On Statistical Model Checking of Stochastic Systems

Koushik Sen, Mahesh Viswanathan, Gul Agha
Department of Computer Science,
University of Illinois at Urbana-Champaign.
{ksen,vmahesh,agha}@uiuc.edu

Abstract. Statistical methods to model check stochastic systems have been, thus far, developed only for a sublogic of continuous stochastic logic (CSL) that does not have steady state operator and unbounded until formulas. In this paper, we present a statistical model checking algorithm that also verifies CSL formulas with unbounded untils. The algorithm is based on Monte Carlo simulation of the model and hypothesis testing of the samples, as opposed to sequential hypothesis testing. We have implemented the algorithm in a tool called VESTA, and found it to be effective in verifying several examples.

1 Introduction

Stochastic models and temporal logics such as continuous stochastic logic (CSL) [1, 3] and probabilistic computation tree logic (PCTL) [9] are widely used to model practical systems and analyze their performance and reliability. There are two primary approaches to analyzing the stochastic behavior of such systems: *numerical* and *statistical*. In the numerical approach, the formal model of the system is *model checked* for correctness with respect to the specification using symbolic and numerical methods. Model checkers for different classes of stochastic processes and specification logics have been developed [10, 13, 12, 4, 5, 14, 2]. Although the numerical approach is highly accurate, it suffers from memory problem due to state-space explosion and being computationally intensive. An alternate method, proposed in [18], is based on Monte Carlo simulation of the model and performing sequential hypothesis testing on the sample generated. In [15], this method was extended to statistically verify black-box, deployed systems that can only be passively observed. Being statistical, these methods are less precise: they only provide probabilistic guarantees of correctness.

Both statistical approaches (presented in [18, 15]), considered a sublogic of continuous stochastic logic (CSL) that excludes steady state operator and *unbounded until operator*. In this paper, we extend the statistical verification method to verify CSL (or PCTL) formulas that may have unbounded until connectives. Specifically, we consider a sublogic of CSL (and PCTL) that contains all the logical connectives, except for the steady-state operator and present a model checking algorithm for it. As in [18], we assume we have a model that can be simulated on a need basis. The samples generated by Monte Carlo simulation are subjected to hypothesis testing. However, unlike [18], we do simple hypothesis testing as opposed to sequential hypothesis testing. Simple hypothesis testing is easily amenable to parallelism, since the sampling and statistical tests can be done in parallel. We exploit parallelism in our implementation of the algorithm.

We make no inherent assumptions about the model that is being verified, other than it can be simulated using discrete event simulation, and that the model checking problem is well defined with respect to CSL (or PCTL). Thus, our algorithm can be successfully applied to Discrete Time Markov Chains, Continuous Time Markov Chains, and Semi Markov Chains. However, it is unclear whether our method can be applied to Generalized Semi Markov Processes (GSMP). This is because there is no well understood definition of a probability space on execution paths of a GSMP such that the model checking problem is well-defined, i.e., path formulas in CSL define measurable sets.

The rest of the paper is organized as follows. In Section 2, we give our assumptions about the system being analyzed, and present the syntax and semantics of CSL (and PCTL). The model checking algorithm is presented in Section 3. The algorithm is inductive, based on the structure of the formula being verified, and we present the details of the algorithm for all the CSL connectives in our sublogic, though our analysis of the previously considered operators (such as conjunction, negation, next, bounded until, and the probabilistic operator) is similar to that presented in [18], the decision procedures we use differ. Section 4 contains details of our implementation in the VESTA tool and the results of our experimental analysis of the tool. Finally, we conclude in Section 5.

2 Model and Logic

We consider stochastic models that meet the following requirements:

1. Sample execution paths can be generated through discrete-event simulation. Execution paths will be a sequences of the form $\pi = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \cdots$ where each s_i is a state of the model and $t_i \in \mathbb{R}_{>0}$ is the time spent in the state s_i before moving to the state s_{i+1}.
2. A probability space can be defined on the execution paths of the model in such a way that the paths satisfying any path formula in our concerned logic (CSL or PCTL), is measurable.
3. The number of states of the system is finite.

It has been shown that commonly used models such as continuous-time Markov chains (CTMC) [17], semi-Markov chains (SMC) [7, 14], which are a generalization of CTMC, meet the above requirements. While we believe our algorithm will work for any model that satisfies the above conditions, in order to establish the mathematical concepts and notation clearly, we focus on SMCs.

Let AP be a set of finite atomic propositions. A labelled semi-Markov chain (SMC) is a tuple $\mathcal{M} = (S, s_I, \mathbf{P}, \mathbf{Q}, L)$ where S is a finite set of states, s_I is the initial state, $\mathbf{P}\colon S \times S \to [0, 1]$ is a *transition probability matrix* such that $\sum_{s' \in S} \mathbf{P}(s, s') = 1$ for each s in S, $\mathbf{Q}\colon S \times S \to (\mathbb{R}_{\geq 0} \to [0, 1])$ is a matrix of continuous cumulative probability distribution functions such that $\mathbf{P}(s, s') = 0$ implies for all t, $\mathbf{Q}(s, s', t) = 1$, and $L\colon S \to 2^{AP}$ is a labelling function that maps every state to a set of atomic propositions.

If for any two states s and s', $\mathbf{P}(s, s') > 0$ then there is a transition from s to s', and the probability of the transition is given by $\mathbf{P}(s, s')$. Thus we can see $(S, s_I, \mathbf{P}, L)$ as the discrete-time Markov chain embedded in the SMC $\mathcal{M}$.

Once a next state s' from the current state s is sampled according to the matrix $\mathbf{P}$, the sojourn time in the state s is determined according to the cumulative probability distribution function $\mathbf{Q}(s, s', t)$. The probability to move from state s to s' within t units of time given that s' is sampled as the next state is given by $\mathbf{Q}(s, s', t)$. Note that if all the probability distribution functions in the matrix $\mathbf{Q}$ are exponential then the SMC becomes a CTMC.

A sequence $\pi = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \cdots$ is called a path of $\mathcal{M}$, if $s_0 = s_I$, $s_i \in S$, $t_i \in \mathbb{R}_{\geq 0}$, and $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $i \geq 0$. We denote the i^{th} state in an execution π by $\pi[i] = s_i$, and the time spent in the i^{th} state by $\delta(\pi, i) = t_i$. The time at which the execution enters state $\pi[i+1]$ is given by $\tau(\pi, i+1) = \sum_{j=0}^{j=i} \delta(\pi, j)$. The state of the execution at time t (if the sum of sojourn times in all states in the path exceeds t), denoted by $\pi(t)$, is the state s_i such that i is the smallest number for which $t \leq \tau(\pi, i+1)$. We let $Path(s)$ be the set of paths starting at state s.

