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

PAUL JENNINGS, ARKA P. GHOSH, and SAMIK BASU, Iowa State University

We have developed a new approximate probabilistic model-checking method for *untimed* properties in probabilistic systems, expressed in a 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 required to verify the until property. The method consists of two phases. In the first phase, a suitable system- and property-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 of the probability of satisfying the original unbounded until property. Both phases require only verification of 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 compare this implementation with sampling-based model-checking techniques implemented in two tools: PRISM and MRMC. We show that for several models these existing tools fail to compute the result, while the two-phase method successfully computes the result efficiently with respect to time and space.

Categories and Subject Descriptors: D.2.4 [**Software Engineering**]: Software/Program Verification—*Formal methods, model checking, 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

ACM Reference Format:
Jennings, P., Ghosh, A. P., and Basu, S. 2012. A two-phase approximation for model checking probabilistic unbounded until properties of probabilistic systems. ACM Trans. Softw. Eng. Methodol. 21, 3, Article 18 (June 2012), 35 pages.
DOI = 10.1145/2211616.2211621 http://doi.acm.org/10.1145/2211616.2211621

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 prespecified probability distribution. Such probabilistic behaviors are often modeled using discrete-time Markov chains (DTMC), continuous-time Markov chains (CTMC), and Markov decision processes (MDP). Several techniques and tools have been developed to prove the correctness (in a probabilistic sense) of

This article is a revised and expanded version of an article presented at the 11th International Conference of Formal Engineering Methods.
This work is supported in part by NSF grant CCF0702758.
Authors' addresses: P. Jennings and S. Basu, Department of Computer Science, Iowa State University; A. P. Ghosh, Department of Statistics, Iowa State University; email: {poj, sbasu, apghosh}@iastate.edu.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2012 ACM 1049-331X/2012/06-ART18 $15.00
DOI 10.1145/2211616.2211621 http://doi.acm.org/10.1145/2211616.2211621

these system models. One such technique that automatically verifies the conformance of probabilistic systems modeled as DTMC, CTMC, or MDP [Roy and Gopinath 2005; Norman and Shmatikov 2006; Duflot 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 the *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 system and applies linear equation solvers to obtain the probability with which the system satisfies a property. In contrast, the other method, referred to as the *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 satisfaction of a property by the whole system using statistical arguments and probabilistic analysis. While the numerical method provides exact solutions, it requires complete knowledge of the system and may fail for systems with a large state space (known as the 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 to simulate 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 of the sample (i.e., the number of sample paths N). 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 a certain prespecified error limit (ϵ) and confidence parameter (δ).

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) of $\varphi_1 \, \mathtt{U}^{\leq k} \, \varphi_2$, that is, φ_2 must be satisfied within k steps from the start state, and φ_1 must be satisfied in all states before that. This implies that finite paths of length k are sufficient to verify such properties.

However, the property of interest may be *unbounded*[1], that is, $\varphi_1 \, \mathtt{U} \, \varphi_2$. The semantics of the property states that a path satisfies it if and only if there exists a state in the path which satisfies φ_2 and state φ_1 is satisfied in all states before that. Note that φ_1 can be satisfied any number of times in a path before φ_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 any extension of the path will eventually satisfy or not satisfy $\varphi_1 \, \mathtt{U} \, \varphi_2$. The existing statistical methods for probabilistic model checking either assume that the bound is given (e.g., [Herault et al. 2004; Younes and Simmons 2006]—in essence verifying a bounded path property—or require some specific knowledge of the system behavior [Sen et al. 2005; Rabih and Pekergin 2009].

In contrast, we introduce a new statistical method which automatically computes bound k_0 on a simulation path length and does not require any prior knowledge of the system. The central theme of our technique is to reduce the problem of verifying ($\varphi_1 \, \mathtt{U} \, \varphi_2$) to that of its bounded counterpart ($\varphi_1 \, \mathtt{U}^{\leq k_0} \, \varphi_2$). The reduction is possible only when a suitable k_0 can be obtained for which the $\mathsf{P}(s, \varphi_1 \, \mathtt{U}^{\leq k_0} \, \varphi_2)$ (i.e., probability of satisfying of $\varphi_1 \, \mathtt{U}^{\leq k_0} \, \varphi_2$ at state s) is a good approximation of $\mathsf{P}(s, \varphi_1 \, \mathtt{U} \, \varphi_2)$. In other words, the bound k_0 is large enough to make the difference between $\mathsf{P}(s, \varphi_1 \, \mathtt{U}^{\leq k_0} \, \varphi_2)$ and $\mathsf{P}(s, \varphi_1 \, \mathtt{U} \, \varphi_2)$ small.

[1]Grunske [2008] proposes patterns for specifying probabilistic properties where he discusses the need for (time-) unbounded until properties for representing various probabilistic properties, for example, invariance, existence, and response.

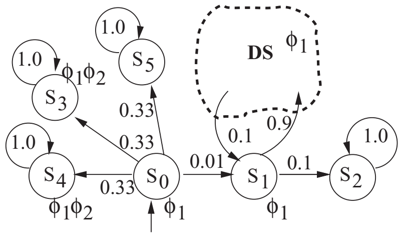

Fig. 1. A simple example.

Such a k_0 provides an approximate upper bound of the sample path length needed for our statistical verification technique. We obtain k_0 using the probability of satisfying a different bounded path property: $\psi_k := (\varphi_1 \, \mathtt{U}^{\leq k} \, \varphi_2) \vee (\neg\varphi_2 \, \mathtt{U}^{\leq k} \, (\neg\varphi_1 \wedge \neg\varphi_2))$. This property states that the original property $(\varphi_1 \, \mathtt{U} \, \varphi_2)$ is either satisfied (first disjunct) or unsatisfied (second disjunct) in at most k steps. We prove that a suitable k_0 is one for which $\mathsf{P}(s, \psi_{k_0})$ is close to 1, and that the degree of closeness is related to ϵ, that is, the overall measure of accuracy of the entire statistical method.

In essence, there are two phases in our method. The first phase estimates $\mathsf{P}(s, \psi_k)$ for $k = 0, 1, 2, \ldots$ and chooses k_0, which satisfies $\mathsf{P}(s, \psi_{k_0}) \geq 1 - \epsilon_0$, where $\epsilon_0 < \epsilon$. In the second phase, $\mathsf{P}(s, (\varphi_1 \, \mathtt{U}^{\leq k_0} \, \varphi_2))$ is estimated, which in turn serves as an estimate of $\mathsf{P}(s, (\varphi_1 \, \mathtt{U} \, \varphi_2))$. The computations for each phase involve only *bounded-path* properties and can be carried out efficiently using the existing sampling techniques, such as those of Herault et al. [2004]. For systems where $\mathsf{P}(s, \psi_k) \not\geq 1 - \epsilon_0$ for any k, we propose an alternate heuristic method for estimating $\mathsf{P}(s, (\varphi_1 \, \mathtt{U} \, \varphi_2))$ based on changes in the estimates of $\mathsf{P}(s, \psi_k)$ 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), for which the proposed method is applied successfully and for which 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 Jennings et al. [2010] for the specification). We will refer to this segment as `DS`. Let proposition φ_1 hold in all states except s_2 and s_5, and proposition φ_2 hold in states s_3 and s_4. The objective is to find the probability of satisfying the property $(\varphi_1 \, \mathtt{U} \, \varphi_2)$ at state s_0. From the probabilities specified in the figure, we know that the resultant probability is 0.66, as the only two paths that satisfy the property are $(s_0, s_3, \ldots)$ and $(s_0, s_4, \ldots)$.

We experimented with the `PRISM` model checker [Hinton et al. 2006] using the preceding 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 ϵ and provides an approximate result within ϵ 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 $\varphi_1 \, \mathtt{U} \, \varphi_2$ be known within some prespecified number of steps. The failure happens when at least one sample path enters `DS` and does not leave `DS`. In this case, φ_1 is satisfied for all states in the path, but φ_2 is not satisfied for any states in the path, such that it cannot be known whether $\varphi_1 \, \mathtt{U} \, \varphi_2$ is satisfied or not.

In terms of ψ_k, as introduced earlier, the previous requirement in PRISM's statistical method is equivalent to $\mathsf{P}(s_0, \psi_k) = 1$, for some prespecified bound k ($k = 10{,}000$ by default in PRISM). In general, it is not possible to find an appropriate value for k such that $\mathsf{P}(s_0, \psi_k) = 1$. By contrast, we claim that it is not necessary to verify whether $\mathsf{P}(s_0, \psi_k)$ 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 $\mathsf{P}(s_0, \psi_k)$ is close to 1. In the preceding example, such a bound can be immediately obtained, as the sample paths $(s_0, s_1, \ldots)$ have very low probability (≤ 0.01). Once such a bound is obtained, we compute $\mathsf{P}(s_0, \varphi_1 \mathtt{U}^{\leq k_0} \varphi_2)$, which approximately coincides with $\mathsf{P}(s_0, \varphi_1 \mathtt{U} \varphi_2)$. In our experiments, with $n \approx 10^8$, the PRISM model checker fails to provide any result, while our method identifies a bound $k_0 = 3343$ and estimates the probability to be equal to 0.6601 in approximately 97 seconds.

1.2. Contributions

(1) *Automation.* We present a methodology for automatically selecting a suitable bound k_0, which allows unbounded until properties for probabilistic systems modeled as DTMC and CTMC to be verified using the corresponding k_0-bounded until properties. The reduction allows us to reuse the existing results for statistical verification of bounded until properties to identify the bound on the sample size N, as required to infer results within a prespecified error margin.
(2) *Universal Application.* The technique is applied to probabilistic model checking of any unbounded (untimed) path properties (expressed in PCTL or CSL), for models expressed as 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 with a prespecified error bound and confidence parameter. When the required condition is not satisfied in a system, our technique (and any other statistical method for probabilistic model-checking that does not require prior knowledge of the complete model) will fail to terminate. We discuss a heuristic for dealing with such systems.
(4) *Effective Implementation and Usability.* We present PRISM-U2B, an optimized implementation of our method based on the well-developed probabilistic model-checking tool PRISM. We leverage PRISM's realization of generating sample simulations from a given DTMC or CTMC model and reuse 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. It is worth mentioning that Jansen et al. [2007] rates PRISM as the most user-friendly tool for probabilistic model checking in terms of modeling features and usability. Being based on PRISM, our tool PRISM-U2B enjoys similar ease of use.
(5) *Experimental Evaluation.* We compare PRISM-U2B and PRISM's statistical method 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. We also compare the tools PRISM-U2B and MRMC. MRMC is faster than PRISM-U2B, as MRMC (unlike PRISM-U2B) utilizes pre-analysis of the model. However, MRMC fails for case studies with large state spaces, where PRISM-U2B successfully computes the result. The tool PRISM-U2B, as well as its documentation and case studies,[2] can be obtained at Jennings et al. [2010].

[2]Most of the case studies are available at http://www.prismmodelchecker.org/casestudies/index.php.

1.3. Organization

Section 2 provides a brief overview of discrete-time Markov chains and the probabilistic temporal logic PCTL. Section 3 discusses related work. 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 this 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. PRELIMINARIES

We proceed with a brief summary on probabilistic systems modeled as DTMC, followed by the syntax and semantics of 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 CTMC.

2.1. Probabilistic Systems: Discrete-Time Markov Chain Models

We will describe the behavior of a system that evolves from one configuration to another based on a certain probability as a state machine 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 DTMC.

Definition 2.1 (*Discrete-Time Markov chain*). 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, $T : S \times S \to [0, 1]$ is a transition probability function such that $\forall s : \sum_{s' \in S} T(s, s') = 1$, and $L : S \to \mathcal{P}(AP)$ is the labeling function which labels each state with a set of atomic propositions $\subseteq AP$ that hold in that state.

Paths and Probability Measures. A path in a DTMC, denoted by π, is a finite or infinite sequence of states $(s_0, s_1, s_2, s_3, \ldots)$ 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 ith state in the path π, and $|\pi|$ is the length of π in terms of the number of transitions in π. For example, for an infinite path π, $|\pi| = \infty$, while for a finite path $\pi = (s_0, \ldots, s_n)$, $|\pi| = n, n \geq 0$. The cylinder set, denoted by $C_s(\pi)$ for a state s and a finite length path π 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 π. For any finite path π with $|\pi| = n$ we define the following.

$$P(\pi) = \begin{cases} 1 \text{ if } n = 0; \\ T(\pi[0], \pi[1]) \times \cdots \times T(\pi[n-1], \pi[n]) \text{ otherwise.} \end{cases} \tag{1}$$

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 σ−algebra of path(s).

