f_T*:* $T \to 2^{Transition\times Prt}$, f_S*:* $S \to [0, 1]$, $f_{T\times S}$*:* $T \times S \to [0, 1]$, $f_{S\times T}$*:* $S \times T \to [0, 1]$, *the value of* $f_{S\times T}$ *is determined by the nondeterminism of transitions, the value of* $f_{T\times S}$ *can be obtained from the probability value of* $f_T(t)$, *i.e.,* $\sigma_{Prt}(f_T(t))$, *and the value of* f_S *except initial place can be computed according to the value of* $f_{S\times T}$ *and* $f_{T\times S}$*; 4)* $\forall t \in T$, $\exists s \in S$, $f_{T\times S}(t \times s) = 1 - \sigma_{Prt}(f_T(t))$. $f_{T\times S}(t \times s) = 0$, *if* $\sigma_{Prt}(f_T(t)) = 1$, *which represents transition* (t, s) *and place* s *are invalid; 5)* C *is the set of nondeterminism classes, and each nondeterminism class is a set comprised of* (s, t_i)*; if* $\{(s, t_1), (s, t_2), \ldots, (s, t_n)\} \in C$, *then* $\sum_{i=1}^{n} f_{S\times T}(s, t_i) = 1$*; 6) AP is a set of atomic propositions; 7)* L*:* $S \to 2^{AP}$ *is the labeling function, which can express the requirements of users, i.e., the property.*

M is thought to be finite if S and T are finite, and the size of M is the number of places and transitions plus the number of pairs (s, t) with $f_{S\times T} > 0$ and (t, s) with $f_{T\times S} > 0$.

Definition 2 (Adversary (Scheduler, or Policy) for LNPPN). *An adversary for LNPPN is a function* $Adv : S^+ \to T$, *such that an* $i > 0$ *for all* $s_0 s_1 s_2 ... s_n \in S^+$. *The path* $\pi = s_0 \xrightarrow{t_1} s_1 \xrightarrow{t_2} \ldots \to s_i \ldots$ *is called an Adv-path if* $t_i = Adv(s_0 s_1 s_2 ... s_{i-1})$ *for all* $i > 0$.

Theorem 1. *LNPPN model is measurable with probability.*

Proof. The measurement of LNPPN is related to the nondeterminism. When the nondeterminism is solved by adversary, the LNPPN M is just a pure LPPN (probabilistic Petri net with label) $M' = (S, T; F, f; AP, L)$. The probability space over LPPN M' can be defined as $(\Omega, \sum Path(M'), Pr_s)$, where Ω is sample space, $\sum Path(M')$ is event set, and Pr_s is probability measure. $\Omega = Paths(M')$, which is the set of infinite paths with initial place. $\sum Path(M')$ is the least σ-algebra on $Paths(M')$ containing the cylinder set $Cylinder(\hat{\pi})$ for all finite paths $\hat{\pi}$ starting with initial place, and $Cylinder(\hat{\pi}) = \{\pi \in Paths(M') | \hat{\pi} \in pref(\pi)\}$ where $pref(\pi)$ denotes the set of prefix paths of path π. Pr_s extends uniquely to a probability measure $Pr_s : \sum Path(M') \to [0, 1]$, and the probability of cylinder set is

$$Pr_{s_0}(Cylinder(\hat{\pi})) = \begin{cases} 1, & \text{if } \hat{\pi} = s_0, \\ P(s_0, s_1) \times \ldots \times P(s_{k-1}, s_k), & \text{if } \hat{\pi} = s_0 s_1 \ldots s_k, \end{cases}$$

where s_0 is the initial place and $P(s_i, s_j) = f_{S\times T}(s_i, t) \times f_{T\times S}(t, s_j)$. Therefore, an LPPN M' is measurable, and an LNPPN M is also measurable. □

2.2 PCTL* Semantics for LNPPN

PCTL* is an extension of CTL*[1], which also can be regarded as extending PCTL by dropping the requirement that any temporal operator has to be proceeded by the state formula. In addition, it allows for Boolean combinations of path formulae. The logic PCTL* can capture quantitative branching property specification about DTMC, MDP and LNPPN. PCTL* state formulae over the atomic propositions set AP can be defined as: $\Phi ::= \text{true}|a|\Phi \wedge \Phi|\neg\Phi|\mathrm{P}_{\sim p}[\Psi]$, where $a \in AP$, Ψ is a path formula, $\sim \in \{<, \leqslant, >, \geqslant\}$ and $p \in [0,1]$ is the rational bound. The path formulae of PCTL* can be defined as: $\Psi ::= \Phi|\Psi \wedge \Psi|\neg\Psi|\mathrm{X}\Psi|\Psi\mathrm{U}\Psi$, where Φ is the PCTL* state formula. Other Boolean operators and temporal modalities can be derived from the above PCTL* definition. The events specified by the PCTL* path formulae are measurable[12].

Because an LNPPN model is measurable, the PCTL* semantics about LNPPN model can be defined as follows.

Definition 3 (PCTL* Semantics about LNPPN). *For an LNPPN model M, path $\pi = s_0 \xrightarrow{t_1} s_1 \xrightarrow{t_2} s_2...$(abbr. $\pi = s_0 s_1 s_2...$, if it is not related to t), place $s_i \in S$, the semantics of PCTL* formulae can be defined by structural induction:*

1) *a satisfaction relation $\vDash$ for place s and PCTL* state formulae*:

$$
\begin{aligned}
&s \vDash \text{true}, && \forall s \in S,\\
&s \vDash a, && iff\ a \in L(s),\\
&s \vDash \neg\Phi, && iff\ s \nvDash \Phi,\\
&s \vDash \Phi_1 \wedge \Phi_2, && iff\ s \vDash \Phi_1 \wedge s \vDash \Phi_2,\\
&s \vDash \mathrm{P}_{\sim p}[\Psi], && iff\ Prob(s,\Psi) \sim p,
\end{aligned}
$$

where $Prob(s,\Psi)$ is the probability measure of path set such that $\pi \vDash \Psi$, i.e., $Prob(s,\Psi) = Pr_s(\{\pi \in Path(s)|\pi \vDash \Psi\})$, and $Path(s)$ is the path set which starts with place s;

2) *a satisfaction relation $\vDash$ for path π in M and PCTL* path formulae*:

$$
\begin{aligned}
&\pi \vDash \Phi, && iff\ s_0 \vDash \Phi,\\
&\pi \vDash \neg\Psi, && iff\ \pi \nvDash \Psi,\\
&\pi \vDash \Psi_0 \wedge \Psi_1, && iff\ \pi \vDash \Psi_0 \wedge \pi \vDash \Psi_1,\\
&\pi \vDash \mathrm{X}\Psi, && iff\ s_1 s_2 s_3... \vDash \Psi,\\
&\pi \vDash \Psi_0 \mathrm{U} \Psi_1, && iff\ there\ exits\ n \geqslant 0, such\ that\\
& && s_n s_{n+1} s_{n+2}... \vDash \Psi_1,\ s_m s_{m+1} s_{m+2}...\\
& && \vDash \Psi_0 \text{ for } 0 \leqslant m < n.
\end{aligned}
$$

3 Game Semantics for PCTL* Stochastic Model Checking

In this section, we describe how to use game semantics to implement PCTL* stochastic model checking, which is the theory foundation for evidence constructing. Any PCTL* formula can be transformed into a canonical form, and the so-called PNF (positive normal form) is characterized by the fact that negations only occur adjacent to atomic propositions. For avoiding the exponential blowup in transforming the PCTL* formulae into PNF, we choose release-PNF as the PNF for PCTL*.

3.1 Release-PNF for PCTL*

Definition 4 (Release-PNF for PCTL*). *PCTL* state formulae in release-PNF are formed according to the following grammar: $\Phi ::= true|false|a|\neg a|\Phi \wedge \Phi|\Phi \vee \Phi|\mathrm{P}_{\sim p}[\Psi]$, where $a \in AP$, AP is the set of atomic propositions, Ψ is the path formula, $\sim \in \{<, \leqslant, >, \geqslant\}$, and $p \in [0,1]$ is the rational bound; PCTL* path formulae in release-PNF are formed according to the following grammar: $\Psi ::= \Phi|\Psi \wedge \Psi|\Psi \vee \Psi|\mathrm{X}\Psi|\Psi\mathrm{U}\Psi|\Psi\mathrm{R}\Psi$, where Φ is the state formula, temporal modality R is dual to the until operator U, formula $\Psi_0\mathrm{R}\Psi_1$ can be read as "Ψ_0 releases Ψ_1". $\Psi_0\mathrm{R}\Psi_1$ holds for a path if Ψ_1 always holds, a requirement that is released as soon as Ψ_0 becomes valid.*

PCTL* formula in release-PNF can rewrite any PCTL* formula, which is facilitated by the following theorem.

Theorem 2. *For each PCTL* formula, there is an equivalent PCTL* formula in release-PNF.*

Proof. Let $Sat(\Phi)$ denote the set of states which satisfy the formula Φ. PCTL* formula Φ and PCTL* formula in release-PNF Φ' are called equivalent, denoted as $\Phi \equiv \Phi'$, if $Sat(\Phi) = Sat(\Phi')$ for all LNPPN M over the atomic proposition set AP. Moreover, the semantics of $\neg$ is: for any place s in LNPPN M, $s \vDash \neg\Phi$ if and only if $s \nvDash \Phi$. Therefore, any PCTL* formula can be transformed into PCTL* formula in release-PNF by the following equivalence laws in Fig.3. □

3.2 Game Definition for PCTL* Stochastic Model Checking

Let M be an LNPPN model, and Φ be a PCTL* formula in release-PNF. The stochastic model checking problem is to decide whether $M \vDash \Phi$, i.e., whether $s_0 \vDash \Phi$, where s_0 is the initial

$$\neg\text{true} \equiv \text{false}$$
$$\neg\neg\Phi \equiv \Phi$$
$$\neg(\Phi_0 \wedge \Phi_1) \equiv \neg\Phi_0 \vee \neg\Phi_1$$
$$\neg\neg\Psi \equiv \Psi$$
$$\neg(\Psi_0 \wedge \Psi_1) \equiv \neg\Psi_0 \vee \neg\Psi_1$$
$$\neg\mathrm{X}\Psi \equiv \mathrm{X}\neg\Psi$$
$$\neg(\Psi_0 \mathrm{U}\, \Psi_1) \equiv \neg\Psi_0 \mathrm{R} \neg\Psi_1$$
$$\neg\mathrm{P}_{\geqslant p}[\Psi] \equiv \mathrm{P}_{\leqslant p}[\Psi]$$
$$\neg\mathrm{P}_{\geqslant p}[\Psi] \equiv \mathrm{P}_{<p}[\Psi]$$

Fig.3. Equivalence laws.

place[5]. The game for PCTL* stochastic model checking can be defined as a two-person game $G_M^\Phi(player, board, rule)$, which can be abbreviated as $G(player, board, rule)$ if M and Φ are clear from the context, where:

1) *player* represents the one who anticipates in the game, which is composed of two players that are named $refuter$ and $verifier$ respectively, and player $verifier$ tries to show model M satisfies Φ, whereas player $refuter$ tries to show model M does not satisfy Φ;

2) *board* presents where the players play, which is the Cartesian product of $S \times Sub(\Phi) \times 2^{Sub(\Phi)}$, where S is the set of places, and $Sub(\Phi)$ is the set of sub-formulae, which is defined in Fig.4;

3) *rule* represents how to play in the game, which is the guidance for which player is going to play and how to play. The play of game G_M^Φ is a sequence of configurations $Con_0 \rightarrow_{player} Con_1 \rightarrow_{player} \ldots \rightarrow_{player} Con_i \rightarrow_{player} \ldots \rightarrow_{player} Con_n$, where Con_i is the form of $(player \times S \times Sub(\Phi) \times \Omega)$, $\Omega = 2^{Sub(\Phi)}$ which can be seen as the insurance for a player to redo the play that he/she did before. Without loss of generality, we assume $\sim$ is $\geqslant$, and the initial player is $verifier$. The rules are shown in Fig.5.

