RAPPORT DE RECHERCHE

LRI

APPROXIMATE PROBABILISTIC MODEL CHECKING

HERAULT T / LASSAIGNE R / MAGNIETTE F / PEYRONNET S

Unité Mixte de Recherche 8623
CNRS-Université Paris Sud-LRI

04/2003

Rapport de Recherche N° 1359

CNRS – Université de Paris Sud
Centre d'Orsay
LABORATOIRE DE RECHERCHE EN INFORMATIQUE
Bâtiment 650
91405 ORSAY Cedex (France)

Approximate Probabilistic Model Checking

Thomas Hérault[1], Richard Lassaigne[2], Frédéric Magniette[1], and Sylvain Peyronnet[1]

[1] University Paris XI
LRI Bâtiment 490,
F-91405 Orsay Cedex, France
{herault,magniett,syp}@lri.fr
[2] University Paris VII
2 place Jussieu,
F-75251 Paris, France
lassaign@logique.jussieu.fr

Abstract. Symbolic model checking methods have been extended recently to the verification of probabilistic processes. However, the representation of the transition matrix may be expensive for very large systems and may induce a prohibitive cost for the model checking algorithm. In this paper, we propose an approximation method to verify quantitative properties on discrete Markov Chains. We give a randomized algorithm to approximate the probability that a property expressed by some positive LTL formula is satisfied with high confidence by a probabilistic system. Our randomized algorithm requires only a succinct representation of the system and is based on an execution sampling method. We also present an implementation and a few classical examples to prove the effectiveness of our approach.

1 Introduction

In this paper, we address the problem of verifying quantitative properties on discrete time markov chains (DTMC). We present an efficient procedure to approximate model checking of positive LTL formulas on probabilistic transition systems. This procedure decides if the probability of a formula is greater than a certain threshold by sampling finite execution paths. It allows us to verify monotone properties on the system with high confidence. For example, we can verify a property such as : "the probability that the message sent will be received without error is greater than 0.99". This method is an improvement on the method described in [14].

The main advantage of this approach is to allow verification of formulas even if the transition system is huge, even without any abstraction. Indeed, we do not have to deal with the state space explosion phenomenon because we verify the property on only one finite execution path at a time. This approach can be used in addition to classical probabilistic model checkers when the verification is intractable.

Our main results are:

- A method that allows the efficient approximation of the satisfaction probability of monotone properties on probabilistic systems.
- A tool named APMC that implements the method. We use it to verify extremely large systems such as the Pnueli and Zuck's 500 dining philosophers.

The paper is organized as follows. In Section 2, we review related work on probabilistic verification of qualitative and quantitative properties. In Section 3, we consider fully probabilistic systems and classical LTL logic. In Section 4, we recall the Bounded Model Checking method and explain how to adapt the main idea to the probabilistic framework. In Section 5, we present a randomized algorithm for the approximation of the satisfaction probability of monotone properties. In Section 6, we present our tool and give experimental results and compare them with the probabilistic model checker PRISM [6].

2 Related work

Several methods have been proposed to verify a probabilistic or a concurrent probabilistic system against LTL formulas. Vardi [21] developed an automata theoretic approach for verifying qualitative properties stating that a linear time formula holds with probability 0 or 1.
Courcoubetis and Yannakakis [4] studied probabilistic verification of quantitative properties expressed in the linear time framework. For the fully probabilistic case, the time complexity of their method is polynomial in the size of the state space, and exponential in the size of the formula. For the concurrent case, the time complexity is linear in the size of the system, and double exponential in the size of the formula.
Hansson and Jonsson [8] introduced the logic $PCTL$ (Probabilistic Computation Tree Logic) and proposed a model checking algorithm for fully probabilistic systems. They combined reachability-based computation, as in classical model checking, and resolution of systems of linear equations to compute the probability associated with the until operator. For concurrent probabilistic systems, Bianco and de Alfaro [2] showed that the minimal and maximal probabilities for the until operator can be computed by solving linear optimization problems. The time complexity of these algorithms are polynomial in the size of the system and linear in the size of the formula.
There are a few model checking tools that are designed for the verification of quantitative specifications. ProbVerus [7] uses $PCTL$ model checking and symbolic techniques to verify $PCTL$ formulas on fully probabilistic systems. PRISM