2.2. Probabilistic Properties

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

$$\varphi \;\to\; tt \mid a \in AP \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathtt{P}_{\bowtie \mathtt{r}}(\psi) \quad \text{and} \quad \psi \;\to\; \varphi \,\mathtt{U}\, \varphi \mid \varphi \,\mathtt{U}^{\leq k}\, \varphi.$$

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

$$s \models tt \text{ for all } s \in S \qquad s \models a \Leftrightarrow a \in L(s) \qquad s \models \neg\varphi \Leftrightarrow s \not\models \varphi$$
$$s \models \varphi_1 \wedge \varphi_2 \Leftrightarrow s \models \varphi_1 \text{ and } s \models \varphi_2 \qquad s \models \mathtt{P}_{\bowtie \mathtt{r}}(\psi) \Leftrightarrow \mathsf{P}(s, \psi) \bowtie r$$

In the preceding, $\mathsf{P}(s, \psi) = Pr(\{\pi \in Path(s) : \pi \models \psi\})$. In other words, $s \models \mathtt{P}_{\bowtie r}(\psi)$ holds if and only if the probability that ψ is true for an outgoing infinite path from state s is $\bowtie r$. For any infinite path π, we define the following.

$$\pi \models \varphi_1 \, \mathtt{U}^{\leq k} \, \varphi_2 \Leftrightarrow \exists 0 \leq i \leq k : \pi[i] \models \varphi_2 \ \wedge \ \forall j < i : \pi[j] \models \varphi_1$$
$$\pi \models \varphi_1 \, \mathtt{U} \, \varphi_2 \Leftrightarrow \exists i \geq 0 : \pi[i] \models \varphi_2 \ \wedge \ \forall j < i : \pi[j] \models \varphi_1$$

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

3. RELATED WORK

Legay et al. [2010] survey and compare different probabilistic model-checking techniques based on statistical sampling. In this section, we elaborate on some of these existing techniques and distinguish the contributions of our technique those of existing ones. We proceed with an overview of sampling-based methods for probabilistic model checking.

3.1. Sampling-Based Methods for Probabilistic Model Checking

The verification problem of whether a DTMC satisfies a probabilistic temporal property expressed in PCTL (Section 2) can be reduced to the problem of solving a set of linear equations over a set of variables, where each variable corresponds to the probability that a state in the DTMC satisfies the given temporal (path) property. As previously mentioned, this method of verification by solving linear equations numerically is referred to as the numerical method for probabilistic model checking [Hansson and Jonsson 1994; Bianco and de Alfaro 1995; Courcoubetis and Yannakakis 1995; Aziz et al. 2000; Baier et al. 2003]. However, as in traditional model checking, the numerical method suffers from state-space explosion, because it requires complete knowledge of the model state space. Furthermore, it is not possible to apply numerical methods in situations where the model transition structure is not available but, rather, only simulation runs of the model can be obtained as needed.

To address the problem of state-space explosion and avoid the necessity of prior complete knowledge of the model transition structure, sampling-based methods have been proposed and developed. The central theme of these methods is to infer from sample simulations whether the probabilistic property is satisfied by the model, while controlling the error in the inference using statistical bounds. In this context, there are two related inferencing methods: in one, the inference involves *estimating* the probability with which a state satisfies a property, while in the other, the inference involves *verifying* (accepting or, more precisely, rejecting) whether the probability that a state satisfies a property is greater (or less) than a prespecified value. The former computes the estimate within certain confidence bounds (defined by error limits), while the latter employs hypothesis testing with an indifference width (interval where the null hypothesis and its alternate remain undecided). Both methods aim to obtain results such that the error in the result can be controlled. The methods use sample simulations associated with to random variables $X_1, X_2, \ldots$ with outcomes of 1 or 0, depending on whether the simulation satisfies the path property or not; X_i corresponds to the outcome of the ith simulation. The proportion

$$\hat{p} = p_N = \frac{\sum_{i=1}^{N} X_i}{N} \tag{2}$$

is used in both the statistical estimation and the hypothesis testing methods.

3.1.1. Confidence-Interval-Based Statistical Estimation. In this method, given a state s in a DTMC and a path property ψ, an appropriate number of samples (say N) are obtained starting from s, and $\hat{p}$ (Equation (2)) is used as the estimate of the probability p with which paths from s satisfy ψ. The estimation method aims to bound the error in the estimate as $Pr(|\hat{p} - p| > \epsilon) < \delta$. Here, ϵ is the error bound of the estimate and δ is the probability of error in the estimate, that is, the probability that the estimate is at least ϵ distance away from p. δ is also referred to as the confidence parameter of the estimate. The interval $[p-\epsilon, p+\epsilon]$ is referred to as the confidence interval. Given ϵ and δ, one can use the Chernoff-Hoeffding inequality [Hoeffding 1963] to precisely compute the lower bound of the number of samples (N) for probabilistic model checking. Smaller values for ϵ and δ result in larger values of this lower bound. Our technique in this article and that at Herault et al. [2004] are based on this method (details are discussed in Section 4).

3.1.2. Hypothesis-Testing-Based Statistical Estimation. In contrast to the previous approach, the hypothesis-testing-based method does not estimate the probability with which s satisfies ψ; instead, it tests whether the probability that s satisfies ψ is $\bowtie r$, where the probabilistic property under consideration is $\mathtt{P}_{\bowtie r}(\psi)$. For the discussion here, assume that $\bowtie$ is $\geq$. In this setting, the null hypothesis $H_0 : p \geq r$ and its alternate $H_1 : p < r$ are considered. Using an appropriate number N of sample simulations starting from s, the proportion of paths $\hat{p}$ (Equation (2)) that satisfy ψ, is computed. The test rejects the null hypothesis if the proportion is less than r and does not reject the null hypothesis if the proportion is greater than or equal to r. There are two types of errors considered in a testing procedure. Type I error (false negative error) is the probability that the null hypothesis is rejected when, in reality, it holds. Type II error (false positive error) is the probability that the null hypothesis is not rejected when, in reality, it does not hold. The maximal bounds on these probabilities are typically denoted by α and β, that is,

$$Pr[\text{Type I error}] = Pr[\text{reject } H_0 \mid H_0 \text{ holds }] \leq \alpha,$$
$$Pr[\text{Type II error}] = Pr[\text{do not reject } H_0 \mid H_0 \text{ does not hold }] \leq \beta.$$

From the preceding, it can be shown that $Pr[\text{do not reject } H_0 \mid H_0 \text{ holds}] \geq 1 - \alpha$. This implies that the test can correctly distinguish between the two cases, that is, p is equal to r and p is smaller than r by an infinitesimally small amount, which in turn, requires that the samples cover the entire model [Younes and Simmons 2002]. This is an unrealistic assumption. Therefore, a new testing procedure is developed, the result of which implies the results of the original test. The new test uses the hypothesis $H_0' : p \geq r + \xi$ and the alternate $H_1' : p \leq r - \xi$. Note that if the new test fails to reject H_0', then it fails to reject H_0. Proceeding as before, let

$$Pr[\text{reject } H_0' \mid H_0' \text{ holds }] \leq \alpha \text{ and } Pr[\text{do not reject } H_0' \mid H_0' \text{ does not hold }] \leq \beta.$$

Therefore, $Pr[\text{do not reject } H_0 \mid H_0' \text{ holds }] \geq 1 - \alpha$, and $Pr[\text{do not reject } H_0 \mid H_1' \text{ holds }] \leq \beta$. The test is not applicable in the event p falls in $[r - \xi, r + \xi]$. This region is referred to as the *indifference region*, and 2ξ is called the indifference width. Finally, the appropriate sample size N is computed from α, and ξ using well-known results on binomial distribution with parameters p and N. Several methods [Younes and Simmons 2002; Younes et al. 2006] for probabilistic model checking are based on this method of hypothesis testing.

Note that in both preceding methods, it is necessary to decide whether a simulation path satisfies or does not satisfy the given property ψ. For an unbounded path property $\psi = \varphi_1 \mathtt{U} \varphi_2$, it is not possible to know a priori the length of the simulation that is necessary to decide whether or not a given simulation satisfies ψ. That is, while the preceding method provides the lower bound on the sample size necessary to obtain results within the prespecified error bounds, the problem of identifying the necessary simulation length still remains to be addressed. In the following section, we discuss the existing work in this context.

3.2. Survey of Sampling-Based Methods for Probabilistic Model Checking

The statistical method based on Monte Carlo simulation and sequential hypothesis testing [Wald 1945] for verifying time-bounded until CSL properties in CTMC was developed by Younes and Simmons [2002]. Similar techniques are proposed and discussed in Younes et al. [2006]. Younes et al. note that their method can handle unbounded until properties $\varphi_1 \mathtt{U} \varphi_2$ only when any path in the model either reaches a deadlocked state or a state satisfying $\neg\varphi_1 \vee \varphi_2$.

Herault et al. [2004] have proposed a method based on Monte Carlo simulation for verifying a subset of LTL formulas, namely the EPF (Essentially Positive Fragment) in DTMC. Their technique checks whether the probability that a state satisfies an unbounded path property ψ is greater than or equal to r. This is performed using the null hypothesis $H_0 : p \geq r$ against the alternate $H_1 : p < r$. The technique relies on estimating p (Section 3.1.1) within a prespecified error bound and uses the Chernoff-Hoeffding inequality [Hoeffding 1963] to obtain the appropriate sample size. However, it fails to completely control the error in the procedure. The sample path length used in the procedure has a prespecified upper bound. If a simulation reaches that bound and fails to infer a decided result (i.e., whether the path satisfies the given path property or not), the technique assumes that the simulation, if allowed to proceed, eventually will not satisfy the path property under consideration. This assumption allows the method to control Type II error within a prespecified limit. However, as Herault et al. [2004] state, proposed technique cannot determine the appropriate upper bound on the simulation path length to control the number of undecided simulations. As such, the method loses control of Type I error (the error that the null hypothesis H_0 holds but the test rejects it).

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 that of 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 the simulation run of each sample will have a decided result; if this bound is too small, PRISM fails to compute any result and requests that the user rerun the experiment with a new bound.

In contrast to the technique proposed in Herault et al. [2004], Sen et al. [2005] introduce a new model-checking algorithm based on hypothesis testing that can control both Type I and Type II errors. The technique involves a basis procedure which verifies $\mathsf{P}(s, \varphi_1 \mathtt{U} \varphi_2) > 0$. The central theme of this basis procedure is that simulations are obtained from a modification of the original model, where the modification incorporates new transitions from any state to a deadlocked state via a prespecified stopping probability. This allows every simulation to eventually terminate with a decided result for $\mathsf{P}(s, \varphi_1 \mathtt{U} \varphi_2)$. There are, however, two important drawbacks of the basis procedure.

First, one of the user-specified parameters necessary for the validity of the procedure to verify $\mathsf{P}(s, \varphi_1 \ \mathtt{U} \ \varphi_2) > 0$ depends directly on the actual probability $\mathsf{P}(s, \varphi_1 \ \mathtt{U} \ \varphi_2)$. Such information is impossible to obtain a priori, and as such, it is impossible to choose the parameter value appropriately in order to ensure the correctness of the technique. Second, the basis procedure is not valid in general for models that contain loops. Younes and Simmons [2006] point out this flaw in the validity of the method; we have also discussed the cause for this invalidity in He et al. [2010], in which we propose an alternative to the basis procedure that is valid for any model. The alternate technique requires prior knowledge of the total number of states in the model and its maximum branching factor. The technique is similar to the one proposed by Grosu and Smolka [2005], which deals with sampling-based model checking of nonprobabilistic models against linear temporal logic (LTL) properties. In contrast to that of Grosu and Smolka [2005], the technique proposed in this article does not depend on the state space of the model and can model check probabilistic properties of probabilistic models.

Zapreev [2008] proposes a hypothesis-testing-based statistical technique for verifying unbounded until properties that also uses estimations based on confidence intervals (see Section 9.3 for details). However, this technique requires knowledge of the model structure (to identify bottom strongly connected components in the model) and relies on a user-specified sample size and/or sample path lengths. The tool MRMC [MRMC 2010] is based on Zapreev [2008]. Katoen and Zapreev [2009] compare MRMC with tools Ymer Younes [2005] (based on Younes and Simmons [2002]) and VESTA (based on [Sen et al. 2005]) and show that both Ymer and VESTA use less space (constant memory) than MRMC, while the verification times of MRMC are mostly several factors (up to ten) smaller than those of Ymer and VESTA. However, the performance of MRMC rapidly decreases with growth in model size. A similar and more detailed comparison is presented by Jansen et al. [2007].

Rabih and Pekergin [2009] propose a new statistical approach for checking both steady-state and unbounded until properties based on perfect simulation. Perfect simulation has been proven a valid technique for generating samples according to the stationary distribution of the target DTMC model. An important constraint that is imposed on the target DTMC is that it must be ergodic (i.e., irreducible and aperiodic). In other words, if the DTMC model is not guaranteed to be ergodic, then the proposed method cannot be applied correctly. Therefore, it is necessary first to verify that the DTMC model under consideration is ergodic before application of the method. This computation has a space complexity polynomial to the model size and, therefore, can result in space-space explosion. The assumption that the model is ergodic also restricts the application of the proposed method if such verification of ergodicity is not possible (due to unavailability of knowledge of the model transition structure) or if such verification fails.