Let $s_0, s_1, \ldots, s_k \in S$ with $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $0 \leq i < k$. Let $I_0, I_1, I_2, \ldots I_{k-1}$ be non-empty intervals in $\mathbb{R}_{\geq 0}$. Then $C(s_0, I_0, s_1, \ldots I_{k-1}, s_k)$ denotes a *cylinder set* consisting of all paths $\pi \in Path(s_0)$ such that $\pi[i] = s_i$ (for $0 \leq i \leq k$), and $\delta(\pi, i) \in I_i$ (for $i < k$). Let $\mathcal{B}$ be the smallest σ-algebra on $Path(s_0)$ which contains all the cylinders $C(s_0, I_0, s_1, \ldots I_{k-1}, s_k)$. The measure μ on cylinder sets can be inductively defined as $\mu(C(s_0)) = 1$ and for $k > 0$ as

$$\begin{aligned}&\mu(C(s_0, I_0, s_1, \ldots I_{k-1}, s_k))\\ &\quad = \mu(C(s_0, I_0, s_1, \ldots s_{k-1})) \cdot \mathbf{P}(s_{k-1}, s_k) \cdot (\mathbf{Q}(s_{k-1}, s_k, u) - \mathbf{Q}(s_{k-1}, s_k, \ell))\end{aligned}$$

where $\ell = \inf I_k$ and $u = \sup I_k$. The probability measure on $\mathcal{B}$ is then defined as the unique measure that agrees with μ (as defined above) on the cylinder sets.

2.1 CSL and PCTL Syntax and Semantics

Continuous stochastic logic (CSL) is introduced in [1] as a logic to express probabilistic properties of continuous time Markov chains (CTMCs). We adopt a sublogic of CSL that excludes the steady-state probabilistic operator. Let ϕ represents a *state* formula and ψ represents a *path* formula. Then:

$$\phi ::= true \mid a \in AP \mid \neg\phi \mid \phi \wedge \phi \mid \mathcal{P}_{\bowtie p}(\psi) \qquad \psi ::= \phi\, \mathcal{U}\, \phi \mid \phi\, \mathcal{U}^{\leq t} \phi \mid \mathbf{X}\phi \mid \mathbf{X}^{\leq t}\phi$$

where AP is the set of atomic propositions, $\bowtie \in \{<, \leq, >, \geq\}$, $p \in [0, 1]$, and $t \in \mathbb{R}_{\geq 0}$. The notion that a state s (or a path π) *satisfies* a formula ϕ is denoted by $s \models \phi$ (or $\pi \models \phi$), and is defined inductively as follows:

$s \models true$		$s \models a$	iff $a \in AP(s)$
$s \models \neg\phi$	iff $s \not\models \phi$	$s \models \phi_1 \wedge \phi_2$	iff $s \models \phi_1$ and $s \models \phi_2$
$s \models \mathcal{P}_{\bowtie p}(\psi)$	iff $Prob\{\pi \in Path(s) \mid \pi \models \psi\} \bowtie p$		
$\pi \models \mathbf{X}\phi$	iff $\tau(\pi, 1) < \infty$ and $\pi[1] \models \phi$		
$\pi \models \mathbf{X}^{\leq t}\phi$	iff $\tau(\pi, 1) \leq t$ and $\pi[1] \models \phi$		
$\pi \models \phi_1\, \mathcal{U}\, \phi_2$	iff $\exists x \in \mathbb{R}_{\geq 0}\ (\pi(x) \models \phi_2$ and $\forall y \in [0, x).\ \pi(y) \models \phi_1)$		
$\pi \models \phi_1\, \mathcal{U}^{\leq t} \phi_2$	iff $\exists x \in [0, t].\ (\pi(x) \models \phi_2$ and $\forall y \in [0, x).\ \pi(y) \models \phi_1)$		

It can shown that for any path formula ψ and any state s, the set $\{\pi \in Path(s) \mid \pi \models \psi\}$ is measurable [14]. A formula $\mathcal{P}_{\bowtie p}(\psi)$ is satisfied by a state

s if $Prob$[path starting at s satisfies ψ] $\bowtie p$. The path formula $\mathbf{X}\phi$ holds over a path if ϕ holds at the second state on the path. The formula $\phi_1\,\mathcal{U}^{\leq t}\phi_2$ is true over a path π if ϕ_2 holds in some state along π at a time $x \in [0,t]$, and ϕ holds at all prior states.

Note that if we change the time domain in the above logic from $\mathbb{R}_{\geq 0}$ to natural numbers $\mathbb{N}$, we get the logic PCTL (stands for probabilistic computation tree logic) [9]. The model-checking algorithm that we describe next is correct for both time domains. Therefore, we can use the model-checking algorithm for verifying properties expressed in both CSL and PCTL. In case of model-checking a PCTL formula, we will assume that the model provided is discrete-time with unit time associated with every transition.

3 Statistical Model Checking

Our model checking algorithm, $\mathcal{A}$, proceeds recursively based on the structure of the formula. Before describing the details of the algorithm, we present the theorem that formally states the correctness of the algorithm. The statement of the theorem is instructive in understanding the subsequent analysis. The algorithm $\mathcal{A}$ takes as input a stochastic model $\mathcal{M}$, a formula ϕ in CSL, error bounds α^* and β^*, and three other parameters δ_1, δ_2, and p_s. The result of model checking on these parameters, denoted by $\mathcal{A}^{\delta_1,\delta_2,p_s}(\mathcal{M},\phi,\alpha^*,\beta^*)$, can be either *true* or *false*. The algorithm provides the following correctness guarantees.

Theorem 1. *If the model $\mathcal{M}$ satisfies the following conditions*

- C1: *For every subformula of the form $\mathcal{P}_{\geq p}\psi$ in the formula ϕ and for every state s in $\mathcal{M}$, the probability that a path from s satisfies ψ must not lie in the range $[\frac{p-\delta_1-\alpha^*}{1-\alpha^*}, \frac{p+\delta_1}{1-\beta^*}]$;*
- C2: *For any subformula of the form $\phi_1\,\mathcal{U}\,\phi_2$ and for every state s in $\mathcal{M}$, the probability that a path from s satisfies $\phi_1\,\mathcal{U}\,\phi_2$ must not lie in the range $(0, \frac{\delta_2}{(1-p_s)^N}]$, where N is the number of states in the model $\mathcal{M}$.*

Then the algorithm provides the following guarantees

$$\begin{array}{ll}\mathrm{R1}: & Prob[\mathcal{A}^{\delta_1,\delta_2,p_s}(\mathcal{M},\phi,\alpha^*,\beta^*) = \mathit{true} \mid \mathcal{M} \not\models \phi] \leq \alpha^* \\ & Prob[\mathcal{A}^{\delta_1,\delta_2,p_s}(\mathcal{M},\phi,\alpha^*,\beta^*) = \mathit{false} \mid \mathcal{M} \models \phi] \leq \beta^*\end{array}$$

Condition C1 requires that the model be such that for any subformula $\mathcal{P}_{\geq p}\psi$, the probability of ψ being satisfied at a state be bounded away from p. Condition C2 requires that either an until formula does not hold in a state or it holds with some probability that is bounded away from 0. Under such circumstances, we guarantee that the probability of error of $\mathcal{A}$ is within the required bounds.