The game G_M^Φ for PCTL* stochastic model checking is played on *board* between $refuter$ and $verifier$ according to *rule*. The play of game G_M^Φ is deterministic according to the above rules. Moreover, a play either ends with rule 0/1, or iterates infinitely with rule 9/10/13/14. The winning criteria for a play are as follows. $verifier$ wins the play if and only if one of the following conditions holds: 1) the play ends with rule 0; 2) the play ends with rule 1, and the configuration is $(player, s, \text{true}/\mathrm{P}_{\geqslant p}(\text{true}), \Omega)$, or $(player, s, a/\neg a/\mathrm{P}_{\geqslant p}(a/\neg a), \Omega)$ and $a/\neg a \in L(s)$; 3) the play iterates infinitely with rule 10; 4) the rule 13 is used for the second time. $refuter$ wins the play if and only if one of the following conditions holds: 1) the play ends with rule 0; 2) the play ends with rule 1, and the configuration is $(player, s, \text{false}/\mathrm{P}_{\geqslant p}(\text{false}), \Omega)$, or $(player, s, a/\neg a/\mathrm{P}_{\geqslant p}(a/\neg a), \Omega)$ and $a/\neg a \notin L(s)$; 3) the play executes infinitely with rule 9; 4) the rule 14 is used for the second time.

Theorem 3. *For every play in the game $G_M^\Phi(player, board, rule)$, there is a determined winner.*

Proof. The LNPPN model M is finite, and the sub-formulae of Φ are also finite, thereby the configurations Con of game G_M^Φ are finite. Every play always reaches the configuration in which Φ is atomic proposition, or the configuration in which Φ is the form of $\mathrm{P}_{\geqslant p}(\Psi_0 \mathrm{U} \Psi_1)/\mathrm{P}_{\geqslant p}(\Psi_0 \mathrm{R} \Psi_1)$ or the configuration is visited for the second time. According to the winning criteria, all of the above cases have the determined winner. Therefore, for every play in the game $G_M^\Phi(player, board, rule)$, there is a determined winner. □

$$Sub(\Phi) = \{\Phi\}, \quad \text{if } \Phi = \text{true}/\text{false}/a/\neg a$$
$$Sub(\Phi_0 \wedge \Phi_1) = \{\Phi_0 \wedge \Phi_1\} \cup Sub(\Phi_0) \cup Sub(\Phi_1)$$
$$Sub(\Phi_0 \vee \Phi_1) = \{\Phi_0 \vee \Phi_1\} \cup Sub(\Phi_0) \cup Sub(\Phi_1)$$
$$Sub(\mathrm{P}_{\sim p}(\Psi_0 \wedge \Psi_1)) = \{\mathrm{P}_{\sim p}(\Psi_0 \wedge \Psi_1)\} \cup Sub(\Psi_0) \cup Sub(\Psi_1)$$
$$Sub(\mathrm{P}_{\sim p}(\Psi_0 \vee \Psi_1)) = \{\mathrm{P}_{\sim p}(\Psi_0 \vee \Psi_1)\} \cup Sub(\Psi_0) \cup Sub(\Psi_1)$$
$$Sub(\mathrm{P}_{\sim p}(\mathrm{X}\Psi)) = \{\mathrm{P}_{\sim p}(\mathrm{X}\Psi)\} \cup Sub(\Psi)$$
$$Sub(\mathrm{P}_{\sim p}(\Psi_0 \mathrm{U} \Psi_1)) = \{expansion(\mathrm{P}_{\sim p}(\Psi_0 \mathrm{U} \Psi_1)) \cup Sub(\Psi_0) \cup Sub(\Psi_1)\}$$
$$Sub(\mathrm{P}_{\sim p}(\Psi_0 \mathrm{R} \Psi_1)) = \{expansion(\mathrm{P}_{\sim p}(\Psi_0 \mathrm{R} \Psi_1)) \cup Sub(\Psi_0) \cup Sub(\Psi_1)\}$$
$$expansion(\Phi) = \begin{cases} \{\Phi, \mathrm{P}_{\sim p}(\Psi_1 \vee (\Psi_0 \wedge \mathrm{X}\Phi)), \mathrm{P}_{\sim p}(\Psi_0 \wedge \mathrm{X}\Phi), \mathrm{P}_{\sim p}(\mathrm{X}\Phi)\}, & \text{if } \Phi = \mathrm{P}_{\sim p}(\Psi_0 \mathrm{U} \Psi_1) \\ \{\Phi, \mathrm{P}_{\sim p}(\Psi_1 \wedge (\Psi_0 \vee \mathrm{X}\Phi)), \mathrm{P}_{\sim p}(\Psi_0 \vee \mathrm{X}\Phi), \mathrm{P}_{\sim p}(\mathrm{X}\Phi)\}, & \text{if } \Phi = \mathrm{P}_{\sim p}(\Psi_0 \mathrm{R} \Psi_1) \end{cases}$$

Fig.4. Elements of $Sub(\Phi)$.

[5] Assume that M has at most one single initial place, which will not lead to weakening its ability to model the stochastic system behaviours.

0) $p = 0$: the play finishes and the player *verifier* wins
1) $Con_i = (player, s, \text{true/false}/a/\neg a/\mathrm{P}_{\geqslant p}(\text{true/false}/a/\neg a), \Omega)$: the play finishes
2) $Con_i = (verifier, s, \Phi_0 \wedge \Phi_1, \Omega)$: player $refuter$ chooses $\Phi_j, j \in \{0,1\}$, and $Con_{i+1} = (verifier, s, \Phi_j, \{\Phi_{1-j}\} \cup \Omega)$
3) $Con_i = (verifier, s, \Phi_0 \vee \Phi_1, \Omega)$: player $verifier$ chooses $\Phi_j, j \in \{0,1\}$, and $Con_{i+1} = (verifier, s, \Phi_j, \Omega)$
4) $Con_i = (refuter, s, \Phi_0 \wedge \Phi_1, \Omega)$: player $refuter$ chooses $\Phi_j, j \in \{0,1\}$, and $Con_{i+1} = (refuter, s, \Phi_j, \Omega)$
5) $Con_i = (refuter, s, \Phi_0 \vee \Phi_1, \Omega)$: player $verifier$ chooses $\Phi_j, j \in \{0,1\}$, and $Con_{i+1} = (refuter, s, \Phi_j, \{\Phi_{1-j}\} \cup \Omega)$
6) $Con_i = (verifier, s, \mathrm{P}_{\geqslant p}(\mathrm{X}\Psi), \Omega)$: player $verifier$ chooses some transitions t in the minimum nondeterministic class by an adversary, and $Con_{i+1} = (verifier, s', \mathrm{P}_{\geqslant ps'}(\Psi))$, where $s \xrightarrow{t} s', \sum P(s,s') \times ps' \geqslant p, P(s,s')$ is the probability from place s to place s' and ps' is the probability of s' satisfying Ψ
7) $Con_i = (verifier, s, \mathrm{P}_{\geqslant p}(\Psi_1 \wedge \Psi_2), \Omega)$: player $refuter$ chooses $\Psi_j, j \in \{0,1\}$, and $Con_{i+1} = (refuter, s, \mathrm{P}_{\geqslant pj}(\Psi_j))$, and $p0 + p1 - 1 < p$
8) $Con_i = (verifier, s, \mathrm{P}_{\geqslant p}(\Psi_0 \vee \Psi_1), \Omega)$: player $verifier$ chooses Ψ_j, and $Con_{i+1} = (verifier, s, \mathrm{P}_{\geqslant pj}(\Psi_j))$, and $p0 + p1 \geqslant p$
9) $Con_i = (verifier, s, \mathrm{P}_{\geqslant p}(\Psi_0 \mathrm{U} \Psi_1), \Omega)$: $Con_{i+1} = (verifier, s, \mathrm{P}_{\geqslant p}(\Psi_1 \vee (\Psi_0 \wedge \mathrm{X}(\Psi_0 \mathrm{U} \Psi_1))))$
10) $Con_i = (verifier, s, \mathrm{P}_{\geqslant p}(\Psi_0 \mathrm{R} \Psi_1), \Omega)$: $Con_{i+1} = (verifier, s, \mathrm{P}_{\geqslant p}(\Psi_1 \wedge (\Psi_0 \vee \mathrm{X}(\Psi_0 \mathrm{R} \Psi_1))))$
11) $Con_i = (player, s, \Phi, \{\mathrm{P}_{\geqslant p}(\Psi)\} \cup \Omega)$: $Con_{i+1} = (player, s, \Phi, \Omega)$
12) $Con_i = (player, s, \Phi, \{\text{true/false}/a/\neg a\} \cup \Omega)$: $Con_{i+1} = (player, s, \Phi, \Omega)$
13) $Con_i = (verifier, s, \Phi_0, \{\Phi_1\} \cup \Omega)$: player $refuter$ chooses a next configuration, and $Con_{i+1} = (verifier, s, \Phi_1, \{\Phi_0\} \cup \Omega)$
14) $Con_i = (refuter, s, \Phi_0, \{\Phi_1\} \cup \Omega)$: player $verifier$ chooses a next configuration, and $Con_{i+1} = (refuter, s, \Phi_1, \{\Phi_0\} \cup \Omega)$

Fig. 5. Move rules of game.

3.3 Game Semantics

The PCTL* model checking game $G_M^{\Phi}(player, board, rule)$ is composed of all the possible plays from initial configuration $(verifier, s, \Phi, \Omega)$, which is the form of tree structure. The graph structure for game G_M^{Φ} is the graph presentation of game tree, in which nodes are configurations, edges are possible moves according to *rule*, and loops are plays of the form rule 9/10/13/14. The winning strategy means that there is a set of moves allowing the player to make every play into a configuration which he/she wins. The game tree of the winner is the refined game tree of game G_M^{Φ}, which only consists of the winning moves for the winner. The game graph of the winner has the similar definition with the game tree of the winner.

Theorem 4. *For every game, there is a winning strategy for one of the players.*

Proof. We use the induction method to prove the theorem. 1) When the PCTL* model checking game G_M^{Φ} is composed of a configuration Con_0, in which the formula Φ is one of the following forms: $\text{true/false}/a/\neg a/\mathrm{P}_{\geqslant p}(\text{true/false}/a/\neg a)$, one of the players has a winning strategy based on winning criteria. 2) Game G_M^{Φ} is composed of a set of configurations Con_0, Con_1, ..., Con_i, ..., Con_n. Assuming there is a winning strategy for one of the players at the game with start configuration Con_i, then Con_{i+1} also has a winning strategy because Con_i may choose Con_{i+1} to play on. If not, the other player will have a winning strategy in the game with start configuration Con_{i+1}. Therefore, for every game, there is a winning strategy for one of the players. □

Using winning strategy in game G_M^{Φ}, we can capture the operational semantics for PCTL* stochastic model checking which is stated in Theorem 5.

