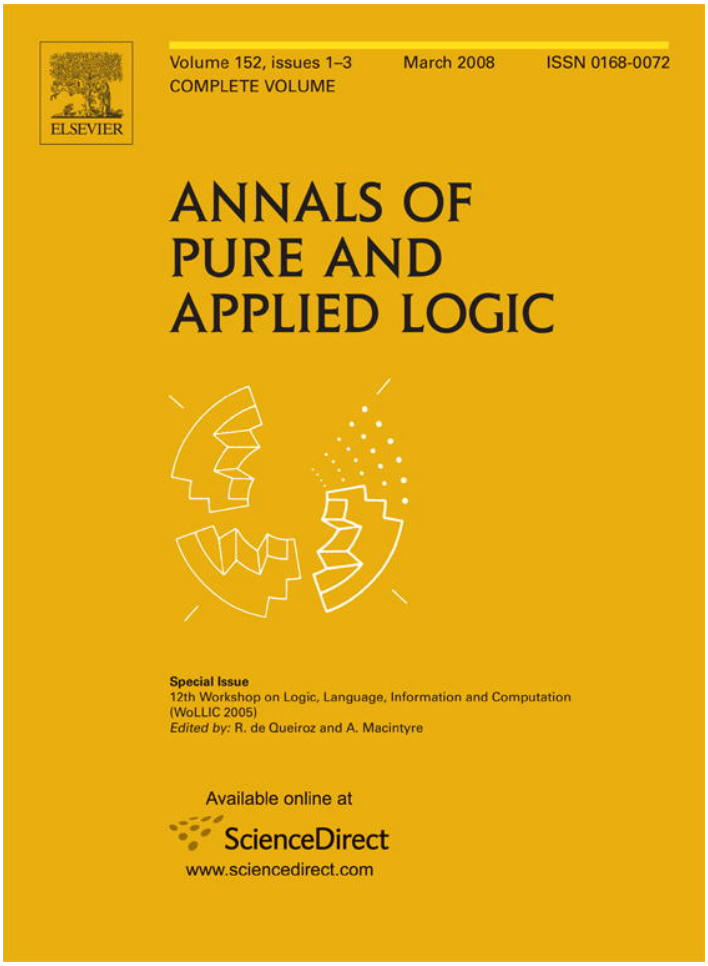

Available online at www.sciencedirect.com
ScienceDirect
Annals of Pure and Applied Logic 152 (2008) 122–131
ANNALS OF PURE AND APPLIED LOGIC
www.elsevier.com/locate/apal

Probabilistic verification and approximation

Richard Lassaigne[a,*], Sylvain Peyronnet[a,b]

[a] *Equipe de Logique, UMR 7056 CNRS, University Paris VII, France*
[b] *Laboratoire d'Informatique de l'Ecole Polytechnique (LIX), France*

Available online 20 December 2007

Abstract

We study the existence of efficient approximation methods to verify quantitative specifications of probabilistic systems. Models of such systems are labelled discrete time Markov chains and checking specifications consists of computing satisfaction probabilities of linear temporal logic formulas. We prove that, in general, there is no polynomial time randomized approximation scheme with relative error for probabilistic verification. However, in many applications, specifications can be expressed by monotone formulas or negation of monotone formulas and randomized approximation schemes with absolute error are sufficient. We show how some simple randomized approximation algorithms can improve the efficiency of verification of such probabilistic specifications and can be implemented in a probabilistic model checker.

MSC: 68Q60; 68W20

Keywords: Probabilistic verification; Randomized approximation algorithms

1. Introduction

Model checking has been successfully used to verify if the behavior of concurrent and reactive systems satisfy some correctness properties, which are usually expressed in temporal logic. While many important system properties can be studied in this framework, others such as reliability and performance, require instead a probabilistic characterization of the system.

In classical model checking, time complexity is linear in the size of the model and in the size of formula for CTL branching time temporal logic [4], and exponential in the size of formula for LTL linear time temporal logic [21]. In many applications, as protocol verification, the representation of systems by Kripke models leads to a "state explosion phenomenon" and symbolic representation methods of the transition system and of the formula have been introduced to overcome this problem.

For probabilistic verification, time complexity is in general polynomial in the size of the model and polynomial or exponential in the size of the formula [5]. As well as for classical model checking, symbolic representation methods have been generalized in the probabilistic framework. However, in spite of using these techniques, for many examples the verification is limited to small values of the main parameters of the model, due to the space complexity of

* Corresponding author.
E-mail addresses: lassaign@logique.jussieu.fr (R. Lassaigne), syp@lix.polytechnique.fr (S. Peyronnet).

0168-0072/$ - see front matter
doi:10.1016/j.apal.2007.11.006

the representation. Thus it seems natural to wonder whether some approximation algorithms could be useful for probabilistic verification.

Since 2001, we have been working on randomized approximation schemes to compute satisfaction probabilities of interesting properties such as reachability and safety. As a consequence, a natural question is: can we efficiently approximate satisfaction probability of an LTL formula?

Unfortunately, there are serious complexity reasons to think that this is not the case in general. One of the main results of this paper is the non-approximability of computing satisfaction probability of a given *LTL* formula in an efficient way, unless some unlikely complexity hypothesis is satisfied.

