

# A Two-phase Approximation for Model Checking Probabilistic Unbounded Until Properties of Probabilistic Systems

Paul Jennings  
Arka P. Ghosh  
Samik Basu

---

We have developed a new approximate probabilistic model checking of probabilistic systems for *untimed* properties expressed in probabilistic temporal logic (PCTL, CSL). This method, in contrast to the existing ones, does not require the untimed until properties to be *bounded* a priori, where the bound refers to the number of discrete steps in the system necessary to verify the until properties. The method has two phases. In the first phase, given an until property, a suitable system-dependent bound  $k_0$  is obtained automatically. In the second phase, the probability of satisfying the  $k_0$ -bounded until property is computed as the estimate for the probability of satisfying the given unbounded until property. Both phases require verification of only bounded until properties which can be effectively performed by simulation based methods. We prove the correctness of the proposed two-phase method and present its optimized implementation in the widely used PRISM model checking engine. We show that for several models existing probabilistic model checking methods, as implemented in PRISM, fail; while the two-phase method successfully computes the result. Second, we show that the implementation of two-phase method is as fast as (typically, 1.5 times as fast as) the PRISM's statistical method.

Categories and Subject Descriptors: D.2.4 [Software Engineering]: Software/Program Verification—*Formal Methods*; D.2.4 [Software Engineering]: Software/Program Verification—*Model Checking*; D.2.4 [Software Engineering]: Software/Program Verification—*Statistical Methods*; F.4.1 [Mathematical Logic and Formal Methods]: Mathematical Logic—*Temporal Logic*

General Terms: Verification

Additional Key Words and Phrases: DTMC, CTMC, PCTL, CSL

---

## 1. INTRODUCTION

A large variety of systems (e.g. communication protocols over lossy channels, client-server protocols with unreliable servers, and distributed leader-election algorithms) exhibit probabilistic behavior in which the systems evolve from one configuration to another following a certain pre-specified probability distribution. Such probabilistic

---

This article is an expanded version of a paper published in the proceedings of 11th International Conference on Formal Engineering Methods, 2009. Authors address: Paul Jennings and Samik Basu, Department of Computer Science, Iowa State University; Arka P. Ghosh, Department of Statistics, Iowa State University; Email: {poj,sbasu,apghosh}@iastate.edu. This work is supported in parts by NSF grants CNS0709217, CCF0702758.

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 20YY ACM 0000-0000/20YY/0000-0001 \$5.00

behaviors are modeled using discrete time Markov chains (DTMC), continuous time Markov chains (CTMC), Markov decision process (MDP). Several techniques and tools have been developed to prove the correctness (in a probabilistic sense) of these system-models. One such technique, which automatically verifies the conformance of probabilistic systems modeled as DTMC, CTMC or MDP ([Roy and Gopinath 2005; Norman and Shmatikov 2006; Dufflot et al. 2006; Kwiatkowska et al. 2008]) against desired properties expressed in probabilistic temporal logic (e.g. PCTL [Hansson and Jonsson 1994], CSL [Aziz et al. 2000]), is called probabilistic model checking.

Broadly, there are two categories of probabilistic model checking methods. The first category, typically referred to as *numerical method*, [Hansson and Jonsson 1994; Bianco and de Alfaro 1995; Courcoubetis and Yannakakis 1995; Aziz et al. 2000; Baier et al. 2003] relies on exploration of the entire state-space of the probabilistic systems and applies linear equation solvers to obtain the exact probability with which the system satisfies a property. In contrast, the other method, referred to as *approximate or statistical method* [Younes and Simmons 2002; Herault et al. 2004; Sen et al. 2005], samples a finite set of paths in the system and infers the approximate probability of satisfiability of a property by the whole system using probabilistic arguments. While the numerical method provides exact solutions, it requires complete knowledge of the system and may fail for systems with large state-space (state-space explosion problem). It is in this situation that sampling based statistical methods are useful. However, two important issues need to be addressed before any statistical verification approach can be applied effectively: the number of sample paths and the length for each sample path.

The sampling method explores only a portion of the state-space of the system and therefore the accuracy of the verification results depends on the size  $N$  of the sample (i.e., the number of sample paths). This value of  $N$  is typically obtained from well-known probabilistic bounds that ascertain the closeness of the estimate to the actual probability with respect to certain pre-specified error limit ( $\epsilon$ ) and confidence parameter ( $\delta$ ).

By definition, the sampling method can consider only paths of finite length. This is not a problem when the property under consideration has a specific bound (*bounded path property*):  $\varphi_1 \text{ U}^{\leq k} \varphi_2$ , i.e.,  $\varphi_2$  must be satisfied within  $k$  steps from the start state. This implies that finite paths of length  $k$  are sufficient to verify such properties.

However, the property of interest may be *unbounded*, i.e.,  $\varphi_1 \text{ U } \varphi_2$ . The semantics of the property states that a path satisfies it if and only if there exists a state in the path which satisfies  $\varphi_2$  and in all states before that state  $\varphi_1$  is satisfied. Note that  $\varphi_1$  can be satisfied any number of times in a path before  $\varphi_2$  is satisfied for the first time. Therefore, in any path of finite length where every state satisfies  $\varphi_1 \wedge \neg \varphi_2$ , it is impossible to infer whether the extension of the path will eventually satisfy or not satisfy  $\varphi_1 \text{ U } \varphi_2$ . The existing statistical methods for probabilistic model checking either assume that the bound is given [Herault et al. 2004] (in essence verifying a bounded path property) or require some specific knowledge of the system behavior [Sen et al. 2005].

In contrast, we introduce a new statistical method which computes the bound  $k$  of simulation path length automatically and does not require any prior knowledge

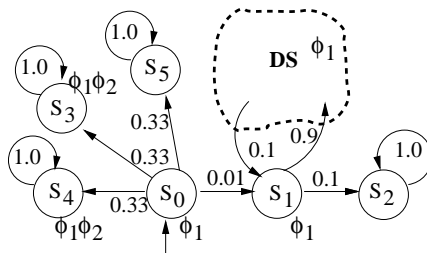


Fig. 1. A simple example.

of the system. The central theme of our technique is to reduce the problem of verifying  $(\varphi_1 \text{ U } \varphi_2)$  into that of its bounded counter-part  $(\varphi_1 \text{ U}^{\leq k_0} \varphi_2)$ . The reduction is possible only when a suitable  $k_0$  can be obtained for which the  $P(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2)$  (i.e., probability of satisfying of  $\varphi_1 \text{ U}^{\leq k_0} \varphi_2$  at state  $s$ ) is a *good* approximation of  $P(s, \varphi_1 \text{ U } \varphi_2)$ . In other words, the bound  $k_0$  is large enough to make the difference between  $P(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2)$  and  $P(s, \varphi_1 \text{ U } \varphi_2)$  small. Such a  $k_0$  provides an approximate upper bound of the sample path lengths needed for our statistical verification technique. We obtain  $k_0$  using the probability of satisfying a different *bounded* path property:  $\psi_k := (\varphi_1 \text{ U}^{\leq k} \varphi_2) \vee (\neg \varphi_2 \text{ U}^{\leq k} (\neg \varphi_1 \wedge \neg \varphi_2))$ . This property states that the property  $(\varphi_1 \text{ U } \varphi_2)$  is either satisfied (first disjunct) or unsatisfied (second disjunct) in at most  $k$  steps. We prove that the suitable  $k_0$  is one for which  $P(s, \psi_{k_0})$  is close to 1, and the degree of “closeness” is related to  $\epsilon$ , the overall measure of accuracy of the entire statistical method.

In essence, there are two phases in our method. The first phase estimates  $P(s, \psi_k)$  for  $k = 0, 1, 2, \dots$  and chooses  $k_0$  which satisfies  $P(s, \psi_{k_0}) \geq 1 - \epsilon_0$ , where  $\epsilon_0 < \epsilon$ . In the second phase,  $P(s, (\varphi_1 \text{ U}^{\leq k_0} \varphi_2))$  is estimated, which in turn serves as an estimate of  $P(s, (\varphi_1 \text{ U } \varphi_2))$ . The computations for each phase involve only *bounded-path* properties and can be carried out efficiently using the existing sampling techniques such as [Herauld et al. 2004]. For systems where  $P(s, \psi_k) \not\geq 1 - \epsilon_0$  for any  $k$ , we propose an alternate heuristic method to estimate  $P(s, (\varphi_1 \text{ U } \varphi_2))$  based on changes in the estimates of  $P(s, \psi_k)$  with  $k$ .

### 1.1 Illustrative Example

To provide an intuitive explanation of why the proposed technique is useful and effective, we present a simple toy example (Figure 1) where the proposed method is applied successfully and where both the numerical method and the existing statistical verification method as implemented in the popular PRISM model checker [Hinton et al. 2006] fail. The example contains a probabilistic transition system containing  $6 + n$  states where  $n$  is some large integer. The state  $s_0$  is the start state of the system. The dotted segment in the figure represents some “complicated” transition structure on  $n$  different states (see [Two Phase Probabilistic Model Checking 2010] for the specification). We will refer to this segment as DS. Let proposition  $\varphi_1$  hold in all states except  $s_2$  and  $s_5$ , and proposition  $\varphi_2$  hold in states  $s_3$  and  $s_4$ . The objective is to find the probability of satisfying the property  $(\varphi_1 \text{ U } \varphi_2)$  at state  $s_0$ . From the probabilities specified in the figure, we know that the resultant probability

is 0.66 as only two paths  $(s_0, s_3, \dots)$  and  $(s_0, s_4, \dots)$  satisfy the property.

We experimented with the PRISM model checker [Hinton et al. 2006] using the above example. PRISM’s numerical method fails as the large state-space (large  $n$ ) results in state-space explosion. PRISM’s statistical method takes as input a parameter  $\epsilon$  and provides an approximate result within  $\epsilon$  error margin. Our experiments with several  $\epsilon > 0$  failed to provide any estimate. This is because PRISM’s statistical method requires that the satisfaction of a property  $\varphi_1 \cup \varphi_2$  be known within some pre-specified number of steps. The failure happens when at least one sample path enters DS. In this case,  $\varphi_1$  is satisfied for all states in the path, but  $\varphi_2$  is not satisfied for any states in the path.

In terms of  $\psi_k$  introduced earlier, the above requirement in PRISM’s statistical method is equivalent to  $P(s_0, \psi_k) = 1$ , for some pre-specified bound  $k$  ( $k = 10,000$  by default in PRISM). In general, it is not possible to find an appropriate value for  $k$  such that  $P(s_0, \psi_k) = 1$ . In contrast, we claim that it is not necessary to verify whether  $P(s_0, \psi_k)$  is equal to 1. The necessary precision for an approximate statistical model checking can be obtained by identifying a  $k$  ( $k_0$  in our terminology) for which  $P(s_0, \psi_k)$  is close to 1. In the above example, such a bound can be immediately obtained as the sample paths  $(s_0, s_1, \dots)$  have very low probability ( $\leq 0.01$ ). Once such a bound is obtained, we compute  $P(s_0, \varphi_1 \cup^{\leq k_0} \varphi_2)$  which approximately coincides with  $P(s_0, \varphi_1 \cup \varphi_2)$ . In our experiments, with  $n \approx 10^8$ , while the PRISM model checker fails to provide any result, our method identifies a bound  $k_0 = 1972$  and estimates the probability to be equal to 0.6605 in approximately 80sec.

## 1.2 Contributions

The contributions of our approach are summarized as follows:

- (1) *Automation.* We present a methodology for automatically selecting a suitable bound  $k_0$  through which unbounded until properties for probabilistic systems modeled as Discrete Time and Continuous Time Markov Chains can be verified using the corresponding  $k_0$ -bounded until properties. The reduction allows us to re-use existing results of statistical verification of bounded until properties to identify the bound on sample size  $N$  required to infer results within a pre-specified error margin.
- (2) *Universal Application.* The technique is applied for probabilistic model checking of any unbounded (untimed) path properties (expressed in PCTL or CSL) for models expressed in DTMC and CTMC.
- (3) *Theoretical Correctness.* We prove the soundness of our technique and discuss the condition under which our technique will always terminate with an estimate within a pre-specified error bound and confidence parameter. When the required condition is not satisfied in a system, our technique (as well as any other statistical method for probabilistic model checking that do not require prior knowledge of complete model structure) will fail to terminate. We discuss a heuristic for dealing with such systems.
- (4) *Effective Implementation and Usability.* We present an optimized implementation PRISM-U2B of our method based on the well-developed probabilistic model checking tool PRISM. We leverage on PRISM’s realization of generating sample simulations from a given DTMC or CTMC model and re-use PRISM’s widely-used

graphical user interface, command-line interface and input specification languages, thereby, reducing the cognitive burden of understanding and using PRISM-U2B.

- (5) *Experimental Evaluation.* We compare PRISM-U2B and PRISM, and empirically show that PRISM-U2B is about 1.5 times faster than PRISM for the examples where both can compute an estimate. We discuss several examples (and present results) where PRISM fails to provide an estimate while PRISM-U2B successfully terminates with an estimate. The tool PRISM-U2B, its documentation and case studies<sup>1</sup> can be obtained at [Two Phase Probabilistic Model Checking 2010].

### 1.3 Organization

Section 2 discusses related work. Section 3 provides a brief overview of discrete time Markov chain and probabilistic temporal logic PCTL. Section 4 presents our solution methodology and its proof of correctness. The implementation is discussed in Section 5. Section 6 discusses the necessary condition for the termination of our technique and presents a heuristic method when such condition is not satisfied. Section 7 discusses the application of our technique for continuous time Markov chains. Section 8 presents a brief summary of the tool PRISM-U2B followed by its empirical evaluation using several examples in Section 9. Finally, Section 10 concludes with the summary and future avenues of research.

## 2. RELATED WORK

The statistical method based on Monte Carlo simulation and sequential hypothesis testing [Wald 1945] for verifying time bounded until CSL properties in CTMC is developed by Younes and Simmons [Younes and Simmons 2002]. Similar techniques are proposed and discussed in [Younes et al. 2006]. Sen et al. [Sen et al. 2005] introduce a new statistical model checking algorithm for handling unbounded until properties based on Monte Carlo simulation and hypothesis testing. However, the technique suffers from the drawback that it requires some prior knowledge of the actual probability of satisfying an unbounded until property in order to obtain valid results. Furthermore, the validity of the technique requires that the system does not contain self-loops in any state [Younes and Simmons 2006]. Zapreev [Zapreev 2008] also proposes a statistical technique for verifying unbounded until properties. However, this technique requires knowledge of the model structure and relies on user-specified sample size and sample path lengths.

Closest to our approach is the technique proposed by Herault et al. in [Herault et al. 2004]. The proposed method based on Monte Carlo simulation uses estimation from the Chernoff-Hoeffding inequality [Hoeffding 1963] to verify a subset of LTL formulae, namely EPF (Essentially Positive Fragment) in DTMC. The algorithm includes the checking of the unbounded until properties. However, it fails to completely control the error in the procedure. The reason is essentially as follows. The sample path length used in the procedure has a pre-specified upper bound. If a simulation reaches that bound and fails to infer a decided result, the technique assumes that the simulation, if allowed to proceed, will eventually have results defending the null hypothesis ([Casella and Berger 2002]) of the testing procedure.

<sup>1</sup>Most of the case studies are available at <http://www.prismmodelchecker.org/casestudies/index.php>.

This assumption allows the method to control Type I Error (the error that the null hypothesis  $H_0$  is true but the test incorrectly rejects  $H_0$ ) within a pre-specified limit. However, as the authors state in [Herault et al. 2004], the proposed technique cannot determine the appropriate upper bound of simulation path length to control the number of the undecided simulations. As such, the method loses the control of Type II Error (the error that the null hypothesis  $H_0$  is false but the test incorrectly accepts  $H_0$ ).

This technique ([Herault et al. 2004]) is incorporated in the popular probabilistic model checker PRISM [Hinton et al. 2006]. The distinguishing feature of PRISM's statistical approach is that unlike [Herault et al. 2004] which allows the undecided simulations, PRISM requires that every simulation terminate with a decided result. This requirement makes it necessary to appropriately identify the bound on sample length for which simulation run of each sample will have a decided result; if such bound is not used, PRISM's simulation based technique fails to compute the result and requests the user to re-run the experiment with a new bound.

Our technique, a natural extension of [Herault et al. 2004], addresses this problem by using a two phase method where in the first phase an appropriate system-dependent bound  $k_0$  in sample length is obtained automatically and in the second phase this bound is used to compute the result for unbounded until properties.

### 3. PRELIMINARIES

We proceed with a brief summary on probabilistic systems modeled as discrete time Markov chains followed by the syntax and semantics probabilistic properties expressed in the logic of PCTL. The proposed method, its explanations and theoretical results will be presented in terms of these concepts. Subsequently, we will show (Section 7) that the proposed method is equally applicable for specific types of reachability properties in Continuous Time Markov Chains.

#### 3.1 Probabilistic Systems: Discrete Time Markov Chain Models

We will describe the behavior of a system which evolves from one configuration to another based on a certain probability as state machines augmented with probabilities labeling the transitions. In its simplest form, where every transition in the state machine represents probabilistic choice and every choice only depends on the current configuration, the representation aligns with discrete time Markov chains.