In this article, we propose a technique which can be viewed as a natural extension of the technique proposed by Herault et al. [2004] and the algorithm used in PRISM. It uses two phases: 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. To the best of our knowledge, this is the first technique that can estimate the probability of satisfying unbounded until properties in probabilistic DTMC and CTMC models using statistical sampling, without any prior knowledge of the model structure to appropriately control the error in estimate.

It should be noted that any of the existing techniques based on either confidence interval estimation or hypothesis testing could be deployed in the second phase of our technique, as the property under consideration is bounded appropriately by k_0, which is computed automatically in the first phase.

4. TWO-PHASE APPROXIMATE PROBABILISTIC MODEL CHECKING

The objective of our work is to reduce unbounded until properties 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 (*Phase I*), and (b) a bound on the number of simulation paths, each of length k_0 (*Phase II*), such that a statistical-sampling-based verification result of the bounded until property approximately coincides with that of the unbounded until property, within a prespecified error limit.

4.1. Rationale

The paths belonging to the semantics of $\varphi_1\ \mathtt{U}\ \varphi_2$ (Section 2.1) 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, that is, the semantics of $\varphi_1\ \mathtt{U}\ \varphi_2$ can be written as

$$\pi \models \varphi_1\ \mathtt{U}\ \varphi_2 \Leftrightarrow \forall k : \begin{bmatrix} \exists 0 \leq i \leq k : \pi[i] \models \varphi_2 \ \wedge\ \forall j < i : \pi[j] \models \varphi_1 \\ \bigvee \\ \exists i > k : \pi[i] \models \varphi_2 \ \wedge\ \forall j < i : \pi[j] \models \varphi_1 \wedge \neg\varphi_2 \end{bmatrix}$$

$$\Leftrightarrow \pi \models \forall k : [(\varphi_1\ \mathtt{U}^{\leq k}\ \varphi_2)\ \vee\ (\varphi_1\ \mathtt{U}^{>k}\ \varphi_2)].$$

Note that we have defined $\pi \models \varphi_1\ \mathtt{U}^{>k}\ \varphi_2$ if and only if the first state $\pi[i]$ that satisfies φ_2 appears in π after at least $k+1$ steps, and φ_1 is satisfied in all states before $\pi[i]$. Since $(\varphi_1\ \mathtt{U}^{\leq k}\ \varphi_2)\ \wedge\ (\varphi_1\ \mathtt{U}^{>k}\ \varphi_2) = ff$, by law of total probability,

$$\mathsf{P}(s, \varphi_1\ \mathtt{U}\ \varphi_2) = \mathsf{P}(s, \varphi_1\ \mathtt{U}^{\leq k}\ \varphi_2)\ +\ \mathsf{P}(s, \varphi_1\ \mathtt{U}^{>k}\ \varphi_2). \tag{3}$$

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

$$0 \leq \mathsf{P}(s, \varphi_1\ \mathtt{U}\ \varphi_2) - \mathsf{P}(s, \varphi_1\ \mathtt{U}^{\leq k}\ \varphi_2)\ =\ \mathsf{P}(s, \varphi_1\ \mathtt{U}^{>k}\ \varphi_2). \tag{4}$$

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

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

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

$$\mathsf{P}(s, \varphi_1\ \mathtt{U}^{>k}\ \varphi_2) \leq \mathsf{P}(s, \neg\psi_k) = 1 - \mathsf{P}(s, \psi_k). \tag{5}$$

From Equations (4) and (5), for any $k \geq 1$, we obtain

$$0 \leq \mathsf{P}(s, \varphi_1\ \mathtt{U}\ \varphi_2) - \mathsf{P}(s, \varphi_1\ \mathtt{U}^{\leq k}\ \varphi_2) \leq 1 - \mathsf{P}(s, \psi_k). \tag{6}$$

Our objective is to select a k_0 such that for any given ϵ_0,

$$\mathsf{P}(s, \psi_{k_0}) \geq 1 - \epsilon_0. \tag{7}$$

In that case,

$$0\ \leq\ \mathsf{P}(s, \varphi_1\ \mathtt{U}\ \varphi_2) - \mathsf{P}(s, \varphi_1\ \mathtt{U}^{\leq k_0}\ \varphi_2)\ \leq\ 1 - \mathsf{P}(s, \psi_{k_0}) \leq \epsilon_0. \tag{8}$$

In other words, by choosing an appropriate k_0, the probability of satisfying the unbounded path property $\varphi_1\ \mathtt{U}\ \varphi_2$ can be made close (within an error margin of ϵ_0, for any arbitrarily small choice of ϵ_0) to the probability of satisfying the bounded path property $\varphi_1 \; \mathtt{U}^{\leq k_0} \; \varphi_2$.

4.2. U2B: Two-Phase Approximate Model Checking for DTMC

The discussion in the previous section (specifically, Equations (7) and (8)) motivates our two-phase method. In the first phase, we determine a suitable k_0, and in the second phase, we estimate $\mathsf{P}(s, \varphi_1 \; \mathtt{U}^{\leq k_0} \; \varphi_2)$. Finally, we use this estimate of $\mathsf{P}(s, \varphi_1 \; \mathtt{U}^{\leq k_0} \; \varphi_2)$ as the estimate of $\mathsf{P}(s, \varphi_1 \; \mathtt{U} \; \varphi_2)$. Our method utilizes confidence-interval-based statistical estimation (see Section 3.1.1) in both phases.

In Phase I, $\mathsf{P}(s, \psi_k)$ is estimated for different values of k using N_1 Monte Carlo simulation paths, in a method similar to the 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 (7).

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

The two phases for computing k_0 and then computing $\mathsf{P}(s, \varphi_1 \; \mathtt{U}^{\leq k_0} \; \varphi_2)$ are carried out independently, that is, 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 controls the errors in each of the phases and combines them to guarantee the correctness of the final estimate within a prespecified error limit (see, Theorem 4.1). In short, our method $\mathtt{U2B}(M, s, \varphi_1 \; \mathtt{U} \; \varphi_2, \epsilon, \delta)$ takes as input the DTMC model M, the state s, the until property under consideration, the error margin ϵ, and the confidence parameter δ, and returns the estimate of $\mathsf{P}(s, \varphi_1 \; \mathtt{U} \; \varphi_2)$ within the ϵ bound, with a high degree of certainty (at least $1 - \delta$). The steps of our method are summarized as follows.

Main Steps. $\mathtt{U2B}(M, s, \varphi_1 \; \mathtt{U} \; \varphi_2, \epsilon, \delta)$.

(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 $i = 1, \ldots, N_1$, let $X_i = 1$ if the ith simulation satisfies ψ_k; $X_i = 0$, otherwise.
   (b) Estimate $\mathsf{P}(s, \psi_k)$ as the proportion of the simulation paths satisfying ψ_k, that is,

$$\widehat{\mathsf{P}}(s, \psi_k) = \frac{1}{N_1} \sum_{i=1}^{N_1} X_i. \tag{9}$$

   (c) Verify if Equation (7) is satisfied by the estimate in Equation (9) with the current value of k and $\epsilon_0 = \frac{\epsilon}{3}$. More precisely, if

$$\widehat{\mathsf{P}}(s, \psi_k) \geq 1 - \epsilon_0 = 1 - \frac{\epsilon}{3}, \tag{10}$$

   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 $X_i, i = 1, \ldots, N_1$ as in Step 1(a) using these extended simulation paths and repeat Steps 1(b)–(c).

(2) *Phase II*. Estimating $\mathsf{P}(s, \varphi_1 \; \mathtt{U}^{\leq k_0} \; \varphi_2)$.
   (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 $i = 1, \ldots, N_2$, let $Y_i = 1$ if the ith simulation path satisfies $\varphi_1 \; \mathtt{U}^{\leq k_0} \; \varphi_2$; $Y_i = 0$, otherwise.

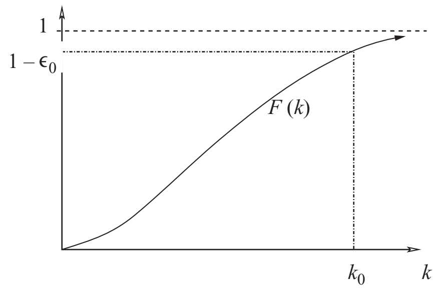

Fig. 2. Choosing k_0 in *Phase I* from $F(k)$.

(b) Estimate $\mathsf{P}(s, \varphi_1 \, \mathtt{U}^{\leq k_0} \, \varphi_2)$ as the proportion of the simulation paths that satisfy $\varphi_1 \, \mathtt{U}^{\leq k_0} \, \varphi_2$, that is,

$$\widehat{\mathsf{P}}(s, \varphi_1 \, \mathtt{U}^{\leq k_0} \, \varphi_2) = \frac{1}{N_2} \sum_{i=1}^{N_2} Y_i. \tag{11}$$

Return $\widehat{\mathsf{P}}(s, \varphi_1 \, \mathtt{U}^{\leq k_0} \, \varphi_2)$ as the estimate for $\mathsf{P}(s, \varphi_1 \, \mathtt{U} \, \varphi_2)$.

4.3. Proof of Correctness

The following theorem states the correctness of our method.

THEOREM 4.1. *Given any precision parameter $\epsilon > 0$ and confidence parameter $\delta > 0$, the estimator $\mathtt{U2B}(M, s, \varphi_1 \, \mathtt{U} \, \varphi_2, \epsilon, \delta)$ with the chosen values of k_0, N_1^*, N_2^* satisfies the following.*

$$Pr(\mid \mathtt{U2B}(M, s, \varphi_1 \, \mathtt{U} \, \varphi_2, \epsilon, \delta) - \mathsf{P}(s, \varphi_1 \, \mathtt{U} \, \varphi_2) \mid > \epsilon) \leq \delta. \tag{12}$$

We begin by discussing auxiliary results in theoretical statistics that will be used in proving this 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 $F(k) = \mathsf{P}(s, \psi_k),\ k \geq 1$. 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 $\varphi_1 \, \mathtt{U} \, \varphi_2$, is *decided* in $\leq k$ steps from the state s. The main challenge in Phase I is that the function $F(\cdot)$ is not typically known. If this function were known, finding a k_0 that satisfies Equation (7) could have been achieved by simply inverting this (nondecreasing) function.

The function $F(\cdot)$ can be thought of as the cumulative distribution function (c.d.f) of a random variable K, where K is the minimum number of transitions required to verify $\varphi_1 \, \mathtt{U} \, \varphi_2$ along a randomly selected simulation path in the given model. In this way, 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 $\widehat{\mathsf{P}}(s, \psi_k)$ (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 for all $k \geq 1$, $\hat{F}_{N_1}(k)$ converges to $F(k)$, as $N_1 \to \infty$ at a suitable rate. For the proof of Theorem 4.1, we need the rate *uniform* in k at which this convergence takes place. This is provided by the celebrated Dvoretzky-Kiefer-Wolfowitz (DKW) inequality (e.g., [Massart 1990]): for each $\epsilon_1 > 0, N_1 \geq 1$, $Pr(\sup_{k \geq 1} |\hat{F}_{N_1}(k) - F(k)| > \epsilon_1) \leq 2e^{-2N_1(\epsilon_1)^2}$. This result, restated in terms of $\mathsf{P}(s, \psi_k) = F(k)$ and $\widehat{\mathsf{P}}(s, \psi_k) = \hat{F}_{N_1}(k)$, for each k, yields the following lemma, which will be needed for our proof of Theorem 4.1.

LEMMA 4.2. *Given any $\epsilon_1 > 0$ and $N_1 \geq 1$,*

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

4.3.2. Phase II: Estimating $P(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2)$. In this phase, we estimate the probability of the bounded until property $\varphi_1 \text{ U}^{\leq k_0} \varphi_2$ in M, with $k_0 \geq 1$, as 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 $\varphi_1 \text{ U}^{\leq k} \varphi_2$ in M. Hence, using the same technique (i.e., using the Chernoff-Hoeffding bound), we get for each $\epsilon_2 > 0$,

$$Pr\left(\mid \widehat{\mathsf{P}}(s, \varphi_1 \text{ U}^{\leq k} \varphi_2) - \mathsf{P}(s, \varphi_1 \text{ U}^{\leq k} \varphi_2) \mid > \epsilon_2\right) \leq 2e^{-N_2(\epsilon_2)^2/4}.$$

Now, since this 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 preceding statement must be true unconditionally as well, for $k = k_0$. Summarizing this discussion, we have the following.

LEMMA 4.3. *Given any $\epsilon_2 > 0$ and $N_2 \geq 1$, for k_0 given by Phase I of U2B,*

$$Pr\left(\mid \widehat{P}(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2) - P(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2) \mid > \epsilon_2\right) \leq 2e^{-N_2(\epsilon_2)^2/4}.$$

Now we use the results of Lemmas 4.2 and 4.3 to complete the proof of Theorem 4.1.