Nevertheless, there exists some efficient algorithms to approximate model checking of expressive classes of *LTL* formulas against probabilistic transition systems. These randomized algorithms allow to obtain two types of results, of independent interest for practical verification:

- polynomial time randomized approximation of bounded-time satisfaction probabilities,
- randomized approximation of non-bounded satisfaction probabilities.

Although in the last case we cannot assure polynomial time in the size of a succinct representation, the main advantage is to eliminate space complexity by using a Monte Carlo method and efficiently bounding sample size. Moreover this approach is highly parallelizable and can be integrated in a classical probabilistic model checker.

Our paper is structured as follows: Section 2 is a review of the main complexity results in probabilistic verification. In Section 3, we recall the linear temporal logical framework for probabilistic verification and we show the hardness of approximating satisfaction probabilities of *LTL* formulas. In Section 4, we restrict the specification language to monotone properties, or negation of monotone properties, and we give a randomized approximation algorithm to compute satisfaction probabilities of formulas expressing such properties.

2. Classical probabilistic verification

2.1. Qualitative verification

The first application of verification methods to probabilistic systems consists in checking if temporal properties are satisfied with probability 1 by systems modeled either as finite discrete time Markov chains or as Markov models enriched with nondeterministic behavior. In the following, the former systems will be denoted by probabilistic sytems and the latter by concurrent probabilistic systems.

In [27], Vardi presented the first method to verify if a linear time temporal property is satisfied by almost all computations of a concurrent probabilistic system. This automata-theoretic method is expensive, since its complexity is doubly exponential in the size of the formula.

The complexity of this work was later addressed by Courcoubetis and Yannakakis [5]. A new model checking method for probabilistic systems was introduced, the complexity of which was proved polynomial in the size of the system and exponential in the size of the formula. For concurrent probabilistic systems they presented an automata-theoretic approach which improved on the Vardi's method by a single exponential in the size of the formula.

2.2. Quantitative verification

The Courcoubetis and Yannakakis's method [5] allows to compute the probability that a probabilistic system, modeled as a Markov chain, satisfies some given linear time temporal formula. The algorithm transforms step by step the Markov chain and the formula, eliminating one-by-one the temporal connectives, while preserving the satisfaction probability of the formula. The elimination of temporal connectives is performed by solving a linear system of equations of the size of the Markov chain.

A temporal logic for the specification of quantitative properties, which refer to a bound of the probability of satisfaction of a formula, is given by Hansson and Jonsson [11]. The authors introduced the logic *PCTL*, which is an extension of branching time temporal logic CTL with some probabilistic quantifiers. A model checking algorithm is also presented: the computation of probabilities for formulas involving probabilistic quantification is performed by solving a linear system of equations.

A model checking method for *PCTL* properties against concurrent probabilistic systems is given by Bianco and de Alfaro [1]. Probabilities are computed by solving an optimisation problem over a system of linear inequalities, the size of which is the model size.

In order to illustrate space complexity problems, we mention the main model checking tool for the verification of quantitative properties. The probabilistic model checker PRISM [7] was designed by the Kwiatkowska's team in Birmingham and allows to check *PCTL* formulas on probabilistic or concurrent probabilistic systems. In many applications, Markov chains are very large and the computation is intractable. Thus the use of PRISM is often limited to small values of model parameters for reason of memory overflow: for example, $N = 25$ for the dining philosophers problem [13].

For theoretical and practical reasons, it is natural to ask the question: **can probabilistic verification be efficiently approximated?** In the following, we study some possible answers for fully probabilistic systems and linear time temporal logic.

We chose to study probabilistic verification in the context of linear time temporal logic for two reasons. The first one is historical: the complexity of probabilistic verification was first studied by Vardi, Coucourbetis and Yannakakis in this logical framework. The second one is practical: in real-life case studies, quantitative specifications to be checked need only to apply one probabilistic operator to a temporal formula. Moreover, it can be useful to obtain a good approximation of the desired probability, and not only the comparison with a given threshold as in the logic *PCTL*.

3. Probabilistic verification and approximation

In this section, we present the linear temporal logical framework which allows to express quantitative properties of probabilistic systems, the results of Courcoubetis and Yannakakis and the hardness of efficiently approximating such probabilities.

3.1. Probabilistic verification

A Discrete Time Markov Chain (DTMC) is a pair $\mathcal{M} = (S, P)$ where S is a finite or countable set of states and $P : S \times S \to [0, 1]$ is a transition probability function, i.e. for all $s \in S$, $\sum_{t \in S} P(s, t) = 1$. In the following, the set of states S is finite and we consider P as a probability matrix.

Definition 1. A probabilistic transition system (*PTS*) is a structure $\mathcal{M} = (S, P, s_0, L)$ given by a Discrete Time Markov chain (S, P) with an initial state s_0 and a function $L : S \to \mathcal{P}(AP)$ labelling each state with a set of atomic propositions in AP.

In linear-time logic *LTL*, formulas are composed from the set AP of atomic propositions by using the Boolean connectives and the temporal operators **X** (*next*) and **U** (*until*). The usual operators **F** (*eventually*) and **G** (*always*) can also be defined. *LTL* formulas are interpreted over paths of a transition system $\mathcal{M}$. A path σ is a finite or countable sequence of states $(s_0, s_1, \ldots, s_i, \ldots)$ such that $P(s_i, s_{i+1}) > 0$ for all $i \geq 0$. We note also that $\sigma(i)$ the $(i + 1)$th state of the path σ and σ^i the path $(\sigma(i), \sigma(i + 1), \ldots)$. We denote by $Path(s)$ the set of paths whose first state is s.