[6, 13] is a probabilistic symbolic model checker that can check $PCTL$ formulas on fully or concurrent probabilistic systems. Reachability-based computation is implemented using BDDs, and numerical analysis may be performed by a choice between three methods: MTBDD-based representation of matrices, conventional sparse matrices, or a hybrid approach. The *Erlangen-Twente Markov Chain Checker* [9] ($E \vdash MC^2$) supports model checking of continuous-time Markov chains against specifications expressed in continuous-time stochastic logic (CSL). Rapture, presented in [5] and [11] uses abstraction and refinement to check a subset of $PCTL$ over concurrent probabilistic systems.

In [22], Younes and Simmons described a procedure for verifying properties of discrete event systems based on Monte-Carlo simulation and statistical hypothesis testing. This procedure uses a refinement technique to build statistical tests for the satisfaction probability of CSL formulas. Their logic framework is more general than ours, but they cannot predict the sampling size, in contrast with our approximation method in which this size is exactly known and tractable.

Monniaux [15] defined abstract interpretation for probabilistic programs and used it to obtain over-approximations for probability measures.

3 Probabilistic Transition Systems

In this section, we introduce the classical concepts for the verification of probabilistic systems.

Definition 1. *A transition system (or a Kripke structure) is a structure* $\mathcal{M} = (S, R, I, L)$ *where* S *is a set of states,* $I \subset S$ *is the set of initial states,* $R \subseteq S \times S$ *is a transition relation between states and* $L : S \to \mathcal{P}(AP)$ *is a function labeling each state with a set* (AP) *of atomic propositions.*

To handle probabilistic systems, we recall the definition of Markov chains.

Definition 2. *A Discrete Time Markov Chain (DTMC) is a pair* $\mathcal{M} = (S, P)$ *where* S *is a finite or enumerable set of states and* $P : S \times S \to [0, 1]$ *is a transition probability function, i.e. for all* $s \in S$, $\sum_{t \in S} P(s, t) = 1$. *If* S *is finite, we can consider* P *to be a transition matrix.*

The notion of DTMC can be extended to the notion of probabilistic transition system by adding a labeling function.

Definition 3. *A probabilistic transition system (PTS) is a structure* $\mathcal{M} = (S, P, I, L)$ *where (S,P) is a DTMC,* I *is the set of initial states and* $L : S \to \mathcal{P}(AP)$ *a function which labels each state with a set of atomic propositions.*

Definition 4. *A path* σ *is a finite or infinite 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 denote by $Path(s)$ the set of paths whose first state is s. We note also $\sigma(i)$ the $(i+1)$-st state of path σ and σ^i the path $(\sigma(i), \sigma(i+1), \dots)$. The length of a path σ is the number of states in the path and is denoted by $|\sigma|$.

Definition 5. *For each PTS $\mathcal{M}$ and state s, we may define a probability measure on the set $Path(s)$ such that: for any finite path $(s_0, s_1, \dots, s_n)$, $Prob(\{\sigma/\sigma$ is a path and $(s_0, s_1, \dots, s_n)$ 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 $\{\sigma/\pi$ is a prefix of $\sigma\}$ where π is a finite path.

Linear Temporal Logic (LTL) formulas are interpreted over paths of a transition system $\mathcal{M}$.