Theorem 5. *Let M be an LNPPN model, $s \in S$, Φ be a PCTL* formula in release-PNF. Then for $\forall s$: $s \vDash \Phi$ if and only if verifier has a winning strategy for game G_M^{Φ} with start configuration $(player, s, \Phi, \Omega)$; $s \vDash \Phi$ if and only if refuter has a winning strategy for game G_M^{Φ} with start configuration $(player, s, \Phi, \Omega)$. In addition, it is independent of the initial player which is verifier or refuter.*

Proof. The "if" part is obvious. It is sufficient to prove the "only if" part which can be implemented by presenting the strategy of player $verifier$ or $refuter$ for $s \vDash \Phi$ or $s \nvDash \Phi$. Moreover, the proof process of "only if" part can be shown in structural induction on $Sub(\Phi)$ and $2^{Sub(\Phi)}$ of PCTL* formula Φ, and all places S of the LNPPN model M.

1) $\Phi = \text{true/false}/a/\neg a/\mathrm{P}_{\geqslant p}(\text{true/false}/a/\neg a)$: if $s \vDash \Phi$, then $\mathrm{P}_{=0}(\Phi)$, and thus $verifier$ wins game G_M^{Φ} right at Con_0 according to winning criteria; if $s \nvDash \Phi$, then $\mathrm{P}_{=0}(\Phi)$, and thus $refuter$ wins the game G_M^{Φ} right at Con_0 according to winning criteria.

2) $\Phi = \Phi_0 \wedge \Phi_1$: if $s \vDash \Phi_0 \wedge \Phi_1$, according to the denotational semantics, $s \vDash \Phi_j$ for each j, where $j \in \{0, 1\}$. Therefore, by the induction hypothesis, *verifier* has a winning strategy for the game with start configuration $(player, s, \Phi_0, \Omega)$ and $(player, s, \Phi_1, \Omega)$. Moreover, the winning strategy of *verifier* which starts from configuration $(player, s, \Phi, \Omega)$ is composed of the union of the winning strategy of *verifier* for the game with start configuration $(player, s, \Phi_0, \Omega)$ or $(player, s, \Phi_1, \Omega)$; if $s \nvDash \Phi_0 \wedge \Phi_1$, according to the denotational semantics, $s \nvDash \Phi_j$ for any j, where $j \in \{0, 1\}$. Therefore, by the induction hypothesis, *refuter* has a winning strategy for the game with start configuration $(player, s, \Phi_0, \Omega)$ or $(player, s, \Phi_1, \Omega)$. Moreover, the winning strategy of *refuter* which starts from configuration $(player, s, \Phi, \Omega)$ is composed of the strategy choosing the next configuration $(player, s, \Phi_j, \Omega)$ and the winning strategy of *refuter* for configuration $(player, s, \Phi_j, \Omega)$.

3) $\Phi = \Phi_0 \vee \Phi_1$: if $s \vDash \Phi_0 \vee \Phi_1$, according to the denotational semantics, $s \vDash \Phi_j$ for any j, where $j \in \{0, 1\}$. Therefore, by the induction hypothesis, *verifier* has a winning strategy for the game with start configuration $(player, s, \Phi_j, \Omega)$. Moreover, the winning strategy of *verifier* which starts from configuration $(player, s, \Phi, \Omega)$ is composed of the strategy choosing the next configuration $(player, s, \Phi_j, \Omega)$ and the winning strategy of *verifier* for configuration $(player, s, \Phi_j, \Omega)$; if $s \nvDash \Phi_0 \vee \Phi_1$, according to the denotational semantics, $s \nvDash \Phi_j$ for each j, where $j \in \{0, 1\}$. Therefore, by the induction hypothesis, *refuter* has a winning strategy for the game with start configuration $(player, s, \Phi_0, \Omega)$ and $(player, s, \Phi_1, \Omega)$. Moreover, the winning strategy of *refuter* which starts from configuration $(player, s, \Phi, \Omega)$ is composed of the union of the winning strategy of *refuter* for the game with start configuration $(player, s, \Phi_0, \Omega)$ or $(player, s, \Phi_1, \Omega)$.

4) $\Phi = \mathrm{P}_{\geqslant p}(\mathrm{X}\Psi)$: if $s \vDash \mathrm{P}_{\geqslant p}(\mathrm{X}\Psi)$, according to the denotational semantics, $Pr_s\{\pi \in Path(s) | \pi \vDash \mathrm{X}\Psi\} \geqslant p$, i.e., there is one or more transitions t which $s \xrightarrow{t} s'$ and $\sum P(s, s') \times ps' \geqslant p$, where $P(s, s')$ is the probability from place s to place s' and ps' is the probability of s' satisfying Ψ. Therefore, by the induction hypothesis, *verifier* has a winning strategy for the game with start configuration $(player, s', \mathrm{P}_{\geqslant ps'}(\Psi), \Omega)$. Moreover, the winning strategy of *verifier* which starts from configuration $(player, s, \Phi, \Omega)$ is composed of the strategy choosing the next configuration $(player, s', \mathrm{P}_{\geqslant ps'}(\Psi), \Omega)$ and the winning strategy for configuration $(player, s', \mathrm{P}_{\geqslant ps'}(\Psi), \Omega)$; if $s \nvDash \mathrm{P}_{\geqslant p}(\mathrm{X}\Psi)$, according to the denotational semantics, $Pr_s\{\pi \in Path(s) | \pi \nvDash \mathrm{X}\Psi\} > 1 - p$, i.e., there is one or more transitions t which $s \xrightarrow{t} s'$ and $\sum P(s, s') \times ps' > 1 - p$, where $P(s, s')$ is the probability from place s to place s' and ps' is the probability of s' satisfying $\neg\Psi$. Therefore, by the induction hypothesis, *refuter* has a winning strategy for the game with start configuration $(player, s', \mathrm{P}_{\geqslant ps'}(\Psi), \Omega)$. Moreover, the winning strategy of *refuter* which starts from configuration $(player, s, \Phi, \Omega)$ is composed of the strategy choosing the next configuration $(player, s', \mathrm{P}_{\geqslant ps'}(\Psi), \Omega)$ and the winning strategy for configuration $(player, s', \mathrm{P}_{\geqslant ps'}(\Psi), \Omega)$.

5) $\Phi = \mathrm{P}_{\geqslant p}(\Psi_0 \wedge \Psi_1)$: if $s \vDash \mathrm{P}_{\geqslant p}(\Psi_0 \wedge \Psi_1)$, according to the denotational semantics, $Pr_s\{\pi \in Path(s) | \pi \vDash \Psi_0 \wedge \Psi_1\} \geqslant p$, i.e., $Pr_s\{\pi \in Path(s) | \pi \vDash \Psi_0 \wedge \pi \vDash \Psi_1\} \geqslant p$. Therefore, by the induction hypothesis, *verifier* has a winning strategy for the game with start configuration $(player, s, \Psi_0 \wedge \Psi_1, \Omega)$. Moreover, the winning strategy of *verifier* which starts from configuration $(player, s, \Phi, \Omega)$ is composed of the strategy choosing the next configuration $(player, s, \Psi_0 \wedge \Psi_1, \Omega)$ and the winning strategy for configuration $(player, s, \Psi_0 \wedge \Psi_1, \Omega)$; if $s \nvDash \mathrm{P}_{\geqslant p}(\Psi_0 \wedge \Psi_1)$, according to the denotational semantics, $Pr_s\{\pi \in Path(s) | \pi \nvDash \Psi_0 \wedge \Psi_1\} > 1 - p$, i.e., $Pr_s\{\pi \in Path(s) | \pi \nvDash \Psi_j\} > 1 - p$ for any j, where $j \in \{0, 1\}$, or $Pr_s\{\pi \in Path(s) | \pi \nvDash \Psi_0\} > p0$ and $Pr_s\{\pi \in Path(s) | \pi \nvDash \Psi_1\} > p1$ and $p0 + p1 > 1 - p$. Therefore, by the induction hypothesis, *refuter* has a winning strategy for the game with start configuration $(player, s, \Psi_j, \Omega)$ or $(player, s, \Psi_0, \Omega)$ and $(player, s, \Psi_1, \Omega)$. Moreover, the winning strategy of *refuter* which starts from configuration $(player, s, \Phi, \Omega)$ is composed of the union of the winning strategy of *refuter* for the game with start configuration $(player, s, \Psi_j, \Omega)$ or $(player, s, \Psi_0, \Omega)$ and $(player, s, \Psi_1, \Omega)$.

6) $\Phi = \mathrm{P}_{\geqslant p}(\Psi_0 \vee \Psi_1)$: if $s \vDash \mathrm{P}_{\geqslant p}(\Psi_0 \vee \Psi_1)$, according to the denotational semantics, $Pr_s\{\pi \in Path(s) | \pi \vDash \Psi_0 \vee \Psi_1\} > p$, i.e., $s \vDash \mathrm{P}_{\geqslant p}(\Psi_j)$ for any j, where $j \in \{0, 1\}$, or $s \vDash \mathrm{P}_{\geqslant p0}(\Psi_0)$ and $s \vDash \mathrm{P}_{\geqslant p1}(\Psi_1)$ where $p0 + p1 \geqslant p$. Therefore, by the induction hypothesis, *verifier* has a winning strategy for the game with start configuration $(player, s, \mathrm{P}_{\geqslant p}(\Psi_j), \Omega)$ or $(player, s, \mathrm{P}_{\geqslant p0}(\Psi_0), \Omega)$ and $(player, s, \mathrm{P}_{\geqslant p1}(\Psi_1), \Omega)$. Moreover, the winning strategy of *verifier* which starts from configuration $(player, s, \Phi, \Omega)$ is composed of the strategy choosing the next configuration $(player, s, \mathrm{P}_{\geqslant p}(\Psi_j), \Omega)$ or $(player, s, \mathrm{P}_{\geqslant p0}(\Psi_0), \Omega)$ and $(player, s, \mathrm{P}_{\geqslant p1}(\Psi_1), \Omega)$ and the winning strat-

egy for configuration $(player, s, \mathrm{P}_{\geqslant p}(\Psi_j), \Omega)$ or $(player, s, \mathrm{P}_{\geqslant p0}(\Psi_0), \Omega)$ and $(player, s, \mathrm{P}_{\geqslant p1}(\Psi_1), \Omega)$; if $s \not\models \mathrm{P}_{\geqslant p}(\Psi_0 \vee \Psi_1)$, according to the denotational semantics, $Pr_s\{\pi \in Path(s) | \pi \not\models \Psi_0 \vee \Psi_1\} > 1-p$, i.e., $Pr_s\{\pi \in Path(s) | \pi \not\models \Psi_0 \vee \pi \not\models \Psi_1\} > 1-p$. Therefore, by the induction hypothesis, *refuter* has a winning strategy for the game with start configuration $(player, s, \Psi_0 \vee \Psi_1, \Omega)$. Moreover, the winning strategy of *refuter* which starts from configuration $(player, s, \Phi, \Omega)$ is composed of the winning strategy choosing the next configuration $(player, s, \Psi_0 \vee \Psi_1, \Omega)$ and the winning strategy of *refuter* for the game with start configuration $(player, s, \Psi_0 \vee \Psi_1, \Omega)$.