For each structure $\mathcal{M}$ and state s, it is possible to define a probability measure Prob on the set $Path(s)$: for any finite path $\pi = (s_0, s_1, \ldots, s_n)$, the measure is defined by $\text{Prob}(\{\sigma/\sigma \text{ is a path and } (s_0, s_1, \ldots, s_n) \text{ is a prefix of } \sigma\}) = \prod_{i=1}^{n} P(s_{i-1}, s_i)$. This measure can be extended uniquely to the Borel family of sets generated by the sets $\{\sigma/\pi \text{ is a prefix of } \sigma\}$ where π is a finite path.

In [27], it is shown that for any *LTL* formula ψ, probabilistic transition system $\mathcal{M}$ and state s, the set of paths $\{\sigma/\sigma(0) = s \text{ and } \mathcal{M}, \sigma \models \psi\}$ is measurable. We denote by $\text{Prob}[\psi]$ the measure of this set. We say that the probabilistic transition system $\mathcal{M}$ satisfy the formula ψ if $\text{Prob}[\psi] = 1$, i.e. if almost all paths in $\mathcal{M}$, whose origin is the initial state, satisfy ψ.

Theorem 1 (*[5]*). *The satisfaction of an LTL formula ψ by a probabilistic transition system $\mathcal{M}$ can be decided in time linear in size of $\mathcal{M}$ and exponential in size of ψ, or in space polylogarithmic in size of $\mathcal{M}$ and polynomial in size of ψ.*
The probability $\text{Prob}[\psi]$ *can be computed in time polynomial in size of $\mathcal{M}$ and exponential in size of ψ.*

The first part of the theorem is about qualitative properties, the second one concerns quantitative properties. In the following, we study quantitative properties and show, in the next subsection, that the problem of computing the number of truth assignments satisfying a propositional formula in conjunctive normal form ($\sharp SAT$) can be reduced to the problem of counting finite paths satisfying *LTL* formulas. Therefore it is unlikely to obtain an approximation algorithm for computing $\text{Prob}[\psi]$ in general.

3.2. Counting problems and approximation

Intuitively, the class $\sharp P$ captures the problems of counting the number of solutions to *NP* problems. The counting versions of all known *NP*-complete problems are $\sharp P$-complete. The well-adapted notion of reduction is parsimonious reduction: it is a polynomial time reduction from the first problem to the second one, recovering via some oracle, the number of solutions for the first problem from the number of solutions for the second one.

No deterministic approximation algorithms are known for $\sharp P$-complete problems. However, randomized versions of such approximation algorithms exist for problems such as counting the number of valuations satisfying a propositional disjunctive normal form formula ($\sharp DNF$) [19] or network reliability problem [17]. We introduce the notion of polynomial randomized approximation scheme which is due to Karp and Luby [18]. Consider a counting problem and let $\#(x)$ be the number of distinct solutions for an instance x of this problem. We note $|x|$ the size of this instance.

Definition 2. A randomized approximation scheme (RAS) for a counting problem is a randomized algorithm $\mathcal{A}$ that takes an input x, two real numbers $\varepsilon, \delta > 0$ and produces a value $A(x, \varepsilon, \delta)$ such that:

$$Pr\big(|A(x, \varepsilon, \delta) - \#(x)| \leq \varepsilon.\#(x)\big) \geq 1 - \delta.$$

A randomized approximation scheme is said to be fully polynomial (FPRAS) [18] if its running time is polynomially bounded in $|x|$, $\frac{1}{\varepsilon}$ and $\log(\frac{1}{\delta})$.

The probability Pr is taken over the random choices of the algorithm. We call ε the *error parameter* and $(1 - \delta)$ the *confidence parameter*. We remark that the error parameter is a multiplicative parameter taking into account the size of $\#(x)$.

We adapt the notion of randomized approximation scheme to the problem of computing probabilities. A probability problem is defined by giving as input a succinct representation of a probabilistic system, a property x and as output the probability measure $\mu(x)$ of the measurable set of execution paths satisfying this property.

Definition 3. A randomized approximation scheme for a probability problem is a randomized algorithm $\mathcal{A}$ that takes an input x, two real numbers $\varepsilon, \delta > 0$ and produces a value $A(x, \varepsilon, \delta)$ such that:

$$Pr\big(|A(x, \varepsilon, \delta) - \mu(x)| < \varepsilon.\mu(x)\big) \geq 1 - \delta.$$

If the running time of $\mathcal{A}$ is polynomial in $|x|$, $\frac{1}{\varepsilon}$ and $\log(\frac{1}{\delta})$, $\mathcal{A}$ is said to be fully polynomial.

In order to obtain the non-approximability result for the general case of probabilistic verification, we recall the randomized complexity class *BPP* which corresponds to the useful class of two-sided error randomized algorithms. Let Σ be some finite alphabet and Σ^* be the set of strings over Σ.

Definition 4. The class *BPP*, for Bounded-error Probabilistic Polynomial time, consists of all languages L that have a randomized polynomial time algorithm $\mathcal{A}$ such that for any input $x \in \Sigma^*$,

- if $x \in L$ then $Pr\big(\mathcal{A}(x) \text{ accepts}\big) \geq \frac{3}{4}$,
- if $x \notin L$ then $Pr\big(\mathcal{A}(x) \text{ accepts}\big) \leq \frac{1}{4}$.