Definition 6. *The satisfaction of LTL formulas is defined by induction by:*

- $\mathcal{M}, \sigma \models a$ *iff* $a \in L(\sigma(0))$.
- $\mathcal{M}, \sigma \models \neg\phi$ *iff* $\mathcal{M}, \sigma \not\models \phi$.
- $\mathcal{M}, \sigma \models \phi \wedge \psi$ *iff* $\mathcal{M}, \sigma \models \phi$ *and* $\mathcal{M}, \sigma \models \psi$.
- $\mathcal{M}, \sigma \models \mathbf{X}\phi$ *iff* $\mathcal{M}, \sigma(1) \models \phi$.
- $\mathcal{M}, \sigma \models \phi\mathbf{U}\psi$ *iff there exists* $j \geq 0$ *s.t.* $\mathcal{M}, \sigma^j \models \psi$ *and for all* $i < j$ $\mathcal{M}, \sigma^i \models \phi$.

Definition 7. *An LTL formula ϕ is universally valid in $\mathcal{M}$, which we write $\mathcal{M} \models \mathbf{A}\phi$, if and only if for all paths σ with $\sigma(0) \in I$, $\mathcal{M}, \sigma \models \phi$.*

An LTL formula ϕ is existentially valid in $\mathcal{M}$, which we write $\mathcal{M} \models \mathbf{E}\phi$, if and only if there is a path σ with $\sigma(0) \in I$ such that $\mathcal{M}, \sigma \models \phi$.

In [21], it is shown that for any LTL formula ϕ, probabilistic transition system $\mathcal{M}$ and state s, the set of paths $\{\sigma/\sigma(0) = s$ and $\mathcal{M}, \sigma \models \phi\}$ is measurable. We denote by $Prob[\phi]$ the measure of this set.

4 Probabilistic bounded model checking

In this section, we review the classical framework for bounded model checking of linear time temporal formulas over transition systems. Then, we show that we cannot directly extend this approach but we use the main idea of checking formulas on paths of bounded length to approximate the target satisfaction probability.

4.1 Bounded model checking

Biere, Cimatti, Clarke and Zhu [3] present a symbolic model checking technique based on SAT procedures instead of BDDs. They introduce bounded model checking (BMC), where the bound correspond to the maximal length of a possible counterexample. First, they give a correspondence between BMC and classical model checking. Then they show how to reduce BMC to propositional satisfiability in polynomial time.

The bounded model checking procedure works as follows. Given a transition system $\mathcal{M}$, an LTL formula ϕ and a bound $k \in \mathbb{N}$, they construct a propositional formula which is satisfiable if and only if there exists a path of length k which is a counterexample to the specification expressed by ϕ. This procedure is well adapted to finding a counterexample, if it exists, by successive incrementation of the bound. If, on the other hand, the transition system satisfies ϕ, the value of k has to be incremented indefinitely and the procedure does not terminate, unless we know some bound on k. The main advantages of this technique are that it finds counterexamples very fast and uses much less space than BDD-based approaches.

Let us review more precisely what BMC is. Given a transition system $\mathcal{M}$, an LTL formula ϕ and a bound k, if we want to verify $\mathcal{M} \models \mathbf{A}\phi$, we consider an LTL formula ψ which is in negative normal form and is equivalent to $\neg\phi$. The translation of the formula ψ to a propositional formula is in two parts: the first component $[\![\mathcal{M}]\!]_k$ requires a sequence $(s_0, s_1, \ldots, s_k)$ to be a path σ in $\mathcal{M}$ and the second component $[\![\psi]\!]_k$ forces σ to satisfy ψ.

The following theorem summarizes the results of [3] for bounded model checking of LTL formulas.