PROOF OF THEOREM 4.1. The triangle inequality (after adding and subtracting suitable terms) yields the following.

$$\begin{aligned}&\mid \mathtt{U2B}(M, s, \varphi_1 \text{ U } \varphi_2, \epsilon, \delta) - \mathsf{P}(s, \varphi_1 \text{ U } \varphi_2) \mid = \mid \widehat{\mathsf{P}}(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2) - \mathsf{P}(s, \varphi_1 \text{ U } \varphi_2) \mid \\ &\leq \mid \widehat{\mathsf{P}}(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2) - \mathsf{P}(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2) \mid + \mid \mathsf{P}(s, \varphi_1 \text{ U } \varphi_2) - \mathsf{P}(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2) \mid .\end{aligned}$$

Recall from Equation (10) that we have $1-\widehat{\mathsf{P}}(s, \psi_k) \leq \epsilon/3$. Hence, using Equation (6) and the triangle inequality, we get the following bound on the last term in Equation (13).

$$\begin{aligned}\mid \mathsf{P}(s, \varphi_1 \text{ U } \varphi_2) - \mathsf{P}(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2) \mid &\leq (1 - \mathsf{P}(s, \psi_{k_0})) \\ &\leq (1 - \widehat{\mathsf{P}}(s, \psi_{k_0})) + \mid \widehat{\mathsf{P}}(s, \psi_{k_0}) - \mathsf{P}(s, \psi_{k_0}) \mid \\ &\leq \frac{\epsilon}{3} + \sup_{k\geq 1} \mid \widehat{\mathsf{P}}(s, \psi_k) - \mathsf{P}(s, \psi_k) \mid . \qquad (13)\end{aligned}$$

Combining Equation (13) with the result of applying the preceding triangle inequality, we get the following bound.

$$\begin{aligned}&\mid \mathtt{U2B}(M, s, \varphi_1 \text{ U } \varphi_2, \epsilon, \delta) - \mathsf{P}(s, \varphi_1 \text{ U } \varphi_2) \mid \\ &\quad \leq \mid \widehat{\mathsf{P}}(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2) - \mathsf{P}(s, \varphi_1 \text{ U}^{\leq k_0} \varphi_2) \mid + \tfrac{\epsilon}{3} + \sup_{k\geq 1} \mid \widehat{\mathsf{P}}(s, \psi_k) - \mathsf{P}(s, \psi_k) \mid\end{aligned}$$

Hence, the fact that the left side of the preceding inequality is greater than ϵ (see Equation (12) in Theorem 4.1) implies that at least one of the terms on the right side is greater than $\epsilon/3$. Therefore, we obtain the following.

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

ALGORITHM 1: Naive Implementation of Phase I

```
1: procedure IDENTIFYK0($N_1$, $\epsilon_0$, $\psi$)
                    ▷ $N_1$: Total number of samples for Phase I (see U2B Step 1(a))
                    ▷ $\epsilon_0$ closeness to 1 (see U2B Step 1(c))
                    ▷ $\psi := (\varphi_1$ U $\varphi_2) \vee (\neg\varphi_2$ U $\neg\varphi_1 \wedge \neg\varphi_2)$
2:     $k := 0$;
3:     workingSet := add $N_1$ copies of start state;          ▷ path length $k = 0$
4:     trueCount := 0;          ▷ Initial number of paths that satisfy $\psi_k$
5:     while (1) do
6:         for each state ∈ workingSet that satisfies $\neg\varphi_1 \vee \varphi_2$ do          ▷ i.e., path satisfies $\psi_k$
7:             remove state from workingSet;
8:             trueCount++;
9:         end for
10:        if ($N_1$(1 - $\epsilon_0$) ≤ trueCount) then
11:            return $k$;          ▷ trueCount $\geq N_1(1-\epsilon_0)$, i.e., $\widehat{\mathsf{P}}(s, \psi_k) = \frac{\text{truecount}}{N_1} \geq 1 - \epsilon_0$
12:        end if
13:        $k$++;
14:        replace current states in workingSet with randomly obtained next states;
15:    end while
16: end procedure
```

The last inequality follows from the bounds in Lemmas 4.2 and 4.3 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, $N_1 \geq \frac{1}{2\epsilon_1^2}log(\frac{4}{\delta}) \geq \frac{9}{2\epsilon^2}log(\frac{4}{\delta})$) and $2e^{-N_2(\epsilon_2)^2/4} \leq \delta/2$ (i.e., $N_2 \geq \frac{4}{\epsilon_2^2}log(\frac{4}{\delta}) \geq \frac{36}{\epsilon^2}log(\frac{4}{\delta})$). This completes the proof of Theorem 4.1. □

Remark 4.4. Observe that there are three error bounds that are derived from ϵ, each of which is assigned to $\frac{\epsilon}{3}$: ϵ_0 (from Equation (10)), ϵ_1 (from Lemma 4.2), and ϵ_2 (from Lemma 4.3). While ϵ_0 is the measure of closeness of $\widehat{\mathsf{P}}(s, \psi_k)$ to 1, ϵ_1 and ϵ_2 capture the closeness of the estimated and true probabilities in each phase. In other words, a smaller ϵ_0 will lead to a larger value for k_0 (sample path length), while smaller ϵ_1 and ϵ_2 values will result in larger values for N_1 and N_2, that is, the sample sizes in each phase. The proof of our theorem holds as long as $\epsilon_0 + \epsilon_1 + \epsilon_2 = \epsilon$. These values can be fine tuned experimentally.

5. EFFICIENT AND PRACTICAL REALIZATION OF THE 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 an optimized variation. Recall that Phase II deals with estimating $\mathsf{P}(s, \varphi_1 \mathtt{U}^{\leq k_0} \varphi_2)$, which can be immediately computed using the existing statistical method present in PRISM; as such, we do not provide any additional optimization for Phase II.

5.1. Naive Implementation of Phase I

The procedure IDENTIFYK0 (see Algorithm 1) describes a naive realization of Phase I in U2B. workingSet contains the set of current states (the last states of the simulation paths) that are examined in Phase I. It is initialized with N_1 copies of the start state of the model; these are the current states in N_1 paths of length 0 (line 3). The variable trueCount, initialized to 0 (line 4), captures the number of paths in workingSet that satisfy ψ_k.

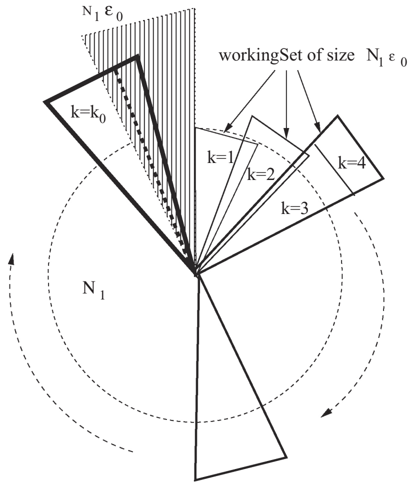

Fig. 3. Optimized realization of Phase I of U2B, workingSet of size $N_1\epsilon_0$: sliding window strategy.

The states in workingSet are verified to check whether ψ_k is satisfied; if satisfied, the states are removed from workingSet and trueCount is incremented (lines 6–9). Observe that trueCount captures the number of paths of length k that satisfy ψ_k. Therefore, trueCount$/N_1$ computes $\widehat{\mathsf{P}}(s, \psi_k)$ (Equation (9)). The loop (line 5) is terminated if $N_1(1-\epsilon_0) \leq$ trueCount (lines 10, 11), that is, if the proportion of paths in the sample of size N_1 that satisfy ψ_k is less than $1-\epsilon_0$.

If the terminating condition is not satisfied, k is incremented, and workingSet is updated by obtaining the next states of the paths in the working set by random selection based on the probability distribution in the model. On termination of the iteration, that is, when trueCount$/N_1 \geq 1-\epsilon_0$, the value k is returned. This concludes Phase I of the U2B and initiates the invocation of Phase II.

Remark 5.1. Algorithm 1 terminates and returns k if and only if $\widehat{\mathsf{P}}(s, \psi_k) \geq 1-\epsilon_0$. This statement holds directly from the facts that $\widehat{\mathsf{P}}(s, \psi_k)$ is equal to the proportion of paths in a sample of size N_1 that satisfy ψ_k, and that Algorithm 1 considers a workingSet of N_1 paths and terminates only when trueCount $\geq N_1(1-\epsilon_0)$.

5.2. Optimization: Reducing the workingSet in Phase I

Algorithm 1 initializes workingSet to a set (of size N_1) of paths of length 0 and maintains a subset of paths that do not satisfy ψ_k until the loop-terminating condition at line 5 is satisfied. This can be expensive in terms of space usage, as N_1 can be large.

An alternative realization of Phase I has been developed in which only a small subset of the sample (of total size N_1) is considered in workingSet. This is achieved from the observation that upon termination of Phase I, the number of paths that do not satisfy ψ_k can be at most $N_1\epsilon_0 - 1$ in the sample of size N_1 (as trueCount$/N_1$ is required to be $\geq 1-\epsilon_0$). Therefore, it is sufficient to consider only $N_1\epsilon_0$ paths in workingSet, at any point in time. Whenever some paths in workingSet satisfy ψ_k, they are replaced with a new set of paths such that the size of workingSet is maintained at $N_1\epsilon_0$. In short, a sliding window workingSet is considered until the total number of paths that satisfy ψ_k is $\geq N_1(1-\epsilon_0)$ (as required in Phase I). This reduces the space usage, as the number of paths stored in workingSet at any time is fixed to $N_1\epsilon_0$, a small proportion 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

ALGORITHM 2: Optimized Implementation of Phase I

```
1: procedure IDENTIFYK0_OPT_1($N_1$, $\epsilon_o$, $\psi$)
                          ▷ $N_1$: Total number of samples for Phase I (see U2B Step 1(a))
                          ▷ $\epsilon_0$: Error bound for Phase I (see U2B Step 1(c))
                          ▷ $\psi := (\varphi_1$ U $\varphi_2) \vee (\neg\varphi_2$ U $\neg\varphi_1 \wedge \neg\varphi_2)$
2:     $k := 0$;                                          ▷ Initial length of paths
3:     $\boxed{\text{workingSet := add } N_1\epsilon_0 \text{ copies of start state;}}$        ▷ paths of length $k = 0$
4:     trueCount := 0;                   ▷ Initial number of paths that satisfy $\psi_k$
5:     while True do
6:         for each state ∈ workingSet that satisfies $\neg\varphi_1 \vee \varphi_2$ do     ▷ i.e., path satisfies $\psi_k$
7:             remove state from workingSet;
8:             trueCount++;
9:         end for
10:        if ($N_1(1 - \epsilon_0) \leq$ trueCount) then
11:            return $k$;      ▷ trueCount $\geq N_1(1-\epsilon_0)$, i.e., $\widehat{\mathsf{P}}(s, \psi_k) = \frac{\text{truecount}}{N_1} \geq 1 - \epsilon_0$
12:        end if
13:        $k$++;
14:        replace current states in workingSet with randomly obtained next states;
15:        $\boxed{\text{run } (N_1\epsilon_0 - |\text{workingSet}|) \text{ random simulations for } k \text{ steps from start state;}}$
16:        $\boxed{\text{add last states to workingSet;}}$
17:    end while
18: end procedure
```

sector denotes the length of paths in the corresponding working set. In the figure, initially a working set containing paths of length 1 is considered. The property ψ_k is satisfied in some of these paths, and as a result, those paths are removed and new paths of length 1 are generated and added to the working set. All paths in the working set are then 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 ψ_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., the path length k becomes 4). This process is repeated until a sector (working set) is obtained such that (a) it overlaps with the shaded sector and (b) contains a subsector of paths (denoted by dotted lines) that satisfy ψ_k and that also overlap with the shaded sector. This implies that the proportion of paths that satisfy ψ_k is $\geq 1 - \epsilon_0$. Algorithm 2 presents the listing of this optimized implementation.

The primary differences between Algorithms 1 and 2 are at lines 3, 15–16 of Algorithm 2. Instead of initializing `workingSet` with N_1 copies of start states (as in Algorithm 1), it is initialized with $N_1\epsilon_0$ copies of start states. Furthermore, if some states are removed from `workingSet` (line 7), it is replenished with the last states of newly extended paths of appropriate length, such that the total number of states in `workingSet` remains equal to $N_1\epsilon_0$ (lines 15, 16).

PROPOSITION 5.2. *Algorithm 2 terminates and returns k if and only if* $\widehat{\mathsf{P}}(s, \psi_k) \geq 1 - \epsilon_0$.