Theorem 2. *If there is a fully polynomial randomized approximation scheme for the problem of computing* $\text{Prob}[\psi]$ *for LTL formula* ψ*, then* $NP \subseteq BPP$.

Proof. We consider the fragment $L(\mathbf{F})$ of *LTL* in which $\mathbf{F}$ is the only temporal operator. The following result is due to Clarke and Sistla [26]: the problem of deciding the existence of some path satisfying a $L(\mathbf{F})$ formula in a transition system is *NP*-complete. Their proof uses a polynomial time reduction of SAT to the problem of deciding satisfaction of $L(\mathbf{F})$ formulas.

From this reduction, we can obtain a one to one reduction between problem $\sharp SAT$ and counting finite paths, of given length, whose extensions satisfy the associated $L(\mathbf{F})$ formula.

Let $\bigwedge_{j=1}^{m} c_j$ be some SAT formula, built with the propositional variables $\{x_i \ / \ i = 1, \ldots, n\}$. Let $\mathcal{M} = (S, P, y_0, L)$ be the following probabilistic transition system:

- $AP = \{c_j \ / \ j = 1, \ldots, m\}$ (set of atomic propositions),
- $S = \{x_i \ / \ i = 1, \ldots, n\} \cup \{x'_i \ / \ i = 1, \ldots, n\} \cup \{y_i \ / \ i = 0, \ldots, n\}$,
- $P(y_{i-1}, x_i) = 1/2$, $P(y_{i-1}, x'_i) = 1/2$ for $i = 1, \ldots, n$,
  $P(x_i, y_i) = 1$, $P(x'_i, y_i) = 1$ for $i = 1, \ldots, n$, $P(y_n, y_n) = 1$,
- $L(x_i) = \{c_j \ / \ x_i$ appears in $c_j\}$ for $i = 1, \ldots, n$,
  $L(x'_i) = \{c_j \ / \ \neg x_i$ appears in $c_j\}$ for $i = 1, \ldots, n$.

Now we consider the *LTL* formula $\bigwedge_{j=1}^{m} \mathbf{F} c_j$. It is easy to see that an assignment satisfying $\bigwedge_{j=1}^{m} c_j$ corresponds to a finite path from y_0 to y_n whose all infinite extensions satisfy the *LTL* formula $\bigwedge_{j=1}^{m} \mathbf{F} c_j$. Therefore, the problem of counting finite paths, of given length, satisfying $L(\mathbf{F})$ formulas is $\sharp P$-complete.

A consequence of this result is the $\sharp P$-hardness of computing probabilities of satisfaction for general *LTL* formulas. Then we remark that if there was a FPRAS for approximating Prob$[\psi]$ for *LTL* formula ψ, we could efficiently approximate $\sharp SAT$.

A polynomial randomized approximation scheme for $\sharp SAT$ could be used to distinguish, for input x, between the case $\sharp(x) = 0$ and the case $\sharp(x) > 0$, thereby implying a randomized polynomial time algorithm for the decision problem SAT. □

As a consequence of a result of Jerrum, Valiant et Vazirani [16] and a remark of Sinclair [25], the existence of an FPRAS for $\sharp SAT$ would also imply $RP = NP$, where RP is the class of languages that admit one-sided error polynomial time randomized algorithms.

4. Approximate probabilistic model checking

In this section, we determine some restrictions on the class of linear temporal formulas ψ, on the value $p =$ Prob$[\psi]$ and on the type of approximation (additive error) in order to obtain efficient randomized approximation schemes for computing such probabilities.

4.1. A positive fragment of LTL

For many natural properties, as reachability, satisfaction by an execution path of finite length implies satisfaction by any extension of this path. Such properties are called monotone. Another important class of properties, namely safety properties, can be expressed as negation of monotone properties. We can reduce the computation of satisfaction probability of a safety property to the same problem for its negation, that is a monotone property.

In the following, we consider a subset of *LTL* formulas which allows to express only monotone properties and for which we can approximate satisfaction probabilities. The *essentially positive fragment* (*EPF*) of *LTL* is the set of formulas constructed from atomic formulas (p) and their negations ($\neg p$), closed under $\vee$, $\wedge$ and the temporal operators $\mathbf{X}$, $\mathbf{U}$.

Definition 5. The set of *EPF* formulas can be defined by induction:

- for all $p \in AP$, p and $\neg p$ are *EPF* formulas,
- if Φ and Ψ are *EPF* formulas, then so are $\Phi \vee \Psi$ and $\Phi \wedge \Psi$,
- if Φ and Ψ are *EPF* formulas, then so are $\mathbf{X}\Phi$ and $\Phi \mathbf{U} \Psi$.

For example, formula $\mathbf{F}p$, that expresses a reachability property, is an *EPF* formula. Formula $\mathbf{G}p$, that expresses a safety property, is equivalent to $\neg\mathbf{F}\neg p$, which is the negation of an *EPF* formula. Formula $\mathbf{G}(p \rightarrow \mathbf{F}q)$, that expresses a liveness property, is not an *EPF* formula, nor equivalent to the negation of an *EPF* formula.
In order to approximate the satisfaction probability Prob$[\psi]$ of an *EPF* formula, we first consider Prob$_k[\psi]$, the probability measure associated to the probabilistic space of execution paths of finite length k. The monotonicity of the property defined by an *EPF* formula gives the following result.