Theorem 1. *[3] Let ψ be an LTL formula and $\mathcal{M}$ be a transition system. Then $\mathcal{M} \models \mathrm{E}\psi$ if and only if there exists $k \in \mathbb{N}$ such that $[\![\mathcal{M}]\!]_k \wedge [\![\psi]\!]_k$ has a satisfying assignment.*

To check the initial property ϕ, one should look for the existence of a counterexample to the negation ψ for a given k, i.e. a satisfying assignment of $[\![\mathcal{M}]\!]_k \wedge [\![\psi]\!]_k$. In [3], the following result is also stated: if one does not find such a counterexample for $k \leq |S| \times 2^{|\psi|}$, then the initial property is true. We cannot hope to find a polynomial bound on k with respect to the size of S and ψ, since the model checking problem for LTL is PSPACE-complete (see [20]) and such a bound would yield a polynomial reduction to propositional satisfiability.

4.2 Satisfaction probabilities on bounded execution paths

We try to check $Prob[\psi] \geq b$ by considering $Prob_k[\psi] \geq b$, i.e., on the probabilistic space limited to the Kripke paths of depth k. Following the BMC approach, we can associate to a formula ψ and depth k the propositional formula $[\![\mathcal{M}]\!]_k \wedge [\![\psi]\!]_k$ in such a way that a path of length k satisfying ψ corresponds to an assignment satisfying $[\![\mathcal{M}]\!]_k \wedge [\![\psi]\!]_k$. Thus determining $Prob_k[\psi]$ could be reduced to a counting version of SAT. Unfortunately, not only are no efficient algorithms known for such counting problems, but they are believed to be strongly intractable (see, for instance[17]). However, it is not necessary to do such a transformation since we can evaluate directly the formula on one finite path. In the following, we use this straightforward evaluation instead of SAT-solving methods.

For many natural formulas, truth at depth k implies truth in the entire model. These formulas are the so-called monotone formulas. We consider a subset of LTL formulas which have this property.

Definition 8. *The essentially positive fragment (EPF) of LTL is the set of formulas built from atomic formulas (p), their negations ($\neg p$), closed under $\vee$, $\wedge$ and the temporal operators X, U.*

These formulas include nested compositions of U but do not allow for negations in front. Nevertheless, this fragment can express various classical properties of transition systems such as reachability and liveness properties. Important properties of protocols like livelock freeness and any convergence property are also expressible by EPF formulas.

If ϕ is a formula of the EPF fragment, we can use a bounded framework to verify whether ϕ is true on a path σ of depth k.

The monotonicity of the property defined by an EPF formula gives the following proposition:

Proposition 1. *For any formula of the essentially positive fragment of LTL and $0 < b \leq 1$, there exists a k such that if $Prob_k[\phi] \geq b$, then $Prob[\phi] \geq b$.*

Indeed, the probability of an EPF formula to be true in the bounded model of depth k is less or equal than the probability of the formula in the bounded model of depth greater than k.

This proposition can be extended to any monotone formula but we restrict our scope to make our method fully automatic.

5 Approximate probabilistic model checking

In order to calculate the satisfaction probability of a monotone formula, we have to verify the inner formula on all paths of depth k. Such a computation is intractable in general since there are exponentially many paths to check. Thus, it is natural to ask: can we approximate $Prob_k[\phi]$? In this section, we propose an efficient procedure to approximate this probability. The running time of this computation is polynomial in the length of paths and the size of the formula.

In order to estimate the probabilities of monotone properties with a simple randomized algorithm, we generate random paths in the probabilistic space underlying the DTMC structure of depth k and compute a random variable A/N which estimates $Prob_k[\psi]$. To verify a statement $Prob_k[\psi] \geq b$, we test whether $A/N > b - \varepsilon$. Our decision is correct with confidence $(1-\delta)$ after a number of samples polynomial in $\frac{1}{\varepsilon}$ and $\log\frac{1}{\delta}$. This result is obtained by using Chernoff-Hoeffding bounds [10] 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 graph, that is a succinct description in an input language, for example Reactive Modules.

Definition 9. *A succinct representation, or diagram, of a PTS* $\mathcal{M} = (S, P, I, L)$ *is an oracle which for any state s, gives the states t such that* $P(s,t) > 0$.