Algorithm 2 terminates if $N_1(1 - \epsilon_0) \leq$ `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 $\widehat{\mathsf{P}}(s, \psi_k) \geq 1 - \epsilon_0$. This proposition holds, as $\widehat{\mathsf{P}}(s, \psi_k)$ is the proportion of paths (i.e., `trueCount`) in a sample

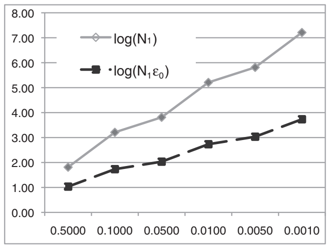

Fig. 4. Logarithms of N_1 and $N_1\epsilon_0$ against ϵ (x-axis).

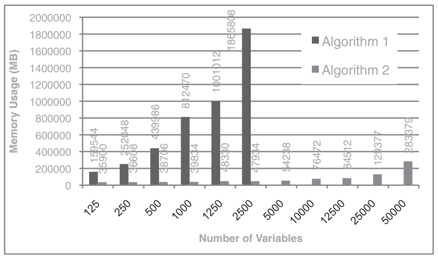

Fig. 5. Memory usage by Algorithm 1 vs. Algorithm 2 for variations of a simple queue model.

of size N_1 that satisfy ψ_k, and the terminating condition of Algorithm 2 ensures that $\texttt{trueCount}/N_1 \geq 1 - \epsilon_0$.

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 4.4 in Section 4.3); however, any other choice of ϵ_0 and ϵ_1 as a function of ϵ would suffice for this discussion. Figure 4 compares the increases (in log scale) in the value of N_1 and $N_1\epsilon_0$ with the decrease in ϵ 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 ϵ 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 the precision of the U2B method is increased, Algorithm 2 uses a considerably smaller working set (and in turn, less space), compared to that used by Algorithm 1; the greater the precision, the greater the benefit in terms of space usage in using Algorithm 2 over Algorithm 1.

The space-saving benefit increases with the increase in the number of variables describing the model (i.e., each model state). We have experimented with different variations of a simple queue model Jennings et al. [2010], where each variation contains a 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 ≥ 2500, Algorithm 1 runs out of memory and terminates without computing any result.

6. DEALING WITH NONTERMINATION IN U2B

Recall that $F(k)$ is the probability that φ_1 U φ_2 is decided in k steps from a state s. In Section 4.3.1, we introduced the notion of the decided graph (D-Graph; Figure 2), presenting the valuations of $F(\cdot)$ for different valuations of k. In this section, we discuss the condition on the valuation of $F(\cdot)$ (and the corresponding shape of the D-Graph) under which the U2B method fails to terminate and propose a heuristic-based alternate method to ensure termination.

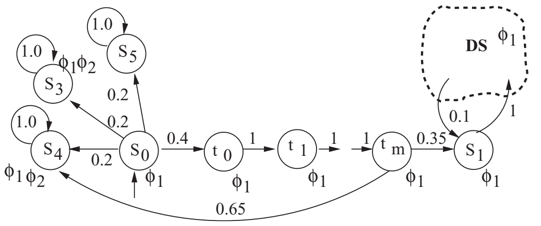

Fig. 6. A modification of the illustrative example of (Figure 1).

Our argument for the applicability of the U2B method requires that as $k \to \infty$, the limit of $F(k) = \mathsf{P}(s, \psi_k) \geq 1 - \epsilon_0$ (see Equation (7)). That is,

$$\lim_{k\to\infty} \mathsf{P}(s, \psi_k) \geq 1 - \epsilon_0. \tag{15}$$

When the limit is 1, Equation (15) is satisfied for any ϵ_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 (see Figure 7), our method works (i.e., Phase I computation terminates) for any ϵ_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 preceding requirement is satisfied. PRISM's statistical method is equivalent to choosing $\epsilon_0 = 0$ in our method and, hence, will fail to provide a 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 $\mathsf{P}(s, \neg(tt \mathtt{\ U\ } (\varphi_2 \vee \neg\varphi_1))) > 0$, which happens if and only if there exists a path to a strongly connected component in the model where every state satisfies $(\varphi_1 \wedge \neg\varphi_2)$. If the probability of such a path (or paths) is greater than ϵ_0, then the Phase I computation will fail to terminate. In such a case, any statistical verification method that does not analyze model transition structure may not be able to compute any result. However, this feature of not analyzing the model allows the application of statistical verification methods (including ours) for the purpose of estimating the probability of properties in systems for which preanalysis of the system model is not possible, due to prohibitively large state space, or for systems for which only sample simulations of the system can be generated (as needed).

Consider the DTMC model in Figure 6. The start state of the model is s_0, and each state is annotated with the property (φ_1, φ_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 (an exact description of the model is available at Jennings et al. [2010]). All states in DS satisfy φ_1. The objective is to compute $\mathsf{P}(s_0, \varphi_1 \mathtt{\ U\ } \varphi_2)$. There are three paths—$s_0, s_3, s_3, \ldots$; $s_0, s_4, s_4, \ldots$; and $s_0, t_0, \ldots, t_m, s_4, \ldots$—that satisfy $\varphi_1 \mathtt{\ U\ } \varphi_2$; therefore, $\mathsf{P}(s, \varphi_1 \mathtt{\ U\ } \varphi_2) = 0.2 + 0.2 + 0.4 \times 0.65 = 0.66$. 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, $\lim_{k\to\infty} \mathsf{P}(s_0, \psi_k) = 0.86$. As PRISM's statistical method requires this limit to be 1, it fails to compute the estimate of $\mathsf{P}(s_0, \varphi_1 \mathtt{\ U\ } \varphi_2)$. Similarly, our proposed method U2B will fail to terminate when $\epsilon_0 < 0.14$. More precisely, Phase I of the U2B method will fail to terminate. In the following, we present a strategy and a corresponding heuristic to address this problem of nontermination.

6.1. Rationale

From the semantics of PCTL (Section 2.2), the probability of satisfying $\varphi_1 \texttt{ U } \varphi_2$ can be expressed in terms of satisfying $\varphi_1 \texttt{ U}^{\leq k} \varphi_2$, as follows.

$$\lim_{k\to\infty} \mathsf{P}(s, \varphi_1 \texttt{ U}^{\leq k} \varphi_2) = \mathsf{P}(s, \varphi_1 \texttt{ U } \varphi_2), \text{ that is, } \mathsf{P}(s, \varphi_1 \texttt{ U}^{\leq \infty} \varphi_2) = \mathsf{P}(s, \varphi_1 \texttt{ U } \varphi_2). \tag{16}$$

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

$$\forall \Delta_k \geq 1 : \widehat{\mathsf{P}}(s, \varphi_1 \texttt{ U}^{\leq k_*} \varphi_2) = \widehat{\mathsf{P}}(s, \varphi_1 \texttt{ U}^{\leq k_*+\Delta_k} \varphi_2). \tag{17}$$

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

THEOREM 6.1. *Given any precision parameter $\epsilon_1 > 0$ and confidence parameter $\delta_1 > 0$,*

$$Pr\left(\mid \widehat{P}(s, \varphi_1\ U^{\leq k_*} \varphi_2) - P(s, \varphi_1\ U \varphi_2) \mid > \epsilon_1\right) \leq \delta_1,$$

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

PROOF.

$$\begin{aligned} \mid \widehat{\mathsf{P}}(s, \varphi_1 \texttt{ U}^{\leq k_*} \varphi_2) - \mathsf{P}(s, \varphi_1 \texttt{ U } \varphi_2) \mid &\leq \mid \widehat{\mathsf{P}}(s, \varphi_1 \texttt{ U}^{\leq k_*} \varphi_2) - \mathsf{P}(s, \varphi_1 \texttt{ U}^{\leq\infty} \varphi_2) \mid \\ &\quad + \mid \mathsf{P}(s, \varphi_1 \texttt{ U}^{\leq\infty} \varphi_2) - \mathsf{P}(s, \varphi_1 \texttt{ U } \varphi_2) \mid \\ &\leq \mid \widehat{\mathsf{P}}(s, \varphi_1 \texttt{ U}^{\leq k_*} \varphi_2) - \mathsf{P}(s, \varphi_1 \texttt{ U}^{\leq\infty} \varphi_2) \mid, \end{aligned}$$

using Equation (16).

Therefore,

$$\begin{aligned} \mid \widehat{\mathsf{P}}(s, \varphi_1 \texttt{ U}^{\leq k_*} \varphi_2) - \mathsf{P}(s, \varphi_1 \texttt{ U } \varphi_2) \mid &\leq \mid \widehat{\mathsf{P}}(s, \varphi_1 \texttt{ U}^{\leq k_*} \varphi_2) - \widehat{\mathsf{P}}(s, \varphi_1 \texttt{ U}^{\leq\infty} \varphi_2) \mid \\ &\quad + \mid \widehat{\mathsf{P}}(s, \varphi_1 \texttt{ U}^{\leq\infty} \varphi_2) - \mathsf{P}(s, \varphi_1 \texttt{ U}^{\leq\infty} \varphi_2) \mid \\ &\leq \mid \widehat{\mathsf{P}}(s, \varphi_1 \texttt{ U}^{\leq\infty} \varphi_2) - \mathsf{P}(s, \varphi_1 \texttt{ U}^{\leq\infty} \varphi_2) \mid, \end{aligned} \tag{18}$$

using Equation (17).

From Lemma 4.2, the following holds for any k, $N_1 \geq 1$, and $\epsilon_1 > 0$.

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

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

$$\begin{aligned} &Pr\left(\mid \widehat{\mathsf{P}}(s, \varphi_1 \texttt{ U}^{\leq\infty} \varphi_2) - \mathsf{P}(s, \varphi_1 \texttt{ U}^{\leq\infty} \varphi_2) \mid > \epsilon_1\right) \leq \delta_1, \text{ that is,} \\ &Pr\left(\mid \widehat{\mathsf{P}}(s, \varphi_1 \texttt{ U}^{\leq k_*} \varphi_2) - \mathsf{P}(s, \varphi_1 \texttt{ U } \varphi_2) \mid > \epsilon_1\right) \leq \delta_1, \end{aligned}$$

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 $\varphi_1 \texttt{ U}^{\leq k_*} \varphi_2$ in N_1 samples (i.e., $\widehat{\mathsf{P}}(s, \varphi_1 \texttt{ U}^{\leq k_*} \varphi_2)$) estimates $\mathsf{P}(s, \varphi_1 \texttt{ U } \varphi_2)$ with precision parameter ϵ_1 and confidence parameter δ_1. □

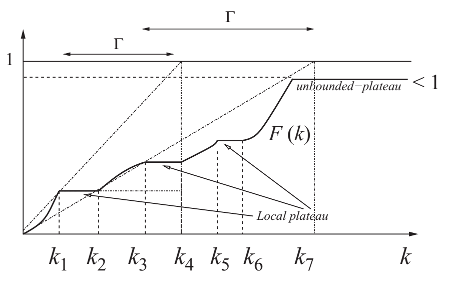

Fig. 7. Plateaus in the D-Graph.

This theorem provides a sound roadmap for estimating $\mathsf{P}(s, \varphi_1 \ \mathtt{U} \ \varphi_2)$, and it does not require that $\lim_{k\to\infty} \mathsf{P}(s, \psi_k) \geq 1 - \epsilon_0$ (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., the resulting $\widehat{\mathsf{P}}(s, \varphi_1 \ \mathtt{U} \ \varphi_2)$ may not be an estimate within a prespecified error bound and confidence parameter) when $\lim_{k\to\infty} \mathsf{P}(s, \psi_k) \not\geq 1 - \epsilon_0$.

6.2. U2B_P: Heuristic for Computing Plateau Detector using the D-Graph

If the valuation of $F(k) = \mathsf{P}(s, \psi_k)$ for a range of values of k (see Figure 7) in a D-Graph remains unaltered, then a plateau in the corresponding valuation of $\mathsf{P}(s, \varphi_1 \ \mathtt{U}^{\leq k} \ \varphi_2)$ 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 $\hat{F}_{N_1}(k) = \widehat{\mathsf{P}}(s, \psi_k)$ (estimate of $F(k)$ from N_1 samples) is detected, starting from $k = \hat{k}_*$. In other words, given Γ—a prespecified 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.

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

The precision of this method of estimation depends on the length Γ of the detected plateau; a longer plateau (larger Γ) is likely to provide a better estimate $\hat{k}_*$. For example, in Figure 7, if Γ is greater than $k_2 - k_1$, $k_3 - k_4$, and $k_6 - k_5$, then $\hat{k}_*$ indeed will be 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 for obtaining a suitable valuation for Γ, 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 to 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 $F(k_x) = 1$.

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

We say that $\Gamma = (k_j + \mathtt{AF})\gamma^{\mathtt{simpoly}}$ where the following hold.

—AF is the additive factor (= 10).
—$\gamma = \mathtt{MINMAX}\big(\mathtt{LB}, \frac{1}{\hat{F}_{N_1}(k_j)} - 1, \mathtt{UB}\big)$.

—simpoly is a user-specified parameter (default value is 1).
—LB $= 2$ and UB $= 5$ are the prespecified lower and upper-bounds of γ.
—$\hat{F}_{N_1}(\cdot)$ 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 arbitrarily set the three parameters AF, UB, and LB. AF is used to ensure that a 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 to control the length of the plateau using the exponent simpoly.