Proposition 1. *Let ψ be an LTL formula of the essentially positive fragment and $\mathcal{M}$ be a probabilistic transition system. Then the sequence* (Prob$_k[\psi]$)$_{k\in\mathbb{N}}$ *converges to* Prob$[\psi]$.

A first idea is to approximate $\text{Prob}_k[\psi]$ and to use a fixed point algorithm to obtain an approximation of $\text{Prob}[\psi]$. This approximation problem is believed to be intractable for deterministic algorithms. In the next section, we give a randomized approximation algorithm to compute $\text{Prob}_k[\psi]$, whose running time is polynomial in the size of a succinct representation of the system and of the formula. Then we deduce a randomized approximation algorithm to compute $\text{Prob}[\psi]$, whose space complexity is logspace.

4.2. Randomized approximation scheme with additive error

We show that we can approximate the satisfaction probability of an *EPF* formula with a simple randomized algorithm. As in practice, randomized approximation with additive error is sufficient and gives simple algorithms, we first explain how to design it. Moreover, this randomized approximation is fully polynomial for bounded properties. Then we will use the estimator theorem [19] and an optimal approximation algorithm [6] in order to obtain randomized approximation scheme with multiplicative error, according to definition 3. However, in this case the randomized approximation is not fully polynomial even for bounded properties.

We generate random paths in the probabilistic space underlying the Kripke structure of depth k and compute a random variable Y which approximates $p = \text{Prob}_k[\psi]$. Our approximation will be correct, i.e $|Y - p| < \varepsilon$ (additive error), with confidence $(1 - \delta)$, after a polynomial number of samples in $\frac{1}{\epsilon}$, $\log \frac{1}{\delta}$. This result is obtained by using the Chernoff–Hoeffding bounds [15] on the tail of the distribution of a sum of independent random variables.

The main advantage of the method is that we can proceed with just a succinct representation of the transition system in some input language, thus eliminating the space complexity problem.

Definition 6. A succinct representation, or diagram, of a *PTS* $\mathcal{M} = (S, P, s_0, L)$ is a representation of the *PTS*, that allows to generate algorithmically, for any state s, the set of states t such that $P(s, t) > 0$.

The size of such a succinct representation is subtantially lower than the size of the corresponding *PTS*. For example, a succinct representation is given by a program representing the *PTS* in the input language of probabilistic models checkers such as PRISM or APMC [13]. Typically the size of a diagram is polylogarithmic in the size of the *PTS*.

The following function **Random Path** uses this succinct representation to generate a random path of length k and to check the formula ψ:

Random Path
Input: $diagram_{\mathcal{M}}, k, \psi$
Output: samples a path π of length k and check formula ψ on π
(1) Generate a random path π of length k (with the diagram)
(2) If ψ is true on π then return 1 else 0

Consider now the random sampling algorithm $\mathcal{GAA}$ designed for the approximate computation of $\text{Prob}_k[\psi]$:

Generic approximation algorithm $\mathcal{GAA}$
Input: $diagram_{\mathcal{M}}, k, \psi, \epsilon, \delta$
Output: approximation of $\text{Prob}_k[\psi]$
$N := \ln(\frac{2}{\delta})/2\varepsilon^2$
$A := 0$
For $i = 1$ to N do
$A := A +$ **Random Path**$(diagram_{\mathcal{M}}, k, \psi)$
Return $Y = A/N$

Theorem 3. *The generic approximation algorithm $\mathcal{GAA}$ is a fully polynomial randomized approximation scheme (with additive error parameter) for computing $p = \text{Prob}_k[\psi]$ whenever ψ is in the EPF fragment of LTL and $p \in]0, 1[$.*

Proof. We want to estimate $p = \text{Prob}_k[\psi]$ by a random sampling of size N. The random variable A is the sum of random variables with a Bernouilli distribution. We use the Chernoff–Hoeffding bound [15] to obtain the result. Let $X_1, \ldots, X_N$ be N independent random variables which take value 1 with probability p and 0 with

probability $(1-p)$, and $Y = \sum_{i=1}^{N} X_i/N$. Then the Chernoff bound gives: $\text{Prob}\big(Y < (p-\varepsilon)\big) < \mathrm{e}^{-2N\cdot\varepsilon^2}$ and $\text{Prob}\big(Y > (p+\varepsilon)\big) < \mathrm{e}^{-2N\cdot\varepsilon^2}$.

In the previous algorithm, if $N \geq \frac{\log(\frac{2}{\delta})}{2\varepsilon^2}$, then: $\text{Prob}\big((p-\varepsilon) \leq A/N \leq (p+\varepsilon)\big) \geq 1-\delta$. □

We can obtain a randomized approximation of $\text{Prob}[\psi]$ by iterating the approximation algorithm described above. Detection of time convergence for this algorithm is hard in general and we will study the particular case of primitive Markov chains, that is important for applications. Another feature, which is crucial for applications, is that space complexity is logspace.