*Definition 1.* A Discrete Time Markov Chain,  $DTMC = (S, s_I, T, L)$ , where  $S$  is a finite set of states,  $s_I \in S$  is the initial or start state,  $T : S \times S \rightarrow [0, 1]$  is a transition probability function such that  $\forall s : \sum_{s' \in S} T(s, s') = 1$ , and  $L : S \rightarrow \mathcal{P}(AP)$  is the labeling function which labels each state with a set of atomic propositions  $\subseteq AP$  that holds in that state.

*Paths and Probability Measures.* A path in DTMC, denoted by  $\pi$ , is a finite or infinite sequence of states  $(s_0, s_1, s_2, s_3, \dots)$  such that for all  $i \geq 0 : s_i \in S$  and  $T(s_i, s_{i+1}) > 0$ . We denote the set of all infinite paths starting from  $s$  as  $Path(s)$ .  $\pi[i]$  denotes the  $i$ -th state in the path  $\pi$  and  $|\pi|$  is the length of  $\pi$  in terms of the number of transitions in  $\pi$ . For example, for an infinite path  $\pi$ ,  $|\pi| = \infty$ , while for a finite path  $\pi = (s_0, \dots, s_n)$ ,  $|\pi| = n, n \geq 0$ . The cylinder set, denoted by  $C_s(\pi)$ , for a state  $s$  and  $\pi$  a finite length path starting from  $s$ , is defined as

$C_s(\pi) = \{\pi' : \pi' \in Path(s) \wedge \pi \text{ is prefix of } \pi'\}$ . Essentially,  $C_s(\pi)$  is the set of all infinite paths  $\in Path(s)$  with the common finite length prefix  $\pi$ . For any finite path  $\pi$  with  $|\pi| = n$  we define

$$P(\pi) = \begin{cases} 1 & \text{if } n = 0 \\ T(\pi[0], \pi[1]) \times \dots \times T(\pi[n-1], \pi[n]) & \text{otherwise} \end{cases} \quad (1)$$

For a cylinder  $C_s(\pi)$ , define  $Pr(C_s(\pi)) = P(\pi)$ . It is well-known that this probability measure  $Pr(\cdot)$  extends uniquely over all sets in the relevant  $\sigma$ -algebra of path(s).

### 3.2 Probabilistic Properties

Properties of DTMC can be expressed using PCTL, an extension of standard CTL augmented with probabilistic specifications. Let  $\varphi$  represent a state formula and  $\psi$  represent a path formula. Then PCTL syntax is defined as follows:

$$\varphi \rightarrow tt \mid a \in AP \mid \neg\varphi \mid \varphi \wedge \varphi \mid P_{\bowtie r}(\psi) \quad \text{and} \quad \psi \rightarrow \varphi \text{ U } \varphi \mid \varphi \text{ U}^{\leq k} \varphi$$

In the above,  $\bowtie \in \{\leq, \geq, <, >\}$ ,  $r \in [0, 1]$  and  $k \in \{0, 1, \dots\}$ . Note that we always use state formulas to specify the properties of a DTMC and path formulas only occur inside  $P_{\bowtie r}(\cdot)$ . A state  $s$  (or a path  $\pi$ ) satisfying a state formula  $\varphi$  (or a path formula  $\psi$ ) is denoted by  $s \models \varphi$  (or  $\pi \models \psi$ ), and is inductively defined as follows:

$$\begin{array}{lll} s \models tt & \text{for all } s \in S & s \models a \Leftrightarrow a \in L(s) & s \models \neg\varphi \Leftrightarrow s \not\models \varphi \\ s \models \varphi_1 \wedge \varphi_2 & \Leftrightarrow s \models \varphi_1 \text{ and } s \models \varphi_2 & s \models P_{\bowtie r}(\psi) & \Leftrightarrow P(s, \psi) \bowtie r \end{array}$$

In the above,  $P(s, \psi) = Pr(\{\pi \in Path(s) : \pi \models \psi\})$ . In other words,  $s \models P_{\bowtie r}(\psi)$  holds if and only if the probability that  $\psi$  is true for an outgoing infinite path from state  $s$  is  $\bowtie r$ . For any infinite path  $\pi$ :

$$\begin{array}{l} \pi \models \varphi_1 \text{ U}^{\leq k} \varphi_2 \Leftrightarrow \exists 0 \leq i \leq k : \pi[i] \models \varphi_2 \wedge \forall j < i : \pi[j] \models \varphi_1 \\ \pi \models \varphi_1 \text{ U } \varphi_2 \Leftrightarrow \exists i \geq 0 : \pi[i] \models \varphi_2 \wedge \forall j < i : \pi[j] \models \varphi_1 \end{array}$$

Note that  $\varphi_1 \text{ U } \varphi_2 \equiv \exists k : \varphi_1 \text{ U}^{\leq k} \varphi_2$ . We refer to properties of the form  $\varphi_1 \text{ U } \varphi_2$  as unbounded as the bound  $k$  is not fixed and not known a priori.

## 4. TWO-PHASE APPROXIMATE PROBABILISTIC MODEL CHECKING

The objective of our work is to reduce unbounded until to bounded until properties in the context of probabilistic model checking. The main problem that needs to be addressed to realize such a reduction involves identifying

- (a) a suitable bound  $k_0$  for checking the bounded until property (in each simulation) (done in *Phase I*), and
- (b) a bound on the number of simulation-paths (each of length  $k_0$ ) (done in *Phase II*),

such that a suitable statistical sampling based verification result of bounded until property approximately coincides with that of the unbounded until property within a pre-specified error limit.

#### 4.1 Rationale

The paths belonging to the semantics of  $\varphi_1 \text{ U } \varphi_2$  (Section 3.1) can be partitioned into two groups for each  $k \geq 1$ : one includes the paths that satisfy the property in  $\leq k$  steps; while the other includes the paths that satisfy the property in  $> k$  steps. I.e., the semantics of  $\varphi_1 \text{ U } \varphi_2$  can be written as

$$\begin{aligned} \pi \models \varphi_1 \text{ U } \varphi_2 & \\ \Leftrightarrow \forall k : & \left[ \begin{array}{l} \exists 0 \leq i \leq k : \pi[i] \models \varphi_2 \wedge \forall j < i : \pi[j] \models \varphi_1 \\ \vee \\ \exists i > k : \pi[i] \models \varphi_2 \wedge \forall j < i : \pi[j] \models \varphi_1 \wedge \neg \varphi_2 \end{array} \right] \\ \Leftrightarrow \pi \models \forall k : & [(\varphi_1 \text{ U}^{\leq k} \varphi_2) \vee (\varphi_1 \text{ U}^{> k} \varphi_2)] \end{aligned}$$

Note that, we have defined that  $\pi \models \varphi_1 \text{ U}^{> k} \varphi_2$  if and only if the first state  $\pi[i]$ , which satisfies  $\varphi_2$ , appears in  $\pi$  after at least  $k + 1$  steps and  $\varphi_1$  is satisfied in all states before  $\pi[i]$ .

Since,  $(\varphi_1 \text{ U}^{\leq k} \varphi_2) \wedge (\varphi_1 \text{ U}^{> k} \varphi_2) = \text{ff}$ , by law of total probability

$$\text{P}(s, \varphi_1 \text{ U } \varphi_2) = \text{P}(s, \varphi_1 \text{ U}^{\leq k} \varphi_2) + \text{P}(s, \varphi_1 \text{ U}^{> k} \varphi_2) \quad (2)$$

In other words, from the fact that probabilities  $\in [0, 1]$ ,

$$0 \leq \text{P}(s, \varphi_1 \text{ U } \varphi_2) - \text{P}(s, \varphi_1 \text{ U}^{\leq k} \varphi_2) = \text{P}(s, \varphi_1 \text{ U}^{> k} \varphi_2) \quad (3)$$

Next consider the property  $\varphi_1 \text{ U}^{> k} \varphi_2$ .

$$\begin{aligned} \pi \models \varphi_1 \text{ U}^{> k} \varphi_2 & \\ \Leftrightarrow \exists i > k : \pi[i] \models \varphi_2 \wedge \forall j < i : \pi[j] \models \varphi_1 \wedge \neg \varphi_2 & \\ \Rightarrow \forall i \leq k : \pi[i] \models \varphi_1 \wedge \neg \varphi_2 & \\ \Leftrightarrow \varphi_1 \text{ U } \varphi_2 \text{ is neither satisfied nor unsatisfied in } k \text{ steps from } \pi[0] & \\ \Leftrightarrow \pi \models \neg(\varphi_1 \text{ U}^{\leq k} \varphi_2) \wedge \neg(\neg \varphi_2 \text{ U}^{\leq k} (\neg \varphi_1 \wedge \neg \varphi_2)) & \end{aligned}$$

Let  $\psi_k = (\varphi_1 \text{ U}^{\leq k} \varphi_2) \vee (\neg \varphi_2 \text{ U}^{\leq k} (\neg \varphi_1 \wedge \neg \varphi_2))$ , i.e.,  $\psi_k$  is the property that is satisfied by a path  $\pi$  only when the satisfiability of  $\varphi_1 \text{ U } \varphi_2$  can be proved or disproved in  $k$  steps from the start state ( $\pi[0]$ ). Therefore, from the above  $(\varphi_1 \text{ U}^{> k} \varphi_2) \Rightarrow \neg \psi_k$  and

$$\text{P}(s, \varphi_1 \text{ U}^{> k} \varphi_2) \leq \text{P}(s, \neg \psi_k) = 1 - \text{P}(s, \psi_k) \quad (4)$$

From Equations 3 and 4, for any  $k \geq 1$  we obtain

$$0 \leq \text{P}(s, \varphi_1 \text{ U } \varphi_2) - \text{P}(s, \varphi_1 \text{ U}^{\leq k} \varphi_2) \leq 1 - \text{P}(s, \psi_k) \quad (5)$$

Our objective is to select a  $k_0$  such that for any given  $\epsilon_0$ .

$$\text{P}(s, \psi_{k_0}) \geq 1 - \epsilon_0 \quad (6)$$

In that case,

$$0 \leq \text{P}(s, \varphi_1 \text{ U } \varphi_2) - \text{P}(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2) \leq 1 - \text{P}(s, \psi_{k_0}) \leq \epsilon_0 \quad (7)$$

In other words, by choosing an appropriate  $k_0$ , the probability of satisfying unbounded path property  $\varphi_1 \text{ U } \varphi_2$  can be made close (within an error margin of  $\epsilon_0$ , for any arbitrarily small choice of  $\epsilon_0$ ) to the probability of satisfying the bounded path property  $\varphi_1 \text{ U}^{\leq k_0} \varphi_2$ .

#### 4.2 U2B: Two-phase Approximate Model Checking for DTMC

The discussion in previous subsection (specifically, Equations 6 and 7) motivates our two phase method. In the first phase, we determine  $k_0$  suitably and in the second phase we estimate  $P(s, \varphi_1 \cup^{\leq k_0} \varphi_2)$ . Finally, we use this estimate of  $P(s, \varphi_1 \cup^{\leq k_0} \varphi_2)$  as the estimate of  $P(s, \varphi_1 \cup \varphi_2)$ .

In *Phase I*,  $P(s, \psi_k)$  is estimated for different values of  $k$  using  $N_1$  Monte Carlo simulation paths similar to GAA (Generic Approximation Algorithm) described in [Herault et al. 2004]. This is done for all  $k \geq 1$  until for some  $k_0$ , the estimate satisfies Equation 6.

Once  $k_0$  is obtained, in *Phase II* we estimate  $P(s, \varphi_1 \cup^{\leq k_0} \varphi_2)$ . This estimate is computed as the proportion of  $N_2$  Monte Carlo simulation paths (each of length at most  $k_0$ ) that satisfy  $\varphi_1 \cup^{\leq k_0} \varphi_2$ . This also can be thought of as a simple application of the GAA algorithm for bounded until properties described in [Herault et al. 2004].

The two phases for computing  $k_0$  and then computing  $P(s, \varphi_1 \cup^{\leq k_0} \varphi_2)$  are carried out “independently”, i.e., involving separate samples (of sizes  $N_1$  and  $N_2$  respectively), which enables us to combine the errors in two phases to guarantee a certain precision. The number of Monte Carlo simulation paths used in the two phases and the value of  $k_0$  are chosen in a way that control the errors in each of the phases and combine them to guarantee the correctness of the final estimate within a pre-specified error limit (see Theorem 1 below). In short, our method,  $U2B(M, s, \varphi_1 \cup \varphi_2, \epsilon, \delta)$ , takes as input the DTMC model  $M$ , the state  $s$ , the until property under consideration, error margin  $\epsilon$  and confidence parameter  $\delta$  and returns the estimate of  $P(s, \varphi_1 \cup \varphi_2)$  within  $\epsilon$  bound with a high degree of certainty (at least  $1 - \delta$ ). The steps of our method are summarized as follows:

Main Steps.  $U2B(M, s, \varphi_1 \cup \varphi_2, \epsilon, \delta)$

- (1) *Phase I*: Obtaining  $k_0$ 
  - (a) Choose  $N_1 \geq N_1^* = 9 \log(\frac{4}{\delta}) / 2\epsilon^2$ . From  $M$ , obtain  $N_1$  Monte Carlo simulation paths of length  $k = 1$ . For  $i = 1, \dots, N_1$ , let  $X_i = 1$  if the  $i$ -th simulation satisfies  $\psi_k$ ;  $X_i = 0$  otherwise.
  - (b) Estimate  $P(s, \psi_k)$  as the proportion of the simulation paths satisfying  $\psi_k$ , i.e.

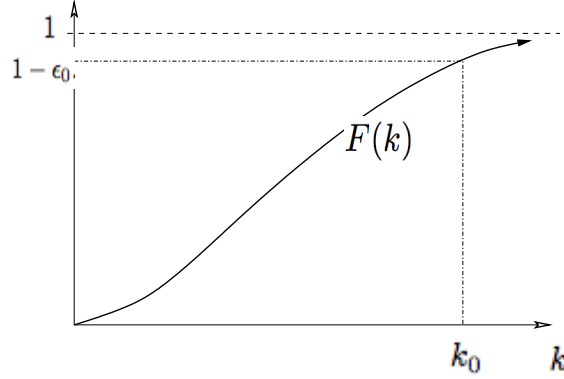
$$\hat{P}(s, \psi_k) = \frac{1}{N_1} \sum_{i=1}^{N_1} X_i. \quad (8)$$

- (c) Verify if Equation 6 is satisfied by the estimate in Equation 8 with the current value of  $k$  and  $\epsilon_0 = \frac{\epsilon}{3}$ . More precisely, if

$$\hat{P}(s, \psi_k) \geq 1 - \epsilon_0 \geq 1 - \frac{\epsilon}{3}, \quad (9)$$

then  $k_0 = k$  and proceed to *Phase II*. Otherwise, increase  $k$  by 1 and generate one more transition for each of the existing  $N_1$  simulation paths, creating  $N_1$  paths of increased (by 1) length. Define  $X_i, i = 1, \dots, N_1$  as in Step 1(a) using these extended simulation paths and repeat the Step 1(b)-(c).

- (2) *Phase II*: Estimating  $P(s, \varphi_1 \cup^{\leq k_0} \varphi_2)$

Fig. 2. Choosing  $k_0$  in *Phase I* from  $F(k)$ 

- (a) Choose  $N_2 \geq N_1^* = 36 \log(\frac{4}{\delta})/\epsilon^2$ . From  $M$ , obtain  $N_2$  Monte Carlo simulation paths (of length at most  $k_0$ ). For  $i = 1, \dots, N_2$ , let  $Y_i = 1$  if the  $i$ -th simulation path satisfies  $\varphi_1 \text{ U}^{\leq k_0} \varphi_2$ ;  $Y_i = 0$  otherwise.
- (b) Estimate  $P(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2)$  as the proportion of the simulation paths that satisfy  $\varphi_1 \text{ U}^{\leq k_0} \varphi_2$ , i.e

$$\hat{P}(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2) = \frac{1}{N_2} \sum_{i=1}^{N_2} Y_i. \quad (10)$$

Return  $\hat{P}(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2)$ , as the estimate for  $P(s, \varphi_1 \text{ U} \varphi_2)$

### 4.3 Proof of Correctness

The following theorem states the correctness of our method.

**THEOREM 1.** *Given any precision parameter  $\epsilon > 0$  and confidence parameter  $\delta > 0$ , the estimator  $\text{U2B}(M, s, \varphi_1 \text{ U} \varphi_2, \epsilon, \delta)$  with the chosen values of  $k_0, N_1^*, N_2^*$  satisfies the following:*

$$Pr(|\text{U2B}(M, s, \varphi_1 \text{ U} \varphi_2, \epsilon, \delta) - P(s, \varphi_1 \text{ U} \varphi_2)| > \epsilon) \leq \delta. \quad (11)$$

We begin by discussing auxiliary results in theoretical statistics that will be used in proving the above theorem. We discuss properties of the estimation procedure separately for the two phases.