A few points about the algorithm are in order. First, the requirement that the model satisfy condition C1, is something that previous stochastic model checking algorithms also have. Second, the error bounds α^* and β^* are parameters to the algorithm. Hence, we can improve the confidence in the algorithm's answer to be as close to 1 as we like. Third, the bounds required in conditions C1 and C2 depend on the parameters δ_1, δ_2, and p_s given to the algorithm. Thus, they can be tuned based on the model and formula being analyzed, to ensure that

C1 and C2 are satisfied. Typically, for our experiments, we picked $\delta_1 = \delta_2 = 0.01$ and $p_s = 0.1$. Note that one may easily pick p_s to be c/N where N is the number of states and c is some positive constant. This will ensure that the upper bound of the range in condition C2 is $\frac{\delta_2}{(1-c/N)^N} \leq \delta_2 2e^c$ (proved in [16]), which can be made as close to 0 as desired by a suitable choice of c. Note that making p_s smaller comes with a price: if we make p_s very small, the expected length of the samples increases. This can increase the computation cost, something we also observed in our experiments. However, techniques such as caching and discounting optimization (discussed in Section 4) helped us to considerably reduce the computation cost for small p_s.

We make the following notational simplifications in the rest of the paper. The parameters δ_1, δ_2, and p_s are global to the algorithm $\mathcal{A}$; therefore, we will omit the superscript δ_1, δ_2, p_s from $\mathcal{A}^{\delta_1,\delta_2,p_s}(\mathcal{M}, \phi, \alpha^*, \beta^*)$ and write it simply as $\mathcal{A}(\mathcal{M}, \phi, \alpha^*, \beta^*)$. The value of the error bounds α and β will change for the invocation of $\mathcal{A}$ on various subformulas; therefore, we will carry them with $\mathcal{A}$. The result of model-checking a state formula ϕ at a state s will be denoted by $\mathcal{A}(s, \phi, \alpha, \beta)$; similarly, the result of model-checking a path formula ψ over a path π will be denoted by $\mathcal{A}(\pi, \psi, \alpha, \beta)$. Note that $\mathcal{A}(\mathcal{M}, \phi, \alpha^*, \beta^*)$ is same as $\mathcal{A}(s_I, \phi, \alpha^*, \beta^*)$.

3.1 Probabilistic Operator: Computing $\mathcal{A}(s, \mathcal{P}_{\bowtie p}(\psi), \alpha, \beta)$

We use statistical hypothesis testing [11] to verify a probabilistic property $\phi = \mathcal{P}_{\bowtie p}(\psi)$ at a given state s. Without loss of generality, we show our procedure for $\phi = \mathcal{P}_{\geq p}(\psi)$. Note that $\mathcal{P}_{<p}(\psi)$ is the same as $\neg\mathcal{P}_{\geq p}(\psi)$ and $<$ (or $>$) is essentially the same as $\leq$ (or $\geq$). Let p' be the probability that ψ holds over a random path starting at s. We say that $s \models \mathcal{P}_{\geq p}(\psi)$ if and only if $p' \geq p$ and $s \not\models \mathcal{P}_{\geq p}(\psi)$ if and only if $p' < p$.

We want to decide whether $s \models \mathcal{P}_{\geq p}(\psi)$ or $s \not\models \mathcal{P}_{\geq p}(\psi)$. By condition C1, we know that p' cannot lie in the range $[\frac{p-\delta_1-\alpha}{1-\alpha}, \frac{p+\delta_1}{1-\beta}]$, which implies that p' cannot lie in the range $[p-\delta_1, p+\delta_1]$. Accordingly, we set up the following experiment. Let $H_0\colon p' < p-\delta_1$ be the *null hypothesis* and $H_1\colon p' > p+\delta_1$ be the *alternative hypothesis*. Let n be the number of execution paths sampled from the state s. We will show how to estimate n from the different given parameters. Let $X_1, X_2, \ldots, X_n$ be a random sample having Bernoulli distribution with unknown mean $p' \in [0,1]$ i.e., for each $i \in [1,n]$, $\mathit{Prob}[X_i = 1] = p'$. Then the sum $Y = X_1 + X_2 + \ldots + X_n$ has binomial distribution with parameters n and p'. We say that x_i, an observation of the random variable X_i, is 1 if the i^{th} sample path from s satisfies ψ and 0 otherwise. In the experiment, we reject $H_0\colon p' < p - \delta_1$ and say $\mathcal{A}(s, \phi, \alpha, \beta) = \mathit{true}$ if $\frac{\sum x_i}{n} \geq p$; otherwise, we reject $H_1\colon p' \geq p$ and say $\mathcal{A}(s, \phi, \alpha, \beta) = \mathit{false}$ if $\frac{\sum x_i}{n} < p$. Given the above experiment, to meet the requirement R1 of $\mathcal{A}$, we must have

$$\begin{array}{l} \mathit{Prob}[\text{accept } H_1 \mid H_0 \text{ holds}] = \mathit{Prob}[Y/n \geq p \mid p' < p - \delta_1] \leq \alpha \\ \mathit{Prob}[\text{accept } H_0 \mid H_1 \text{ holds}] = \mathit{Prob}[Y/n < p \mid p' > p + \delta_1] \leq \beta \end{array}$$

Accordingly, we can choose the unknown parameter n for this experiment such that $Prob[Y/n \geq p \mid p' < p - \delta_1] \leq Prob[Y/n \geq p \mid p' = p - \delta_1] \leq \alpha$ and $Prob[Y/n < p \mid p' \geq p + \delta_1] \leq Prob[Y/n < p \mid p' = p + \delta_1] \leq \beta$. In other words, we want to choose the smallest n such that both $Prob[Y/n \geq p] \leq \alpha$ when Y is binomially distributed with parameters n and $p - \delta_1$, and $Prob[Y/n < p] \leq \beta$ when Y is binomially distributed with parameters n and $p + \delta_1$, holds. Such an n can be chosen by standard statistical methods.

3.2 Nested Probabilistic Operators: Computing $\mathcal{A}(s, \mathcal{P}_{\bowtie p}(\psi), \alpha, \beta)$

The above procedure for hypothesis testing works if the truth value of ψ over a sample path determined by the algorithm is the same as the actual truth value. However, in the presence of nested probabilistic operators in ψ, $\mathcal{A}$ cannot determine the satisfaction of ψ over a sample path exactly. Therefore, we modify the hypothesis test so that we can use the inexact truth values of ψ over the sample paths.

Let the random variable X be 1 if a sample path π from s actually satisfies ψ in the model and 0 otherwise. Let the random variable Z be 1 for a sample path π if $\mathcal{A}(\pi, \psi, \alpha, \beta) = \mathit{true}$ and 0 if $\mathcal{A}(\pi, \psi, \alpha, \beta) = \mathit{false}$. In our algorithm, we cannot get samples from the random variable X; instead, our samples come from the random variable Z. Let X and Z have Bernoulli distributions with parameters p' and p'' respectively. Let $Z_1, Z_2, \ldots, Z_n$ be a random sample from the Bernoulli distribution with unknown mean $p'' \in [0, 1]$. We say that z_i, an observation of the random variable Z_i, is 1 if $\mathcal{A}(\pi_i, \psi, \alpha, \beta) = \mathit{true}$ for i^{th} sample path π_i from s and 0 otherwise.