Consider an example illustration in Figure 7. At k_1, the value of Γ is $k_4 - k_1$ (assume that there is no preset LB, UB, AF), that is, 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 k_1, and the corresponding value of $\widehat{\mathsf{P}}(s, \varphi_1\ \mathtt{U}^{\leq k_1}\ \varphi_2)$ is identified as the $\widehat{\mathsf{P}}(s, \varphi_1\ \mathtt{U}\ \varphi_2)$ (see Equation (19)). Similarly at k_3, Γ computed using the preceding formula is $k_7 - k_3$.

We have developed a method, referred to as U2B_P, based upon the preceding heuristic to compute $k_j = \hat{k}_*$. In U2B_P, if a plateau in $\hat{F}_{N_1}(\cdot)$ is detected starting from k_j and if $\hat{F}_{N_1}(k_j) \not> 1 - \epsilon_0$, then $\widehat{\mathsf{P}}(s, \varphi_1\ \mathtt{U}^{\leq k_j}\ \varphi_2)$ is returned as the estimate of $\mathsf{P}(s, \varphi_1\ \mathtt{U}\ \varphi_2)$ (see Theorem 6.1). $\widehat{\mathsf{P}}(s, \varphi_1\ \mathtt{U}^{\leq k_j}\ \varphi_2)$ is the proportion of paths in N_1 samples that satisfy $\varphi_1\ \mathtt{U}^{\leq k_j}\ \varphi_2$. On the other hand, if $\hat{F}_{N_1}(k_j) > 1 - \epsilon_0$, 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 6.2. As U2B_P is based on a heuristic, it cannot provide any correctness guarantees. We give users the option of deploying either U2B (with the 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 willing to wait, the user can terminate the process and deploy U2B_P. We have not deployed U2B_P in any of the experimental results presented in Section 8.

The U2B_P method is realized as follows in Algorithm 3. In addition to keeping track of the proportion of paths that satisfy ψ_k (trueCount, i.e., $\hat{F}_{N_1}(k)$), the algorithm also keeps information regarding the number of paths that satisfy $\varphi_1\ \mathtt{U}^{\leq k}\ \varphi_2$ (satCount in lines 6, 12). The value of k, for which there is some change in the valuation of trueCount, is recorded in lastk (line 14). Γ is computed at line 22 following the preceding description. If trueCount does not change for a large range of values of k (i.e, the difference between the current value of k and lastk is $> \Gamma$), then for the same range, the value of satCount also will 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 $\mathsf{P}(s, \varphi_1\ \mathtt{U}\ \varphi_2)$, along with a warning to the user that a heuristic has been used to satisfy Equation (17) (line 25). The computation can be redone with larger values of Γ, 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 ϵ_0 close to 1. In that case, the algorithm returns k as k_0 (line 17) 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 $\mathsf{P}(s, \varphi_1\ \mathtt{U}\ \varphi_2)$ for this model, because the limit[3] $\lim_{k\to\infty} \mathsf{P}(s, \psi_k) = 0.86 < 1 - \epsilon_0$, where $\epsilon_0 = 0.0025$. That is, $\mathsf{P}(s, \psi_k)$ or $F(k)$ has an

[3]Recall that ψ_k denotes the property stating that $\varphi_1\ \mathtt{U}\ \varphi_2$ is decided in k steps. The $\lim_{k\to\infty} \mathsf{P}(s, \psi_k) \geq 1 - \epsilon_0$ (see Equation (15)) is required for the termination of Algorithm 2.

ALGORITHM 3: Implementation of U2B_P

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

unbounded plateau for some value of k, and the valuation of $\mathsf{P}(s, \psi_k)$ at the plateau is $< 1 - \epsilon_0$.

The U2B_P method as implemented in Algorithm 3, on the other hand, terminates and successfully estimates $\mathsf{P}(s_0, \varphi_1 \mathtt{U} \varphi_2)$. 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 $t_0, \ldots, t_m$ (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 results), and the satCount (number of paths that satisfy $\varphi_1 \mathtt{U} \varphi_2$). The execution starts with $k = 0$ and $|\text{workingSet}| = N_1 = 19{,}171$ ($N_1 = 19{,}171$ for $\epsilon_0 = 0.0025$ and $\delta = 0.01$). The value of trueCount remains at $11,492$ until the value of k becomes 20, that is, $\forall k \in [1, 20] : \hat{F}_{N_1}(k) = \widehat{\mathsf{P}}(s, \psi_k) = 0.59942$. This is because the length of paths considered is ≤ 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

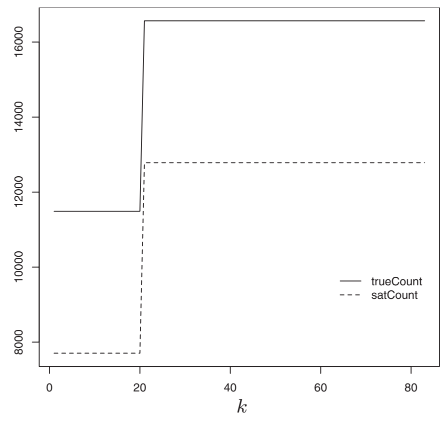

Fig. 8. Execution of Algorithm 3 for the DTMC model in Figure 6: trueCount and satCount vs. k.

to be sufficient, as per the heuristic, because $\forall \texttt{lastk} \in [1, 19] : \texttt{lastk} + \Gamma \geq 23$ (see the computation of Γ 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 long plateau has been detected and, therefore, would terminate. In the current example, that does not happen, as $\hat{F}_{N_1}(20) < \hat{F}_{N_1}(21)$.[4]

The value of trueCount remains at $16,566$ for $k \geq 21$, that is, the last time trueCount value changes is when $k = 21$. In this case, when $\texttt{lastk} = 21$, $\Gamma = 62$. In other words, if the trueCount value does not change for k between 21 and $21 + 62 = 83$, U2B_P terminates. Observe that changes in satCount follow a similar pattern as changes in trueCount; a plateau in the valuation of trueCount (i.e., $\hat{F}_{N_1}(k)$) implies a plateau in the valuation of satCount (i.e., $\widehat{\mathsf{P}}(s_0, \varphi_1 \ \texttt{U}^{\leq k} \ \varphi_2)$). As such, on termination, U2B_P returns $\widehat{\mathsf{P}}(s_0, \varphi_1 \ \texttt{U}^{\leq \texttt{lastk}} \ \varphi_2) = \texttt{satCount}/N_1 = 0.66665$ as the estimate for $\mathsf{P}(s_0, \varphi_1 \ \texttt{U} \ \varphi_2)$; recall that $\mathsf{P}(s, \varphi_1 \ \texttt{U} \ \varphi_2) = 0.66$.

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. The U2B method is also applicable in verifying similar properties for CTMC models. Unlike a DTMC model, in which a 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. Properties of interest for CTMC include transient and steady state properties which are expressed in Continuous Stochastic Logic (CSL), developed by Aziz et al. [1996]. However, the semantics of *time unbounded* until CSL properties of the form $\texttt{P}_{\bowtie r}(\varphi_1 \ \texttt{U}^{[0,\infty)} \ \varphi_2)$ is identical to the semantics of $\texttt{P}_{\bowtie r}(\varphi_1 \ \texttt{U} \ \varphi_2)$ given in terms of the embedded DTMC of the CTMC, where the embedded DTMC captures the probability of each transition in the CTMC, independent of the time at which it occurs [Baier et al. 2003; Kwiatkowska et al. 2007]. Hence, a CSL property of the form $\texttt{P}_{\bowtie r}(\varphi_1 \ \texttt{U}^{[0,\infty)} \ \varphi_2)$ can be verified at any state s in a CTMC by verifying the satisfiability of $\texttt{P}_{\bowtie r}(\varphi_1 \ \texttt{U} \ \varphi_2)$ in the corresponding embedded DTMC, at the same state

[4]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 $\mathsf{P}(s_0, \varphi_1 \ \texttt{U} \ \varphi_2)$ incorrectly. That is why U2B_P produces a warning message on termination and recommends that the user rerun U2B_P using different values of simpoly.

s. That is, our U2B algorithm is directly applicable in the verification of $\mathtt{P}_{\bowtie r}(\varphi_1 \; \mathtt{U}^{[0,\infty)} \; \varphi_2)$ for CTMC models.

8. OVERVIEW OF PRISM-U2B TOOL

We have realized the U2B and U2B_P methods in a tool called PRISM-U2B, which is developed by leveraging the existing implementation of the PRISM model checker [Hinton et al. 2006] (available under GNU GPL). PRISM-U2B reuses PRISM's user interface, parsers, and simulation engine. Specifically, PRISM-U2B uses PRISM's input specification language and presents a user interface similar to that of PRISM. This significantly minimizes the cognitive burden of learning and understanding the usage of PRISM-U2B. While simple changes to the PRISM user interface were sufficient to allow for the additional error bounds and confidence parameters used in the two phases of PRISM-U2B, the internal changes to the PRISM simulator are significant. In the following, we discuss the distinguishing aspects of PRISM-U2B with respect to realization of sample path generation.

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 rules and probabilities for transitions, as 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 number 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 generating one path up to a prespecified 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 unsatisfiability 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 Phase I of the U2B method in PRISM-U2B. First, the PRISM simulator requires that there be a prespecified upper bound on 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 any restriction on the path length, the preceding is not a good strategy for random generation of paths in Phase I. For instance, for the property $\varphi_1 \; \mathtt{U} \; \varphi_2$, if the path being generated is part of an unconditional loop in the model and satisfies the property $\varphi_1 \wedge \neg\varphi_2$, then this path generation method will lead to the 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 generates and keeps track of multiple paths using the last state in the corresponding path (where the number of states corresponds to the *window size* in Algorithms 2, 3) at a time.

Implementation of Phase II in PRISM-U2B. These preceding features of the 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 the $[0, k_0]$ time interval. That is, the bound on until properties in CSL corresponds to a *time* bound as opposed to a *step* bound, as in PCTL and as is required for Phase II in the U2B method. As such, when the model under consideration is a CTMC, the corresponding CSL until property is left unaltered for Phase II and the PRISM simulator is directed to consider sample paths of length up to k_0. This faithfully satisfies the requirement of verifying the k_0-step bounded until property in the embedded DTMC of the given CTMC.

9. EXPERIMENTAL EVALUATION

We evaluate the proposed U2B method and its realization in PRISM-U2B using a number of case studies available in the PRISM example suite. The objective of this empirical study is to show the effectiveness of the 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 a detailed analysis of empirical results in Sections 9.2 and 9.3.

9.1. Case Studies

As noted before, PRISM-U2B reuses PRISM's input specification language and language parsers, and therefore, our experiments directly use several case studies from the PRISM example suite. The case studies include probabilistic models of the randomized dining philosopher (a classic illustration of the multiprocess synchronization problem), the IPv4 Zeroconf protocol, the crowds protocol for capturing the behavior of anonymous Web browsing, a gossip-based broadcast protocol, an embedded control system, contract signing, and a cyclic polling system for client-server conversation. The software programs implementing these protocols take into consideration the stochastic nature of the environment and, as such, the problem of model checking whether these protocols exhibit desired behavior (in a probabilistic sense) is an important software engineering problem.

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[5].

This is a DTMC model representing 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=?[ "allhungry" U ("othershungry" & "eat") ],
```

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

[5]Jennings et al. [2010].

9.1.2. Simple Queue[6]. This is a DTMC model of a queue in which requests are either received or serviced with a particular probability distribution. If too many requests are received before they can be serviced, then there is an *overflow* of requests. The property considered computes the probability with which the queue eventually reaches such a state.

```
P=?[ true U "overflow"].
```

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

9.1.3. IPv4 Zeroconf Protocol. This a simplified DTMC model [Sen et al. 2005] representing the IPv4 Zeroconf Protocol. Similar to the overflow state in the 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=?[ true U "error"].
```

As in the 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. Crowds Protocol[7]. This is a DTMC model of a protocol [Reiter and Rabin 1998] for anonymous Web browsing that attempts to minimize the chance that eavesdropping adversaries will be effective in identifying the individual users by spying on network traffic. Experiments are done with models containing 50, 100, and 200 hosts (size of the crowd) and four adversaries. The property query involves estimating the probability that the identify of an user is divulged, which happens when at least two adversaries observe (agree on) the user's identity.

```
P=?[true U (observe0>1)],
```

where, `observe[`i`]` denotes the number of adversaries that have observed the user i.

9.1.5. Broadcast Protocol[8]. This is a 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 prespecified probability. Two variations of the protocol are considered. In the *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 collisions 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=?[ true U (active1=0) ]` and `P=?[ true U (active2=0) ]`,

where `active[i]` is equal to 0 when a node goes to sleep after receiving (and possibly forwarding) message.