Corollary 1. *The fixed point algorithm defined by iterating the approximation algorithm $\mathcal{GAA}$ is a randomized approximation scheme, whose space complexity is logspace, for the probability problem $p = \text{Prob}[\psi]$ whenever ψ is in the EPF fragment of LTL and $p \in]0, 1[$.*

Proof. In order to compute an ε-approximation of $\text{Prob}_k[\psi]$, we have to store the $N := \ln(\frac{2}{\delta})/2\varepsilon^2$ Boolean results of the **Random Path** function. The space needed is $log\ N$ and is independent of an appropriate bound for the length k of generated execution paths to obtain a good approximation of $\text{Prob}[\psi]$. □

In the following, we recall some basic facts of Markov chain theory [2] and we give a speed convergence criterion for the important case of irreducible, aperiodic Markov chains. Let P be the transition matrix of a finite discrete-time Markov chain. A Markov chain is called irreducible if the underlying graph is strongly connected. A Markov chain is called primitive if some power of the transition matrix has only positive elements. It is well known that a Markov chain is primitive if and only if it is irreducible and aperiodic [2].

Proposition 2. *Let P be the transition matrix of a primitive Markov chain and v an arbitrary probability vector. Then $lim_{k\to\infty} vP^k = u$ where u is the unique left eigenvector associated to the eigenvalue $\lambda = 1$.*

If we start a Markov chain with initial probabilities given by v, then the probability vector vP^k gives the probabilities of being in the various states after k steps. The previous proposition establishes the fact that for this general class of processes, the probability of being in state s_j is u_j.
The following theorem, which is known as the Perron–Frobenius theorem, is a classical result of matrix theory [24].

Theorem 4. *Let P be a non-negative primitive matrix of order n. There exists a real eigenvalue λ_1 with multiplicity 1 such that $\lambda_1 > 0$ and $\lambda_1 > |\lambda_j|$ for any other eigenvalue λ_j. Let $\lambda_2, \lambda_3, \ldots, \lambda_n$ the eigenvalues of P other than λ_1 ordered in such a way that $\lambda_1 > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|$ and if $|\lambda_2| = |\lambda_j|$ for some $j \geq 3$, then $m_2 \geq m_j$, where m_j is the multiplicity of λ_j. Then*

$$P^k = \lambda_1^k vu + O(k^{m_2-1}|\lambda_2|^k)$$

, where u, v are the left and the right eigenvectors associated with λ_1.
If in addition, P is a Markov matrix, then $\lambda_1 = 1$.

Thus the convergence rate is geometrical and the question is to determine k such that $O(k^{m_2-1}|\lambda_2|^k) \leq \varepsilon$. If the multiplicity of the second eigenvalue λ_2 is one, the number k of iterations sufficient to obtain an ε-approximation is $O(ln(1/\varepsilon))$. In general, there is no known polynomial time convergence criterion. The probabilistic model checker APMC, that implements the previous randomized approximation scheme, uses a practical stopping criterion as explained in the APMC section. Number of iterations can be exponential in the size of a succinct representation of the model. However, complexity space is logspace and the approximation method can be used to verify very large systems [13].

4.3. Randomized approximation scheme with multiplicative error

We use a generalization of the zero-one estimator theorem [19] to estimate the expectation μ of a random variable X distributed in the interval [0, 1]. The generalized zero-one estimator theorem [6] proves that if the random

variables $X_1, X_2, \ldots, X_N$ are independent and identically distributed according to X, $S = \sum_{i=1}^{N} X_i$, $\epsilon < 1$, and $N = 4(e-2).\ln(\frac{2}{\delta}).\rho/(\epsilon.\mu)^2$, then S/N is an (ϵ, δ)-approximation of μ, i.e.:

$$Pr\big(\mu(1-\epsilon) \leq S/N \leq \mu(1+\epsilon)\big) \geq 1-\delta$$

, where $\rho = max(\sigma^2, \epsilon\mu)$ is a parameter used to optimize the number N of experiments and σ^2 denotes the variance of X.

In [6], an optimal approximation algorithm, running in three steps, is described:

- using a stopping rule, the first step outputs an (ϵ, δ)-approximation $\hat{\mu}$ of μ after expected number of experiments proportional to Γ/μ, where $\Gamma = 4(e-2).\ln(\frac{2}{\delta})/\epsilon^2$,
- the second step uses the value of $\hat{\mu}$ to set the number of experiments in order to produce an estimate $\hat{\rho}$ that is within a constant factor of ρ with probability at least $(1-\delta)$,
- the third step uses the values of $\hat{\mu}$ and $\hat{\rho}$ to set the number of experiments and runs the experiments to produce an (ϵ, δ)-approximation of μ.

We obtain a randomized approximation scheme with multiplicative error by applying the optimal approximation algorithm $\mathcal{OAA}$ with input parameters ϵ, δ and the sample given by the function **Random Path** to a succinct representation of $\mathcal{M}$, the parameter k and the formula ψ.

Theorem 5. *The optimal approximation algorithm $\mathcal{OAA}$ is a randomized approximation scheme (with multiplicative error parameter) for computing $p = \mathrm{Prob}_k[\psi]$ whenever ψ is in the EPF fragment of LTL and $p \in]0, 1[$.*

The optimal approximation algorithm was also used by Grosu and Smolka to approximate classical model checking [9]. We remark that the optimal approximation algorithm is not an *FPRAS* as the expected number of experiments Γ/μ can be exponential for small values of μ.