7) $\Phi = \mathrm{P}_{\geqslant p}(\Psi_0 \mathrm{U} \Psi_1)$: if $s \models \mathrm{P}_{\geqslant p}(\Psi_0 \mathrm{U} \Psi_1)$, according to the denotational semantics, $Pr_s\{\pi \in Path(s) | \pi \models \Psi_0 \mathrm{U} \Psi_1\} \geqslant p$, i.e., $s \models \mathrm{P}_{\geqslant p}(\Psi_1)$, or $s \models \mathrm{P}_{\geqslant p}(\Psi_1 \wedge \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{U} \Psi_1))$. Thus by the induction hypothesis, *verifier* has a winning strategy for the game with start configuration $(player, s, \mathrm{P}_{\geqslant p}(\Psi_1), \Omega)$ or $(player, s, \mathrm{P}_{\geqslant p}(\Psi_1 \wedge \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{U} \Psi_1)), \Omega)$. Moreover, the winning strategy of *verifier* which starts from configuration $(player, s, \Phi, \Omega)$ is composed of the winning strategy choosing the next configuration $(player, s, \mathrm{P}_{\geqslant p}(\Psi_1), \Omega)$ or $(player, s, \mathrm{P}_{\geqslant p}(\Psi_0 \wedge \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{U} \Psi_1)), \Omega)$ and the winning strategy of *verifier* for the game with start configuration $(player, s, \mathrm{P}_{\geqslant p}(\Psi_1), \Omega)$ or $(player, s, \mathrm{P}_{\geqslant p}(\Psi_1 \wedge \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{U} \Psi_1)), \Omega)$; if $s \not\models \mathrm{P}_{\geqslant p}(\Psi_0 \mathrm{U} \Psi_1)$, according to the denotational semantics, $Pr_s\{\pi \in Path(s) | \pi \not\models \Psi_0 \mathrm{U} \Psi_1\} > 1-p$, i.e., $Pr_s\{\pi \in Path(s) | \pi \not\models \Psi_1 \wedge \pi \not\models \mathrm{P}_{\geqslant p}(\Psi_0 \wedge \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{U} \Psi_1))\} > 1-p$. Therefore, by the induction hypothesis, *refuter* has a winning strategy for the game with start configuration $(player, s, \Psi_1 \vee (\Psi_0 \wedge \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{U} \Psi_1)), \Omega)$. Moreover, the winning strategy of *refuter* which starts from configuration $(player, s, \Phi, \Omega)$ is composed of the winning strategy choosing the next configuration $(player, s, \Psi_1 \vee (\Psi_0 \wedge \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{U} \Psi_1)), \Omega)$ and the winning strategy of *refuter* for the game with start configuration $(player, s, \Psi_1 \vee (\Psi_0 \wedge \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{U} \Psi_1)), \Omega)$.

8) $\Phi = \mathrm{P}_{\geqslant p}(\Psi_0 \mathrm{R} \Psi_1)$: if $s \models \mathrm{P}_{\geqslant p}(\Psi_0 \mathrm{R} \Psi_1)$, according to the denotational semantics, $Pr_s\{\pi \in Path(s) | \pi \models \Psi_0 \mathrm{R} \Psi_1\} \geqslant p$, i.e., $s \models \mathrm{P}_{\geqslant p}(\Psi_1 \wedge (\Psi_0 \vee \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{R} \Psi_1)))$. Thus by the induction hypothesis, *verifier* has a winning strategy for the game with start configuration $(player, s, \mathrm{P}_{\geqslant p}(\Psi_1), \Omega)$ and $(player, s, \mathrm{P}_{\geqslant p}(\Psi_0 \vee \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{R} \Psi_1)), \Omega)$. Moreover, the winning strategy of *verifier* which starts from configuration $(player, s, \Phi, \Omega)$ is composed of the winning strategy choosing the next configuration $(player, s, \mathrm{P}_{\geqslant p}(\Psi_1), \Omega)$ and $(player, s, \mathrm{P}_{\geqslant p}(\Psi_0 \vee \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{R} \Psi_1)), \Omega)$, and the winning strategy of *verifier* for the game with start configuration $(player, s, \mathrm{P}_{\geqslant p}(\Psi_1), \Omega)$ and $(player, s, \mathrm{P}_{\geqslant p}(\Psi_0 \vee \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{R} \Psi_1)), \Omega)$; if $s \not\models \mathrm{P}_{\geqslant p}(\Psi_0 \mathrm{R} \Psi_1)$, according to the denotational semantics, $Pr_s\{\pi \in Path(s) | \pi \not\models \Psi_0 \mathrm{R} \Psi_1\} > 1-p$, i.e., $Pr_s\{\pi \in Path(s) | \pi \not\models \Psi_1 \vee \pi \not\models | \neq \mathrm{P}_{\geqslant p}(\Psi_0 \vee \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{R} \Psi_1))\} > 1-p$. Therefore, by the induction hypothesis, *refuter* has a winning strategy for the game with start configuration $(player, s, \mathrm{P}_{\geqslant p}(\Psi_1), \Omega)$ or $(player, s, \mathrm{P}_{\geqslant p}(\Psi_0 \vee \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{R} \Psi_1)), \Omega)$. Moreover, the winning strategy of *refuter* which starts from configuration $(player, s, \Phi, \Omega)$ is composed of the winning strategy choosing the next configuration $(player, s, \mathrm{P}_{\geqslant p}(\Psi_1), \Omega)$ or $(player, s, \mathrm{P}_{\geqslant p}(\Psi_0 \vee \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{R} \Psi_1)), \Omega)$ and the winning strategy of *refuter* for the game with start configuration $(player, s, \mathrm{P}_{\geqslant p}(\Psi_1), \Omega)$ or $(player, s, \mathrm{P}_{\geqslant p}(\Psi_0 \vee \mathrm{XP}_{\geqslant ps'}(\Psi_0 \mathrm{R} \Psi_1)), \Omega)$. □

4 Strategy Solving in PCTL* Stochastic Model Checking Game

The game process for PCTL* stochastic model checking can be presented by the game graph $Gg(N, E, w)$. Node set N is the set of configurations in the game, $N \subseteq (player \times S \times Sub(\Phi) \times \Omega)$; edge set E is the set of possible move in the game, $E \subseteq N \times N$; w is the probability value, and $w = P(s, s')$. The game graph $Gg(N, E, w)$ can be constructed from initial configuration as the initial node in a BFS (breadth first search) or DFS (depth first search) manner, and it owns the characteristics stated in Theorem 6.

Theorem 6. *The game graph for game G_M^Φ can be partitioned into MSCCs (maximal strongly connected components). Moreover, every play never leaves $MSCC_m$ into $MSCC_n$ with $m < n$.*

Proof. The algorithm for finding MSCCs with the order is basically the same with standard algorithm for finding a topological order on the set of connected components of a directed graph in the DFS method. Let $Con_0 \rightarrow_{player} Con_1 \rightarrow_{player} \ldots \rightarrow_{player} Con_i \rightarrow_{player} \ldots Con_n$ be a game process with $Con_0 = (player \times S \times Sub(\Phi) \times \Omega)$, then all the intermediate configurations are all of the form $(player \times S \times Sub(\Phi) \times \Omega)$. Let $m = 1$, then the configurations are generated by rule 1, and the other MSCCs are labeled as $m{+}{+}$ one by one in the bottom-up manner. Moreover, the other configurations from Con_i and all successor configurations Con_n which are not used in rule 1 and have a move return

to Con_i are labeled as m. There is no edge from an MSCC with a lower index to one with a higher index, because the generation of MSCCs is the application of game rules which strictly increase the path quantifiers' number in the configuration.

Based on Theorem 6, we can alter the coloring algorithm[19] to color each node in the MSCCs of game graph, which depends on whether $verifier$ or $refuter$ has a winning strategy for the game with the node as the initial configuration. That is to say, if the node is colored in white/dark, then $verifier/refuter$ wins the game, and the witness is shown by all the nodes which are colored in white/dark. Therefore, the winning strategy solving algorithm is how to color the game graph. The coloring algorithm colors MSCCs by their order. Let MSCC_i be the smallest MSCC at present, i.e., the MSCC_m with $m < i$ have all been colored before, and the coloring process of a node in MSCC_i is as follows.

1) The node at which Φ is the form of true/false/a/$\neg a$/$\mathrm{P}_{\geqslant p}$(true/false/a/$\neg a$) is colored in white, if $verifier$ wins in the node, and dark otherwise.

2) The node at which Φ is the form of $\Phi_0 \wedge \Phi_1$ is colored in white, if all its sons at which Φ is the form of Φ_j are colored in white, and dark if it has at least one son colored in dark.

3) The node at which Φ is the form of $\Phi_0 \vee \Phi_1$ is colored in white if at least one of its sons at which Φ is the form of Φ_j is colored in white, and dark if all its sons are colored in dark.

4) The node at which Φ is the form of $\mathrm{P}_{\geqslant p}(\mathrm{X}\Psi)$ is colored in white, if the sum of $P(s, s') \times ps'$ is equal to or greater than p, where $P(s, s')$ is the probability of moving from the place at the node to its son that is colored in white and satisfies Ψ at the probability ps', and dark if the sum of $P(s, s') \times ps'$ is greater than $1 - p$, where $P(s, s')$ is the probability of moving from the place at the node to its son which does not satisfy Ψ at the probability ps' and is colored in dark.

5) The node at which Φ is the form of $\mathrm{P}_{\geqslant p}(\Psi_0 \wedge \Psi_1)$ is colored in white, if the probability of moving from the place at the node to its sons at which Φ is true for the form of $\Psi_0 \wedge \Psi_1$, is equal to or greater than p. It is colored in dark, if the probability of moving from the place at the node to its sons at which Φ is true for the form of $\Psi_0 \wedge \Psi_1$, is greater than $1 - p$.

6) The node at which Φ is the form of $\mathrm{P}_{\geqslant p}(\Psi_0 \vee \Psi_1)$ is colored in white, if the probability of moving from the place at the node to its sons at which Φ is true for the form of $\Psi_0 \vee \Psi_1$, is equal to or greater than p. It is colored in dark, if the probability of moving from place at the node to its sons at which Φ is true for the form of $\Psi_0 \vee \Psi_1$, is greater than $1 - p$.

7) The node which is the witness of the form $\mathrm{P}_{\geqslant p}(\Psi_0 \mathrm{U} \Psi_1)$ is colored in dark, and the node which is the witness of the form $\mathrm{P}_{\geqslant p}(\Psi_0 \mathrm{R} \Psi_1)$ is colored in white. □

Strategy solving algorithm for PCTL* stochastic model checking game combines finding MSCCs with order algorithm and coloring algorithm. The complexity of strategy solving algorithm is shown in Theorem 7.

Theorem 7. *Strategy solving algorithm for PCTL* stochastic model checking game is in PSPACE.*

Proof. Let $|\Phi|$ be the length of Φ, and $|M|$ the number of states in LNPPN model M. Because the PCTL* formula Φ contains $|\Phi|/2$ irredundant path quantifiers, the game graph can contain $|M| \times |\Phi|/2$ MSCCs according to Theorem 6. The coloring algorithm may be used for $|M| \times |\Phi|/2$ times, but the space it occupies can be reused. Moreover, the coloring algorithm is recursive, thus it has to store the constant value of the number of MSCCs, the polynomial size of the number of nodes in each MSCC, and two configurations of linear size of Φ. Thus, according to Savitch's theorem[20], the strategy solving algorithm can be transformed into a deterministic one only with a quadratic trade-off in the polynomial space complexity only. □

5 Evidence Constructing for PCTL* Stochastic Model Checking Results

At present, the mainstream tools of stochastic model checking, such as PRISM[21], just return the answer "yes" or "no", which can be viewed as the decision result without any further explanation information. For understanding the stochastic model checking results better, we argue that it will be appreciative to show why the model satisfies (or does not satisfy) the property, i.e., the evidence for verification or refutation. In this section, we present the evidence constructing algorithm for stochastic model checking results.

Intuitively speaking, in stochastic model checking, an evidence for verification is a part of the stochastic system model responsible for the quantitative temporal property being satisfied. Dually, when the stochastic system model does not satisfy a quantitative temporal property, an evidence for refutation is provided as a portion of the stochastic system model responsible for the quantitative temporal property being violated.