We want to test the null hypothesis $H_0\colon p' < p - \delta_1$ against the alternative hypothesis $H_1\colon p' > p + \delta_1$. Using the samples from Z we can estimate p''. However, we need an estimation for p' in order to decide whether $\phi = \mathcal{P}_{\geq p}(\psi)$ holds in state s or not. To get an estimate for p' we note that the random variables X and Z are related as follows: $Prob[Z = 1 \mid X = 0] \leq \alpha'$ and $Prob[Z = 0 \mid X = 1] \leq \beta'$, where α' and β' are the error bounds within which $\mathcal{A}$ verifies the formula ψ over a sample path from s. We can set $\alpha' = \alpha$ and $\beta' = \beta$. By elementary probability theory, we have

$$Prob[Z = 1] = Prob[Z = 1 \mid X = 0]Prob[X = 0] + Prob[Z = 1 \mid X = 1]Prob[X = 1]$$

Therefore, we can approximate $p'' = Prob[Z = 1]$ as follows:

$$\begin{array}{l} Prob[Z = 1] \leq \alpha(1 - p') + 1 \cdot p' = p' + (1 - p')\alpha \\ Prob[Z = 1] \geq Prob[Z = 1 \mid X = 1]Prob[X = 1] \geq (1 - \beta)p' = p' - \beta p' \end{array}$$

This gives the following range in which p'' lies: $p' - \beta p' \leq p'' \leq p' + (1 - p')\alpha$.

By condition C1, we know that p' cannot lie in the range $[\frac{p-\delta_1-\alpha}{1-\alpha}, \frac{p+\delta_1}{1-\beta}]$. Accordingly, we set up the following experiment. Let $H_0\colon p' < \frac{p-\delta_1-\alpha}{1-\alpha}$ be the *null hypothesis* and $H_1\colon p' > \frac{p+\delta_1}{1-\beta}$ be the *alternative hypothesis*. Let us say that we accept H_1 if our observation is $\frac{\sum z_i}{n} \geq p$ and we accept H_0 if $\frac{\sum z_i}{n} < p$. By the requirement of algorithm $\mathcal{A}$, we want $Prob[\text{accept } H_1 \mid H_0 \text{ holds}] \leq \alpha$ and $Prob[\text{accept } H_0 \mid H_1 \text{ holds}] \leq \beta$. Hence, we want $Prob[\frac{\sum Z_i}{n} \geq p \mid p' <$

$\frac{p-\delta_1-\alpha}{1-\alpha}] \leq Prob[\frac{\sum Z_i}{n} \geq p \mid \frac{p''-\alpha}{1-\alpha} \leq \frac{p-\delta_1-\alpha}{1-\alpha}] = Prob[\frac{\sum Z_i}{n} \geq p \mid p'' < p - \delta_1] \leq Prob[\frac{\sum Z_i}{n} \geq p \mid p'' = p - \delta_1] \leq \alpha$. Similarly, we want $Prob[\frac{\sum Z_i}{n} < p \mid p'' = p + \delta_1] \leq \beta$. Note that $\sum Z_i$ is distributed binomially with parameters n and p''. We choose the smallest n such that the above requirements for $\mathcal{A}$ are satisfied.

3.3 Negation and Conjunction: $\mathcal{A}(s, \neg\phi, \alpha, \beta)$ and $\mathcal{A}(s, \phi_1 \wedge \phi_2, \alpha, \beta)$

For the verification of a formula $\neg\phi$ at a state s, we recursively verify ϕ at state s. If we know the decision of $\mathcal{A}$ for ϕ at s, we can say that $\mathcal{A}(s, \neg\phi, \alpha, \beta) = \neg\mathcal{A}(s, \phi, \beta, \alpha)$.

For conjunction, we first compute $\mathcal{A}(s, \phi_1, \alpha_1, \beta_1)$ and $\mathcal{A}(s, \phi_2, \alpha_2, \beta_2)$. If one of $\mathcal{A}(s, \phi_1, \alpha_1, \beta_1)$ or $\mathcal{A}(s, \phi_2, \alpha_2, \beta_2)$ is *false*, we say $\mathcal{A}(s, \phi_1 \wedge \phi_2, \alpha, \beta) = \mathit{false}$. Now:

$$
\begin{aligned}
& Prob[\mathcal{A}(s, \phi_1 \wedge \phi_2, \alpha, \beta) = \mathit{false} \mid s \models \phi_1 \wedge \phi_2]^1 \\
& = Prob[\mathcal{A}(s, \phi_1, \alpha_1, \beta_1) = \mathit{false} \vee \mathcal{A}(s, \phi_2, \alpha_2, \beta_2) = \mathit{false} \mid s \models \phi_1 \wedge \phi_2] \\
& \leq Prob[\mathcal{A}(s, \phi_1, \alpha_1, \beta_1) = \mathit{false} \mid s \models \phi_1 \wedge \phi_2] + Prob[\mathcal{A}(s, \phi_2, \alpha_2, \beta_2) = \mathit{false} \mid s \models \phi_1 \wedge \phi_2] \\
& = Prob[\mathcal{A}(s, \phi_1, \alpha_1, \beta_1) = \mathit{false} \mid s \models \phi_1] + Prob[\mathcal{A}(s, \phi_2, \alpha_2, \beta_2) = \mathit{false} \mid s \models \phi_2] \\
& \leq \beta_1 + \beta_2 = \beta \text{ [by the requirement R1 of } \mathcal{A}]
\end{aligned}
$$

The equality of the expressions in the third and fourth line of the above derivation follows from the fact that if $s \models \phi_1 \wedge \phi_2$ then the state s actually satisfies $\phi_1 \wedge \phi_2$; hence, $s \models \phi_1$ and $s \models \phi_2$. We set $\beta_1 = \beta_2 = \beta/2$.

If both $\mathcal{A}(s, \phi_1, \alpha_1, \beta_1)$ and $\mathcal{A}(s, \phi_2, \alpha_2, \beta_2)$ are *true*, we say $\mathcal{A}(s, \phi_1 \wedge \phi_2, \alpha, \beta) = \mathit{true}$. Then, we have

$$
\begin{aligned}
& Prob[\mathcal{A}(s, \phi_1 \wedge \phi_2, \alpha, \beta) = \mathit{true} \mid s \not\models \phi_1 \wedge \phi_2] \\
& \leq \max(Prob[\mathcal{A}(s, \phi_1 \wedge \phi_2, \alpha, \beta) = \mathit{true} \mid s \not\models \phi_1], Prob[\mathcal{A}(s, \phi_1 \wedge \phi_2, \alpha, \beta) = \mathit{true} \mid s \not\models \phi_2]) \\
& \leq \max(Prob[\mathcal{A}(s, \phi_1, \alpha_1, \beta_1) = \mathit{true} \mid s \not\models \phi_1], Prob[\mathcal{A}(s, \phi_2, \alpha_2, \beta_2) = \mathit{true} \mid s \not\models \phi_2] \\
& \leq \max(\alpha_1, \alpha_2)
\end{aligned}
$$

We set $\alpha_1 = \alpha_2 = \alpha$.