4.4. APMC: An implementation

In 2003, we started the design of a tool that implements the randomized approximation scheme with additive error described in this paper. This tool [13], called APMC for Approximate Probabilistic Model Checker, is freely available under GPL (Gnu Public License). APMC is now a probabilistic distributed model checker that uses a distributed computation model in order to speed up the verification process by distributing the **Random Path** function on a cluster of workstations [12]. Extensive tests with hundreds of machines were done. We have also collaborated with the Kwiatkowska's team to achieve the integration of APMC with PRISM.

Now we would like to explain the practical stopping criterion that is used by APMC. Let $p = \mathrm{Prob}[\psi] = lim_{k\to\infty}\mathrm{Prob}_k[\psi]$ be the probability to approximate. If ψ is a monotone formula, then $q = 1 - p = \mathrm{Prob}[\neg\psi]$ can be approximated by a slight modification of the MC^2 algorithm [10] for probabilistic model checking. Let $\hat{p}_k$ be an $\varepsilon/3$-approximation of $p_k = \mathrm{Prob}_k[\psi]$ computed by the $\mathcal{GAA}$ randomized approximation algorithm and $\hat{q}$ be an $\varepsilon/3$-approximation of q computed by the MC^2 algorithm. The stopping criterion is : $|\hat{p}_k - (1-\hat{q})| \leq \varepsilon/3$. If this test is satisfied, then p_k is an ε-approximation of p.

Since 2003, numerous experiments have been done, such as the verification of various probabilistic distributed algorithms (mutual exclusion, dining philosophers, leader election...) [13], of the IEEE 802.3 CSMA/CD protocol [8], that is part of the ethernet protocol, and of very large sensor networks [3]. Recently, we have extended the random path generator of APMC to the verification of continuous time Markov chains [14].

5. Related work

To our knowledge, there exists no similar non-approximability result for the probabilistic verification of general linear temporal formula.

In this section, we compare approximation methods with statistical approaches to probabilistic model checking. Other sampling methods have been used for statistical model checking. In [28], a procedure is described for verifying properties of discrete event systems, that are expressed in a fragment of CSL continuous stochastic logic, restricted to bounded time formulas. This acceptance sampling method, that is based on Monte Carlo simulation and statistical

hypothesis testing, is well adapted to determine whether the satisfaction probability is above or below a given threshold. The objective is different from approximating such a probability with any degree of accuracy and the comparison with randomized approximation methods is difficult. In a more recent paper [29], Younes states that the sample size used in the acceptance method is smaller than the path generation size of our randomized approximation algorithm. In fact, we believe that hypothesis testing is better to deal with threshold problems, whereas approximation methods are better to obtain good estimation of unknown probabilities. Moreover, the acceptance method is limited to verify bounded time properties. In [23], a statistical method is proposed to model checking black-box probabilistic systems against specifications given in a fragment of continuous stochastic logic. This approach also uses statistical hypothesis testing and is limited to bounded time properties.

The closest approach to ours is the MC^2 algorithm [10], a randomized algorithm for classical model checking of linear time logic formulas. A straightforward modification of this algorithm can be used for probabilistic model checking. It provides a randomized algorithm to estimate the satisfaction probability of safety properties over probabilistic transition systems. The MC^2 algorithm uses the optimal approximation algorithm of [6] and differs from ours by the random generation of finite execution paths containing cycles, named lassos. Their approximation method is complementary and we are collaborating with them since 2005.

6. Conclusion

In this paper, we first addressed the general problem of approximating the probabilistic verification of any linear time temporal formula against probabilistic systems. We showed that satisfaction probabilities of such formulas are non-approximable by a polynomial time randomized algorithm with relative error unless some unlikely complexity conjecture holds. Nevertheless, we presented a polynomial time randomized approximation scheme, with absolute error, for the quantitative verification of bounded time monotone *LTL* formulas, and we deduced a randomized approximation scheme for non-bounded monotone formulas. We obtained also a randomized approximation scheme with relative error by using an optimal approximation algorithm. The randomized approximation schemes have been implemented in a probabilistic model checker that is very efficient in practice. We started the study of such methods in 2001 [20,22], and to our knowledge, it was the first time that such randomized approximation methods were used for probabilistic verification. The main advantage of this approximation method is to eliminate the space complexity problem due to the state explosion phenomenon that occurs in the representation of protocols as probabilistic transition systems.

Acknowledgements

We would like to thank Thomas Hérault for his important participation to the development of APMC, and an anonymous referee for his pertinent remarks and suggestions.