Definition 5 (Evidence in General Form). *If* $\mathrm{P}_{\leqslant p}(\Psi)$ *or* $\mathrm{P}_{<p}(\Psi)$ *is a failed formula in stochastic*

system model M, then Ev is an evidence for refutation for $\mathrm{P}_{\leqslant p}(\Psi)$ or $\mathrm{P}_{\leqslant p}(\Psi)$ in M if and only if Ev is a $\sum Path(M')$ with the biggest adversary such that $Ev \models \mathrm{P}_{>p}(\Psi)$ or $Ev \models \mathrm{P}_{\geqslant p}(\Psi)$. If $\mathrm{P}_{\geqslant p}(\Psi)$ or $\mathrm{P}_{>p}(\Psi)$ is a successful formula in stochastic system model M, then Ev is an evidence for verification for $\mathrm{P}_{\geqslant p}(\Psi)$ or $\mathrm{P}_{>p}(\Psi)$ in M if and only if Ev is a $\sum Path(M')$ with the smallest adversary such that $Ev \models \mathrm{P}_{\geqslant p}(\Psi)$ or $Ev \models \mathrm{P}_{>p}(\Psi)$.

As the evidence for refutation, if a formula Φ is the form of $\mathrm{P}_{\geqslant p}(\Psi)$ or $\mathrm{P}_{>p}(\Psi)$, Φ can be translated into the form $\mathrm{P}_{\leqslant p}(\Psi)$ or $\mathrm{P}_{<p}(\Psi)$ equivalently. As the evidence for verification, if a formula Φ is the form of $\mathrm{P}_{\leqslant p}(\Psi)$ or $\mathrm{P}_{<p}(\Psi)$, Φ can be translated in the form $\mathrm{P}_{\geqslant p}(\Psi)$ or $\mathrm{P}_{>p}(\Psi)$ equivalently. Actually, the corresponding evidence for verification or refutation can also be defined for them, but it makes no sense because even the empty set trivially fulfills $Prob(\sum Path(M')) < p$ (or $Prob(\sum Path(M')) \leqslant p$).

Informally, as an evidence for verification or refutation, it is expected to fulfill the following conditions: 1) it verifies or falsifies the property formula; 2) it holds enough information for explaining why the stochastic system model satisfies (or does not satisfy) the property formula; 3) it is the minimality which means it is precise without redundancies, i.e., every place and every transition are needed to maintain 1) and 2).

Generally speaking, the sub-model of a stochastic system model and its unwinding can be the evidence for verification or refutation, but the information it holds is implicit. For PCTL*, having a sub-model as an evidence for verification or refutation is probably not what we want. We need it to be annotated with further explanatory information. For showing the information explicitly, we take full advantage of the game graph constructed in Section 4 and use the sub-graph of corresponding game graph to present the evidence for verification or refutation of PCTL* property formula. Moreover, this can define the evidence for verification or refutation unifiedly. In the sub-graph, each node is marked by the place and the sub-formula, the color of a node means whether it satisfies the sub-formula, and the value on the outgoing edge represents the probability between the connected nodes. Therefore, we define an evidence for verification or refutation to be a sub-graph of the corresponding game graph, rather than a sub-model.

Definition 6 (Evidence in Unified Form). *Let Gg be a game graph constructed for an LNPPN model M and a PCTL* formula Φ in release-PNF, and Con_0 be the initial node of Gg. The sub-graph Ev of game graph Gg from Con_0 is an evidence for verification or refutation of Φ, if it meets the following conditions:* 1) *for each node Con_i in Ev, its color is the same with that of Con_0;* 2) *sub-graph Ev is independent of Gg;* 3) *sub-graph Ev is minimal with regard to* 1) *and* 2). *Moreover, if the color of Con_0 is white, the sub-graph Ev is the evidence for verification of Φ; if the color of Con_0 is dark, the sub-graph Ev is the evidence for refutation of Φ.*

Condition 1 requires color consistency with the initial node, because the color of initial node decides whether the model satisfies the property, i.e., sub-graph Ev is the evidence for verification or refutation of Φ. In condition 2, the independence means that the color of a node in sub-graph does not have any relationship with the color of other nodes in the game graph. Thus, condition 2 guarantees the sub-graph Ev holds sufficient information for explaining the reason why initial node is colored in white or dark, i.e., why the property formula Φ is verified or refuted by the stochastic system model M. Condition 3 claims there is not a strict sub-graph of Ev satisfying the conditions 1 and 2, i.e., the nodes in the sub-graph Ev are necessary.

The evidence for verification or refutation is indeed the refinement of winning strategy of *verifier* or *refuter*, which is sufficient to win game G_M^{Φ}. The algorithm of constructing evidence for verification or refutation is shown in Appendix, and the soundness is stated in Theorem 8 and Theorem 9.

Theorem 8. *The constructed evidence Ev for verification is in accordance with Definition* 6.

Proof. It is obviously correct that the constructed evidence Ev for verification meets condition 1 of Definition 6, which is shown by the coloring algorithm and the induction on the construction of Ev.

The constructed evidence Ev for verification meets condition 2) of Definition 6, i.e., sub-graph Ev is independent of the game graph Gg, which can be proved respectively according to the following cases.

1) Con_i in Ev is colored in white by case 1 of the coloring algorithm. It is colored in white only depending on itself, which has nothing to do with others.

2) Con_i in Ev is colored in white by case 2 of the coloring algorithm. Supposing some other nodes in the game graph Gg can give rise to coloring Con_i in dark, there is at least one son Con_j of Con_i being colored in dark. However, in the light of the evidence Ev construction process, all sons of Con_i, certainly including Con_j, are in Ev and colored in white. Therefore, this

results in contradiction.

3) Con_i in Ev is colored in white by case 3 of the coloring algorithm. Supposing some other nodes in the game graph Gg can give rise to coloring Con_i in dark, all sons of Con_i are colored in dark. However, in the light of the evidence Ev construction process, at least there is one son of Con_i being colored in white. Therefore, this results in contradiction.

4) Con_i in Ev is colored in white by case 4 of the coloring algorithm. Supposing some other nodes in the game graph Gg can give rise to coloring Con_i in dark, there are some sons of Con_i being colored in dark and $\sum P(s, s_j) \times ps_j > 1 - p$. However, in the light of the evidence Ev construction process, there are some sons of Con_i being colored in white and $\sum P(s, s_j) \times ps_j \geqslant p$. Therefore, this results in contradiction.

5) Con_i in Ev is colored in white by case 5 of the coloring algorithm. Supposing some other nodes in the game graph Gg can give rise to coloring Con_i in dark, there are some sons of Con_i being colored in dark and $\sum ps_j > 1 - p$. However, in the light of the evidence Ev construction process, there are some sons of Con_i being colored in white and $\sum ps_j \geqslant p$. Therefore, this results in contradiction.

6) Con_i in Ev is colored in white by case 6 of the coloring algorithm. Supposing some other nodes in the game graph Gg can give rise to coloring Con_i in dark, there are some sons of Con_i being colored in dark and $\sum ps_j > 1 - p$. However, in the light of the evidence Ev construction process, there are some sons Con_j of Con_i being colored in white and $\sum ps_j \geqslant p$. Therefore, this results in contradiction.

7) Con_i in Ev is colored in white by case 7 (or case 8) of the coloring algorithm, the proof process is similar to 6 and 5 (or 5 and 6).

For proving the constructed evidence Ev for verification meets condition 3 of Definition 6, we can use the method of proof by contradiction, i.e., if any node Con_i or any edge between nodes in Ev is removed from Ev, then it will lead to a situation that the sub-graph is not independent of the game graph. The abbreviated proof process is: assume that a node or an edge between nodes in Ev is removed from Ev, then it can be colored in dark by other coloring algorithm, which is contradictory to all kinds of coloring process, because constructed evidence Ev for verification is independent of game graph. □

Theorem 9. *The constructed evidence Ev for refutation is in accordance with Definition* 6.

Proof. The proof process is similar to the proof of Theorem 8. □

Actually, the evidence for refutation constructed according to Definition 6 and the algorithm of Appendix, is somewhat like the counterexample defined by Clarke *et al.*[22] Clarke *et al.* pointed out that the counterexample has to satisfy three criteria: the counterexample should 1) serve as an explanation of why the model violates the property, 2) be rich enough to explain the violation of a large class of properties, 3) be simple and specific enough to identify bugs, and be amenable to efficient generation and analysis. Strictly speaking, the evidence for refutation is just a quasi-counterexample, because it may not conform to "simple enough" in criterion 3. Chadha and Viswanathan[23] showed that counterexample generation according to criteria of Clarke *et al.* in stochastic model checking is an NPC problem. That is to say, generating the smallest counterexample is NP-complete. Moreover, it is unlikely to be efficiently approximable. The related work about counterexample will be discussed in Section 7. Note that an evidence for refutation constructed in this paper can be viewed as a (minimal) part of the winning strategy of $refuter$ that is sufficient to guarantee its victory. The evidence for refutation in this paper is not the smallest counterexample, but it is the minimal counterexample. Intuitively, a minimal counterexample has the property that removing any edge from the underlying graph of the counterexample will result in the evidence which is no longer a counterexample.

6 Case Study

We implement a prototype tool SMC-NPPN with explicit state search for the game-based PCTL* stochastic model checking algorithm, which includes all functions of NPNMV[16,24-25] and can use game semantic algorithm to quantitatively check the system model with nondeterminism and probabilistic behaviours. It is the first tool for PCTL* stochastic model checking in game semantics, and it can present the evidence for stochastic model checking result. In order to demonstrate the game-based PCTL* stochastic model checking process, an illustrative example is checked in SMC-NPPN. Let M be the LNPPN model to be checked, which is shown in Fig.6, and $\mathrm{P}_{\geqslant 0.4}(q\mathrm{UG}r)$ be the PCTL* property formula. In Fig.6, S1 is the single initial place, {(S1, T0), (S1, T1), (S1, T2)} is a nondeterministic class, {(S1, T6)} is another nondeterministic class. The labeling functions of M are

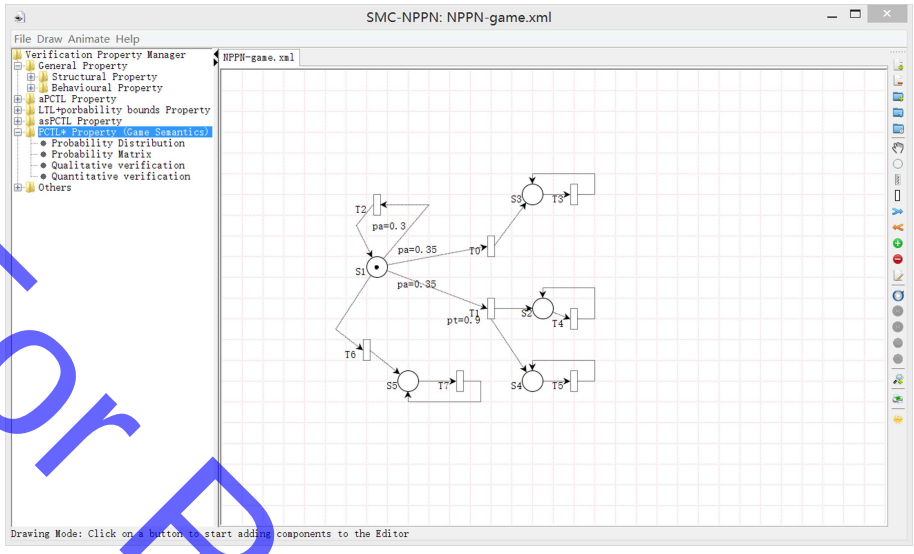


Fig.6. LNPPN model to be checked.

as follows: $L(\mathrm{S3}) = L(\mathrm{S4}) = \text{null}$, $L(\mathrm{S1}) = q$, $L(\mathrm{S2}) = L(\mathrm{S5}) = r$, and the probability matrixes of M are shown in Fig.7. In fact, system model M models a very simple communication protocol. The workflow of communication protocol is as follows. When it starts trying to send a message (S1), there is a nondeterministic choice between: 1) sending the message direct (T2) as the channel is safe and ready; and 2) sending the message in an unreliable way. If the latter, it waits for (T6) with probability 0.3 because the channel is unready, it sends failed (T0) with probability 0.35 and stops, and with probability 0.35 it chooses successful sending (T1) and stops.