In order to prove our result, we introduce the notion of fully polynomial randomized approximation scheme (FPRAS) for probability problems. This notion is analogous to randomized approximation schemes [12, 16] for counting problems. Our probability problem is defined by giving as input x a succinct representation of a probabilistic system, a formula and a positive integer k. The succinct representation is used to generate a set of execution paths of length k. The solution of the probability problem is the probability measure $\mu(x)$ of the formula over the set of execution paths. The difference with randomized approximation schemes for counting problems is that for approximating probabilities, which are rational numbers in the interval $[0, 1]$, we only require approximation with additive error.

Definition 10. *A fully polynomial randomized approximation scheme (FPRAS) for a probability problem is a randomized algorithm* $\mathcal{A}$ *that takes an input* x*, two real numbers* $0 < \varepsilon, \delta < 1$ *and produces a value* $A(x, \varepsilon, \delta)$ *such that:*

$$Prob\big[|A(x,\varepsilon,\delta) - \mu(x)| \leq \varepsilon\big] \;\geq\; 1 - \delta.$$

The running time of $\mathcal{A}$ *is polynomial in* $|x|$, $\frac{1}{\varepsilon}$ *and* $\log\frac{1}{\delta}$.

The probability is taken over the random choices of the algorithm. We call ε the *approximation ratio* and δ the *confidence ratio.* By verifying the formula on $O(\frac{1}{\varepsilon^2}.\log\frac{1}{\delta})$ paths, we obtain an answer with confidence $(1-\delta)$.

Consider the following randomized algorithm designed to approximate $Prob_k[\psi]$, that is the probability of an LTL formula over bounded DTMC of depth k:

Generic approximation algorithm $\mathcal{GAA}$
Input: $diagram, \psi, k, \varepsilon, \delta$
$N := 4\log(\frac{2}{\delta})/\varepsilon^2$
$A := 0$
For $i = 1$ to N do
1. Generate a random path σ of depth k with the diagram
2. If ψ is true on σ then $A := A + 1$

Return A/N

Theorem 2. *The generic approximation algorithm* $\mathcal{GAA}$ *is a fully randomized approximation scheme for the probability* $p = Prob_k[\psi]$ *for an LTL formula* ψ *and* $p \in]0, 1[$.

Proof. The random variable A is the sum of independent random variables with a Bernouilli distribution. We use the Chernoff-Hoeffding bound [10] to obtain the result. Let $X_1, ..., X_N$ 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-Hoeffding bound gives $Prob\big[|Y-p| > \varepsilon\big] < 2e^{-\frac{N\varepsilon^2}{4}}$. In our case, if $N \geq 4\log(\frac{2}{\delta})/\varepsilon^2$, then $Prob\big[|A/N - p| \leq \varepsilon\big] \geq 1-\delta$ where $p = Prob_k[\psi]$.

The time needed to verify if a given path verify ψ is polynomial in the size of the formula. The number N of iterations is polynomial in $\frac{1}{\varepsilon}$ and $\log\frac{1}{\delta}$. So $\mathcal{GAA}$ is a fully polynomial randomized approximation scheme.

This algorithm provides a method to verify quantitative properties expressed by EPF formulas. To check the property $Prob_k[\psi] \geq b$, we can test if the result of the approximation algorithm is greater than $b - \varepsilon$. If $Prob_k[\psi] \geq b$ is true, then the monotonicity of the property guarantees that $Prob[\psi] \geq b$ is true. Otherwise, we increment the value of k within a certain bound, for example the diameter of the system for reachability formulas, to conclude that $Prob[\psi] \not\geq b$.

6 APMC : an implementation

In this section, we present some experimental results of our approximate model checking method. These results were obtained with a tool we developed. This tool, APMC, works in a distributed framework and allows the verification of extremely large systems such as the 300 dining philosophers problem. We compare the performance of our method to the performance of PRISM. These results are promising, showing that large systems can be approximately verified in seconds, using very little memory.