**4.3.1 Phase I: Estimating  $k_0$ .** Let  $F(\cdot)$  be a function where  $F(k) = P(s, \psi_k)$ ,  $k \geq 1$ . Figure 2 illustrates the sample valuations of  $F(\cdot)$  for different valuations of  $k$ ; we refer to this graph as the *Decided Graph* (D-Graph);  $F(k)$  being the probability that  $\varphi_1 \text{ U} \varphi_2$  is *decided* in  $\leq k$  steps from the state  $s$ . The main challenge in *Phase I* is that the function  $F(\cdot)$  is not known in a typical situation. If this function were known, finding a  $k_0$  that satisfies Equation 6 could have been achieved by simply inverting this (non-decreasing) function.

The function  $F(\cdot)$  can be thought of as the cumulative distribution function (c.d.f) of a random variable  $K =$  the minimum number of transitions required to

verify  $\varphi_1 \cup \varphi_2$  along a randomly selected simulation path in the given model. In that case, our estimation process in *Phase I* is equivalent to estimating this c.d.f using  $N_1$  independent samples collected from the distribution of this variable  $K$ . In fact, our estimate  $\hat{P}(s, \psi_k)$  (as a function of  $k$ ) is the usual empirical c.d.f estimator  $\hat{F}_{N_1}(\cdot)$  of the true c.d.f  $F(\cdot)$ . It is well-known that  $k \geq 1$ ,  $\hat{F}_{N_1}(k)$  converges to  $F(k)$ , as  $N_1 \rightarrow \infty$  at a suitable rate, for *each*  $k$ . For the proof of Theorem 1, we need the rate *uniform* in  $k$  at which this convergence takes place. This is provided by the celebrated Dvoretzky-Kiefer-Wolfowitz (DKW) inequality (see for example, [Massart 1990]): For each  $\epsilon_1 > 0$ ,  $N_1 \geq 1$ ,  $Pr\left(\sup_{k \geq 1} |\hat{F}_{N_1}(k) - F(k)| > \epsilon_1\right) \leq 2e^{-2N_1(\epsilon_1)^2}$ .

This result, restated in terms of  $P(s, \psi_k) = F(k)$  and  $\hat{P}(s, \psi_k) = \hat{F}_{N_1}(k)$ , for each  $k$ , yields the following lemma, which will be needed for our proof of Theorem 1.

LEMMA 1. *Given any  $\epsilon_1 > 0$  and  $N_1 \geq 1$ ,*

$$Pr\left(\sup_{k \geq 1} |\hat{P}(s, \psi_k) - P(s, \psi_k)| > \epsilon_1\right) \leq 2e^{-2N_1(\epsilon_1)^2}.$$

4.3.2 *Phase II: Estimating  $P(s, \varphi_1 \cup^{\leq k_0} \varphi_2)$ .* In this phase, we estimate the probability of the *bounded* until property  $\varphi_1 \cup^{\leq k_0} \varphi_2$  in  $M$ , with  $k_0 \geq 1$  determined in *Phase I*. For any given  $k \geq 1$ , our algorithm in *Phase II* is simply the GAA algorithm (c.f. [Herault et al. 2004]) of estimating the probability of a *bounded* until property  $\varphi_1 \cup^{\leq k} \varphi_2$  in  $M$ . Hence using the same technique (i.e., using Chernoff-Hoeffding bound) we get for each  $\epsilon_2 > 0$ ,

$$Pr\left(|\hat{P}(s, \varphi_1 \cup^{\leq k} \varphi_2) - P(s, \varphi_1 \cup^{\leq k} \varphi_2)| > \epsilon_2\right) \leq 2e^{-N_2(\epsilon_2)^2/4}.$$

Now since the above inequality is true for all  $k \geq 1$ , it is true *conditional* on the simulations of *Phase I*, for  $k = k_0$ . But the two phases are carried out independently, which means the above statement must be true *unconditionally* as well, for  $k = k_0$ . Summarizing this discussion, we have

LEMMA 2. *Given any  $\epsilon_2 > 0$  and  $N_2 \geq 1$ , for  $k_0$  given by Phase I of U2B*

$$Pr\left(|\hat{P}(s, \varphi_1 \cup^{\leq k_0} \varphi_2) - P(s, \varphi_1 \cup^{\leq k_0} \varphi_2)| > \epsilon_2\right) \leq 2e^{-N_2(\epsilon_2)^2/4}.$$

Now we use the results in Lemmas 1 and 2 to complete the proof of Theorem 1.

PROOF OF THEOREM 1. Triangle inequality (after adding and subtracting suitable terms) yields the following

$$\begin{aligned} & |U2B(M, s, \varphi_1 \cup \varphi_2, \epsilon, \delta) - P(s, \varphi_1 \cup \varphi_2)| = |\hat{P}(s, \varphi_1 \cup^{\leq k_0} \varphi_2) - P(s, \varphi_1 \cup \varphi_2)| \\ & \leq |\hat{P}(s, \varphi_1 \cup^{\leq k_0} \varphi_2) - P(s, \varphi_1 \cup^{\leq k_0} \varphi_2)| \\ & \quad + |P(s, \varphi_1 \cup \varphi_2) - P(s, \varphi_1 \cup^{\leq k_0} \varphi_2)|. \end{aligned} \quad (12)$$

Recall that, from Equation 9, we have  $1 - \hat{P}(s, \psi_k) \leq \epsilon/3$ . Hence, using Equation 5 and triangle inequality, we get the following bound on the last term in Equation 12

$$\begin{aligned} |P(s, \varphi_1 \cup \varphi_2) - P(s, \varphi_1 \cup^{\leq k_0} \varphi_2)| & \leq (1 - P(s, \psi_{k_0})) \\ & \leq (1 - \hat{P}(s, \psi_{k_0})) + |\hat{P}(s, \psi_{k_0}) - P(s, \psi_{k_0})| \\ & \leq \frac{\epsilon}{3} + \sup_{k \geq 1} |\hat{P}(s, \psi_k) - P(s, \psi_k)|. \end{aligned} \quad (13)$$

Combining Equation 12 with Equation 13, we get the following bound

$$\begin{aligned} & | \text{U2B}(M, s, \varphi_1 \cup \varphi_2, \epsilon, \delta) - \text{P}(s, \varphi_1 \cup \varphi_2) | \leq \\ & | \widehat{\text{P}}(s, \varphi_1 \cup^{\leq k_0} \varphi_2) - \text{P}(s, \varphi_1 \cup^{\leq k_0} \varphi_2) | + \frac{\epsilon}{3} + \sup_{k \geq 1} | \widehat{\text{P}}(s, \psi_k) - \text{P}(s, \psi_k) | \end{aligned}$$

Hence, the left side of the above inequality is greater than  $\epsilon$  (see Equation 11 in Theorem 1) implies that at least one of the terms on the right side is greater than  $\epsilon/3$ . Therefore, we obtain the following:

$$\begin{aligned} & \Pr(| \text{U2B}(M, s, \varphi_1 \cup \varphi_2, \epsilon, \delta) - \text{P}(s, \varphi_1 \cup \varphi_2) | > \epsilon) \\ & \leq \Pr\left(| \widehat{\text{P}}(s, \varphi_1 \cup^{\leq k_0} \varphi_2) - \text{P}(s, \varphi_1 \cup^{\leq k_0} \varphi_2) | > \frac{\epsilon}{3}\right) \\ & \quad + \Pr\left(\sup_{k \geq 1} | \widehat{\text{P}}(s, \psi_k) - \text{P}(s, \psi_k) | > \frac{\epsilon}{3}\right) \leq \delta. \end{aligned} \quad (14)$$

The last inequality follows from the bounds in Lemmas 1 and 2 with  $\epsilon_1 = \epsilon/3$  and  $\epsilon_2 = \epsilon/3$ , since with  $N_i \geq N_i^*, i = 1, 2$ , we have  $2e^{-2N_1(\epsilon_1)^2} \leq \delta/2$  (i.e.,  $N_1 \geq \frac{1}{2\epsilon_1^2} \log(\frac{4}{\delta}) \geq \frac{9}{2\epsilon_2^2} \log(\frac{4}{\delta})$ ) and  $2e^{-N_2(\epsilon_2)^2/4} \leq \delta/2$  (i.e.,  $N_2 \geq \frac{4}{\epsilon_2^2} \log(\frac{4}{\delta}) \geq \frac{36}{\epsilon_2^2} \log(\frac{4}{\delta})$ ). This completes the proof of Theorem 1.  $\square$

*Remark 1.* Observe that there are three error bounds that are derived from  $\epsilon$ , each of which is assigned to  $\frac{\epsilon}{3}$ :  $\epsilon_0$  (From Equation 9),  $\epsilon_1$  (From Lemma 1) and  $\epsilon_2$  (From Lemma 2). While  $\epsilon_0$  is the measure of closeness of  $\widehat{\text{P}}(s, \psi_k)$  to 1,  $\epsilon_1$  and  $\epsilon_2$  capture the closeness of the estimated and true probabilities in each phase. In other words, smaller  $\epsilon_0$  will lead to larger value for  $k_0$  (sample path length), while smaller  $\epsilon_1$  and  $\epsilon_2$  values will result in larger values for sample size in each phase,  $N_1$  and  $N_2$ . The proof of our theorem holds as long as  $\epsilon_0 + \epsilon_1 + \epsilon_2 = \epsilon$ . The choice of these values can be fine tuned in the experiments.

## 5. EFFICIENT & PRACTICAL REALIZATION OF U2B MODEL CHECKER

As discussed in Section 4.2, our method U2B involves two phases. In this section, we present a direct (naive) implementation of *Phase I* followed by a variation that optimizes the computation of  $k_0$  in *Phase I*. Recall that, *Phase II* deals with estimating  $\text{P}(s, \varphi_1 \cup^{\leq k_0} \varphi_2)$  which can be immediately computed using the existing statistical method present in PRISM; as such, we do not provide any additional optimization for the *Phase II*.

### 5.1 Naive Implementation of Phase I

Algorithm 1 lists the procedure IDENTIFYK0 describing a naive realization of the *Phase I* in U2B. The `workingSet` contains the set of paths that are examined in the *Phase I*. It is initialized with  $N_1$  paths of length 0 (Line 3), i.e.,  $N_1$  copies of the start state  $s$  of the probabilistic system under consideration. The variable `trueCount`, initialized to 0 (Line 4), captures the number of paths in the `workingSet` that satisfy  $\psi_k$ . The paths in `workingSet` are iteratively extended and checked for satisfiability of  $\psi_k$  (Lines 5–13). The length of paths in the `workingSet` to be examined for satisfiability is incremented in Lines 6–7, and a corresponding bounded until property  $\psi_k$  is generated in Line 8. The `trueCount` is updated to capture the number of paths of length  $k$  that satisfy  $\psi_k$  (Lines 9–11). Therefore, `trueCount`/ $N_1$

**Algorithm 1** Naive Implementation of *Phase I*


---

```

1: procedure IDENTIFYK0( $N_1, \epsilon_0, \psi$ )
     $\triangleright N_1$ : Total number of samples for Phase I (see U2B Step 1(a))
     $\triangleright \epsilon_0$  closeness to 1 (see U2B Step 1(c))
     $\triangleright \psi := (\varphi_1 \ U \ \varphi_2) \vee (\neg\varphi_2 \ U \ \neg\varphi_1 \wedge \neg\varphi_2)$ 
2:    $k := 0$ ;
3:   workingSet := add  $N_1$  paths of length  $k$ ;
4:   trueCount := 0;  $\triangleright$  Initial number of paths that satisfy  $\psi_k$ 
5:   while ( $N_1(1 - \epsilon_0) > \mathbf{trueCount}$ ) do  $\triangleright$  Not enough paths sampled
6:      $k++$ ;
7:     extend existing paths in workingSet to length  $k$ ;
8:     generate  $\psi_k$  from  $\psi$ ;
9:     for each path  $\in$  workingSet that satisfies  $\psi_k$  do
10:      remove path from workingSet;
11:      trueCount++;
12:   end for
13: end while
14: return  $k$ ;  $\triangleright \mathbf{trueCount} \geq N_1(1 - \epsilon_0)$ , i.e.,  $\widehat{P}(s, \psi_k) = \frac{\mathbf{trueCount}}{N_1} \geq 1 - \epsilon_0$ 
15: end procedure

```

---

computes  $\widehat{P}(s, \psi_k)$  (Equation 8). The loop (Lines 5–13) is executed if  $N_1(1 - \epsilon_0) > \mathbf{trueCount}$ , i.e., if the proportion of paths in the sample of size  $N_1$  that satisfy  $\psi_k$  is less than  $1 - \epsilon_0$ . On termination of the iteration, i.e., when  $\mathbf{trueCount}/N_1 \geq 1 - \epsilon_0$ , the value  $k$  of the length of the paths in **workingSet** is returned. This concludes *Phase I* of the U2B and initiates the invocation of *Phase II*.

*Remark 2.* Algorithm 1 terminates and returns  $k$  if and only if  $\widehat{P}(s, \psi_k) \geq 1 - \epsilon_0$ . The above statement holds directly from the facts that  $\widehat{P}(s, \psi_k)$  is equal to the proportion of paths in sample of size  $N_1$  that satisfy  $\psi_k$ , and Algorithm 1 considers a **workingSet** of  $N_1$  paths and terminates only when  $\mathbf{trueCount} \geq N_1(1 - \epsilon_0)$ .

## 5.2 Optimization: Reducing the **workingSet** in Phase I

Algorithm 1 requires that the **workingSet** be initialized to a set (of size  $N_1$ ) of paths of length  $k$ , and maintains a subset of paths that do not satisfy  $\psi_k$  until the loop terminating condition at Line 5 is satisfied. This can be expensive in terms of space usage when  $\psi_k$  is (only) satisfied for a very large  $k$  or the majority of the paths in the **workingSet** satisfy  $\psi_k$  for large  $k$ .

An alternative realization of *Phase I* can be developed where only a small subset of the sample (of total size  $N_1$ ) is considered in the **workingSet**. This is achieved from the observation that on termination of *Phase I* the number of paths that do not satisfy  $\psi_k$  can be at most  $N_1\epsilon_0 - 1$  in the sample of size  $N_1$  (as  $\mathbf{trueCount}/N_1$  is required to be  $\geq 1 - \epsilon_0$ ). It is, therefore, sufficient to consider  $N_1\epsilon_0$  number of paths in the **workingSet** at any point of time. Whenever some paths in the **workingSet** satisfy  $\psi_k$ , they are replaced with a new set of paths such that the size of the **workingSet** is maintained at  $N_1\epsilon_0$ . In short, a sliding window **workingSet** is considered until the total number of paths that satisfy  $\psi_k$  is  $\geq N_1(1 - \epsilon_0)$  (as required in *Phase I*). This reduces the space usage as the number of paths stored the **workingSet** and checked for satisfiability of  $\psi_k$  is fixed to  $N_1\epsilon_0$ , a small proportion

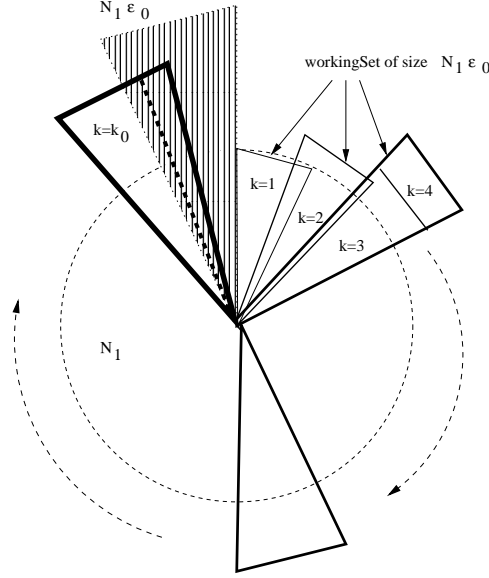


Fig. 3. Optimized realization of *Phase I* of U2B, **workingSet** of size  $N_1 \epsilon_0$ : Sliding window strategy.

of  $N_1$ .

Figure 3 illustrates this strategy. Each sector represents the working set of paths examined at each iteration. The working set size is fixed to  $N_1 \epsilon_0$ . The radius of each sector denotes the length of paths in the corresponding working set. In the figure, initially working set containing paths of length 1 is considered. The property  $\psi_k$  is satisfied in some of these paths and as a result new paths are added to the working set and all paths in the working set are extended by one step. This is repeated in iteration 2 at the end of which the working set contains paths of length 3. As per the figure, none of the paths in the working set satisfy  $\psi_k$  and as such, no new paths are added to the working set; instead all paths in the working set are extended by one step (i.e., path length  $k$  becomes 4). The above process is repeated till a sector (working set) is obtained such that (a) it overlaps with the shaded sector and in addition (b) it contains a sub-sector (denoted by dotted lines) that satisfy  $\psi_k$  also overlaps with the shaded sector. This implies that the proportion of paths that satisfy  $\psi_k$  is  $\geq 1 - \epsilon_0$ . Algorithm 2 presents the listing of this optimized implementation.

The differences between Algorithms 1 and 2 are at Lines 3 and 8 of Algorithm 2. Instead of initializing the **workingSet** with  $N_1$  paths (as in Algorithm 1), it is initialized with  $N_1 \epsilon_0$  paths. Furthermore, if some paths are removed from the **workingSet** (Line 11), the **workingSet** is replenished with new paths of appropriate length such that the total number of paths in **workingSet** remains equal to  $N_1 \epsilon_0$  (Line 8).