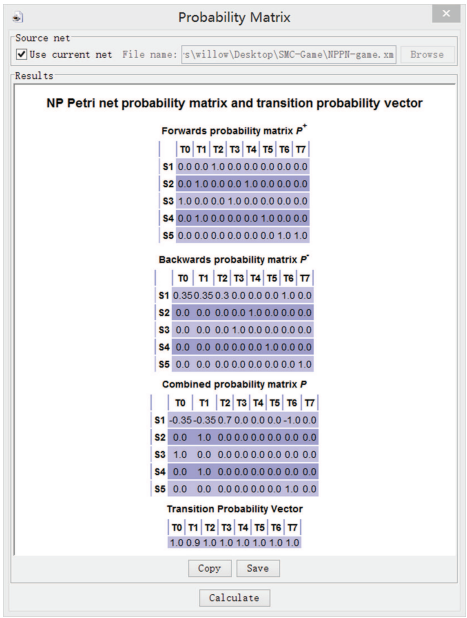


Fig.7. Probability matrixes of the LNPPN model. Si corresponds to s_i, and Tj corresponds to t_j.

Fig.8 shows the stochastic model checking result based on game semantics, and indicates that the colored game graph is saved as Fig.9, and the evidence is saved as Fig.10. If the color is not considered, Fig.9 shows a game graph Gg that is constructed for the given LNPPN model M and the PCTL* property formula $\mathrm{P}_{\geqslant 0.4}(q\mathrm{UG}r)$. The model M has a single initial place S1, thus Gg has a single initial node $(s_1, \mathrm{P}_{\geqslant 0.4}(q\mathrm{UG}r))$. The other nodes are derived according to game semantics in Section 3. The dashed and solid edges represent moves of players, and the solid edges also denote the actual transition in M. Taking the color into consideration, Fig.9 presents a coloring game graph that is colored by the coloring algorithm in Section 4. The coloring process colors MSCCs of game graph Gg by their order. If the MSCC only contains one node, it will be colored according to coloring process 1. The other nodes of MSCCs are colored according to coloring process 2∼7. The white node means it satisfies the corresponding property formula, and the dark node means it does not satisfy the corresponding property formula. The initial node $(s_1, \mathrm{P}_{\geqslant 0.4}(q\mathrm{UG}r))$ is colored in white, which means that the given model M satisfies formula $\mathrm{P}_{\geqslant 0.4}(q\mathrm{UG}r)$. Therefore, from Fig.9, we can say that 1) *verifier* has a winning strategy, 2) the sub-graph $(s_1, \mathrm{P}_{\geqslant 0.4}(q\mathrm{UG}r)) \to (s_1, \mathrm{P}_{\geqslant 0.4}(\mathrm{G}r \vee (q \wedge \mathrm{XP}_{\geqslant ps'}(q\mathrm{UG}r)))) \to (s_1, \mathrm{P}_{\geqslant 0.4}(q \wedge \mathrm{XP}_{\geqslant ps'}(q\mathrm{UG}r))) \to (s_1, q) \wedge (s_1, \mathrm{P}_{\geqslant 0.4}(q \wedge (s_1, \mathrm{P}_{\geqslant 0.4}(q \wedge \mathrm{XP}_{\geqslant ps'}(q\mathrm{UG}r))) \to (s_5, \mathrm{P}_{\geqslant ps5}(q\mathrm{UG}r)) \to (s_5, \mathrm{P}_{<ps5}(\mathrm{G}r \vee (q \wedge$

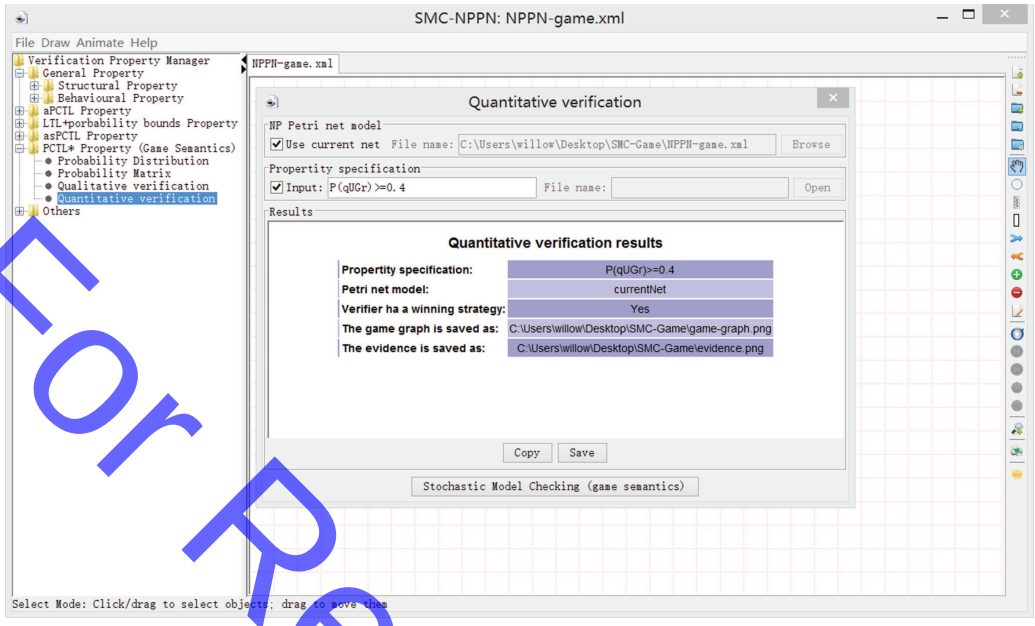


Fig.8. Stochastic model checking result.

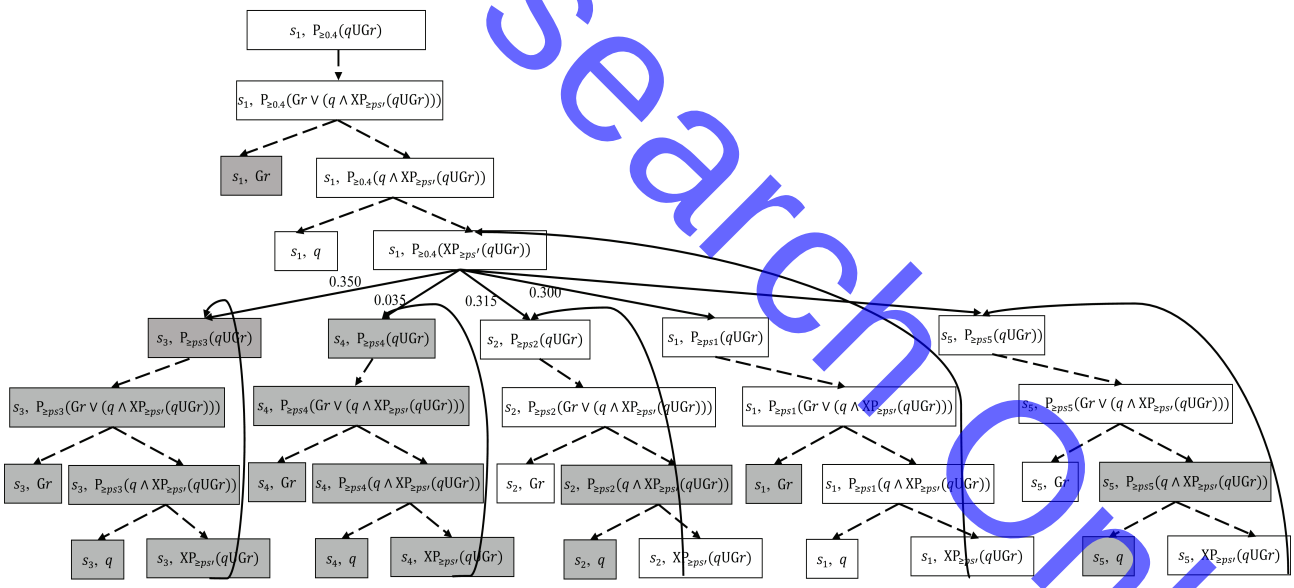


Fig.9. Colored game graph of LNPPN and PCTL*.

$\mathrm{XP}_{\geqslant ps'}(q\mathrm{UG}r)))) \rightarrow (s_5, \mathrm{G}r) \rightarrow (s_5, \mathrm{P}_{\geqslant ps5}(q\mathrm{UG}r)) \rightarrow (s_5, \mathrm{P}_{<ps5}(\mathrm{G}r \vee (q \wedge \mathrm{XP}_{<ps'}(q\mathrm{UG}r)))) \rightarrow (s_5, \mathrm{G}r)$ and Fig.10 are colored in white, and they are both independent of other nodes in the game graph. It means that they are the independent winning strategies for *verifier*. However, according to Definition 5 and Definition 6, Fig.10 is the evidence for verification because it is the event set with the smallest adversary. It denotes the corresponding sub-LNPPN is composed of place set {S1, S2}, transition set {T1, T2, T4} and flow relations among them in the LNPPN model.

If the PCTL* property formula is $\mathrm{P}_{<0.4}(q\mathrm{UG}r)$, the game graph has the same structure with Fig.9, but its colors are opposite to those in Fig.9. Therefore, *refuter* has a winning strategy, and the evidence for refutation is the following sub-graph that is colored in dark: $(s_1, \mathrm{P}_{<0.4}(q\mathrm{UG}r)) \rightarrow (s_1, \mathrm{P}_{<0.4}(\mathrm{G}r \vee (q \wedge \mathrm{XP}_{<ps'}(q\mathrm{UG}r)))) \rightarrow (s_1, \mathrm{P}_{<0.4}(q \wedge \mathrm{XP}_{<ps'}(q\mathrm{UG}r))) \rightarrow (s_1, q) \wedge (s_1, \mathrm{P}_{<0.4}(q \wedge \mathrm{XP}_{<ps'}(q\mathrm{UG}r))) \rightarrow (s_5, \mathrm{P}_{<ps5}(q\mathrm{UG}r)) \rightarrow$

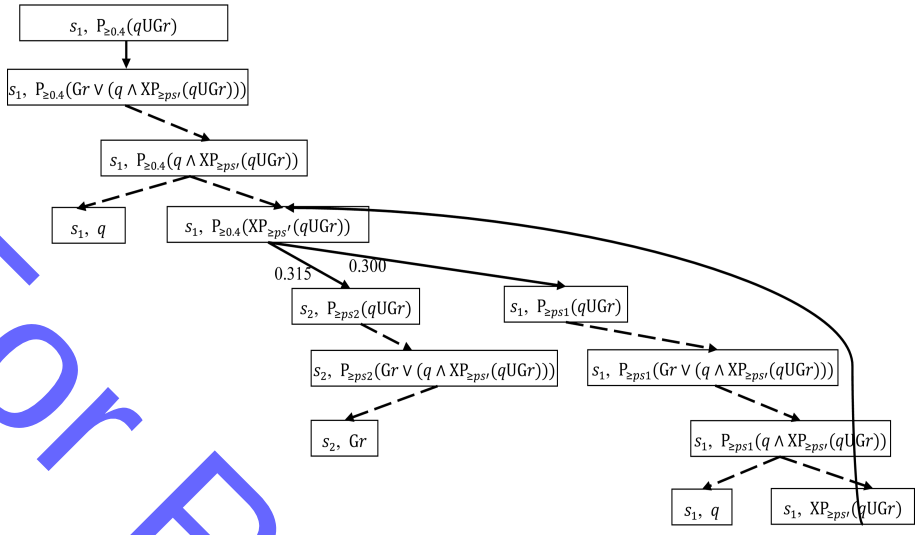


Fig.10. Evidence for verification of PCTL*.