References

[1] A. Bianco, L. de Alfaro, Model checking of probabilistic and nondeterministic systems, in: Proc. 15th Conf. Foundations of Software Technology and Theoretical Computer Science, in: LNCS, vol. 1026, 1995, pp. 499–513.
[2] P. Brémaud, Markov chains, Gibbs fields, Monte Carlo simulation, and queues, in: Texts in Applied Mathematics, Springer, 1999, 445 pages.
[3] M. Cadilhac, T. Herault, R. Lassaigne, S. Peyronnet, Sebastien Tixeuil, Evaluating complex MAC protocols for sensor networks with APMC. Proc. 6th Int. Workshop on Automated Verification of Critical Systems, 2006.
[4] E.M. Clarke, E.A. Emerson, A.P. Sistla, Automatic verification of finite-state concurrent systems using temporal logic specifications, ACM Transactions on Programming Languages and Systems 8 (2) (1986) 244–263.
[5] C. Courcoubetis, M. Yannakakis, The complexity of probabilistic verification, Journal of the ACM 42 (4) (1995) 857–907.
[6] P. Dagum, R. Karp, M. Luby, S. Ross, An optimal algorithm for Monte Carlo estimation, SIAM Journal of Computing 29 (5) (2000) 1484–1496.
[7] L. de Alfaro, M. Kwiatkowska, G. Norman, D. Parker, R. Segala, Symbolic model checking of concurrent probabilistic processes using MTBDDs and the kronecker representation, in: Proc. 6th Int. Conf. Tools and Algorithms for Construction and Analysis of Systems, in: LNCS, vol. 1785, 2000, pp. 395–410.
[8] M. Duflot, L. Fribourg, T. Herault, R. Lassaigne, F. Magniette, S. Messika, S. Peyronnet, C. Picaronny, Probabilistic model checking of the CSMA/CD protocol using PRISM and APMC, in: Proc. 4th Int. Workshop on Automated Verification of Critical Systems, ENTCS 128 (6) (2005) 195–214.
[9] R. Grosu, S.A. Smolka, Quantitative model checking, in: Proc. 1st Int. Symposium on Leveraging Applications of Formal Methods, 2004.

[10] R. Grosu, S.A. Smolka, Monte-Carlo model checking, in: Proc. 11th Int. Conf. Tools and Algorithms for Construction and Analysis of Systems, in: LNCS, vol. 3440, 2005, pp. 271–286.
[11] H. Hansson, B. Jonsson, A logic for reasoning about time and reliability, Formal Aspects of Computing 6 (1994) 512–535.
[12] G. Guirado, T. Herault, R. Lassaigne, S. Peyronnet, Distribution, approximation and probabilistic model checking, in: Proc. Int. Workshop on Parallel and Distributed Methods in Verification, ENTCS 135 (2) (2006) 19–30.
[13] T. Herault, R. Lassaigne, F. Magniette, S. Peyronnet, Approximate probabilistic model checking, in: Proc. 5th Int. Conf. Verification, Model Checking, and Abstract Interpretation, in: LNCS, vol. 2937, 2004, pp. 73–84.
[14] T. Herault, R. Lassaigne, S. Peyronnet, APMC 3.0: Approximate verification of discrete and continuous time Markov chains, in: Proc. 3rd Int. Conf. on Quantitative Evaluation of Systems, IEEE Computer Society, 2006, pp. 129–130.
[15] W. Hoeffding, Probability inequalities for sums of bounded random variables, Journal of the American Statistical Association 58 (1963) 13–30.
[16] M.R. Jerrum, L.G. Valiant, V.V. Vazirani, Random generation of combinatorial structures from a uniform distribution, Theoretical Computer Science 43 (1986) 169–188.
[17] D.R. Karger, A randomized fully polynomial time approximation scheme for the all terminal network reliability problem, SIAM Journal on Computing 29 (1999) 492–514.
[18] R.M. Karp, M. Luby, Monte-Carlo algorithms for enumeration and reliability problems, in: Proc. 24th Annual IEEE Symposium on Foundations of Computer Science, 1983, pp. 56–64.
[19] R.M. Karp, M. Luby, N. Madras, Monte-Carlo algorithms for enumeration and reliability problems, Journal of Algorithms 10 (1989) 429–448.
[20] R. Lassaigne, S. Peyronnet, Approximate verification of probabilistic systems, in: Proc. 2nd Int. Workshop Process Algebra and Probabilistic Methods, Performance Modeling and Verification, in: LNCS, vol. 2399, 2002, pp. 213–214.
[21] O. Lichtenstein, A. Pnueli, Checking that finite state concurrent programs satisfy their linear specification, in: Proc. 12th ACM Symposium on Principles of Programming Languages, ACM CS Press, 1985, pp. 97–107.
[22] S. Peyronnet, Can verification be approximated? in: Proc. 1st Int. Workshop on Automated Verification of Critical Systems, Programming Research Group Research, Report RR-01-07, Oxford University, 2001.
[23] K. Sen, M. Viswanathan, G. Agha, Statistical model checking of black-box probabilistic systems, in: Proc. 16th Int. Conf. Computer Aided Verification, in: LNCS, vol. 3114, 2004, pp. 202–215.
[24] E. Seneta, Non-negative Matrices and Markov Chains. Springer Series in Statistics, 1973.
[25] A. Sinclair, Algorithms for random generation and counting: A Markov chain approach, in: Progress in Theoretical Computer Science, Birkhauser, Boston, 1993.
[26] A.P. Sistla, E.M. Clarke, The complexity of propositional linear temporal logics, Journal of the ACM 32 (3) (1985) 733–749.
[27] M.Y. Vardi, Automatic verification of probabilistic concurrent finite-state programs, in: Proc. 26th Annual IEEE Symposium on Foundations of Computer Science, 1985, pp. 327–338.
[28] H.L.S. Younes, R.G. Simmons, Probabilistic verification of discrete event systems using acceptance sampling, in: Proc. 14th Int. Conf. Computer Aided Verification, in: LNCS, vol. 2404, 2002, pp. 223–235.
[29] H.L.S. Younes, Error control for probabilistic model checking, in: Proc. 7th Int. Conf. Verification, Model Checking, and Abstract Interpretation, in: LNCS, vol. 3855, 2006, pp. 142–156.