**PROPOSITION 1.** *Algorithm 2 terminates and returns  $k$  if and only if  $\widehat{P}(s, \psi_k) \geq 1 - \epsilon_0$ .*

**Algorithm 2** Optimized Implementation of *Phase I*


---

```

1: procedure IDENTIFYK0_OPT_1( $N_1, \epsilon_0, \psi$ )
     $\triangleright N_1$ : Total number of samples for Phase I (see U2B Step 1(a))
     $\triangleright \epsilon_0$ : Error bound for Phase I (see U2B Step 1(c))
     $\triangleright \psi := (\varphi_1 \cup \varphi_2) \vee (\neg\varphi_2 \cup \neg\varphi_1 \wedge \neg\varphi_2)$ 
     $\triangleright$  Initial length of paths
2:    $k := 0$ ;
3:   workingSet := add  $N_1\epsilon_0$  paths of length  $k$ ;
4:   trueCount := 0;  $\triangleright$  Initial number of paths that satisfy  $\psi_k$ 
5:   while ( $N_1(1 - \epsilon_0) > \mathbf{trueCount}$ ) do  $\triangleright$  Not enough paths sampled
6:      $k++$ ;
7:     extend existing paths in workingSet to length  $k$ ;
8:     add ( $N_1\epsilon_0 - |\mathbf{workingSet}|$ ) paths of length  $k$  to workingSet;
9:     generate  $\psi_k$  from  $\psi$ ;
10:    for each path  $\in$  workingSet that satisfy  $\psi_k$  do
11:      remove path from workingSet;
12:      trueCount++;
13:    end for
14:  end while
15:  return  $k$ ;  $\triangleright \mathbf{trueCount} \geq N_1(1 - \epsilon_0)$ , i.e.,  $\widehat{P}(s, \psi_k) = \frac{\mathbf{trueCount}}{N_1} \geq 1 - \epsilon_0$ 
16: end procedure

```

---

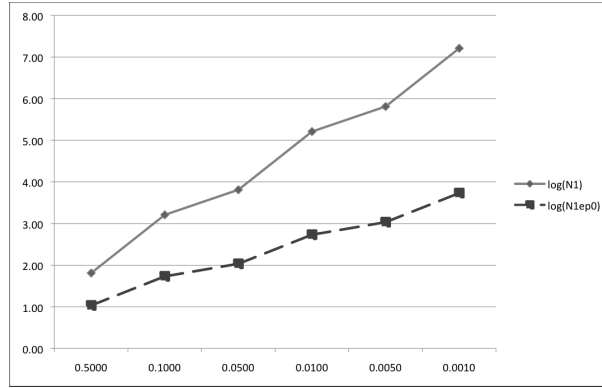


Fig. 4. Logarithms of  $N_1$  and  $N_1\epsilon_0$  (y-axis) against  $\epsilon$  (x-axis).

Algorithm 2 terminates if  $N_1(1 - \epsilon_0) \leq \mathbf{trueCount}$ . Therefore, it must examine at least  $N_1(1 - \epsilon_0) + 1$  paths before terminating and returning a  $k$ . That is, at most  $N_1\epsilon_0 - 1$  paths are not considered by Algorithm 2 for returning a  $k$  and inferring that  $\widehat{P}(s, \psi_k) \geq 1 - \epsilon_0$ . The above proposition holds as  $\widehat{P}(s, \psi_k)$  is the proportion of paths (i.e.,  $\mathbf{trueCount}$ ) in a sample of size  $N_1$  that satisfy  $\psi_k$  and the terminating condition of the Algorithm 2 ensures that  $\mathbf{trueCount}/N_1 \geq 1 - \epsilon_0$ .

5.2.1 *Discussion: Algorithm 1 Vs. Algorithm 2.* Recall that Algorithm 1 uses  $N_1 = \frac{1}{2\epsilon_1^2} \log(\frac{4}{\delta})$  samples while Algorithm 2 requires  $N_1\epsilon_0$  samples. We have considered  $\epsilon_0 = \epsilon_1 = \frac{\epsilon}{3}$  (see Remark 1 in Section 4.3); however any other choice of  $\epsilon_0$  and  $\epsilon_1$  as a function of  $\epsilon$  would suffice for this discussion. Figure 4 compares the

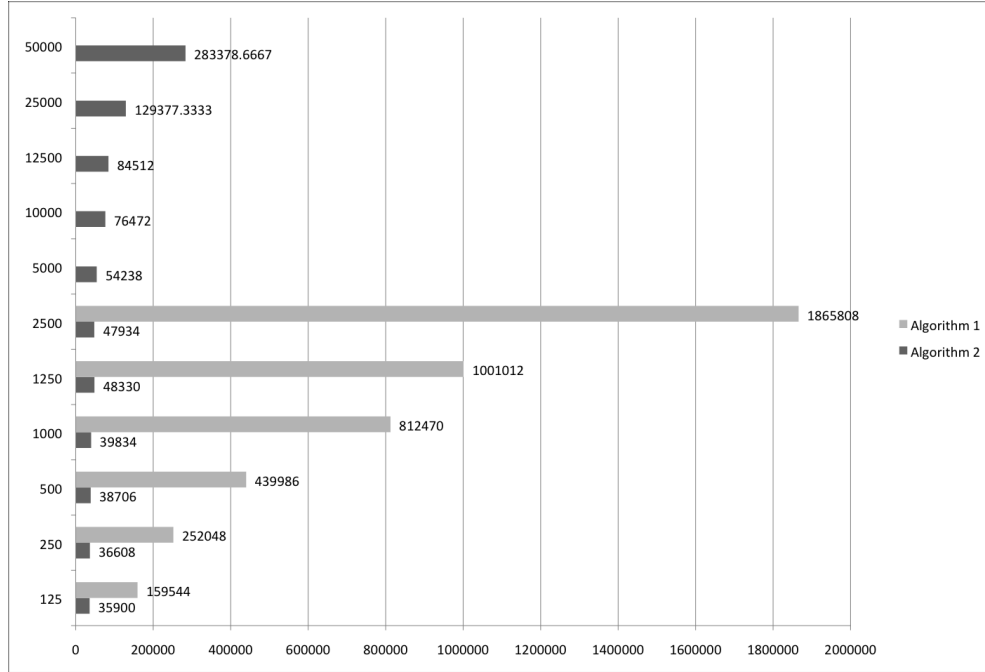


Fig. 5. Memory usage by Algorithm 1 Vs. Algorithm 2 for variations of a simple queue model (x-axis: Memory usage (MB), y-axis: Number of variables).

increases (in log scale) in the value of  $N_1$  and  $N_1\epsilon_0$  with the decrease in  $\epsilon$  for a specific confidence parameter  $\delta = 0.001$ . Observe that  $N_1$  increases *faster* than  $N_1\epsilon_0$  with the decrease in error bound  $\epsilon$  of the U2B method (for instance, with  $\epsilon = 0.001$ ,  $N_1 = 1.6 \times 10^7$  while  $N_1\epsilon_0 = 5403$ ). In other words, as precision of U2B method is increased, Algorithm 2 uses considerably smaller working set (and in turn, lesser space) compared to that used by Algorithm 1; larger the precision greater is the benefit in terms of space usage in using Algorithm 2 over Algorithm 1.

A similar benefit in space usage is realized when the number of variables describing the model (i.e., each model state) under consideration is increased. We have experimented with different variations of simple queue model (see [Two Phase Probabilistic Model Checking 2010]) where each variation contains different number of variables. Figure 5 shows that with the increase in the number of variables in the model, the increase in the space usage (in MB) by Algorithm 1 is larger than that in Algorithm 2. In fact, when the number of variables is  $\geq 2500$ , Algorithm 1 reports out of memory problem and terminates without computing any result.

## 6. DEALING WITH NON-TERMINATION IN U2B

Recall that  $F(k)$  is the probability that  $\varphi_1 \mathbf{U} \varphi_2$  is decided in  $k$  steps from a state  $s$  under consideration. In Section 4.3.1, we introduced the notion of Decided Graph (D-Graph; Figure 2) presenting the valuations of  $F(\cdot)$  for different valuations of  $k$ . In this section, we discuss the condition over the valuation of  $F(\cdot)$  (and the

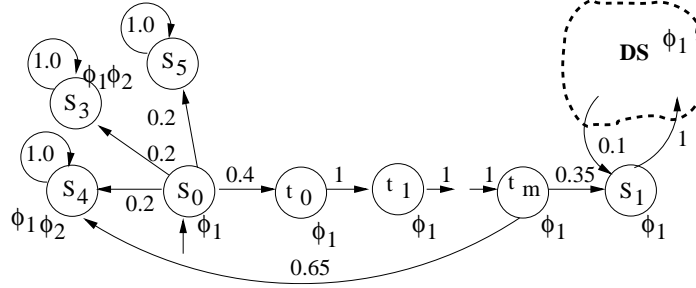


Fig. 6. A modification of illustrative example (Figure 1).

corresponding shape of D-Graph) under which the proposed U2B method fails to terminate and propose a heuristic based alternate method to ensure termination.

Our argument for the applicability of the U2B method requires that as  $k \rightarrow \infty$ , limit of  $F(k) = P(s, \psi_k) \geq 1 - \epsilon_0$  (see Equation 6). That is,

$$\lim_{k \rightarrow \infty} P(s, \psi_k) \geq 1 - \epsilon_0 \quad (15)$$

When the limit is 1, Equation 15 is satisfied for any  $\epsilon_0$  and hence our method is valid for any  $\epsilon_0 \geq 0$ . (See Figure 2 which illustrates the case when the limit is 1).

When the limit is not equal to 1 (Figure 7 illustrates a D-Graph where the limit of  $F(\cdot)$  is not equal to 1), our method works i.e., *Phase I* computation terminates for any  $\epsilon_0$  that satisfies the inequality in Equation 15. Note that, the requirement of satisfying this inequality does not restrict the applicability of our technique to any specific class of properties or models; our technique can be applied for any until property and for any model as long as the above requirement is satisfied. It is worth mentioning that PRISM's statistical method is equivalent to choosing  $\epsilon_0 = 0$  in our method and hence will fail to provide result whenever this limit is not equal to 1. In other words, whenever PRISM's statistical verification method is successful in computing a result, our method also terminates with a result, and furthermore, our method is able to estimate probabilities for many cases where PRISM's statistical method fails (see discussion in Section 9).

The implication of the limit not being equal to 1 is that  $P(s, \neg(ttU(\varphi_2 \vee \neg\varphi_1))) > 0$ , which happens if and only if there exists an infinite path in the model with positive probability where every state satisfies  $(\varphi_1 \wedge \neg\varphi_2)$ . In this case, any statistical verification method that does not analyze model transition structure may not be able to provide result. However, this feature of not analyzing the model allows application of statistical verification methods (including ours) for the purpose of estimating the probability of properties in black-box systems, where information regarding complete transition structure is not available.

Consider the DTMC model in Figure 6. The start state of the model is  $s_0$  and each state is annotated with the property  $(\varphi_1, \varphi_2)$  that holds at that state. As in the illustrative example introduced in Section 1.1, the dotted segment, denoted by DS, represents some complicated transition structure on  $n$  different states (exact description of the model available at [Two Phase Probabilistic Model Checking

2010]). All states in DS satisfy  $\varphi_1$ . The objective is to compute  $P(s_0, \varphi_1 \cup \varphi_2)$ . There are three paths  $s_0, s_3, s_3, \dots$ ,  $s_0, s_4, s_4, \dots$  and  $s_0, t_0, \dots, t_m, s_4, \dots$  that satisfy  $\varphi_1 \cup \varphi_2$ ; therefore,  $P(s, \varphi_1 \cup \varphi_2) = 0.2 + 0.2 + 0.4 \times 0.65 = 0.66$ . Observe that there exists an infinite path in the model with positive probability where every state satisfies  $(\varphi_1 \wedge \neg\varphi_2)$ . In fact,  $\lim_{k \rightarrow \infty} P(s_0, \psi_k) = 0.86$ . As PRISM's statistical method requires the above limit to be 1, it fails to compute the estimate of  $P(s_0, \varphi_1 \cup \varphi_2)$ . Similarly, our proposed method U2B will fail to terminate when  $\epsilon_0 < 0.14$ . More precisely, the *Phase I* of U2B method will fail to terminate. In the following, we present a strategy and the corresponding heuristic to address this problem of non-termination.

### 6.1 Rationale

We proceed by discussing the theoretical basis of our alternate strategy and provide a (heuristic) method to realize it in practice.

From the semantics of PCTL (Section 3.2), the probability of satisfying  $\varphi_1 \cup \varphi_2$  can be expressed in terms of satisfying  $\varphi_1 \cup^{\leq k} \varphi_2$  as follows:

$$\lim_{k \rightarrow \infty} P(s, \varphi_1 \cup^{\leq k} \varphi_2) = P(s, \varphi_1 \cup \varphi_2), \text{ i.e., } P(s, \varphi_1 \cup^{\leq \infty} \varphi_2) = P(s, \varphi_1 \cup \varphi_2) \quad (16)$$

Next, suppose there exists an method by which a  $k_*$  can be computed such that

$$\forall \Delta_k \geq 1 : \hat{P}(s, \varphi_1 \cup^{\leq k_*} \varphi_2) = \hat{P}(s, \varphi_1 \cup^{\leq k_* + \Delta_k} \varphi_2) \quad (17)$$

We refer to any  $k_*$  that satisfies the above equation as a *unbounded-plateau-detector* (the valuation of estimate  $\hat{P}(s, \varphi_1 \cup^{\leq k} \varphi_2)$  remains unchanged for all  $k \geq k_*$ ). Note that, there is at most one unbounded-plateau and infinite number of unbounded-plateau-detectors. The above equation implies that  $\hat{P}(s, \varphi_1 \cup^{\leq k_*} \varphi_2) = \hat{P}(s, \varphi_1 \cup^{\leq \infty} \varphi_2)$ . Proceeding further, we prove the following theorem which forms the basis of our method.

**THEOREM 2.** *Given any precision parameter  $\epsilon_1 > 0$  and confidence parameter  $\delta_1 > 0$ ,*

$$Pr \left( \left| \hat{P}(s, \varphi_1 \cup^{\leq k_*} \varphi_2) - P(s, \varphi_1 \cup \varphi_2) \right| > \epsilon_1 \right) \leq \delta_1$$

where  $k_*$  is a plateau-detector and  $\delta_1 \geq 2e^{-2N_1(\epsilon_1)^2}$ .

**PROOF.**

$$\begin{aligned} \left| \hat{P}(s, \varphi_1 \cup^{\leq k_*} \varphi_2) - P(s, \varphi_1 \cup \varphi_2) \right| &\leq \left| \hat{P}(s, \varphi_1 \cup^{\leq k_*} \varphi_2) - P(s, \varphi_1 \cup^{\leq \infty} \varphi_2) \right| \\ &\quad + \left| P(s, \varphi_1 \cup^{\leq \infty} \varphi_2) - P(s, \varphi_1 \cup \varphi_2) \right| \\ &\leq \left| \hat{P}(s, \varphi_1 \cup^{\leq k_*} \varphi_2) - P(s, \varphi_1 \cup^{\leq \infty} \varphi_2) \right| \\ &\quad \text{using Equation 16} \end{aligned}$$

Therefore,

$$\begin{aligned}
|\widehat{P}(s, \varphi_1 \text{ U}^{\leq k_*} \varphi_2) - P(s, \varphi_1 \text{ U} \varphi_2)| &\leq |\widehat{P}(s, \varphi_1 \text{ U}^{\leq k_*} \varphi_2) - \widehat{P}(s, \varphi_1 \text{ U}^{\leq \infty} \varphi_2)| \\
&\quad + |\widehat{P}(s, \varphi_1 \text{ U}^{\leq \infty} \varphi_2) - P(s, \varphi_1 \text{ U}^{\leq \infty} \varphi_2)| \\
&\leq |\widehat{P}(s, \varphi_1 \text{ U}^{\leq \infty} \varphi_2) - P(s, \varphi_1 \text{ U}^{\leq \infty} \varphi_2)| \\
&\quad \text{using Equation 17} \\
&\quad (18)
\end{aligned}$$

From Lemma 1, the following holds for any  $k$ ,  $N_1 \geq 1$  and  $\epsilon_1 > 0$

$$Pr\left(\sup_{k \geq 1} |\widehat{P}(s, \varphi_1 \text{ U}^{\leq k} \varphi_2) - P(s, \varphi_1 \text{ U}^{\leq k} \varphi_2)| > \epsilon_1\right) \leq 2e^{-2N_1(\epsilon_1)^2}$$

where  $\widehat{P}(s, \varphi_1 \text{ U}^{\leq k} \varphi_2)$  is the proportion of paths (starting from  $s$ ) in  $N_1$  samples that satisfy  $\varphi_1 \text{ U}^{\leq k} \varphi_2$ . Therefore, from Equation 18

$$\begin{aligned}
Pr\left(|\widehat{P}(s, \varphi_1 \text{ U}^{\leq \infty} \varphi_2) - P(s, \varphi_1 \text{ U}^{\leq \infty} \varphi_2)| > \epsilon_1\right) &\leq \delta_1, \text{ i.e.,} \\
Pr\left(|\widehat{P}(s, \varphi_1 \text{ U}^{\leq k_*} \varphi_2) - P(s, \varphi_1 \text{ U} \varphi_2)| > \epsilon_1\right) &\leq \delta_1
\end{aligned}$$

where  $\delta_1 \geq 2e^{-2N_1(\epsilon_1)^2}$ .