APMC (Approximate Probabilistic Model Checker) is a GPL (Gnu Public License) tool written in C with lex and yacc. It uses a client/server computation model (described in Subsection 6.2) to distribute path generation and verification on a cluster of machines.

APMC is simple to use: the user enters an LTL formula and a description of a system written in the same variant of Reactive Modules as used by PRISM. The user enters the target satisfaction probability for the property, the length of the paths to consider and the approximation and confidence parameters ε and δ. These parameters can be changed through a Graphical User Interface (GUI), represented in Figure 1. These are the basic parameters, there are advanced parameters such as the choice of a specific strategy for the speed/space compromise to use, but one can use a "by default" mode which is sufficiently efficient in general. After this, the user clicks on "go" and waits for the result. APMC is a fully automatic verification tool.

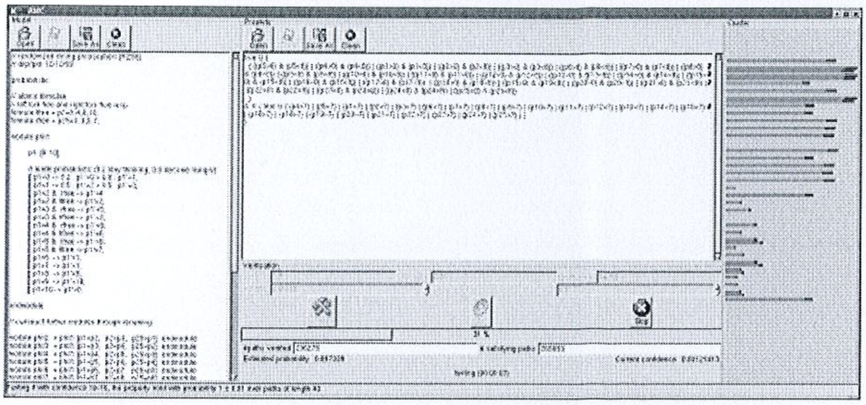

Figure 1. The Graphical User Interface.

6.1 Standalone use and comparison with PRISM

We first consider a classical problem from the PRISM examples library [19]: the dining philosophers problem. Let us quickly recall the problem: n philosophers are sitting around a table, each philosopher spends most of its time thinking, but sometimes gets hungry and wants to eat. To eat, a philosopher needs both its right and left forks, but there are only n forks shared by all philosophers.
The problem is to find a protocol for the philosophers without livelock. Pnueli and Zuck [18] give a protocol that is randomized. We ran experiments on a fully probabilistic version of this protocol (that is, a DTMC version): there are no non-deterministic transitions and the scheduling between philosophers is randomized. For this protocol, we checked the following liveness property: "If a philosopher is hungry, then with probability one, some philosopher will eventually eat". This property guarantees that the protocol is livelock free. The following table shows our results using APMC and those of PRISM (model construction and model checking time) on one 1.8 GHz Pentium 4 workstation with 512 MB of memory under the Linux operating system. For this experiment, we let $\varepsilon = 10^{-2}$ and $\delta = 10^{-10}$.

number of phil.	depth	APMC (time in sec.)	PRISM (time in sec.)	PRISM (states)
3	20	35	0.394	770
5	23	56	0.87	64858
10	30	125	11.774	4.21×10^{9}
15	42	242	64.158	2.73×10^{14}
20	50	387	137.185	1.77×10^{19}
25	55	531	2469.56	1.14×10^{24}
30	65	823	out of mem.	out of mem.
50	130	3579	out of mem.	out of mem.
100	148	8364	out of mem	out of mem.

On this example, we see that we can handle larger systems than PRISM, more than 30 philosophers for Pnueli and Zuck's philosophers, without having to construct the entire model which contains 10^{24} states for 25 philosophers. Note that during the computation, our tool uses very little memory. This is due to the fact that the verification process never stores more than one path at a time.