3.4 Unbounded Until: Computing $\mathcal{A}(\pi, \phi_1 \,\mathcal{U}\, \phi_2, \alpha, \beta)$

Consider the problem of checking if a path π satisfies an until formula $\phi_1 \,\mathcal{U}\, \phi_2$. We know that if π satisfies $\phi_1 \mathcal{U} \phi_2$ then there will be a finite prefix of π which will witness this satisfaction; namely, a finite prefix terminated by a state satisfying ϕ_2 and preceded only by states satisfying ϕ_1. On the other hand, if π does not satisfy $\phi_1 \mathcal{U} \phi_2$ then π may have no finite prefix witnessing this fact; in particular it is possible that π only visits states satisfying $\phi_1 \wedge \neg\phi_2$. Thus, to check the non-satisfaction of an until formula, it seems that we have to sample infinite paths.

Our first important observation in overcoming this challenge is to note that set of paths with non-zero measure that do not satisfy $\phi_1 \mathcal{U} \phi_2$ have finite prefixes that are terminated by states s from which there is *no* path satisfying $\phi_1 \,\mathcal{U}\, \phi_2$, i.e., $s \models \mathcal{P}_{=0}(\phi_1 \mathcal{U} \phi_2)$. We therefore set about trying to first address the problem of statistically verifying if a state s satisfies $\mathcal{P}_{=0}(\phi_1 \,\mathcal{U}\, \phi_2)$. It turns out that this special combination of a probabilistic operator and an unbounded until is indeed

[1] Note that this is not a conditional probability, because $s \models \phi_1 \wedge \phi_2$ is not an event.

easier to statistically verify. Observe that by sampling finite paths from a state s, we can witness the fact that s does not satisfy $\mathcal{P}_{=0}(\phi_1 \mathcal{U} \phi_2)$. Suppose we have a model that satisfies the following promise: either states satisfy $\mathcal{P}_{=0}(\phi_1 \mathcal{U} \phi_2)$ or states satisfy $\mathcal{P}_{>\delta}(\phi_1 \mathcal{U} \phi_2)$, for some positive real δ. Now, in this promise setting, if we sample an adequate number of finite paths and none of those witness the satisfaction then we can statistically conclude that the state satisfies $\mathcal{P}_{=0}(\phi_1 \mathcal{U} \phi_2)$ because we are guaranteed that either a significant fraction of paths will satisfy the until formula or none will.

There is one more challenge: we want to sample finite paths from a state s to check if $\phi_1 \mathcal{U} \phi_2$ is satisfied. However, we do not know *a priori* a bound on the lengths of paths that may satisfy the until formula. We provide a mechanism to sample finite paths of any length by sampling paths with a stopping probability.

We are now ready to present the details of our algorithm for the unbounded until operator. We first show how the special formula $\mathcal{P}_{=0}(\phi_1 \mathcal{U} \phi_2)$ can be statistically checked at a state. We then show how to use the algorithm for the special case to verify unbounded until formulas.

Computing $\boldsymbol{\mathcal{A}(s, \mathcal{P}_{=0}(\phi_1 \mathcal{U} \phi_2), \alpha, \beta)}$ To compute $\mathcal{A}(s, \mathcal{P}_{=0}(\phi_1 \mathcal{U} \phi_2), \alpha, \beta)$, we first compute $\mathcal{A}(s, \neg\phi_1 \wedge \neg\phi_2, \alpha, \beta)$. If the result is *true*, we say $\mathcal{A}(s, \mathcal{P}_{=0}(\phi_1 \mathcal{U} \phi_2), \alpha, \beta) = \textit{true}$. Otherwise, if the result is *false*, we have to check if the probability of a path from s satisfying $\phi_1 \mathcal{U} \phi_2$ is non-zero. For this we set up an experiment as follows.

Let p be the probability that a random path from s satisfies $\phi_1 \mathcal{U} \phi_2$. Let the *null hypothesis* be $H_0 \colon p > \delta_2$ and the *alternative hypothesis* be $H_1 \colon p = 0$ where δ_2 is the small real, close to 0, provided as parameter to the algorithm. The above test is one-sided: we can check the satisfaction of the formula $\phi_1 \mathcal{U} \phi_2$ along a path by looking at a finite prefix of a path; however, if along a path $\phi_1 \wedge \neg\phi_2$ holds only, we do not know when to stop and declare that the path does not satisfy $\phi_1 \mathcal{U} \phi_2$. Therefore, checking the violation of the formula along a path may not terminate if the formula is not satisfied by the path. To mitigate this problem, we modify the model by associating a stopping probability p_s with every state s in the model. While sampling a path from a state, we stop and return the path so far simulated with probability p_s. This allows one to generate paths of finite length from any state in the model.

Formally, we modify the model $\mathcal{M}$ as follows: we add a terminal state $s_\perp$ to the set S of all states of $\mathcal{M}$. Let $S' = S \cup \{s_\perp\}$. For every state $s \in S$, we define $\mathbf{P}(s, s_\perp) = p_s$, $\mathbf{P}(s_\perp, s_\perp) = 1$, and for every pair of states $s, s' \in S$, we modify $\mathbf{P}(s, s')$ to $\mathbf{P}(s, s')(1 - p_s)$. For every state $s \in S$, we pick some arbitrary probability distribution function for $\mathbf{Q}(s, s_\perp, t)$ and $\mathbf{Q}(s_\perp, s_\perp, t)$. We further assume that $L(s_\perp)$ is the set of atomic propositions such that $s_\perp \not\models \phi_2$. This in turn implies that any path (there is only one path) from $s_\perp$ do not satisfy $\phi_1 \mathcal{U} \phi_2$. Let us denote this modified model by $\mathcal{M}'$. Given this modified model, the following result holds:

Theorem 2. *If a path from any state $s \in S$ in the model $\mathcal{M}$ satisfies $\phi_1 \mathcal{U} \phi_2$ with some probability, p, then a path sampled from the same state in the modified model $\mathcal{M}'$ will satisfy the same formula with probability at least $p(1-p_s)^N$, where $N = |S|$.*

Proof is given in [16].

By condition C2 of algorithm $\mathcal{A}$, p does not lie in the range $(0, \frac{\delta_2}{(1-p_s)^N}]$. In other words, the modified probability $p(1-p_s)^N$ $(= p'$, say) of a path from s satisfying the formula $\phi_1 \,\mathcal{U}\, \phi_2$ does not lie in the range $(0, \delta_2]$. To take into account the modified model with stopping probability, we modify the experiment to test whether a path from s satisfies $\phi_1 \,\mathcal{U}\, \phi_2$ as follows. We change the *null hypothesis* to $H_0 \colon p' > \delta_2$ and the *alternative hypothesis* to $H_1 : p' = 0$.