It follows that if  $N_1 \geq \frac{1}{2\epsilon_1^2} \log(\frac{\delta_1}{2})$  samples are considered to compute a  $k_*$  such that Equation 17 is satisfied, then the proportion of paths that satisfy  $\varphi_1 \text{ U}^{\leq k_*} \varphi_2$  in  $N_1$  samples (i.e.,  $\widehat{P}(s, \varphi_1 \text{ U}^{\leq k_*} \varphi_2)$ ) estimates  $P(s, \varphi_1 \text{ U} \varphi_2)$  with precision parameter  $\epsilon_1$  and confidence parameter  $\delta_1$ .  $\square$

The above theorem provides a sound roadmap to estimate  $P(s, \varphi_1 \text{ U} \varphi_2)$  and it does not require that  $\lim_{k \rightarrow \infty} P(s, \psi_k) \geq 1 - \epsilon_0$  (as is necessary for the termination of *Phase I* of our U2B method). However, this roadmap suffers from the drawback that it assumes the existence of a method for computing  $k_*$ , an unbounded-plateau-detector. No such method for computing  $k_*$  can be realized in practice. As such, we propose a heuristic for estimating  $k_*$  which ensures termination of U2B at the cost of precision (i.e., resulting  $\widehat{P}(s, \varphi_1 \text{ U} \varphi_2)$  may not be an estimate within pre-specified error bound and confidence parameter) when  $\lim_{k \rightarrow \infty} P(s, \psi_k) \not\geq 1 - \epsilon_0$ .

## 6.2 U2B.P: Heuristic for Computing Plateau-Detector using the D-Graph

If in a D-Graph the valuation of  $F(k) = P(s, \psi_k)$  for a range of values of  $k$  (see Figure 7) remains unaltered, then a plateau in the corresponding valuation of  $P(s, \varphi_1 \text{ U}^{\leq k} \varphi_2)$  occurs for the same  $k$  values. Recall that, *Phase I* of U2B does not terminate when there exists an unbounded plateau in  $F(\cdot)$  such that the value of the plateau is less than  $1 - \epsilon_0$ . For instance  $F(\cdot)$  for the model in Figure 6 has an unbounded plateau with valuation 0.86.

The basis of the heuristic, therefore, is as follows. An estimate  $\hat{k}_*$  of  $k_*$  is said to be “good” if a “long” plateau in  $\hat{F}_{N_1}(k) = \widehat{P}(s, \psi_k)$  (estimate of  $F(k)$  from  $N_1$  samples) is detected starting from  $k = \hat{k}_*$ . In other words, given  $\Gamma$ , a pre-specified length of a plateau,  $\hat{k}_*$  is an estimate of  $k_*$  if  $\forall k : \hat{k}_* \leq k \leq \hat{k}_* + \Gamma$ , the valuation of  $\hat{F}_{N_1}(k)$  remains unaltered.

$$\forall k : k_* \leq k < (k_* + \Gamma) : \hat{F}_{N_1}(k) = \hat{F}_{N_1}(k+1) \Rightarrow \widehat{P}(s, \varphi_1 \text{ U}^{\leq k} \varphi_2) = \widehat{P}(s, \varphi_1 \text{ U}^{\leq k+1} \varphi_2) \quad (19)$$

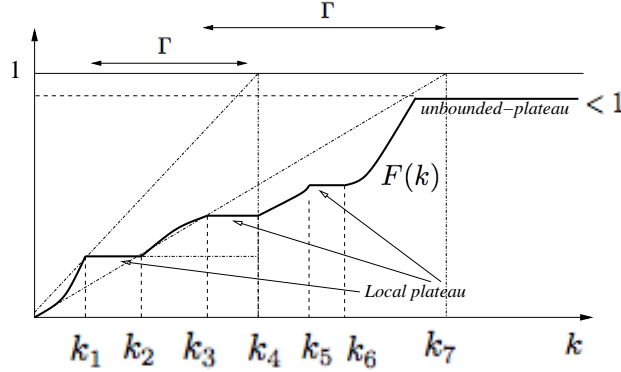


Fig. 7. Plateaus in the D-Graph.

The precision of this method of estimation depends on the length,  $\Gamma$ , of the detected plateau; a longer plateau (larger  $\Gamma$ ) is likely to provide a better estimate  $\hat{k}_*$ . For example, in Figure 7, if  $\Gamma$  is greater than  $k_2 - k_1$ ,  $k_3 - k_4$ , and  $k_6 - k_5$ , then  $\hat{k}_*$  will be indeed an unbounded-plateau-detector; otherwise, one of the local plateaus may result in an incorrect  $\hat{k}_*$  computation. In the following, we provide a heuristic to obtain a suitable valuation for  $\Gamma$  based on the valuations of  $k$ .

Let  $k_i$  be the current valuation of the simulation length and  $k_j$  be the smallest value of  $k$  for which  $\hat{F}_{N_1}(k_i) = \hat{F}_{N_1}(k_j)$ . The objective is decide whether the difference  $k_i - k_j$  is a long enough ( $\geq \Gamma$ ) to infer that a *global* plateau is detected. Assuming that  $F(\cdot)$  increases at a constant rate given by  $F(k_j)/k_j$ , we estimate the  $k_x$  for which  $F(k_x) = 1$

$$\frac{1 - F(k_j)}{k_x - k_j} = \frac{1}{k_x} \quad \Leftrightarrow \quad k_x - k_j = k_j \left( \frac{1}{F(k_j)} - 1 \right)$$

We say that  $\Gamma = (k_j + \text{AF})\gamma^{\text{simpoly}}$  where

- AF is the additive factor (= 10).
- $\gamma = \text{MINMAX} \left( \text{LB}, \frac{1}{\hat{F}_{N_1}(k_j)} - 1, \text{UB} \right)$ ,
- simpoly** is the user-specified parameter (default value is 1),
- LB = 2 and UB = 5 are the pre-specified lower- and upper-bounds of  $\gamma$ , and
- $\hat{F}_{N_1}(\cdot)$  is the estimator of  $F(\cdot)$  (see Section 4.3.1)

The value  $k_j$  is said to be an estimate  $\hat{k}_*$  when  $k_i - k_j > \Gamma$  and  $\hat{F}_{N_1}(k_i) = \hat{F}_{N_1}(k_j)$ . Note that we have artificially set three parameters AF, UB and LB. AF is used to ensure long enough plateau is considered even when  $k_j$  is small (e.g.,  $k_j = 1$ ). LB (and UB) ensures that the required plateau length grows reasonably by exponents of 2 (or 5) if the slope  $\hat{F}_{N_1}(k_j)/k_j$  is too large (or too small). We have allowed users the control the length of the plateau using the exponent **simpoly**.

Consider an example illustration in Figure 7. At  $k_1$ , the value of  $\Gamma$  is  $k_4 - k_1$  (assume that there is no pre-set LB, UB, AF), i.e., if the value of  $\hat{F}_{N_1}(\cdot)$  remains unaltered between  $k_1$  and  $k_4$  (inclusive), we say that a plateau is identified starting from

**Algorithm 3** Implementation of U2B\_P

---

```

1: procedure PLATEAUKSTAR( $N_1, \epsilon_0, \psi, \text{simpoly}$ )
     $\triangleright N_1$ : Total number of samples for phase 1 (see U2B Step 1(a))
     $\triangleright \epsilon_0$ : Error bound for phase 1 (see U2B Step 1(c))
     $\triangleright \psi := (\varphi_1 \cup \varphi_2) \vee (\neg\varphi_2 \cup \neg\varphi_1 \wedge \neg\varphi_2)$ 
     $\triangleright \text{simpoly}$ : Exponent for plateau length calculation
2:    $k := 0$ ;  $\triangleright$  Initial length of paths
3:    $\text{lastk} := 0$ ;  $\triangleright$  last value of  $k$  for which  $\psi_k$  was satisfied
4:    $\text{workingSet} := \text{add } N_1 \text{ paths of length } k$ ;
5:    $\text{trueCount} := 0$ ;  $\triangleright$  Initial number of paths that satisfy  $\psi_k$ 
6:    $\text{satCount} := 0$ ;  $\triangleright$  Initial number of paths that satisfy  $\varphi_1 \cup^{\leq k} \varphi_2$ 
7:   while ( $N_1(1 - \epsilon_0) > \text{trueCount}$ ) do
8:      $k := k++$ ;
9:     extend existing paths in  $\text{workingSet}$  to length  $k$ ;
10:    generate  $\psi_k$  from  $\psi$ ;  $\text{lastkChg} := 0$ ;
11:    for each path  $\in \text{workingSet}$  that satisfies  $\psi_k$  do
12:      remove path from  $\text{workingSet}$ ;
13:       $\text{trueCount}++$ ;
14:      if (the path satisfies  $\varphi_1 \cup^{\leq k} \varphi_2$ ) then
15:         $\text{satCount}++$ ;  $\triangleright$  update  $\text{satCount}$ 
16:      end if
17:       $\text{lastk} := k$ ;  $\triangleright$  record value of  $k$ 
18:    end for
19:    if ( $\text{trueCount} > 0$ ) then  $\triangleright$  Calculation of plateau length
20:       $\Gamma := (\text{lastk} + 10) \times [\text{MINMAX}(2, N_1/\text{trueCount} - 1, 5)]^{\text{simpoly}}$ ;
21:    end if
22:    if ( $\Gamma > 0$ ) && ( $k > \text{lastk} + \Gamma$ ) then  $\triangleright$  Condition for plateau: heuristic
23:      return  $k$  and  $\text{satCount}/N_1$  as  $\widehat{P}(s, \varphi_1 \cup \varphi_2)$ 
24:    end if
25:  end while
26:  return  $k$ ;  $\triangleright \text{trueCount} \geq N_1(1 - \epsilon_0)$ , i.e.,  $\widehat{P}(s, \psi_k) = \frac{\text{trueCount}}{N_1} \geq 1 - \epsilon_0$ 
     $\triangleright$  Invoke Phase II of U2B using  $k$ 
27: end procedure

```

---

$k_1$  and the corresponding value of  $\widehat{P}(s, \varphi_1 \cup^{\leq k_1} \varphi_2)$  is identified as the  $\widehat{P}(s, \varphi_1 \cup \varphi_2)$  (see Equation 19). Similarly, at  $k_3$ ,  $\Gamma$  computed using the above formula is  $k_7 - k_3$ .

We have developed a method, referred to as the U2B\_P, based on the above heuristic to compute  $k_j = \hat{k}_*$ . In U2B\_P, if a plateau in  $\widehat{F}_{N_1}(\cdot)$  is detected starting from  $k_j$  and if  $\widehat{F}_{N_1}(k_j) \not\geq 1 - \epsilon_0$ , then  $\widehat{P}(s, \varphi_1 \cup^{\leq k_j} \varphi_2)$  is returned as the estimate of  $P(s, \varphi_1 \cup \varphi_2)$  (see Theorem 2).  $\widehat{P}(s, \varphi_1 \cup^{\leq k_j} \varphi_2)$  is the proportion of paths in  $N_1$  samples that satisfy  $\varphi_1 \cup^{\leq k_j} \varphi_2$ . On the other hand, if  $\widehat{F}_{N_1}(k_j) > 1 - \epsilon_0$ , then U2B\_P is said to have successfully estimated  $k_0 (= k_j)$  and therefore, U2B\_P reduces to U2B and invokes *Phase II* computation (as described in U2B method; see Section 4.3.2).

*Remark 3.* It should be noted that as U2B\_P is based on a heuristic and cannot provide any correctness guarantees. We give users the option to deploy either the U2B (with Algorithm 2 implementation for *Phase I*) or U2B\_P. In the event that U2B fails to return an estimated probability within the amount of time the user is

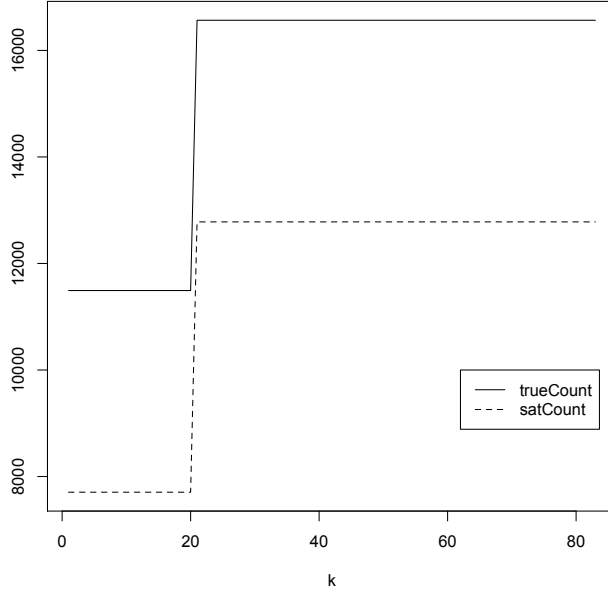


Fig. 8. Execution of Algorithm 3 for DTMC model in Figure 6: `trueCount` and `satCount` vs.  $k$ .

willing to wait, the user can terminate the process and deploy `U2B_P`.

The `U2B_P` method is realized as follows in Algorithm 3. In addition to keeping track of the proportion of paths that satisfy  $\psi_k$  (`trueCount`, i.e.,  $\hat{F}_{N_1}(k)$ ), the algorithm also keeps information regarding the number of paths that satisfy  $\varphi_1 \cup^{\leq k} \varphi_2$  (`satCount` in Lines 6, 15). The value of  $k$  for which there is some change in the valuation of `trueCount` is recorded in `lastk` (Line 17). The  $\Gamma$  is computed at Line 20 following the above description. If `trueCount` does not change for a large range of values of  $k$  (i.e., difference between current value of  $k$  and `lastk` is  $> \Gamma$ ), then for the same range the value of `satCount` will also not change (Equation 19). If a plateau is detected in the value of `trueCount` (i.e.,  $F_{N_1}(\cdot)$ ), then `satCount`/ $N_1$  is returned as an estimate for  $P(s, \varphi_1 \cup \varphi_2)$  along with a warning to the user that a heuristic has been used to satisfy Equation 17 (Line 23). The computation can be re-done with larger values of  $\Gamma$ , if the user so chooses, by increasing `simpoly`. If the termination condition of the while-loop at Line 8 is satisfied, then it implies that  $\hat{F}_{N_1}(k)$  is  $\epsilon_0$  close to 1. In that case, the algorithm returns  $k$  as  $k_0$  (Line 26) and *Phase II* of `U2B` is invoked.

6.2.1 *Discussion: U2B vs U2B\_P*. Consider the example model in Figure 6. The `U2B` method will fail to terminate and estimate  $P(s, \varphi_1 \cup \varphi_2)$  for this model due to

the following reason. The limit<sup>2</sup>  $\lim_{k \rightarrow \infty} P(s, \psi_k) = 0.86 < 1 - \epsilon_0$  where  $\epsilon_0 = 0.0025$ . That is,  $P(s, \psi_k)$  or  $F(k)$  has an unbounded plateau for some value of  $k$  and the valuation of  $P(s, \psi_k)$  at the plateau is  $< 1 - \epsilon_0$ .

The U2B\_P method as implemented in Algorithm 3, on the other hand, terminates and successfully estimates  $P(s_0, \varphi_1 \cup \varphi_2)$ . In the example, we have considered  $m = 20$  (number of  $t_i$ -states in the model) and used `simpoly` = 1. Algorithm 3 identifies two plateaus—one where the simulation paths end up in the states  $t_0, \dots, t_m$  (local plateau) and the other where the simulation paths end up in the DS (global plateau). Figure 8 shows these plateaus in terms of the changes in the valuation of the `trueCount` (number of paths with decided result), `satCount` (number of paths that satisfy  $\varphi_1 \cup \varphi_2$ ).

The execution starts with  $k = 0$  and `workingSet` =  $N_1 = 19,171$  ( $N_1 = 19,171$  for  $\epsilon_0 = 0.0025$  and  $\delta = 0.01$ ). The value of `trueCount` remains at 11,492 till the value of  $k$  becomes 20, i.e.,  $\forall k \in [1, 20] : \hat{F}_{N_1}(k) = \hat{P}(s, \psi_k) = 0.59942$ . This is because the length of paths considered is  $\leq 20$  and as such the transitions  $t_m$  to  $s_4$  and  $t_m$  to  $s_1$  are yet to be explored. However, the length (20) of this plateau in  $\hat{F}_{N_1}(\cdot)$  is not considered to be sufficient as per the heuristic because  $\forall \text{lastk} \in [1, 19] : \text{lastk} + \Gamma \geq 23$  (see the computation of  $\Gamma$  described in Section 6.2). That is, if  $\hat{F}_{N_1}(\cdot)$  remained unaltered for paths of length 23, then U2B\_P would infer that a reasonably large size plateau have been detected and, therefore, would terminate. In the current example, that does not happen as  $\hat{F}_{N_1}(20) < \hat{F}_{N_1}(21)$ <sup>3</sup>.