9.1.6. EGL Contract Signing Protocol[9]. 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 either both participants obtain the other's secret or neither does. A DTMC model of the EGL contract signing protocol is considered. The property query under consideration demonstrates that the protocol is flawed with respect to

[6]Jennings et al. [2010].
[7]http://www.prismmodelchecker.org/casestudies/crowds.php.
[8]http://www.prismmodelchecker.org/casestudies/prob_broadcast.php.
[9]http://www.prismmodelchecker.org/casestudies/contract_egl.php.

fairness. That is, the property captures the fact that one participant gets access to other party's secret without communicating its own secrets.

```
P=?[ true U (!"kA" & "kB") ],
```

where, kA and kB denote the states where participants A and B have knowledge of the others secret.

9.1.7. Simple Dice Protocol[10]. This is a DTMC model in which the outcome of rolling a fair die 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 roll of the die. For instance,

```
P=?[true U ("s=7" & "d=6")],
```

is a PCTL property querying the probability of obtaining a roll of six on a single die.

9.1.8. Embedded Control System[11]. 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=?[ !"down" U "fail_sensors" ].
```

9.1.9. Cyclic Server Polling System[12]. 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

```
P=?[ !(s=2 & a=1) U (s=1 & a=1) ],
```

which queries 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.

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 an Intel Core 2 Duo 3GHz CPU with 2GB of memory. The results for PRISM-U2B are obtained by using Algorithm 2, as all the examples considered from the PRISM example suite satisfy the condition $\lim_{k\to\infty} \mathsf{P}(s, \psi_k) \geq 1 - \epsilon_0$, as required for the termination of Phase I in the U2B method.

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

Table I provides a summary of our results. For each of the methods, the table presents the probability, 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, and k_0 denotes the maximum path length considered by PRISM-U2B (the bound obtained automatically from Phase I of U2B).

[10]http://www.prismmodelchecker.org/casestudies/dice.php.
[11]http://www.prismmodelchecker.org/casestudies/embedded.php.
[12]http://www.prismmodelchecker.org/casestudies/polling.php.

Table I. Summary of Results: `PRISM` vs. `PRISM-U2B`

Model	Variations	PRISM					PRISM-U2B			
		Numerical		Statistical			Phase I		Phase II	
		p	Time	$\hat{p}$	$\bar{k}$	Time	k_0	Time	$\hat{p}$	Time
(Fig. 1)		No result		No result			3343	**7.74**	0.6601	**88**.74
Philosopher	# philos: 10	0.1	5.98	0.0999	75	**29.21**	48	**1.5**	0.0997	**16**.85
	# philos: 20	No result		0.0497	103	**89.6**	68	**4.31**	0.0496	**51**.97
	# philos: 40	No result		0.0247	141	**252.49**	98	**11.76**	0.0246	**146**.47
	# philos: 60	No result		0.0163	176	**469.29**	123	**21.75**	0.0166	**271**.69
	# philos: 100	No result		0.0099	227	**1,044.12**	164	**48.29**	0.0097	**606**.7
Queue	$n:80, q:0.9, r:0.5$	0.1022	<1	0.1067	20,156	**214.71**	8,066	**9.92**	0.105	**126**.27
	$n:100, q:0.9, r:0.5$	0.0832	<1	0.0871	29,906	**275.2**	12,083	**12.62**	0.0862	**162**.16
	$n:140, q:0.9, r:0.5$	0.0607	<1	0.0638	55,523	**395.55**	20,774	**18.19**	0.0627	**235**.46
	$n:180, q:0.9, r:0.5$	0.0477	<1	0.0502	83,910	**507.59**	32,336	**23.75**	0.0491	**303**.97
	$n:200, q:0.9, r:0.5$	0.0431	<1	No result at $\bar{k}=10^6$			38,928	**26.45**	0.0444	**339**.01
Zeroconf	$n:10, q:0.9, r:0.9$	0.8098	<1	0.8118	99	**4.18**	44	<1	0.8098	**2**.27
	$n:20, q:0.9, r:0.9$	0.5975	<1	0.6015	360	**11.16**	173	<1	0.5998	**6**.54
	$n:60, q:0.9, r:0.9$	0.0215	<1	0.0222	1,469	**30.72**	636	**1.57**	0.0221	**17**.98
	$n:80, q:0.9, r:0.9$	0.0027	<1	0.0027	1,483	**31.58**	670	**1.61**	0.0028	**18**.43
Crowds	$n:50$	0.0483	175.42	0.0483	220	**73.88**	135	**3.68**	0.0484	**45**.16
	$n:100$	No result		0.046	218	**135.85**	135	**6.79**	0.0462	**83**.97
	$n:200$	No result		0.045	219	**249.39**	135	**13.03**	0.045	**163**.16
Broadcast Protocol	Synch. no Collision	0.8	<1	0.8001	3	**34.83**	3	**1.79**	0.8003	**19**.98
		0.7324	<1	0.7329	8	**45.14**	8	**2.27**	0.7326	**26**.02
	Synch. collision, lossy	0.5815	9.3	0.5814	20	**163.26**	20	**7.89**	0.581	**94**.95
		0.3462	9.61	0.3465	22	**225.99**	20	**10.84**	0.3463	**131**.27
Contract Signing		1.0	<1	1.0	36	**60.99**	36	**3.12**	1.0	**33**.88
Dice		0.1667	<1	0.1646	23	**2.39**	13	<1	0.1642	**1**.21
ECS	$MAX\_COUNT: 20,000$	0.7555	24,568.49	No result at $\bar{k}=10^6$			122,231	**1,520.05**	0.7532	**25,029**.29
Polling	$N:2$	0.5	<1	0.5	2,581	**178.76**	1,195	**8.09**	0.4983	**103**.73
	$N:20$	0.538	8,713.28	0.5384	9,714	**3,847.5**	4,261	**179.05**	0.5377	**2,230**.01

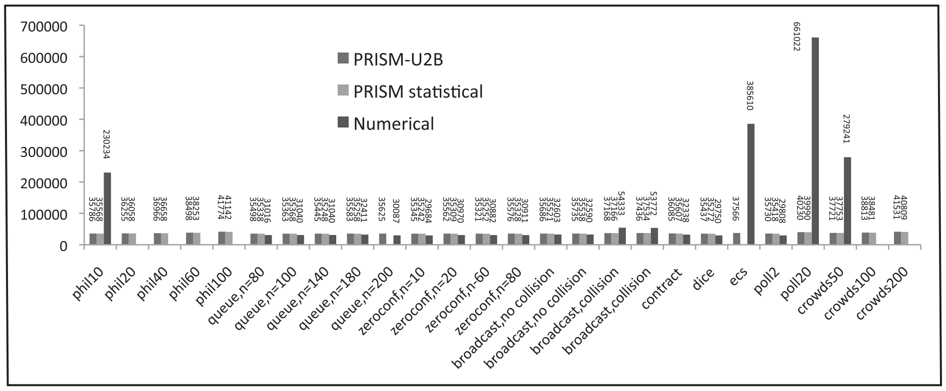

Fig. 9. Memory usage (in KB) for case studies.

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 ≥ 12, due to the 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 both require updates to the default value for maximum number of Jacobi iterations or the maximum simulation path length. The U2B method does not require any such user guidance.

Observe that for the rest of the case studies, PRISM-U2B typically outperforms PRISM's statistical method (timing results shown in bold). While both U2B in Phase II and PRISM's statistical method use the same sampling technique, the former uses a 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 the default value for sample-path-length in PRISM's statistical method is $10,000$ and may require the user to specify a greater value for some examples. The U2B method uses Phase I to compute a model-dependent path-length-bound k_0. 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 spent computing k_0 in Phase I.

9.2.2. Memory Usage. Another important consideration is the memory consumed by each of the methods. Figure 9 is a graph of the memory usage (in terms of KB) for the case studies for each of the methods. As all the methods are realized in the PRISM model-checking engine, each incurs a minimal overhead of ~30MB. The memory usage is comparable for all the methods for models that have a small state space. As expected, for models in which the state-space is large (e.g., randomized dining philosopher and embedded control system), PRISM's numerical method uses much more memory than the statistical methods (in PRISM and PRISM-U2B).

Figure 10 shows the increase in the memory usage by PRISM's numerical method with the increase in the number of philosophers in the randomized dining philosopher case study. PRISM's numerical method fails, due to the large memory requirement when the number of philosophers is greater than 11. Observe that the statistical methods do not suffer from such an increase in the memory usage with the increase in the state space of the model. It is worth mentioning that the U2B method in general is expected to use more memory 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 $\delta = 0.001$, $N_1\epsilon_0 = 48$).

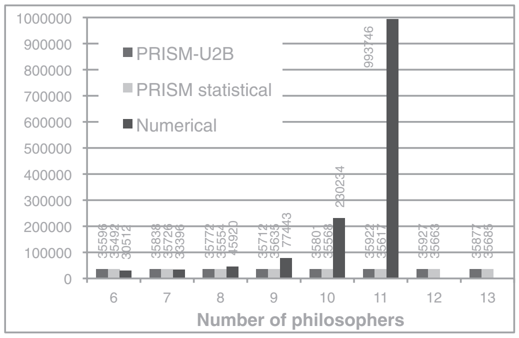

Fig. 10. Memory usage (in KB) for the randomized dining philosopher model.

9.2.3. Summary of Results. The results empirically show that the U2B method, as implemented in PRISM-U2B, is (a) as time efficient as PRISM's statistical method (and in many cases, outperforms PRISM's statistical method), (b) as precise as PRISM's statistical method, and (c) successfully computes results automatically for cases where PRISM fails or requires user guidance (to increase the maximum Jacobi iterations for the numerical method or the maximum sample path length for the numerical method). Finally, as PRISM-U2B utilizes the model specification language GUI and command-line interface of PRISM, it enjoys (as does PRISM) a high rating among all the probabilistic model checkers in terms of usability [Jansen et al. 2007]. 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 in case studies where PRISM fails.

9.3. Comparison with the MRMC Tool

Katoen and Zapreev [2009] have performed extensive comparisons of the statistical model-checking tools MRMC (based on Zapreev [2008]), VESTA (based on Sen et al. [2005]), and Ymer (based on Younes and Simmons [2002]), and have concluded that MRMC is the fastest among the three. Ymer [Younes 2005] does not allow verification of (time) unbounded until properties, while VESTA and MRMC do support verification of these properties. Recall from Section 3.2 that the statistical method in VESTA suffers from two main drawbacks which restrict its correct application in models with loops (see Younes and Simmons [2006] and He et al. [2010] for details). Furthermore, we were unable to obtain the tool from the authors (personal communication with Koushik Sen, 11/04/2010). While the tool MRMC does not have any such restriction, it requires prior analysis of the model transition structure to correctly verify unbounded until properties, even when using the sampling-based method. In the following, we compare MRMC, and PRISM-U2B and discuss their distinguishing aspects.

The Markov reward model checker [MRMC 2010] supports verification of probabilistic temporal logic properties (PCTL, CSL, etc.) against different types of probabilistic models (DTMC, CTMC, etc.) using both numerical and statistical sampling-based methods. MRMC's statistical method utilizes confidence-interval-based estimation to control the error in inferring whether the null hypothesis is correctly rejected or not. As this method utilizes both confidence-interval-based statistical estimation (Section 3.1.1) and hypothesis testing (Section 3.1.2), we present an overview of the technique to clearly explain the user-specified inputs that are necessary for correct application of MRMC's statistical method.

Recall that PRISM and PRISM-U2B are meant to compute the probability of satisfying a given path property within a prespecified error margin and with a certain confidence bound. In contrast, probabilistic queries in MRMC are of the form $\mathrm{P}_{\bowtie r}(\psi)$, where $\bowtie \in \{<, >, \leq, \geq\}$, and ψ is a path property. Therefore, the result or output of MRMC is boolean. More precisely, MRMC uses the statistical estimate $\hat{p}$ using the confidence interval and infers that the relation $\hat{p} \bowtie r$ holds when the confidence interval does not include r. For instance, if $\bowtie$ is equal to $\geq$, then the method can provide a definitive answer (true) when the lower bound of the confidence interval is greater than r, and a definitive answer (false) when the upper bound of the confidence interval is less than r. However, due to the probabilistic nature of the confidence interval, it is necessary to assume that r is at least ξ distance from the true probability (where the indifference width is 2ξ; see Section 3.1.2), and that the confidence interval is tighter than ξ. In short, MRMC draws from techniques used in hypothesis testing (as applied in probabilistic model checking) as well as confidence-interval-based statistical estimation. MRMC therefore requires the following user inputs: confidence level δ for estimation using the confidence interval and $2\xi'$, the confidence interval (such that $\xi' < \xi$). That is, ξ' corresponds to ϵ used in PRISM-U2B.

Zapreev [2008] proposes a technique (realized in the tool MRMC) that deals with unbounded until properties by pre-analyzing the model under consideration. More specifically, for the property $\varphi_1 \ \mathtt{U} \ \varphi_2$, the sampling-based method in MRMC first identifies (and discards) the states that never lead to any state satisfying $\varphi_1 \ \mathtt{U} \ \varphi_2$. Such reachability analysis requires complete knowledge of the model transition structure, which may not be possible when the model state space is prohibitively large or the model of the system is not available (e.g., if only execution paths are available, rather than the transition structure).