Let n be the number of finite execution paths sampled from the state s in the modified model. Let $X_1, X_2, \ldots, X_n$ be a random sample having Bernoulli distribution with mean $p' \in [0,1]$ i.e., for each $j \in [1,n]$, $Prob[X_j = 1] = p'$. Then the sum $Y = X_1 + X_2 + \ldots + X_n$ has binomial distribution with parameters n and p'. We say that x_j, an observation of the random variable X_j, is 1 if the j^{th} sample path from s satisfies $\phi_1 \,\mathcal{U}\, \phi_2$ and 0 otherwise. In the experiment, we reject $H_0 \colon p' > \delta_2$ if $\frac{\sum x_j}{n} = 0$; otherwise, if $\frac{\sum x_j}{n} > 0$, we reject $H_1 \colon p' = 0$. Given the above experiment, to make sure that the error in decisions is bounded by α and β, we must have

$$\begin{array}{l} Prob[\text{accept } H_1 \mid H_0 \text{ holds}] = Prob[Y/n = 0 \mid p' > \delta_2] \leq \alpha \\ Prob[\text{accept } H_0 \mid H_1 \text{ holds}] = Prob[Y/n \geq 1 \mid p' = p] = 0 \leq \beta \end{array}$$

Hence, we can choose the unknown parameter n for this experiment such that $Prob[Y/n = 0 \mid p' > \delta_2] \leq Prob[Y/n = 0 \mid p' = \delta_2] \leq \alpha$ i.e., n is the smallest natural number such that $(1-\delta_2)^n \leq \alpha$.

Note that in the above analysis we assumed that $\phi_1 \,\mathcal{U}\, \phi_2$ has no nested probabilistic operators; therefore, it can be verified over a path without error. However, in the presence of nested probabilistic operators, we need to modify the experiment in a way similar to that given in section 3.2.

Computing $\mathcal{A}(\pi, \phi_1 \;\mathcal{U}\; \phi_2, \alpha, \beta)$ Once we know how to compute $\mathcal{A}(s, \mathcal{P}_{=0}(\phi_1 \,\mathcal{U}\, \phi_2), \alpha, \beta)$, we can give a procedure to compute $\mathcal{A}(\pi, \phi_1 \,\mathcal{U}\, \phi_2, \alpha, \beta)$ as follows. Let S be the set of states of the model. We partition S into the sets S^{true}, S^{false}, and $S^?$ and characterize the relevant probabilities as follows:

$$\begin{array}{ll} S^{true} = \{s \in S \mid & s \models \phi_2\} \\ S^{false} = \{s \in S \mid & \text{it is not the case that } \exists k \text{ and } \exists s_1 s_2 \ldots s_k \text{ such that } s = s_1 \\ & \text{and there is a non-zero probability of transition from } s_i \text{ to } s_{i+1} \text{ for } 1 \leq i < k \\ & \text{and } s_i \models \phi_1 \text{ for all } 1 \leq i < k, \text{ and } s_k \in S^{true}\} \\ S^? = S - S^{true} - S^{false} & \end{array}$$

Theorem 3.

$$\begin{array}{ll} Prob[\pi \in Path(s) \mid & \pi \models \phi_1 \,\mathcal{U}\, \phi_2] \\ = Prob[\pi \in Path(s) \mid & \exists k \textit{ and } s_1 s_2 \ldots s_k \textit{ such that } s_1 s_2 \ldots s_k \textit{ is a prefix of } \pi \textit{ and} \\ & s_1 = s \textit{ and } s_i \in S^? \textit{ for all } 1 \leq i < k \textit{ and } s_k \in S^{true}] \\ Prob[\pi \in Path(s) \mid & \pi \not\models \phi_1 \,\mathcal{U}\, \phi_2] \\ = Prob[\pi \in Path(s) \mid & \exists k \textit{ and } s_1 s_2 \ldots s_k \textit{ such that } s_1 s_2 \ldots s_k \textit{ is a prefix of } \pi \textit{ and} \\ & s_1 = s \textit{ and } s_i \in S^? \textit{ for all } 1 \leq i < k \textit{ and } s_k \in S^{false}] \end{array}$$

Proof of a similar theorem is given in [8].

Therefore, to check if a sample path $\pi = s_1 s_2 s_3 \ldots$ (ignoring the time-stamps on transitions) from state s satisfies (or violates) $\phi_1 \mathcal{U} \phi_2$, we need to find a k such that $s_k \in S^{true}$ (or $s_k \in S^{false}$) and for all $1 \leq i < k$, $s_i \in S^?$. This is done iteratively as follows:

$i \leftarrow 1$
<u>**while**</u>(<u>**true**</u>){
  <u>**if**</u> $s_i \in S^{true}$ <u>**then**</u> <u>**return**</u> *true*;
  <u>**else**</u> <u>**if**</u> $s_i \in S^{false}$ <u>**then**</u> <u>**return**</u> *false*;
  <u>**else**</u> $i \leftarrow i + 1$; }

The above procedure will terminate with probability 1 because, by Theorem 3, the probability of reaching a state in S^{true} or S^{false} after traversing a finite number of states in $S^?$ along a random path is 1.

To check whether a state s_i belongs to S^{true}, we compute $\mathcal{A}(s, \phi_2, \alpha_i, \beta_i)$; if the result is *true*, we say $s_i \in S^{true}$. The check for $s_i \in S^{false}$ is essentially computing $\mathcal{A}(s_i, \mathcal{P}_{=0}(\phi_1 \mathcal{U} \phi_2), \alpha_i, \beta_i)$. If the result is *true* then $s_i \in S^{false}$; else, we sample the next state s_{i+1} and repeat the loop as in the above pseudo-code.

The choice of α_i and β_i in the above decisions depends on the error bounds α and β with which we wanted to verify $\phi_1 \mathcal{U} \phi_2$ over the path π. By arguments similar to conjunction, it can be shown that we can choose each α_i and β_i such that $\alpha = \sum_{i \in [1,k]} \alpha_i$ and $\beta = \sum_{i \in [1,k]} \beta_i$ where k is the length of the prefix of π that has been used to compute $\mathcal{A}(\pi, \phi_1 \mathcal{U} \phi_2, \alpha, \beta)$. Since, we do not know the length k before-hand we choose to set $\alpha_i = \alpha/2^i$ and $\beta_i = \beta/2^i$ for $1 \leq i < k$, and $\alpha_k = \alpha/2^{k-1}$ and $\beta_k = \beta/2^{k-1}$.