The value of `trueCount` remains at 16,566 for  $k \geq 21$ , i.e., the last time `trueCount` value changes is when  $k = 21$ . In this case, when `lastk` = 21,  $\Gamma = 62$ . In other words, if the `trueCount` value does not change for  $k$  between 21 and  $21 + 62 = 83$ , the U2B\_P terminates. Observe that, changes in the `satCount` follows similar pattern as that in the `trueCount`; plateau in the valuation of `trueCount` (i.e.,  $\hat{F}_{N_1}(k)$ ) implies plateau in the valuation of `satCount` (i.e.,  $\hat{P}(s_0, \varphi_1 \cup^{\leq k} \varphi_2)$ ). As such, on termination U2B\_P returns  $\hat{P}(s_0, \varphi_1 \cup^{\leq \text{lastk}} \varphi_2) = \text{satCount}/N_1 = 0.66665$  as the estimate for  $P(s_0, \varphi_1 \cup \varphi_2)$ ; recall that  $P(s, \varphi_1 \cup \varphi_2) = 0.66$ .

## 7. APPLICATION TO CONTINUOUS TIME MARKOV CHAIN MODELS

So far, we have shown the application of U2B in the context of probabilistic model checking of unbounded until properties against DTMC models. In this section, we discuss its applicability in verifying similar properties for CTMC models. Unlike a DTMC model where transition from one state to another captures the probability of a *discrete* time step, a Continuous Time Markov Chain (CTMC) model describes the probability with which a system evolves from one state (configuration) to another within  $t$  time units. Formally, a CTMC is defined as:

*Definition 2.* A Continuous Time Markov Chain CTMC =  $(S, s_I, R, L)$ , where  $S$  is a finite set of states,  $s_I \in S$  is the initial or start state,  $R : S \times S \rightarrow \mathbb{R}_{\geq 0}$  is the

<sup>2</sup>Recall that  $\psi_k$  denotes the property stating that  $\varphi_1 \cup \varphi_2$  is decided in  $k$  steps. The  $\lim_{k \rightarrow \infty} P(s, \psi_k) \geq 1 - \epsilon_0$  (see Equation 15) is required for the termination of Algorithm 2.

<sup>3</sup>If  $m$  is increased to  $> 23$  and `simpoly` is set to 1, then the heuristic based method U2B\_P would terminate, incorrectly infer a plateau, and estimate  $P(s_0, \varphi_1 \cup \varphi_2)$  incorrectly. That is why U2B\_P produces a warning message on termination and recommends that the user re-run U2B\_P using different values of `simpoly`.

rate matrix, and  $L : S \rightarrow \mathcal{P}(AP)$  is the labeling function which labels each state with a set of atomic propositions  $\subseteq AP$  that holds in that state.

The rate matrix  $R$  in the above definition denotes the rate at which the system evolves from one state to another, while for any  $s \in S$ ,  $E(s) = \sum_{s' \in S} R(s, s')$  denotes the rate at which the system moves out of the state  $s$ . This is used as the parameter to the exponential distribution capturing the probability  $1 - e^{-E(s) \cdot t}$  of taking an outgoing transition from  $s$  within  $t$  time units. If  $R(s, s') > 0$  for multiple  $s'$ , then there exists a *race* between the moves from  $s$  to its multiple next states. The probability  $T(s, s')$  with which a state  $s$  in CTMC moves to a state  $s'$  in single step is equal to the probability that the delay of moving from  $s$  to  $s'$  is less than those for any other moves from  $s$ ;  $T(s, s') = R(s, s')/E(s)$ . Based on this observation, a CTMC *contains* an embedded DTMC which captures the probability of each transition independent of the time at which it occurs.

*Definition 3.* [Baier et al. 2003] Given a CTMC  $= (S, s_I, R, L)$ , the corresponding embedded DTMC,  $\text{emb}(\text{DTMC}) = (S, s_I, T, L)$ , where  $T : S \times S \rightarrow [0, 1]$  such that

$$T(s, s') = \begin{cases} R(s, s')/E(s) & \text{if } E(s) > 0 \\ 1 & \text{if } E(s) = 0 \text{ and } s' = s \\ 0 & \text{if } E(s) = 0 \text{ and } s' \neq s \end{cases}$$

*Paths and Probability Measures.* A path  $\sigma$  in CTMC is a finite or infinite sequence of tuples  $(s_0 t_0, s_1 t_1, s_2 t_2, \dots)$  where  $t_i$  denotes the amount of time the system remains in state  $s_i$ . As in DTMC, the set of all infinite paths starting from state  $s$  is denoted by  $\text{Path}_{\text{CTMC}}(s)$ . Let  $\sigma[i]$  denote the  $i$ -th state in the path,  $\sigma^i$  denote the suffix of  $\sigma$  starting from  $\sigma[i]$  and  $\sigma@t$  denote the state at time  $t$ . Then,  $\sigma@t = \sigma[i]$  where  $i = \text{MIN}(k \mid t \leq \sum_{j=0}^k t_j)$ . We will use  $\sigma_{\text{emb}(\text{DTMC})}$  to denote  $(s_0, s_1, s_2, \dots)$  which represents a path corresponding to  $\sigma$  in the  $\text{emb}(\text{DTMC})$ . Observe that  $\sigma[i] = \sigma_{\text{emb}(\text{DTMC})}[i]$  for  $i \geq 0$ .

$C(s_0 I_0, s_1 I_1, \dots, s_{n-1} I_{n-1}, s_n)$  denotes the cylinder set consisting of all paths of the form  $\sigma = (s_0 t_0, s_1 t_1, \dots) \in \text{Path}_{\text{CTMC}}(s_0)$  such that  $\sigma[i] = s_i$  and  $t_i \in I_i$  for  $i < n$  and  $I \in \mathbb{R}_{\geq 0}$ . Proceeding further,  $\text{Pr}_{\text{CTMC}}(C(s_0 I_0, s_1 I_1, \dots, s_n))$  is recursively defined as follows.

$$\text{Pr}_{\text{CTMC}}(C(s_0 I_0, s_1 I_1, \dots, s_n)) \begin{cases} 1 & \text{if } n = 0 \\ \text{Pr}_{\text{CTMC}}(C(s_0 I_0, s_1 I_1, \dots, s_{n-1})) \cdot T(s_{n-1}, s_n) \cdot \int_{I_{n-1}} E(s_{n-1}) \cdot e^{-E(s_{n-1})t} dt & (20) \\ \text{otherwise} \end{cases}$$

Observe that  $T(s_{n-1}, s_n)$  in the above equation corresponds to probability of moving from  $s_{n-1}$  to  $s_n$  in the  $\text{emb}(\text{DTMC})$  (Definition 3). The last term in the above equation captures the probability of exiting  $s_{n-1}$  in the interval  $I_{n-1}$ .

*CTMC properties.* Properties of interest for CTMC include transient and steady state properties which are expressed in Continuous Stochastic Logic (CSL) developed by Aziz et al. [Aziz et al. 1996]. In this paper, we focus on verification of a specific type of CSL property: time unbounded until property of the form  $\text{P}_{\times r}(\varphi_1 \text{U}^{[0, \infty)} \varphi_2)$

which is satisfied by  $s \in S$  such that  $P_{\text{CTMC}}(s, \varphi_1 \text{U}^{[0, \infty)} \varphi_2) \bowtie r$ , where

$$\begin{aligned} P_{\text{CTMC}}(s, \varphi_1 \text{U}^{[0, \infty)} \varphi_2) &= Pr_{\text{CTMC}}(\{\sigma \in Path_{\text{CTMC}}(s) : \sigma \models \varphi_1 \text{U}^{[0, \infty)} \varphi_2\}), \text{ and} \\ \sigma \models \varphi_1 \text{U}^{[0, \infty)} \varphi_2 &\Leftrightarrow \exists a \in [0, \infty) : \sigma @ a \models \varphi_2 \wedge \forall b < a : \sigma @ b \models \varphi_1 \end{aligned} \quad (21)$$

We will use the fact that  $P_{\text{CTMC}}(s, \varphi_1 \text{U}^{[0, \infty)} \varphi_2)$  is equal to  $P(s, \varphi_1 \text{U} \varphi_2)$  in the corresponding  $\text{emb}(\text{DTMC})$ . For the completeness of discussion, we present here the proof of the above statement.

$$\begin{aligned} P_{\text{CTMC}}(s, \varphi_1 \text{U}^{[0, \infty)} \varphi_2) &= Pr_{\text{CTMC}}(\{\sigma \in Path_{\text{CTMC}}(s) : \sigma \models \varphi_1 \text{U}^{[0, \infty)} \varphi_2\}) \\ &= \sum_{\sigma \in \Upsilon} Pr_{\text{CTMC}}(\sigma) \\ &\quad \text{where } \Upsilon = \{\sigma \in Path_{\text{CTMC}}(s) : \sigma \models \varphi_1 \text{U}^{[0, \infty)} \varphi_2\} \end{aligned}$$

In the above, consider any path  $\sigma = s_0 t_0, s_1 t_1, \dots$ . From Equation 21,  $\exists a \in [0, \infty) : \sigma @ a \models \varphi_2$  and  $\forall b < a : \sigma @ b \models \varphi_1$ . Let  $\sigma @ a = \sigma[k] = s_k$ . Therefore,

$$Pr_{\text{CTMC}}(\sigma^k) = 1 \text{ as } \sigma @ a = \sigma[k] \wedge \sigma @ a \models \varphi_2 \quad (22)$$

and  $\forall i : 0 \leq i < k$

$$Pr_{\text{CTMC}}(\sigma^i) = \int_0^\infty T(s_i, s_{i+1}) \cdot E(s_i) \cdot e^{-E(s_i) \cdot t_i} \cdot Pr_{\text{CTMC}}(\sigma^{i+1}) dt_i \quad (23)$$

From the above,

$$Pr_{\text{CTMC}}(\sigma^0) = \int_0^\infty T(s_0, s_1) \cdot E(s_0) \cdot e^{-E(s_0) \cdot t_0} \cdot Pr_{\text{CTMC}}(\sigma^1) dt_0 = T(s_0, s_1) \cdot Pr_{\text{CTMC}}(\sigma^1)$$

and

$$Pr_{\text{CTMC}}(\sigma^1) = \int_0^\infty T(s_1, s_2) \cdot E(s_1) \cdot e^{-E(s_1) \cdot t_1} \cdot Pr_{\text{CTMC}}(\sigma^2) dt_1 = T(s_1, s_2) \cdot Pr_{\text{CTMC}}(\sigma^2)$$

Proceeding further,

$$Pr_{\text{CTMC}}(\sigma) = Pr_{\text{CTMC}}(\sigma^0) = T(s_0, s_1) \cdot T(s_1, s_2) \cdot \dots \cdot T(s_{k-1}, s_k) = \prod_{i=0}^{k-1} T(s_i, s_{i+1})$$

Note that for any  $\sigma$  in  $\text{CTMC}$  there is a corresponding path  $\sigma_{\text{emb}(\text{DTMC})}$  in the  $\text{emb}(\text{DTMC})$  such that  $\sigma_{\text{emb}(\text{DTMC})}[i] = \sigma[i]$  for all  $i \geq 0$ , and  $\sigma \models \varphi_1 \text{U}^{[0, \infty)} \varphi_2 \Rightarrow \sigma_{\text{emb}(\text{DTMC})} \models \varphi_1 \text{U} \varphi_2$  in the  $\text{emb}(\text{DTMC})$ . Hence,

$$Pr_{\text{CTMC}}(\sigma) = \prod_{i=0}^{k-1} T(s_i, s_{i+1}) = Pr(\sigma_{\text{emb}(\text{DTMC})}) \text{ (see Equation 1 in Section 3.1)} \quad (24)$$

Furthermore,

$$\begin{aligned} \sigma \in Path_{\text{CTMC}}(s) &\Rightarrow \sigma_{\text{emb}(\text{DTMC})} \in Path_{\text{emb}(\text{DTMC})}(s) \text{ and} \\ \pi \in Path_{\text{emb}(\text{DTMC})}(s) &\Rightarrow \exists \sigma \in Path(s) : \sigma_{\text{emb}(\text{DTMC})} = \pi \end{aligned} \quad (25)$$

where  $Path_{\text{emb}(\text{DTMC})}(s)$  denotes the set of paths starting from  $s$  in the  $\text{emb}(\text{DTMC})$ . Recall that,  $\Upsilon = \{\sigma \in Path_{\text{CTMC}}(s) : \sigma \models \varphi_1 \text{U}^{[0, \infty)} \varphi_2\}$ . Let,  $\Xi = \{\sigma_{\text{emb}(\text{DTMC})} : \sigma \in \Upsilon\}$ .

$\Upsilon$ }. From the above, it follows that  $\Xi = \{\pi : \pi \in \text{Path}_{\text{emb}(\text{DTMC})}(s) \wedge \pi \models \varphi_1 \text{ U } \varphi_2\}$ . Therefore, from Equations 24 and 25

$$\begin{aligned} P_{\text{CTMC}}(s, \varphi_1 \text{ U}^{[0, \infty)} \varphi_2) &= \sum_{\sigma \in \Upsilon} Pr_{\text{CTMC}}(\sigma) = \sum_{\pi \in \Xi} Pr(\pi) \\ &= P(s, \varphi_1 \text{ U } \varphi_2) \text{ in the } \text{emb}(\text{DTMC}) \text{ (see Section 3.1)} \end{aligned}$$

Hence, for a CSL property of the form  $P_{\bowtie r}(\varphi_1 \text{ U}^{[0, \infty)} \varphi_2)$  can be verified at any state  $s$  in a CTMC by verifying satisfiability of  $P_{\bowtie r}(\varphi_1 \text{ U } \varphi_2)$  in the corresponding embedded DTMC,  $\text{emb}(\text{DTMC})$ , at the same state  $s$ . That is, our U2B algorithm is applicable in the verification of  $P_{\bowtie r}(\varphi_1 \text{ U}^{[0, \infty)} \varphi_2)$  for CTMC models.

## 8. OVERVIEW OF PRISM-U2B TOOL

We have realized the U2B and U2B\_P methods in a tool PRISM-U2B which is developed leveraging on the existing implementation of PRISM model checker [Hinton et al. 2006] (available under GNU GPL). Figure 9 presents the overview of various modules used in PRISM-U2B. Observe that PRISM-U2B re-uses PRISM’s user interface, parsers and simulation engine. Especially, PRISM-U2B uses PRISM’s input specification language and presents an user interface similar to PRISM. This significantly minimizes the cognitive burden to learn and understand the usage of PRISM-U2B. While the simple changes to the PRISM user interface were sufficient to allow for additional inputs of error bounds and confidence parameters used in two phases of PRISM-U2B, the changes to the PRISM simulator are significant. In the following sections, we discuss the distinguishing aspects of PRISM-U2B with respect to realization of sample path generation and provide an overview of PRISM-U2B usage.

### 8.1 Implementation of PRISM-U2B

The core of PRISM’s statistical method is the simulation generator engine. It takes as input (a) the model (e.g., DTMC) given in terms of its probability transition matrix and translated from PRISM’s model specification language, and (b) the property (e.g., in PCTL) translated from PRISM’s property specification language. The simulator then calculates the set of paths ( $N$ ) required to compute the probability estimate given the user-specified error bound and confidence parameter. Finally, it iterates  $N$  times, in each iteration randomly generates one path up to a pre-specified maximum length  $k$  (by default, 10,000) and returns as the probability estimate the proportion among these  $N$  paths that satisfy the given property. In the event that the simulator obtains a path in which the given property is not decided (i.e., satisfiability and un-satisfiability cannot be inferred), PRISM fails and requests that the user specify a larger value for  $k$ .

**Implementation of *Phase I* in PRISM-U2B.** Two primary limitations in the default PRISM simulator make it unsuitable for its direct use in the *Phase I* of the U2B method in PRISM-U2B. First, the PRISM simulator requires that there be a pre-specified upper bound in the path length even for unbounded until path properties. *Phase I* of the U2B method, on the other hand, requires truly unbounded paths in order to calculate an appropriate  $k_0$ . Second, the PRISM simulator operates by creating single paths and extending them until the property under consideration is decided, or until the maximum path length is reached. As U2B does not put

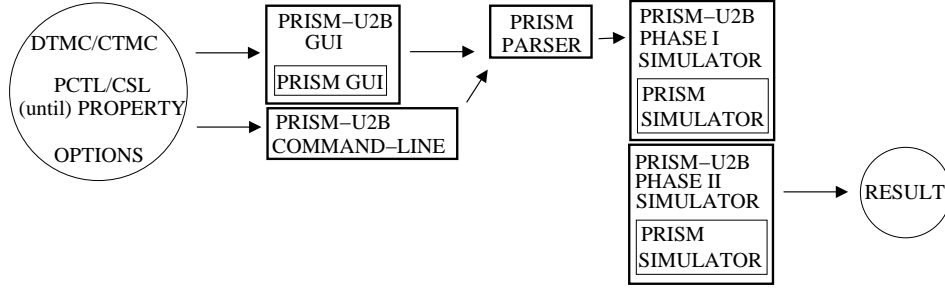


Fig. 9. Architectural overview and module dependencies in PRISM-U2B.

any restriction on the path length, the above is not a good strategy for random generation of paths in the *Phase I*. For instance, for the property  $\varphi_1 \text{ U } \varphi_2$ , if the path being generated is a part of an unconditional loop in the model and satisfies the property  $\varphi_1 \wedge \neg\varphi_2$ , then the above path generation method will lead to generation of an infinite length path and therefore, never terminate. Algorithms proposed in Sections 5 and 6 provide a better strategy for path generation in *Phase I* of U2B (and U2B\_P). This requires that the simulator in PRISM-U2B randomly generate and keep track of multiple paths (where the number of paths corresponds to the *window size* in Algorithms 2, 3) at a time.