Another important difference between MRMC and both PRISM and PRISM-U2B (which possibly stems from MRMC's requirement of pre-analyzing the model) is that MRMC reads/loads the entire transition matrix of the probabilistic model for its sampling-based method. Therefore, the memory usage of the sampling-based method in MRMC is similar to that of numerical methods (e.g., in PRISM's numerical method modulo, the differences that stem from a different implementation framework). This significantly reduces the applicability of MRMC's sampling-based method for large systems. In fact, for several examples in Table I, we cannot obtain results using MRMC, as it leads to an out-of-memory error. Table II summarizes the results of our experiments with MRMC. We have used an identical $\delta = 0.01$ (confidence parameter) and $\epsilon = \xi' = 0.025$ for the experiments. As MRMC tests whether the estimate is related to a given value r using the $\bowtie$ relation and requires that the true p be $\xi > \xi'$ distance away from r, we have set r to be equal to $p + \xi$, where $\xi = 0.026 > \xi'$. This allows MRMC to provide definitive answers (true or false) in all the cases. If ξ' is chosen to be greater than ξ, MRMC fails to compute a definitive answer. Note that no such assumption is required for the validity of our method, as implemented in PRISM-U2B.

In the table, the column p corresponds to the actual probability of satisfying the property being considered. It is either computed manually, computed by PRISM's numerical method, or estimated by PRISM's statistical method. The query input to MRMC tests whether the estimated probability $\bowtie p \pm 0.026$. The boundaries of the confidence interval estimated by MRMC are presented in the columns Lower Bound and Upper Bound. MRMC produces results faster than PRISM-U2B in all the cases where it is successful in computing the result. This is because (as noted previously) MRMC analyzes the model before performing sampling-based confidence-interval estimation. That is, MRMC does not deploy a pure sampling-based statistical method. This is advantageous for smaller models for which reachability can be performed. In fact, in some cases, sampling may not be even necessary. For instance, for the contract signing protocol, MRMC produces

Table II. Summary of results: MRMC vs. PRISM-U2B

Model	Variations	p	MRMC			PRISM-U2B			
						Phase I		Phase II	
			Lower Bound	Upper Bound	Time	k_0	Time	$\hat{p}$	Time
(Fig 1)		0.66	**Cannot load**			3,343	7.74	0.6601	88.74
Philosopher	# philos: 10	0.1	**Cannot load**			48	1.5	0.0997	16.85
Queue	$n: 80, q: 0.9, r: 0.5$	0.1022	0.0929	0.1047	< 1	8,066	9.92	0.105	126.27
	$n: 100, q: 0.9, r: 0.5$	0.0832	0.0737	0.0901	< 1	12,083	12.62	0.0862	162.16
	$n: 140, q: 0.9, r: 0.5$	0.0607	0.0416	0.0784	< 1	20,774	18.19	0.0627	235.46
	$n: 180, q: 0.9, r: 0.5$	0.0477	0.0273	0.0714	< 1	32,336	23.75	0.0491	303.97
	$n: 200, q: 0.9, r: 0.5$	0.0431	0.0230	0.0693	< 1	38,928	26.45	0.0444	339.01
Zeroconf	$n: 10, q: 0.9, r: 0.9$	0.8098	0.8070	0.8152	< 1	44	< 1	0.8098	2.27
	$n: 20, q: 0.9, r: 0.9$	0.5975	0.5849	0.6072	< 1	173	< 1	0.5998	6.54
	$n: 60, q: 0.9, r: 0.9$	0.0215	0.0180	0.0230	< 1	636	1.57	0.0221	17.98
	$n: 80, q: 0.9, r: 0.9$	0.0027	0.0006	0.0047	< 1	670	1.61	0.0028	18.43
Crowds	$n: 50$	0.0483	**Cannot model check**			135	3.68	0.0484	45.16
	$n: 100$	0.0450	PRISM **cannot export**			135	6.79	0.0462	83.97
Broadcast Protocol	Synch. no collision	0.800	0.7969	0.8108	< 1	3	1.79	0.8003	19.98
		0.7324	0.7247	0.7478	< 1	8	2.27	0.7326	26.02
	Synch. collision, lossy	0.5815	0.5716	0.5919	< 1	20	7.89	0.5810	94.95
		0.3462	0.3365	0.3602	< 1	20	10.84	0.3463	131.27
Contract Signing		1.0	**Result with no simulation**			36	3.12	1.0	33.88
ECS	MAX_COUNT: 20,000	0.7555	PRISM **cannot export**			122,231	1,520.05	0.7532	25,029.29
Poll	$N: 2$	0.500	0.4894	0.5077	< 1	1,195	8.09	0.4983	103.73
	$N: 20$	0.0538	PRISM **cannot export**			4,261	179.05	0.5377	2,230.01

a definitive answer (infers that the property is satisfied) without using any samples. However, if such pre-analysis of the model is not possible, then MRMC fails. This happens for several examples when the transition matrix is too large to be exported from PRISM to MRMC (Figure 1 model, Crowds protocol of size $\geq$ 100, Embedded Control System, Cyclic Server Polling System of size 20), or when the reachability analysis fails in MRMC (dining philosopher protocol with $\geq$ ten philosophers, Crowds protocol of size 50).

In summary, the following aspects distinguish PRISM-U2B from MRMC.

(1) MRMC relies on reachability analysis in addition to samples to compute the results. PRISM-U2B deploys a pure sampling-based method. For the case studies for which MRMC can perform reachability analysis, MRMC computes verification result faster than PRISM-U2B.
(2) MRMC fails for models where the transition matrix is too large for loading or for performing reachability analysis. PRISM-U2B can be used for comparatively larger models.
(3) MRMC requires that the user input ξ' (the parameter specifying the confidence interval size) be smaller than the distance (half of the indifference width) of r from the true p in the query of the form $\mathrm{P}_{\bowtie r}(\psi)$, where $\bowtie$ belongs to $\{<, \leq, >, \geq\}$. There is no such requirement for PRISM-U2B. While MRMC may not be able to compute a definitive answer when the confidence interval size is not less than the indifference width, the U2B algorithm in PRISM-U2B may not always terminate (Section 6). None of the case studies, however, led to the nontermination of the U2B algorithm in PRISM-U2B.

10. CONCLUSION

Summary. 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 setting where only sample simulations of the model can be generated. 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 would not be able to estimate a result, and the U2B method would suffer from nontermination. We have explained the cause of such problems in terms of the notion of a plateau in the D-Graph. We have developed U2B_P, a heuristic-based plateau-detection method, to deal with the nontermination problem. Finally, we have incorporated the U2B and U2B_P methods in the PRISM tool to develop PRISM-U2B and conducted a detailed experimental study using different probabilistic models. The experiments show the effectiveness of the tool PRISM-U2B in terms of precision, computation time, and broader applicability (for large probabilistic models).

Future work. As part of future work, we plan to study the feasibility of identifying and precomputing 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 nondeterministic 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 nondeterministic 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 *Proceedings of the International Conference on Computer Aided Verification*. 269–276.

Aziz, A., Sanwal, K., Singhal, V., and Brayton, R. 2000. Model checking continuous time Markov chains. *ACM Trans. Computat. Logic 1*, 1, 162–170.

Baier, C., Haverkort, B., Hermanns, H., and Katoen, J.-P. 2003. Model-checking algorithms for continuous-time markov chains. *IEEE Trans. Softw. Eng. 29, 6*, 524–541.

Bianco, A. and de Alfaro, L. 1995. Model checking of probabilistic and nondeterministic systems. In *Proceedings of the Conference on Foundations of Software Technology and Theoretical Computer Science*.

Courcoubetis, C. and Yannakakis, M. 1995. The complexity of probabilistic verification. *J. ACM 42, 4*, 857–907.

Duflot, M., Kwiatkowska, M., Norman, G., and Parker, D. 2006. A formal analysis of Bluetooth device discovery. *Int. J. Softw. Tools Technol. Transfer 8*, 621–632.

Grosu, R. and Smolka, S. A. 2005. Monte Carlo model checking. In *Proceedings of the International Conference on Tools and Algorithms for the Contruction and Analysis of Systems*. 271–286.

Grunske, L. 2008. Specification patterns for probabilistic quality properties. In *Proceedings of the International Conference on Software Engineering*. 31–40.

Hansson, H. and Jonsson, B. 1994. A logic for reasoning about time and reliability. *Formal Aspects Comput. 6, 5*, 512–535.

He, R., Jennings, P., Basu, S., Ghosh, A., and Wu, H. 2010. A bounded statistical approach for model checking of unbounded until properties. In *Proceedings of the International Conference on Automated Software Engineering*. 225–234.

Herault, T., Lassaigne, R., Magniette, F., and Peyronnet, S. 2004. Approximate probabilistic model checking. In *Proceedings of the 5th International Conference on Verification, Model Checking, and Abstract Interpretation*.

Hinton, A., Kwiatkowska, M., Norman, G., and Parker, D. 2006. PRISM: A tool for automatic verification of probabilistic systems. In *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*.

Hoeffding, W. 1963. Probability inequalities for sums of bounded random variables. *J. Amer. Stat. Assoc. 58,* 301, 13–30.

Jansen, D. N., Katoen, J.-P., Oldenkamp, M., Stoelinga, M., and Zapreev, I. 2007. How fast and fat is your probabilistic model checker. In *Proceedings of the Haifa Verification Coference*. 69–85.

Jennings, P., Basu, S., and Ghosh, A. 2010. Two-phase PMCK. http://www.cs.iastate.edu/∼poj/prism-u2b.

Katoen, J.-P. and Zapreev, I. 2009. Simulation-based CTMC model checking. In *Proceedings of the International Conference on the Quantitative Evaluation of Systems*. IEEE Computer Society Press, Los Alamitos, CA.

Kwiatkowska, M., Norman, G., and Parker, D. 2007. Stochastic model checking. In *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation*, M. Bernardo and J. Hillston, Eds. Lecture Notes in Computer Science (Tutorial Volume), vol. 4486. Springer, Berlin, 220–270.

Kwiatkowska, M., Norman, G., and Parker, D. 2008. Using probabilistic model checking in systems biology. *ACM SIGMETRICS Perform. Eval. Rev. 35*, 14–21.

Legay, A., Delahaye, B., and Bensalem, S. 2010. Statistical model checking: An overview. In *Proceedings of the International Conference on Runtime Verification*. H. Barringer, Y. Falcone, B. Finkbeiner, K. Havelund, I. Lee, G. J. Pace, G. Rosu, O. Sokolsky, and N. Tillmann, Eds. Lecture Notes in Computer Science, vol. 6418. Springer, Berlin, 122–135.

Massart, P. 1990. The tight constant in the Dvoretzky-Kiefer-Wolfowitz inequality. *Ann. Probab. 18*, 1269–1283.

MRMC 2010. Markov reward model checker. http://www.mrmc-tool.org/trac.

Norman, G. and Shmatikov, V. 2006. Analysis of probabilistic contract signing. *J. Comput. Secur. 14*, 561–589.

Rabih, D. and Pekergin, N. 2009. Statistical model checking using perfect simulation. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis*. 120–134.

Reiter, M. K. and Rabin, A. D. 1998. Crowds: Anonymity for web transactions. *ACM Trans. Inform. Syst. Secur. 1,* 1, 66–92.

Roy, A. and Gopinath, K. 2005. Improved probabilistic models for 802.11 protocol verification. In *Proceedings of the 14th International Conference on Computer Aided Verification*.

Sen, K., Viswanathan, M., and Agha, G. 2005. On statistical model checking of stochastic systems. In *Proceedings of the 4th International Conference on Computer Aided Verification*.

Wald, A. 1945. Sequential tests of statistical hypotheses. *Ann. Math. Stat. 16,* 2, 117–186.

Younes, H. L., Kwiatkowska, M., Norman, G., and Parker, D. 2006. Numerical vs. statistical probabilistic model checking. *Int. J. Softw. Tools Technol. Transfer 8,* 3, 216–228.

Younes, H. L. S. 2005. Ymer: A statistical model checker. In *Proceedings of the International Conference on Computer Aided Verification*.

Younes, H. L. S. and Simmons, R. G. 2002. Probabilistic verification of discrete event systems using acceptance sampling. In *Proceedings of the 14th International Conference on Computer Aided Verification*.

Younes, H. L. S. and Simmons, R. G. 2006. Statistical probabilistic model checking with a focus on time-bounded properties. *Inform. Comput. 204,* 9, 1368–1409.

Zapreev, I. S. 2008. Model checking Markov chains: Techniques and tools. Ph.D. dissertation, University of Twente, The Netherlands.

Received June 2010; revised December 2010; accepted February 2011