An interesting and simple technique for the verification of the unbounded until proposed by H. Younes (personal communications) based on theorem 2 is as follows. Let p denote the probability measure of the set of paths that start in s and satisfy $\phi_1 \mathcal{U} \phi_2$. Let p' be the corresponding probability measure for the modified model with stopping probability p_s in each state. Then by theorem 2, we have $p \geq p' \geq p(1 - p_s)^N$, where N is the number of states in the model. These bounds on p can be used to verify the formula $\mathcal{P}_{\geq \theta}(\phi_1 \mathcal{U} \phi_2)$ in the same way as we deal with nested probabilistic operators.

However, there are trade-offs between these two approaches. The simple approach described in the last paragraph has the advantage of being conceptually clearer. The disadvantage of the simpler approach, on the other hand, is that we have to provide the exact value of N as input to the algorithm, which may not be available for a complex model. Our original algorithm does not expect the user to provide N; rather, it expects that the user will provide a suitable value of p_s so that condition C2 in theorem 1 holds. Moreover, the bounds on p' given in theorem 2 holds for the worst case. If we consider the worst case lower bound for p', which is dependent exponentially on N, then the value of p_s that needs to be picked to ensure that $\theta - \delta < (\theta + \delta)(1 - p_s)^N$ might be very small and sub-optimal resulting in large verification time. Note that our method for the verification of $\mathcal{P}_{=0}(\phi_1 \mathcal{U} \phi_2)$ can be used as a technique for verifying properties of the form $\mathcal{P}_{\geq 1}(\psi)$ and $\mathcal{P}_{\leq 0}(\psi)$ which were not handled by any previous statistical approaches.

3.5 Bounded Until: Computing $\mathcal{A}(\pi, \phi_1\, \mathcal{U}^{\leq t} \phi_2, \alpha, \beta)$

The satisfaction or violation of a bounded until formula $\phi_1\, \mathcal{U}^{\leq t} \phi_2$ over a path π can be checked by looking at a finite prefix of the path. Specifically, in the worst case, we need to consider all the states $\pi[i]$ such that $\tau(\pi, i) \leq t$. The decision procedure can be given as follows:

$i \leftarrow 0$
<u>while</u>(<u>true</u>){
  <u>if</u> $\tau(\pi, i) > t$ <u>then</u> <u>return</u> *false*;
  <u>else</u> <u>if</u> $\pi[i] \models \phi_2$ <u>then</u> <u>return</u> *true*;
  <u>else</u> <u>if</u> $\pi[i] \not\models \phi_1$ <u>then</u> <u>return</u> *false*;
  <u>else</u> $i \leftarrow i + 1$; }

where the checks $\pi[i] \models \phi_2$ and $\pi[i] \not\models \phi_1$ are replaced by $\mathcal{A}(\pi[i], \phi_2, \alpha_i, \beta_i)$ and $\mathcal{A}(\pi[i], \neg\phi_1, \alpha_i, \beta_i)$, respectively. The choice of α_i and β_i are done as in the case of unbounded until.

3.6 Bounded and Unbounded Next : Computing $\mathcal{A}(\pi, \mathrm{X}^{\leq t}\phi, \alpha, \beta)$ and $\mathcal{A}(\pi, \mathrm{X}\phi, \alpha, \beta)$

For unbounded next, $\mathcal{A}(\pi, \mathbf{X}\phi, \alpha, \beta)$ is same as the result of $\mathcal{A}(\pi[1], \phi, \alpha, \beta)$. For bounded next, $\mathcal{A}(\pi, \mathbf{X}^{\leq t}\phi, \alpha, \beta)$ returns *true* if $\mathcal{A}(\pi[1], \phi, \alpha, \beta) = \mathit{true}$ and $\tau(\pi, 1) \leq t$. Otherwise, $\mathcal{A}(\pi, \mathbf{X}^{\leq t}\phi, \alpha, \beta)$ returns *false*.

3.7 Computational Complexity

The expected length of the samples generated by the algorithm depends on the various probability distributions associated with the stochastic model in addition to the parameters $\alpha, \beta, p_s, \delta_1$, and δ_2. Therefore, an upper bound on the expected length of samples cannot be estimated without knowing the probability distributions associated with the stochastic model. This implies that the computational complexity analysis of our algorithm cannot be done in a model independent way. However, in the next section and in [16], we provide experimental results which illustrate the performance of the algorithm.

4 Implementation and Experimental Evaluation

We have implemented the above algorithm in Java as part of the tool called VeStA (available from `http://osl.cs.uiuc.edu/~ksen/vesta/`). A stochastic model can be specified by implementing a Java interface, called `State`. The model-checking module of VeStA implements the algorithm $\mathcal{A}$. It can be executed in two modes: single-threaded mode and multithreaded mode. The single threaded mode is suitable for a single processor machine; the multithreaded mode exploits the parallelism of the algorithm when executed on a multi-processor machine. While verifying a formula of the form $\mathcal{P}_{\bowtie p}(\psi)$, the verification of ψ over each sample path is independent of each other. This allows us to run the verification of ψ over each sample path in a separate thread, possibly running on a separate processor.

We successfully used the tool to verify several DTMC (discrete-time Markov chains) and CTMC (continuous-time Markov chains) models. We report the performance of our tool in the verification of unbounded until formulas over a DTMC model. The performance of our tool in verifying two CTMC model is provided in the [16]. The experiments were done on a single-processor 2GHz Pentium M laptop with 1GB SDRAM running Windows XP.

IPv4 ZeroConf Protocol: We picked the DTMC model of the IPv4 ZeroConf Protocol described in [6]. We next describe the model briefly without explaining its actual relation to the protocol. The DTMC model has $N+3$ states: $\{s_0, s_1, \ldots, s_n, ok, err\}$. From the initial state s_0, the system can go to two states: state s_1 with probability q and state ok with probability $1-q$. From each of the states s_i ($i \in [1, N-1]$) the system can go to two possible states: state s_{i+1} with probability r and state s_0 with probability $1-r$. From the state s_N the system can go to the state err with probability r or return to state s_0 with probability $1-r$. Let the atomic proposition a be true if the system is in the state err and false in any other state. The property that we considered is $\mathcal{P}_{\bowtie p}(true\,\mathcal{U}\,a)$.

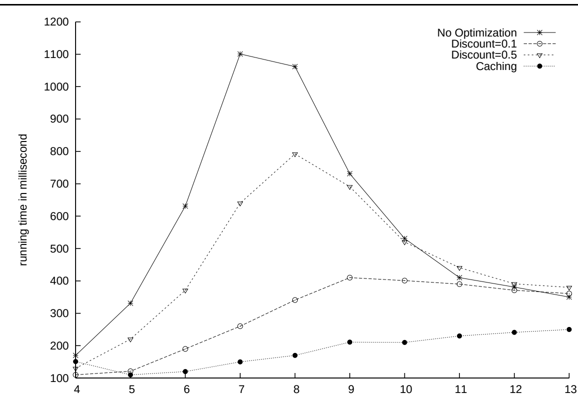

Fig. 1. Performance Measure for Verifying Unbounded Until Formula

The result of our experiment is plotted in Figure 1. In the plot x–axis represents N in the above model and y–axis represents the running time of the algorithm. The solid line represents the performance of the tool when it is used without any optimization. We noticed that computing $\mathcal{A}(s, \mathcal{P}_{=0}(\phi_1\,\mathcal{U}\,\phi_2), \alpha, \beta)$ at every state along a path while verifying an unbounded until formula has a large performance overhead. Therefore, we used the following optimization that reduces the number of times we compute $\mathcal{A}(s, \mathcal{P}_{=0}(\phi_1\,\mathcal{U}\,\phi_2), \alpha, \beta)$.