**Implementation of *Phase II* in PRISM-U2B.** The above features of PRISM simulator, however, do not affect *Phase II* of PRISM-U2B when the input model is a DTMC. In this phase, only  $k_0$  bounded until properties are considered where  $k_0$  is obtained from *Phase I*. Therefore, for DTMC models, the PRISM simulator is directly used in *Phase II* to estimate the  $k_0$  bounded until property using  $N_2$  (as prescribed by *Phase II* of U2B) samples of paths with length  $k_0$ .

In the event the input model is a CTMC, properties are expressed in the logic of CSL. In CSL, unlike PCTL, a  $k_0$  bounded until property represents until property that is required to be satisfied in  $[0, k_0]$  time interval. That is, the bound on until properties in CSL corresponds to *time* bound as opposed to *step* bound as in PCTL and as is required for *Phase II* in U2B method. As such, when the model under consideration is CTMC, in *Phase II* the corresponding CSL until property is left unaltered and the PRISM simulator is directed to consider sample paths of length  $k_0$ . This faithfully satisfies the requirement of verifying  $k_0$ -step bounded until property in the the embedded DTMC of the given CTMC.

## 8.2 PRISM-U2B Usage Description

The interfaces (command line and graphical) of PRISM-U2B follows closely that of PRISM. We proceed by describing various command line options that are added to the existing ones. While in PRISM, the statistical model checking can be invoked by

```
prism -sim <model-file> <property-file>
```

the U2B method based statistical model checking can be invoked in PRISM-U2B with an additional qualification of the type of statistical method

```
prism -sim 2 <model-file> <property-file>.
```

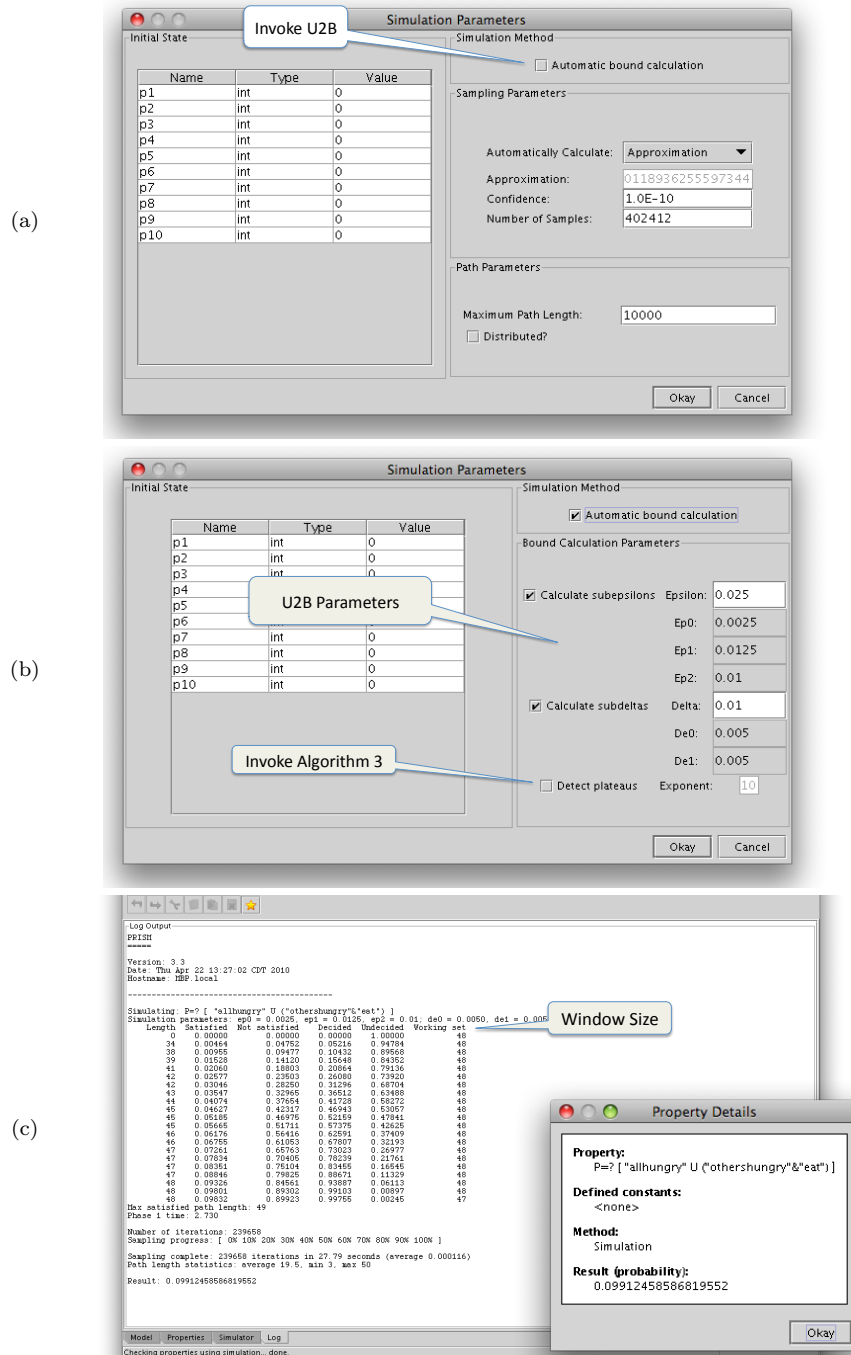


Fig. 10. PRISM-U2B GUI

PRISM’s statistical method can be invoked by setting `sim` flag to 1. In the event that all input properties are (step or time) bounded until properties, PRISM-U2B directly invokes PRISM’s existing sampling technique.

PRISM’s statistical method allows users to input error bound, confidence parameter and maximum path lengths (if no such value is specified, PRISM uses default values of error bound (0.01), confidence parameter ( $10^{-10}$ ) and maximum path lengths (10,000)). In similar fashion, PRISM-U2B uses default values for the error bounds and the confidence parameters in two phases of U2B method:  $\epsilon_0 = 0.0025$ ,  $\epsilon_1 = 0.0125$ ,  $\epsilon_2 = 0.01$  and  $\delta_1 = \delta_2 = 0.005$ . PRISM-U2B also allows for user-specified error bounds and confidence parameters for U2B method. The user can provide specific values for individual  $\epsilon$ s (and/or  $\delta$ s) or can provide a single global  $\epsilon$  (and/or a global  $\delta$ ). In the latter case, the global error bound is partitioned using default factors:  $\epsilon_0 = \frac{\epsilon}{10}$ ,  $\epsilon_1 = \frac{\epsilon}{2}$ ,  $\epsilon_2 = 2\frac{\epsilon}{5}$ , and the global confidence parameter is partitioned into two equal halves for  $\delta_1$  and  $\delta_2$ . For instance,

```
prism -sim 2 -simepsilon 0.002 0.001 0.01 -simdelta 0.0001
                                <model-file> <property-file>
```

This provides as input  $\epsilon_0 = 0.002$ ,  $\epsilon_1 = 0.001$ ,  $\epsilon_2 = 0.01$ , and a global  $\delta = 0.0001$ . In the above, PRISM-U2B uses Algorithm 2 for computing  $k_0$  in *Phase I*. If the user chooses to use method U2B\_P, he/she can invoke

```
prism -sim 2 -simpoly 3 <model-file> <property-file>
```

The value of the switch `simpoly` controls the length of plateau (as discussed in Section 6.2) and ensures the termination of statistical sampling based method.

Inputs can be supplied in similar fashion when PRISM-U2B is invoked via the graphical user interface (Figure 10). The tool, PRISM-U2B, along with its source code, sample examples and documentation, is available from [Two Phase Probabilistic Model Checking 2010].

## 9. EXPERIMENTAL EVALUATION

We evaluate the proposed U2B method and its realization in PRISM-U2B using a number of case studies available in PRISM example suit. The objective of this empirical study is to show the effectiveness of U2B method with respect to (a) precision of estimate, (b) computation time and (c) memory usage. We proceed by describing in brief the various case studies in Section 9.1 followed by detailed analysis of empirical results in Section 9.2.

### 9.1 Case Studies

As noted before, PRISM-U2B re-uses PRISM’s input specification language and language parsers, and therefore, our experiments directly use several case studies from the PRISM example suite. PRISM (and by extension PRISM-U2B) allows for a specific type of PCTL and CSL property syntax, which essentially *queries* the probability with which certain path property is satisfied by the system under consideration. The syntax for such a query is `P=?[path-property]`. The result of such a query is the probability with which the paths from the start state of the system satisfies `path-property`. In the following, we will discuss case studies and consider several property queries. The results obtained by applying PRISM’s numerical method

(whenever possible) will be used to evaluate and compare the precision of results obtained by applying PRISM's statistical method and the U2B method of PRISM-U2B.

9.1.1 *Randomized Dining Philosopher.* This is a DTMC model represented a probabilistic variation of the standard dining philosopher protocol. The model is analyzed against a PCTL until property query to obtain the probability that the first philosopher is able to eat before any other philosophers. That is,

$$P=?[ \text{"allhungry"} \text{ U } ( \text{"othershungry"} \ \& \ \text{"eat"} ) ]$$

where **allhungry** denotes none of the philosophers have eaten, **othershungry** denotes all but the first philosopher have not eaten and **eat** denotes the first philosopher is able to eat. Experiments are conducted by varying the number of philosophers in the model.

Available at: [Two Phase Probabilistic Model Checking 2010].

9.1.2 *Simple Queue.* This is DTMC model of a queue which captures the probabilistic behavior of requests being serviced. There is a specific state in the model such that when the queue moves to that state, it implies there is an *overflow* of requests. The property considered is used to compute the probability with which the queue eventually reaches such a state.

$$P=?[ \text{true U "overflow"} ]$$

Experiments are conducted by varying the size of the buffer (queue states) in the model.

Available at: [Two Phase Probabilistic Model Checking 2010].

9.1.3 *IPv4 Zeroconf Protocol.* This a simplified DTMC model [Sen et al. 2004] representing the IPv4 Zeroconf Protocol. Similar to overflow state in queue model, it has an *error* state; it is not desirable for the system executing the protocol to be in such a state. The property query used in our experiments is

$$P=?[ \text{true U "error"} ]$$

As in queue, experiments are conducted by varying the number of states (denoting the number of intermediate nodes in the network) in the Zeroconf protocol.

9.1.4 *Broadcast Protocol.* This is DTMC model of a simple broadcast protocol based on *gossiping*, where each node in the network forwards the message it receives to its neighbors with a pre-specified probability. Two variations of the protocol are considered. In *synchronous, no-collision* variation, the nodes send and receive messages simultaneously over independent channels (thereby ensuring freedom from collision). The *synchronous, lossy* variation, on the other hand, assumes that the neighboring nodes share a channel, which may result in collision and message loss. The property query of interest involves computing the probability with which certain nodes (e.g., nodes 1 and 2) receive the broadcast message.

$$P=?[ \text{true U (active1=0)} ] \text{ and } P=?[ \text{true U (active2=0)} ]$$

In the above **active<i>** is equal to 0 when a node goes to sleep after receiving (and possibly forwarding) message.

Available at: [http://www.prismmodelchecker.org/casestudies/prob\\_broadcast.php](http://www.prismmodelchecker.org/casestudies/prob_broadcast.php).

9.1.5 *EGL Contract Signing Protocol.* EGL Contract signing protocol [] describes a method by which two parties can reach an agreement by exchanging their respective secrets. The protocol is said to be fair if and only if the exchange of messages following the protocol guarantees that all participants obtain the others secret or none do. A DTMC model of EGL contract signing protocol is considered ([]). A property query is considered to demonstrate the weakness in the protocol with respect to fairness. That is, the property captures the fact that one participant gets access to other party's secret without communicating its own to the latter.

$$P=?[ \text{true} \text{ U } (!\text{"kA"} \ \& \ \text{"kB"}) ]$$

In the above,  $\text{kA}$  and  $\text{kB}$  denotes the states where participants A and B *has knowledge* of the others secret.

Available at: [http://www.prismmodelchecker.org/casestudies/contract\\_egl.php](http://www.prismmodelchecker.org/casestudies/contract_egl.php)

9.1.6 *Simple Dice Protocol.* This is a DTMC model where the possible outcomes of rolling a fair dice is captured using multiple flips of a fair coin. The result of the property query, against which the model is analyzed, is the probability of obtaining a specific output of the dice. For instance,

$$P=?[\text{true} \text{ U } (\text{"s=7"} \ \& \ \text{"d=6"})]$$

is a PCTL property querying the probability of obtaining a result of 6 in a dice roll. Available at: <http://www.prismmodelchecker.org/casestudies/dice.php>

9.1.7 *Embedded Control System.* This is a CTMC model of an embedded data collection system with built-in redundancy. The controller ensures that the system shuts down if any module in the system is down (due to possible malfunction) for a certain number of cycles (e.g., 20,000). The property queries for the probability that the cause of shut down is attributed to a sensor failure. It is represented in PCTL query as follows.

$$P=?[ !\text{"down"} \text{ U } \text{"fail_sensors"} ]$$

Available at: <http://www.prismmodelchecker.org/casestudies/embedded.php>

9.1.8 *Workstation Cluster.* This is a CTMC model representing two clusters of  $N$  workstations connected to a switch in a star-topology. The switches are connected via a backbone. The workstations, switches and the backbone can fail at any time. There is a repairing mechanism such that each component is repaired at a certain rate. A CSL property query is used to compute the probability with which the system is able to offer *premium* quality of service (i.e., at least  $N$  workstations are operational and corrected via switches and backbone) at some point in future. That is,

$$P=?[ \text{true} \text{ U } \text{"premium"} ]$$

Experiments are conducted by varying the number of workstations in the cluster. Available at <http://www.prismmodelchecker.org/casestudies/cluster.php>

9.1.9 *Cyclic Server Polling System.* This is a CTMC model of a polling system consisting of multiple workstations that are being served by a single server. The CSL property that is used for our evaluation is

Model	Variations	PRISM			PRISM-U2B			
		Numerical $p$	Time	Statistical $\hat{p}$	Phase I $k_0$	Time	Phase II $\hat{p}$	Time
Example (Figure. 1)	No result	No result		No result	2349	5.42	0.6605	71.94
	# philos: 10	0.1	5.98	0.0999	75	<b>29.21</b>	0.0994	<b>16.87</b>
	# philos: 20	No result		0.0497	103	<b>89.6</b>	0.0499	<b>52.51</b>
Dining philosopher	# philos: 40	No result		0.0247	141	<b>252.49</b>	0.0246	<b>147.27</b>
	# philos: 60	No result		0.0163	176	<b>469.29</b>	0.0165	<b>272.14</b>
	# philos: 100	No result		0.0099	227	<b>1044.12</b>	0.0098	<b>608.06</b>
Queue	$n : 80, q : 0.9, r : 0.5$	0.1022	< 1	0.1061	20579	<b>212.95</b>	0.1073	<b>1293.45</b>
	$n : 100, q : 0.9, r : 0.5$	0.0852	< 1	0.0874	31253	<b>273.36</b>	0.0852	<b>1889.64</b>
	$n : 140, q : 0.9, r : 0.5$	0.0607	< 1	0.0637	59527	<b>392.61</b>	0.0625	<b>3341.12</b>
	$n : 180, q : 0.9, r : 0.5$	0.0477	< 1	0.0506	77549	<b>521.69</b>	0.049	<b>948.95</b>
	$n : 200, q : 0.9, r : 0.5$	0.0431	< 1	No result at $k = 10^6$	39899	26.33	0.0445	<b>6489.19</b>
Zeroconf	$n : 10, q : 0.9, r : 0.9$	0.8098	< 1	0.8118	99	<b>4.18</b>	0.8093	<b>2.22</b>
	$n : 20, q : 0.9, r : 0.9$	0.5975	< 1	0.6015	360	<b>11.16</b>	0.5998	<b>6.36</b>
	$n : 60, q : 0.9, r : 0.9$	0.0215	< 1	0.0222	1469	<b>30.72</b>	0.0219	<b>17.85</b>
	$n : 80, q : 0.9, r : 0.9$	0.0027	< 1	0.0027	1483	<b>31.58</b>	0.0027	<b>18.34</b>
Broadcast Protocol	Synch. no Collision	0.8	< 1	0.8001	3	<b>34.83</b>	0.8	<b>20.89</b>
	Synch. collision, lossy	0.7324	< 1	0.7329	8	<b>45.14</b>	0.7322	<b>31.6</b>
		0.5815	9.3	0.5814	20	<b>163.26</b>	0.5817	<b>120.88</b>
		0.3462	9.61	0.3465	22	<b>225.99</b>	0.3464	<b>167.57</b>
Contract Signing		1.0	< 1	1.0	36	<b>60.99</b>	1.0	<b>33.77</b>
Dice		0.1667	< 1	0.1646	23	2.39	0.1608	2.57
ECS	MAX.COUNT : 20, 000	0.7555	24568.49	No result at $k = 10^6$	120781	1507.93	0.7544	25003.39
Cluster	$N : 8$	1.0	< 1	1.0	1	< 1	1.0	< 1
	$N : 128$	1.0	5.66	1.0	1	< 1	1.0	< 1
	$N : 1024$	No result		1.0	1	< 1	1.0	< 1
Polling	$N : 2$	0.5	< 1	0.5	2581	178.76	0.4984	104.94
	$N : 20$	0.538	8713.28	0.5384	9714	3847.5	0.5389	2221.5

Table I. Summary of Results: Computed/estimated probabilities, Time (in secs), Maximum Sample path length. Total time used by PRISM-U2B is sum of times used in *Phase I* and in *Phase II*. Entries in bold shows the cases where the total time used by PRISM-U2B is less than that used by PRISM's statistical method.

$$P=?[ !(s=2 \ \& \ a=1) \cup (s=1 \ \& \ a=1) ]$$

which queries for the probability with which the first workstation in the system is served before the second one. Experiments are conducted by varying the number of workstations in the system.