$(s_5, \mathrm{P}_{<ps5}(\mathrm{G}r \vee (q \wedge \mathrm{XP}_{<ps'}(q\mathrm{UG}r)))) \rightarrow (s_5, \mathrm{G}r)$. It denotes the corresponding sub-LNPPN is composed of place set {S1, S5}, transition set {T6, T7} and flow relations among them in the LNPPN model.

7 Related Work

Game theory has been used as a powerful paradigm for giving the formal semantics to a variety of programming languages and logical systems[26-30]. It manages to be both the denotational semantics and operational semantics[31]. For seeking the better result of classical model checking, the game theory was introduced in bisimulation and μ-calculus model checking by Stirling *et al.*[32-34] Then, Lange and Stirling[35] defined and examined the model checking game for the branching time temporal logic, which employed the focus to enrich configuration sets by picking out one distinguished element. Shoham and Grumberg[36] extended the game-based CTL model checking[33] for counterexample generation and incremental abstraction-refinement, which produced the annotated counterexamples for full CTL and provided an iterative 3-valued abstraction-refinement framework. The work most closely related to ours is the PCTL model checking as winning strategies in games proposed by Fecher *et al.*[37] They provided a game foundation for producing diagnostics information in the setting of Markov chains and PCTL in weak until-PNF.

On the other hand, the idea of providing the evidence for justifying the result of classical model checking has appeared in some references[38-41]. Tan and Cleaveland[38] presented how to modify the classical model checkers to return the support sets as the evidence for mu-calculus. Namjoshi[39] used deductive proofs to encode evidence for the modal mu-calculus model checking, and Peled *et al.*[40-41] considered encoding evidence for the linear-time temporal-logic. However, in the setting of stochastic model checking, the related work is little, and the closest work related to ours is counterexample generation. Han *et al.*[42] originally defined the minimal and smallest counterexample, used the probabilistic path traces or regular expression to present counterexample, and adopted the k-shortest paths algorithms to generate counterexample for PCTL formula on discrete-time Markov chain model. Komuravelli *et al.*[43] defined counterexample as strong probabilistic simulation, used stochastic tree to represent counterexample, and exploited it to compositional verification. Chatterjee *et al.*[44] and Hermanns *et al.*[45] suggested viewing the general DTMC as a counterexample, and used the counterexample for planning and abstraction-refinement. Chadha and Viswanathan[23] showed that the above proposals were expressively inadequate, defined the notion of counterexample to simply be small MDP, and first proved generating the smallest counterexample was an NP-complete problem. Up to now, all the research work about counterexample in stochastic model checking is just suitable to the subset of PCTL (e.g., safe-PCTL), and the generating counterexample algorithm is dedicated, which is independent of stochastic model checking process.

We summarize the above work as follows. 1) In

stochastic model checking area, game theory is exploited for PCTL model checking DTMC, and specifically, PCTL in weak until-PNF. It is difficult to be extended to PCTL* which is the superset of PCTL and LTL with probability bounds, because transforming PCTL* into weak until-PNF will easily lead to the exponential blowup. Moreover, the winning strategy holds all the relevant information for the result of the stochastic model checking, but it has redundancies. 2) The state-of-the-art work about counterexample in stochastic model checking is just suitable for subset of PCTL (e.g., safe-PCTL, reachability), which cannot be used for PCTL*. When considering PCTL*, we face properties expressed by Boolean combinations of path formulae. If we use "probabilistic path traces" as a counterexample, the counterexample for PCTL* may be very large or an infinite set of path traces. If we use "stochastic tree" as a counterexample, it cannot express all counterexamples for PCTL*, because the family of tree-like counterexample is not rich enough. If we use "general DTMC" as a counterexample, the stochastic model checker will be used as a decision procedure in each step of the counterexample generation. 3) To the best of our knowledge, there is not any related work about the evidence constructing in unified form for stochastic model checking result.

This paper is the first presentation of game semantics for PCTL* stochastic model checking with evidence. Compared with the above work, the checked model is LNPPN which is the high-level model for system with nondeterministic and probabilistic behaviours. For avoiding the exponential blowup in transforming a PCTL* formula into PNF, we present the game semantics for PCTL* in release-PNF. Moreover, the evidence in unified form for PCTL* stochastic model checking result is constructed by refining winning strategy, which makes full use of the information in a single run of the stochastic model checker, and it can be seen as a minimal part of winning strategy that is sufficient to explain the stochastic model checking result. The algorithm presented for solving strategy and constructing evidence is more intuitive, and it is implemented in a prototype tool SMC-NPPN.

8 Conclusions

The paper made the following contributions. An intuitive and succinct game-theoretic approach for PCTL* stochastic model checking was firstly proposed. More importantly, it can provide the evidence in a unified form to certify the stochastic model checking result. In the point of view of game theory, a stochastic system model satisfies a PCTL* formula, which can be seen as the player $verifier$ having a winning strategy; otherwise, it can be seen as the player $refuter$ having a winning strategy. The evidence for verification or refutation is indeed the refinement of winning strategy of $verifier$ or $refuter$, which is minimal part of the winning strategy to win the PCTL* stochastic model checking game. Therefore, the algorithm has the advantage of providing evidence for the result of stochastic model checking easily, which makes full use of the information in stochastic model checking process and does not increase the complexity. However, the evidence for verification or refutation is minimal (may not be the smallest), and evidence presentation is complex if the model is too big, thus there is the trade-off between the detailed results and the model scale. It may be improved and optimized in several ways. In the future work, we will attempt to optimize the strategy solving and strategy refining (evidence constructing) algorithm by heuristic[46], and on the other hand, we will generalize the game-based stochastic model checking to check the continuous time stochastic system model.

References

[1] Clarke E M, Emerson E A. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. the Workshop on Logic of Programs*, May 1981, pp.52-71.

[2] Queille J P, Sifakis J. Specification and verification of concurrent systems in CESAR. In *Proc. the 5th Colloquium on International Symposium on Programming*, April 1982, pp.337-351.

[3] Lin H M, Zhang W H. Model checking: Theories, techniques and applications. *Acta Electronica Sinica*, 2002, 30(12A): 1907-1912. (in Chinese)

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

[5] Clarke E M, Emerson E A, Sifakis J. Model checking: Algorithmic verification and debugging. *Communications of the ACM*, 2009, 52(11): 74-84.

[6] Kwiatkowska M, Norman G, Parker D. Stochastic model checking. In *Proc. the 7th International Conference on Formal Methods for Performance Evaluation*, May 28-June 2, 2007, pp.220-270.

[7] Ndukwu U, Mclver A. An expectation transformer approach to predicate abstraction and data independence for probabilistic programs. In *Proc. the 8th Workshop on Quantitative Aspects of Programming Languages*, Mar. 2010, pp.129-143.

[8] Baier C, Haverkort B R, Hermanns H, Katoen J P. Performance evaluation and model checking join forces. *Communication of the ACM*, 2010, 53(9): 74-85.

[9] Duflot M, Kwiatkowska M, Norman G, Parker D, Peyronnet S, Picaronny C, Sproston J. Practical applications of probabilistic model checking to communication protocols. In *Formal Methods for Industrial Critical Systems: A Survey of Applications*, Gnesi S, Margaria T (eds.), John Wiley & Sons, Inc., 2012, pp.133-150

[10] Calinescu R, Grunske L, Kwiatkowska M, Mirandola R, Tamburrelli G. Dynamic QoS management and optimisation in service-based systems. *IEEE Transactions on Software Engineering*, 2011, 37(3): 387-409.

[11] Kwiatkowska M, Norman G, Parker D. Probabilistic model checking for systems biology. In *Symbolic Systems Biology*, Iyengar M S (ed.), Jones and Bartlett, 2010, pp.31-59.

[12] Hansson H, Jonsson B. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 1994, 6(5): 512-535.

[13] Pnueli A. The temporal logic of programs. In *Proc. the 18th IEEE Symposium on Foundations of Computer Science*, Oct.31-Nov.2, 1977, pp.46-67.

[14] Aziz A, Sanwal K, Singhal V, Brayton R. Model-checking continuous time Markov chains. *ACM Transactions on Computational Logic*, 2000, 1(1): 162-170.

[15] Baier C, Haverkort B, Hermanns H, Katoen J P. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 2003, 29(6): 524-541.

[16] Liu Y, Miao H, Zeng H, Ma Y, Liu P. Nondeterministic probabilistic Petri net — A new method to studying qualitative and quantitative behaviors of system. *Journal of Computer Science and Technology*, 2013, 28(1): 203-216.

[17] Hintikka J. Logic, Language-Games and Information: Kantian Themes in the Philosophy of Logic. Clarendon Press, Oxford, 1973.

[18] Petri C A. Introduction to general net theory. In*Lecture Notes in Computer Science 84*, Brauer, W (ed.), Springer-Verlag, 1980, 84: 1-19.

[19] Bollig B, Leucker M, Weber M. Local parallel model checking for the alternation-free μ-calculus. In *Proc. the 9th International SPIN Workshop on Model Checking of Software*, April 2002, pp.128-147.

[20] Savitch W J. Deterministic simulation of non-deterministic turing machines. In *Proc. the 1st ACM Symposium on Theory of Computing*, May 1969, pp.247-248.

[21] Kwiatkowska M, Norman G, Parker D. PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. the 23rd International Conference on Computer Aided Verification*, Jul. 2011, pp.585-591.

[22] Clarke E M, Jha S, Lu Y, Veith H. Tree-like counterexamples in model checking. In *Proc. the 17th IEEE Symposium on Logic in Computer Science*, Jul. 2002, pp.19-29.

[23] Chadha R, Viswanathan M. A counterexample-guided abstraction-refinement framework for Markov decision processes. *ACM Transactions on Computational Logic*, 2010, 12(1): 1:1-1:49.

[24] Bonet P, Llado C M, Puijaner R, Knottenbelt W J. PIPE v2.5: A Petri net tool for performance modelling. In *Proc. the 23rd Latin American Conference on Informatics* (*CLEI 2007*), October 2007.

[25] Dingle N J, Knottenbelt W J, Suto T. PIPE2: A tool for the performance evaluation of generalised stochastic Petri nets. *ACM SIGMETRICS Performance Evaluation Review,* 2009, 36(4): 34-39.

[26] Ong C H L. Verification of higher-order computation: A game-semantic approach. In *Proc. the 17th European Symposium on Programming*, Mar.29-Apr.6, 2008, pp.299-306.

[27] Abramsky S, Ghica D, Murawski A, Ong C H L. Applying game semantics to compositional software modelling and verification. In *Proc. the 10th International Conference Tools and Algorithms for the Construction and Analysis of Systems*, Mar.29-Apr.2, 2004, pp.421-435.

[28] Abramsky S, Jagadeesan R. Game semantics for access control. *Electronic Notes in Theoretical Computer Science*, 2009, 249: 135-156.