Discount Optimization: Instead of computing $\mathcal{A}(s, \mathcal{P}_{=0}(\phi_1\,\mathcal{U}\,\phi_2), \alpha, \beta)$ at every state along a path, we can opt to perform the computation with certain probability say $p_d = 0.1$, called *discount probability*. Note that once a path reaches a state $s \in S^{false}$, any other state following s in the path also belongs to S^{false}. Therefore, this way of *discounting the check* of $s \in S^{false}$, or computing $\mathcal{A}(s, \mathcal{P}_{=0}(\phi_1\,\mathcal{U}\,\phi_2), \alpha, \beta)$, does not influence the correctness of the algorithm. However, the average length of sample paths required to verify unbounded until increases. The modified algorithm for checking unbounded until becomes

$i \leftarrow 1$
<u>**while**</u>(<u>**true**</u>){
  <u>**if**</u> $s_i \in S^{true}$ <u>**then**</u> <u>**return**</u> *true*;
  <u>**else**</u> <u>**if**</u> $rand(0.0, 1.0) \leq p_d$ <u>**then**</u> <u>**if**</u> $s_i \in S^{false}$ <u>**then**</u> <u>**return**</u> *false*;
  <u>**else**</u> $i \leftarrow i + 1;$ }

The two dashed lines in the plot show the performance of the algorithm when the discount probability is $p_d = 0.1$ and $p_d = 0.5$.

Caching Optimization: If the algorithm has already computed and cached $\mathcal{A}(s, \phi, \alpha, \beta)$, any future computation of $\mathcal{A}(s, \phi, \alpha', \beta')$ can use the cached value provided that $\alpha \leq \alpha'$ and $\beta \leq \beta'$. However, note that we must maintain a constant size cache to avoid state-space explosion problem. The plot shows the performance of the tool with caching turned on (with no discount optimization).

The experiments show that the tool is able to handle a relatively large state space; it does not suffer from memory problem due to state-explosion because states are sampled as required and discarded when not needed. Specifically, it can be shown that the number of states stored in the memory at any time is linearly proportional to the maximum depth of nesting of probabilistic operators in a CSL formula. Thus the implementation can scale up with computing resources without suffering from traditional memory limitation due to state-explosion problem.

5 Conclusion

The statistical model-checking algorithm we have developed for stochastic models has at least four advantages over previous work. First, our algorithm can model check CSL formulas which have *unbounded untils*. Second, boundary case formulas of the form $\mathcal{P}_{\geq 1}(\psi)$ and $\mathcal{P}_{\leq 0}(\psi)$ can be verified using the technique presented for the verification of $\mathcal{P}_{=0}(\phi_1 \mathcal{U} \phi_2)$. Third, our algorithm is inherently parallel. Finally, the algorithm does not suffer from memory problem due to state-space explosion, since we do not need to store the intermediate states of an execution. However, our algorithm also has at least two limitations. First, the algorithm cannot guarantee the accuracy that numerical techniques achieve. Second, if we try to increase the accuracy by making the error bounds very small, the running time increases considerably. Thus our technique should be seen as an alternative to numerical techniques to be used only when it is infeasible to use numerical techniques, for example, in large-scale systems.

Acknowledgements

The second author was supported in part by DARPA/AFOSR MURI Award F49620-02-1-0325 and NSF 04-29639. The other two authors were supported in part by ONR Grant N00014-02-1-0715.

References

1. A. Aziz, K. Sanwal, V. Singhal, and R. K. Brayton. Verifying continuous-time Markov chains. In *Proc. of Computer Aided Verification (CAV'96)*, volume 1102 of *LNCS*, pages 269–276, 1996.

2. R. Alur, C. Courcoubetis, and D. Dill. Model-checking for probabilistic real-time systems (extended abstract). In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming (ICALP'91)*, volume 510 of *LNCS*, pages 115–126. Springer, 1991.
3. A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model-checking continuous-time Markov chains. *ACM Transactions on Computational Logic*, 1(1):162–170, 2000.
4. C. Baier, E. M. Clarke, V. Hartonas-Garmhausen, M. Z. Kwiatkowska, and M. Ryan. Symbolic model checking for probabilistic processes. In *Proc. of the 24th International Colloquium on Automata, Languages and Programming (ICALP'97)*, volume 1256 of *LNCS*, pages 430–440, 1997.
5. C. Baier, J. P. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In *International Conference on Concurrency Theory*, volume 1664 of *LNCS*, pages 146–161, 1999.
6. H. C. Bohnenkamp, P. van der Stok, H. Hermanns, and F. W. Vaandrager. Cost-optimization of the ipv4 zeroconf protocol. In *International Conference on Dependable Systems and Networks (DSN'03)*, pages 531–540. IEEE, 2003.
7. E. Cinlar. *Introduction to Stochastic Processes*. Prentice-Hall Inc., 1975.
8. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of ACM*, 42(4):857–907, 1995.
9. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
10. H. Hermanns, J. P. Katoen, J. Meyer-Kayser, and M. Siegle. A Markov chain model checker. In *Tools and Algorithms for Construction and Analysis of Systems (TACAS'00)*, pages 347–362, 2000.
11. R. V. Hogg and A. T. Craig. *Introduction to Mathematical Statistics*. Macmillan, New York, NY, USA, fourth edition, 1978.
12. M. Z. Kwiatkowska, G. Norman, and D. Parker. Prism: Probabilistic symbolic model checker, 2002.
13. M. Z. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Verifying quantitative properties of continuous probabilistic timed automata. In *Conference on Concurrency Theory (CONCUR'00)*, volume 1877 of *LNCS*, pages 123–137, 2000.
14. G. G. I. López, H. Hermanns, and J.-P. Katoen. Beyond memoryless distributions: Model checking semi-markov chains. In *Proceedings of the Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification*, volume 2165 of *LNCS*, pages 57–70. Springer-Verlag, 2001.
15. K. Sen, M. Viswanathan, and G. Agha. Statistical model checking of black-box probabilistic systems. In *16th conference on Computer Aided Verification (CAV'04)*, volume 3114 of *LNCS*, pages 202–215. Springer, July 2004.
16. K. Sen, M. Viswanathan, and G. Agha. On statistical model checking of probabilistic systems. Technical Report UIUCDCS-R-2004-2503, University of Illinois at Urbana Champaign, 2005.
17. W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton, 1994.
18. H. L. S. Younes and R. G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *Proc. of Computer Aided Verification (CAV'02)*, volume 2404 of *LNCS*, pages 223–235, 2002.