Available at: <http://www.prismmodelchecker.org/casestudies/polling.php>

## 9.2 Precision, Computation Time and Memory Usage

The experimental evaluation is based on the illustrative example discussed in Section 1 and the case studies described in Section 9.1. All experiments are conducted on Red Hat Enterprise Linux 5.1 running on Intel Core 2 Duo 3GHz CPU and 2GB memory. The results for PRISM-U2B are obtained by using the Algorithm 2.

*9.2.1 Precision & Computation Time.* We show that estimate obtained using U2B method is as precise as the statistical method provided by PRISM tool given the error bounds and confidence parameters. Furthermore, U2B method is as efficient (typically about 1.5 times faster) than the PRISM’s statistical method.

Table I provides a summary of our results. For each of the methods: PRISM’s numerical method, PRISM’s statistical method and PRISM-U2B, the table presents the probability and its estimates, and the computation time. For the statistical methods, the bounds on the sample path lengths are also reported:  $\bar{k}$  denotes the maximum path length considered by PRISM’s statistical method (default upper bound is 10,000 and can be user-specified) and  $k_0$  denotes the maximum path length considered by PRISM-U2B (the bound obtained automatically from *Phase I* of U2B).

For the illustrative example, PRISM fails to compute any result, while PRISM-U2B successfully terminates with a good estimate. For the randomized dining philosopher example, PRISM’s numerical method fails to compute any result when the number of philosophers is  $\geq 12$  due to prohibitively large state-space of the DTMC model (details in Section 9.2.2), whereas PRISM’s statistical method and U2B successfully compute the estimate. For the simplified queue model, PRISM’s numerical method and statistical method requires updates to the default value for maximum number of Jacobi iterations or the maximum simulation path length, respectively. On the other hand, U2B method does not require any such user guidance. For rest of the case studies, PRISM-U2B typically outperforms PRISM’s statistical method (timing results shown in bold).

*9.2.2 Memory Usage.* Another important aspect of consideration is the memory consumed by the methods: PRISM’s numerical method, PRISM’s statistical method and U2B method. Figure 11 illustrates the bar graph of the memory usage (in terms of KB) for the case studies by each of the above methods. As all the methods are realized in PRISM model checking engine, each incurs a minimal overhead of  $\sim 30$ MB. The memory usage is comparable for all the methods for models that have small state-space. As expected, for models in which the state-space is large (e.g., randomized dining philosopher, embedded control system, and workstation cluster with 128 hosts), PRISM’s numerical method uses much more memory than the statistical methods (in PRISM and PRISM-U2B). Figure 12 shows the increase in the memory usage by the PRISM’s numerical method with the increase the number

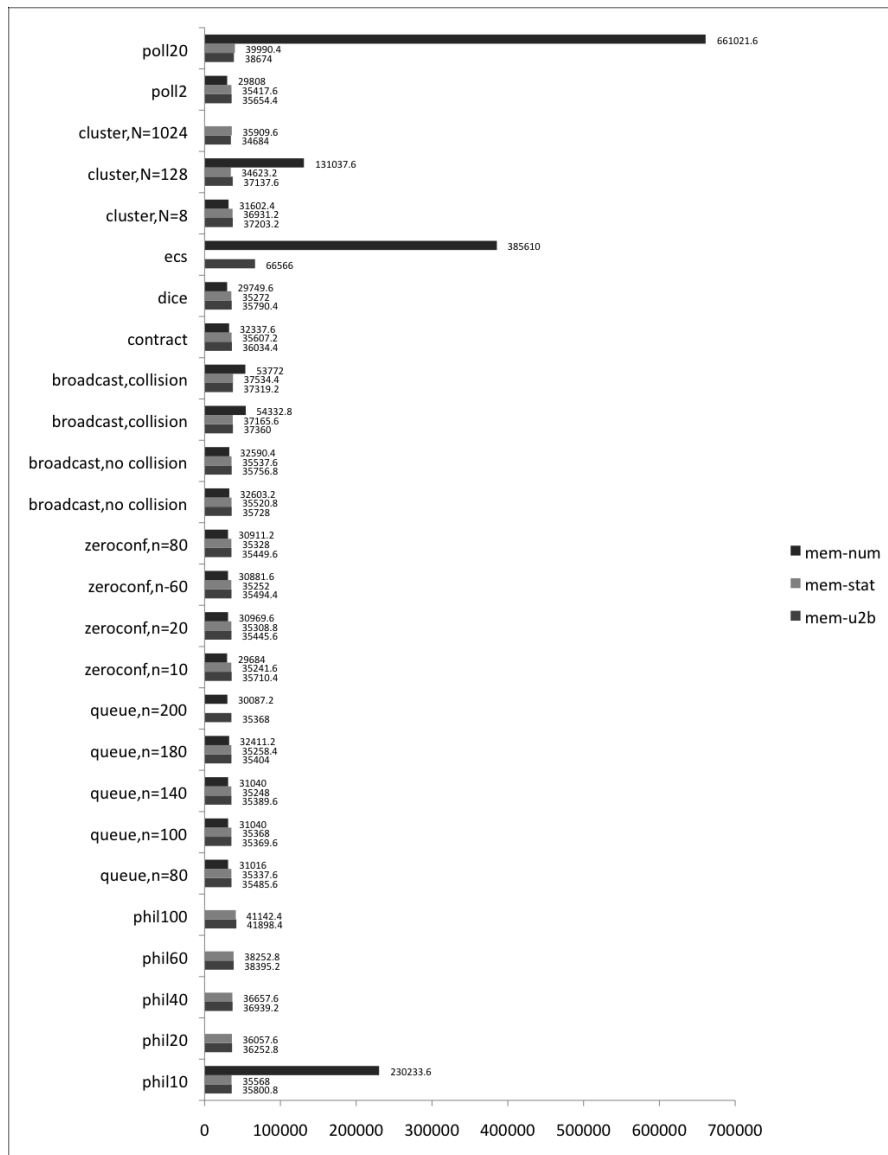


Fig. 11. Memory usage (in KB) for Case Studies (mem-num: memory usage by PRISM's numerical method; mem-stat: memory usage by PRISM's statistical method; mem-u2b: memory usage by PRISM-U2B's U2B method).

of philosophers in randomized dining philosopher case study. PRISM's numerical method fails due to large memory requirement when the number of philosophers is greater than 11. Observe that statistical methods do not suffer from such increase in the memory usage with the increase the state-space of the model.

It is worth mentioning that U2B method in general is expected to use more mem-

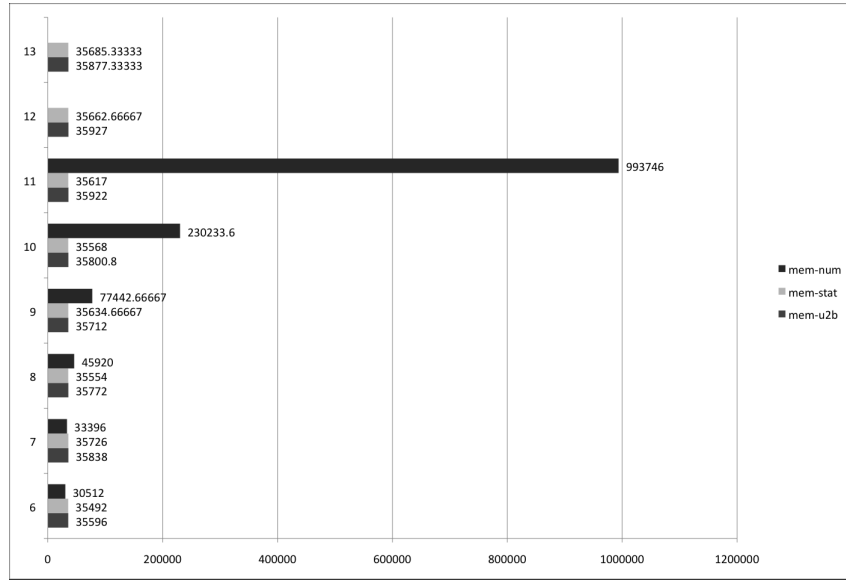


Fig. 12. Memory usage for (in KB) Randomized Dining Philosopher (mem-num: memory usage by PRISM’s numerical method; mem-stat: memory usage by PRISM’s statistical method; mem-u2b: memory usage by PRISM-U2B’s U2B method).

ory than that used by the statistical method of PRISM as U2B method stores  $N_1\epsilon_0$  states for the *Phase I* computation (for  $\epsilon_0 = 0.0025$  and  $\delta = 0.001$ ,  $N_1\epsilon_0 = 48$ ). However, U2B method is typically faster than PRISM’s statistical method due to the following. While both U2B in *Phase II* and PRISM’s statistical method uses similar sampling technique, the former uses much smaller sample path length bound ( $k_0$  obtained via *Phase I*) compared to PRISM’s statistical method (which uses  $\bar{k}$ ; see Table I). Recall that, default value for sample path length in PRISM’s statistical method is 10,000 and it may require user specified ones for some examples. On the other hand, U2B method uses *Phase I* to compute a model dependent path length bound  $k_0$ . As a result, the gain in computation time achieved by using  $k_0$  in *Phase II* of U2B instead of the sample path lengths used in PRISM’s statistical method surpasses the loss in computation time due to *Phase I* overhead.

**9.2.3 Summary of results.** The results empirically show that U2B method as implemented in PRISM-U2B is (a) as time-efficient as PRISM’s statistical method (in many cases outperforms PRISM’s statistical method), (b) as precise as PRISM’s statistical method, and (c) successfully computes result *automatically* for cases where PRISM fails or requires user guidance (to pre-specify up maximum Jacobi iterations for numerical method or maximum sample path length for numerical method). In short, PRISM-U2B broadens the scope of application of approximate probabilistic model checking based on statistical methods; it is not only more efficient and effective than one of the most widely used statistical method as implemented in PRISM but also has been proven (theoretically and empirically) to be applicable for case studies where PRISM fails.

## 10. CONCLUSION

**Summary.** In this paper, we have presented U2B, an approximate probabilistic model checking method, based on statistical sampling. The method does not require any prior information regarding the structure of the model being verified and therefore, can be used in a black-box setting. The method can be applied for verifying untimed properties in both DTMC and CTMC models. We have proved the correctness of the U2B method and have discussed the optimized realization of the U2B method. We have also explored a class of examples where PRISM’s statistical method as well as U2B method will suffer from non-termination and explained the cause of such non-termination in terms of the notion of “plateau” in D-Graph. We have developed U2B.P, a heuristic-based plateau-detection method, to deal with the non-termination problem. Finally, we have incorporated the U2B and U2B.P methods in PRISM tool to develop PRISM-U2B and conducted detailed experimental study using different probabilistic models. The experiments show the effectiveness of U2B both in terms of precision and computation time.

**Future work.** As part of future work, we plan to study the feasibility of identifying and pre-computing the *necessary* information (regarding the model transition structure) required to guarantee the termination of *Phase I* of the U2B method. We also plan to develop statistical sampling based methods to verify untimed, unbounded path properties of Markov Decision Processes (MDP). An MDP is a probabilistic model which includes both probabilistic and non-deterministic choices in its transition system. The verification objective for an MDP is to compute the maximum or the minimum probability with which a state satisfies a given path property. Such computation requires identifying a strategy for selecting next states at non-deterministic choice points, where each strategy results in a DTMC embedded in the corresponding MDP.

## REFERENCES

- AZIZ, A., SANWAL, K., SINGHAL, V., AND BRAYTON, R. 1996. Verifying continuous time markov chains. In *8th Intl. Conf. on Computer Aided Verification*. Vol. 1102.
- AZIZ, A., SANWAL, K., SINGHAL, V., AND BRAYTON, R. 2000. Model checking continuous time markov chains. *ACM Transactions on Computational Logic* 1(1), 162–170.
- BAIER, C., HAVERKORT, B., HERMANN, H., AND KATOEN, J.-P. 2003. Model-checking algorithms for continuous-time markov chains. *IEEE Transactions on Software Engineering* 29(6), 524–541.
- BIANCO, A. AND DE ALFARO, L. 1995. Model checking of probabilistic and nondeterministic systems. In *Foundations of Software Technology and Theoretical Computer Science*. Vol. 1026.
- CASELLA, G. AND BERGER, R. L. 2002. *Statistical Inference*. Duxbury.
- COURCOUBETIS, C. AND YANNAKAKIS, M. 1995. The complexity of probabilistic verification. *Journal of ACM* 42(4), 857–907.
- DUFLOT, M., KWIATKOWSKA, M., NORMAN, G., AND PARKER, D. 2006. A formal analysis of bluetooth device discovery. *Intl. Journal on Software Tools for Technology Transfer* 8, 621 – 632.
- HANSSON, H. AND JONSSON, B. 1994. A logic for reasoning about time and reliability. *Formal Aspects of Computing* 6(5), 512–535.
- HERAULT, T., LASSAIGNE, R., MAGNIETTE, F., AND PEYRONNET, S. 2004. Approximate probabilistic model checking. In *5th Intl. Conf. on Verification, Model Checking, and Abstract Interpretation*. Vol. 2937. Springer.

- HINTON, A., KWIATKOWSKA, M., NORMAN, G., AND PARKER, D. 2006. PRISM: A tool for automatic verification of probabilistic systems. In *12th Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*.
- HOEFFDING, W. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association* 58.
- KWIATKOWSKA, M., NORMAN, G., AND PARKER, D. 2008. Using probabilistic model checking in systems biology. *ACM SIGMETRICS Perf. Eval. Review* 35, 14–21.
- MASSART, P. 1990. The tight constant in the Dvoretzky-Kiefer-Wolfowitz inequality. *Annals of Probability* 18, 1269–1283.
- NORMAN, G. AND SHMATIKOV, V. 2006. Analysis of probabilistic contract signing. *Journal of Computer Security* 14, 561–589.
- ROY, A. AND GOPINATH, K. 2005. Improved probabilistic models for 802.11 protocol verification. In *14th Intl. Conf. on Computer Aided Verification*.
- SEN, K., VISWANATHAN, M., AND AGHA, G. 2004. Statistical model checking of black-box probabilistic systems. In *16th Intl. Conf. on Computer Aided Verification*. Vol. 3114. Springer.
- SEN, K., VISWANATHAN, M., AND AGHA, G. 2005. On statistical model checking of stochastic systems. In *14th Intl. Conf. on Computer Aided Verification*. Vol. 3576. Springer.
- Two Phase Probabilistic Model Checking 2010. Two-phase pmck. Available at <http://www.cs.iastate.edu/~poj/pmck>.
- WALD, A. 1945. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics* 16, 2.
- YOUNES, H. L., KWIATKOWSKA, M., NORMAN, G., AND PARKER, D. 2006. Numerical vs. statistical probabilistic model checking. *Intl. Journal on Software Tools for Technology Transfer* 8, 3.
- YOUNES, H. L. S. AND SIMMONS, R. G. 2002. Probabilistic verification of discrete event systems using acceptance sampling. In *14th Intl. Conf. on Computer Aided Verification*. Vol. 2404. Springer.
- YOUNES, H. L. S. AND SIMMONS, R. G. 2006. Statistical probabilistic model checking with a focus on time-bounded properties. *Information and Computation* 204, 9.
- ZAPREEV, I. S. 2008. Model checking markov chains: Techniques and tools. Ph.D. thesis, University of Twente, The Netherlands.