[29] Fredriksson O, Ghica D R. Abstract machines for game semantics, revisited. In *Proc. the 28th Annual IEEE/ACM Symposium Logic in Computer Science*, Jun. 2013, pp.560-569.

[30] Stirling C. Proof systems for retracts in simply typed lambda calculus. In *Proc. the 40th International Conference on Automata, Languages, and Programming*, Jul. 2013, pp.398-409.

[31] Ghica D R. Applications of game semantics: From program analysis to hardware synthesis. In *Proc. the 24th Annual IEEE Symposium on Logic in Computer Science*, Aug. 2009, pp.17-26.

[32] Zhu X Y, Zhang W H, Li G Y, Lv Y, Lin H M. Report on advances in model checking. In *Report on Advances in Computer Science and Technology* (2011-2012), China Science and Technology Press, 2012. (in Chinese)

[33] Stirling C. Bisimulation, model checking and other games. http://homepages.inf.ed.ac.uk/cps/mathfit.pdf, Dec. 2015.

[34] Stirling C. Modal and Temporal Properties of Processes. Springer-Verlag New York, 2001.

[35] Lange M, Stirling C. Model checking games for branching time logics. *Journal of Logic Computation*, 2002, 12(4): 623-939.

[36] Shoham S, Grumberg O. Game-based framework for CTL counterexamples and 3-valued abstraction-refinement. *ACM Transactions on Computational Logic* (*TOCL*), 2007, 9(1): 1:1-1:52.

[37] Fecher H, Huth M, Piterman N, Wagner D. PCTL model checking of Markov chains: Truth and falsity as winning strategies in games. *Performance Evaluation*, 2010, 67(9): 858-872.

[38] Tan L, Cleaveland R. Evidence-based model checking. In *Proc. the 14th International Conference on Computer Aided Verification*, Jul. 2002, pp.455-470.

[39] Namjoshi K. Certifying model checkers. In *Proc. the 13th International Conference on Computer Aided Verification*, Jul. 2001, pp.2-13.

[40] Peled D, Pnueli A, Zuck L. From falsification to verification. In *Lecture Notes in Computer Science 2245*, Hariharan R, Vinay V, Mukund M (eds.), Springer-Verlag, 2001, pp.292-304.

[41] Peled D, Zuck L. From model checking to a temporal proof. In *Proc. the 8th International SPIN Workshop on Model Checking of Software*, May 2001, pp.1-14.

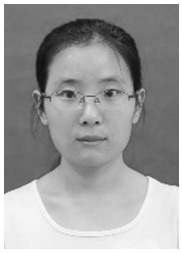
[42] Han T T, Katoen J P, Damman B. Counterexample generation in probabilistic model checking. *IEEE Transactions on Software Engineering*, 2009, 35(2): 241-257.

[43] Komuravelli A, Păsăreanu C S, Clarke E M. Assume-guarantee abstraction refinement for probabilistic systems. In *Proc. the 24th International Conference on Computer Aided Verification*, Jul. 2012, pp.310-326.

[44] Chatterjee K, Henzinger T, Jhala R, Majumdar R. Counter example-guided planning. In*Proc. the 21st International Conference on Uncertainty in Artificial Intelligence*, Jul. 2005, pp.104-111.
[45] Hermanns H, Wachter B, Zhang L Z. Probabilistic CEGAR. In *Proc. the 20th International Conference on Computer Aided Verification*, Jul. 2008, pp.162-175.
[46] Aljazzar H, Leue S. Directed explicit state-space search in the generation of counterexamples for stochastic model checking. *IEEE Transactions on Software Engineering*, 2010, 36(1): 37-60.

Yang Liu received his Ph.D. degree in computer science and technology from Shanghai University in July 2012. He is a post-doctoral researcher in the State Key Laboratory for Novel Software Technology at Nanjing University, and he is also a visitor post-doctoral researcher in the Department of Computer Science at National University of Singapore. His research interests include software engineering and formal verification. He is a member of CCF and ACM.

Xuan-Dong Li received his M.S. and Ph.D. degrees from Nanjing University, Nanjing, in 1991 and 1994, respectively. He is a full professor at the Department of Computer Science and Technology and the State Key Laboratory for Novel Software Technology of Nanjing University. Currently, he is the dean of both the Department of Computer Science and Technology and the Software Institute of Nanjing University. His research interests include software modeling and analysis, and software testing and verification.



Yan Ma received her M.S. degree in computer science and technology from Jiangnan University, Wuxi, in 2007. She is a Ph.D. candidate in the College of Computer Science and Technology at Nanjing University of Aeronautics and Astronautics. Her research interests include software engineering and hybrid system verification.

Appendix

Algorithm of Constructing Evidence for PCTL* Stochastic Model Checking Result

consEvidence (Gg, Con_0){
N'={Con_0}
$Temp1$={Con_0}
$Temp2$=∅
if the color of Con_0 is white{
while $Temp1 \neq \emptyset$ do{
node=remove ($Temp1$)
$Temp2$= $Temp2$ ∪{*node*}
if the Φ of *node* is the form of true/false/a/$\neg a$/$P_{\geq p}$(true/false/a/$\neg a$) then continue
if the Φ of *node* is the form of $\Phi_0 \wedge \Phi_1$ {
for each (*node*, Con_i) in Gg do{
if $Con_i \notin Temp2$ then{
$Temp2$= $Temp2$ ∪{Con_i}; N'=N' ∪{Con_i};E'=E' ∪{(*node*, Con_i)} } }
if the Φ of *node* is the form of $\Phi_0 \vee \Phi_1$ {
Con_i is the reason for coloring the *node*
if $Con_i \notin Temp2$ then{
$Temp2$= $Temp2$ ∪{Con_i}; N'=N' ∪{Con_i};E'=E' ∪{(*node*, Con_i)} }}
if the Φ of *node* is the form of $P_{\geq p}$(XΨ){
for each Con_i colored in white of (*node*, Con_i) in the smallest adversary of Gg do{
pick Con_i from big to small according to the value of $P(node, Con_i) \times P(Con_i \models \Psi)$
while ($\sum_i P(node, Con_i) \times P(Con_i \models \Psi) < p$){

if $Con_i \notin Temp2$ then{
$Temp2 = Temp2 \cup \{Con_i\}$; $N' = N' \cup \{Con_i\}$; $E' = E' \cup \{(node, Con_i)\}$ }}}
if the Φ of *node* is the form of $P_{\geq p}(\Psi_1 \wedge \Psi_2)${
for each Con_i colored in white of $(node, Con_i)$ in Gg do{
pick Con_i from big to small according to the value of $P(Con_i \vDash \Psi_1 \wedge \Psi_2)$
while ($P(Con_i \vDash \Psi_1 \wedge \Psi_2) < p$){
if $Con_i \notin Temp2$ then{
$Temp2 = Temp2 \cup \{Con_i\}$; $N' = N' \cup \{Con_i\}$; $E' = E' \cup \{(node, Con_i)\}$ }}}
if the Φ of *node* is the form of $P_{\geq p}(\Psi_1 \vee \Psi_2)${
for each Con_i colored in white of $(node, Con_i)$ in Gg do{
pick Con_i from big to small according to the value of $P(Con_i \vDash \Psi_1 \vee \Psi_2)$
while ($P(Con_i \vDash \Psi_1 \vee \Psi_2) < p$){
if $Con_i \notin Temp2$ then{
$Temp2 = Temp2 \cup \{Con_i\}$; $N' = N' \cup \{Con_i\}$; $E' = E' \cup \{(node, Con_i)\}$ }}}
if the Φ of *node* is the form of $P_{\geq p}(\Psi_0 U \Psi_1)${
construct according to the Φ of *node* is the form of $P_{\geq p}(\Psi_1 \vee \Psi_2)$ and $P_{\geq p}(\Psi_1 \wedge \Psi_2)$}
if the Φ of *node* is the form of $P_{\geq p}(\Psi_0 R \Psi_1)${
construct according to the Φ of *node* is the form of $P_{\geq p}(\Psi_1 \wedge \Psi_2)$ and $P_{\geq p}(\Psi_1 \vee \Psi_2)$}
}return evidence $Ev(N', E', w')$ for verification}
if the color of Con_0 is dark{
while $Temp1 \neq \emptyset$ do{
node=remove ($Temp1$)
$Temp2 = Temp2 \cup \{node\}$
if the Φ of *node* is the form of true/false/a/$\neg a$/$P_{\geq p}$(true/false/a/$\neg a$) then continue
if the Φ of *node* is the form of $\Phi_0 \vee \Phi_1${
for each $(node, Con_i)$ in Gg do{
if $Con_i \notin Temp2$ then{
$Temp2 = Temp2 \cup \{Con_i\}$; $N' = N' \cup \{Con_i\}$; $E' = E' \cup \{(node, Con_i)\}$ }}
if the Φ of *node* is the form of $\Phi_0 \wedge \Phi_1${
Con_i is the reason for coloring the *node*
if $Con_i \notin Temp2$ then{
$Temp2 = Temp2 \cup \{Con_i\}$; $N' = N' \cup \{Con_i\}$; $E' = E' \cup \{(node, Con_i)\}$ }}
if the Φ of *node* is the form of $P_{\geq p}(X\Psi)${
for each Con_i colored in dark of $(node, Con_i)$ in the biggest adversary of Gg do{
pick Con_i from big to small according to the value of $P(node, Con_i) \times P(Con_i \nvDash \Psi)$
while ($\sum_i P(node, Con_i) \times P(Con_i \nvDash \Psi) < 1-p$){
if $Con_i \notin Temp2$ then{
$Temp2 = Temp2 \cup \{Con_i\}$; $N' = N' \cup \{Con_i\}$; $E' = E' \cup \{(node, Con_i)\}$ }}}
if the Φ of *node* is the form of $P_{\geq p}(\Psi_1 \vee \Psi_2)${
for each Con_i colored in dark of $(node, Con_i)$ in Gg do{
pick Con_i from big to small according to the value of $P(Con_i \nvDash \Psi_1 \vee \Psi_2)$
while ($P(Con_i \nvDash \Psi_1 \vee \Psi_2) < 1-p$){
if $Con_i \notin Temp2$ then{
$Temp2 = Temp2 \cup \{Con_i\}$; $N' = N' \cup \{Con_i\}$; $E' = E' \cup \{(node, Con_i)\}$ }}}

if the Φ of *node* is the form of $P_{\geq p}(\Psi_1 \wedge \Psi_2)$ {
  for each Con_i colored in dark of $(node, Con_i)$ in Gg do{
  pick Con_i from big to small according to the value of $P(Con_i \not\models \Psi_1 \wedge \Psi_2)$
  while ($P(Con_i \not\models \Psi_1 \wedge \Psi_2)$<1- p){
    if $Con_i \notin Temp2$ then{
    $Temp2 = Temp2 \cup \{Con_i\}$; $N' = N' \cup \{Con_i\}$; $E' = E' \cup \{(node, Con_i)\}$ }}}
    if the Φ of *node* is the form of $P_{\geq p}(\Psi_0 U \Psi_1)$ {
      construct according to the Φ of *node* is the form of $P_{\geq p}(\Psi_1 \vee \Psi_2)$ and $P_{\geq p}(\Psi_1 \wedge \Psi_2)$}
    if the Φ of *node* is the form of $P_{\geq p}(\Psi_0 R \Psi_1)$ {
      construct according to the Φ of *node* is the form of $P_{\geq p}(\Psi_1 \wedge \Psi_2)$ and $P_{\geq p}(\Psi_1 \vee \Psi_2)$}
  }return evidence $Ev(N', E', w')$ for refutation}
}