6.2 Cluster use

In the previous subsection, we showed that APMC can be used on a single machine, but to increase the efficiency of the verification, APMC can distribute the computation on a cluster of machines using a client/server architecture.

Let us quickly describe the client/server architecture of APMC. The model, formula and other parameters are entered by the user via the Graphical User Interface which runs on the server (master). Both the model and formula are translated into C source code, compiled and sent to clients (the workers) when they request a job. Regularly, workers send current verification results, getting an acknowledgment from the master, to know if they have to continue or stop the computation. Since the workers only need memory to store the generated code and one path, the verification requires very little memory space. Furthermore, since each path is verified independently, there is no problem of load balancing. Figure 4 shows the scalability of the implementation on Pnueli and Zuck's dining philosophers algorithm for 25 philosophers: computation time is divided by two when we double the size of the cluster. This is a consequence of very low communications overhead in the computation.

We used APMC to check properties of several fully probabilistic systems modeled as DTMCs. In Figure 2, we consider Pnueli and Zuck's Dining Philosophers algorithm [18] for which we verify the liveness property and in figure 3, we consider a fully probabilistic version of the randomized mutual exclusion of Pnueli and Zuck [18]. All the experiments were done with a cluster of 20 workers (all are ATHLON XP1800+ under Linux) with $\varepsilon = 10^{-2}$ and $\delta = 10^{-10}$.

phil.	depth	time (sec.)	max. memory (KBytes)
15	38	11	324
25	55	25	340
50	130	104	388
100	145	418	484
200	230	1399	676
300	295	4071	1012

Figure 2. Dining philosophers: run-time and memory for 20 workers.

We are able to verify very large systems using a reasonable cluster of workers and very little memory for each of them. In a extra experiment, with an hetero-

proc.	depth	time (sec.)	max. memory (KBytes)
3	120	13	316
5	250	35	328
10	520	146	408
15	1000	882	548
20	1400	1499	660

Figure 3. Mutual exclusion: run-time and memory for 20 workers.

geneous cluster of 32 machines, we were able to verify the Pnueli and Zuck's 500 philosophers in about four hours.

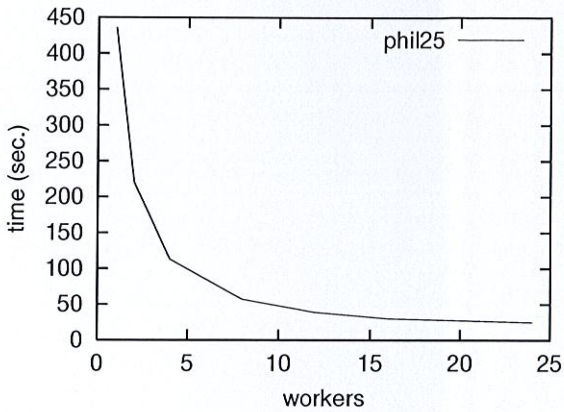

Figure 4. Scalability of the implementation: time vs workers for 25 dining philosophers.

7 Conclusion

To our knowledge, this work is the first to apply randomized approximation schemes to probabilistic model checking. We estimate the probability with a randomized algorithm and conclude that satisfaction probabilities of EPF formulas can be approximated. This fragment is sufficient to express reachability and liveness properties. Our implementation was used to investigate the effectiveness of this method. Our experiments point to an essential advantage of the method: the use of very little memory. In practice, this means that we are able to verify very large fully probabilistic models, such as the dining philosopher's problem with 500 philosophers. This method seems to be very useful when classical verification is intractable.

Acknowledgments

We would like to thank Sophie Laplante for many helpful discussions and suggestions. The second author would like to thank Marta Kwiatkowska and her group for their advices on PRISM.

References

1. APMC homepage. http://www.lri.fr/~syp/APMC
2. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. *Proc. FST&TCS*, Lectures Notes in Computer Science, 1026:499-513, 1995.
3. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDD's. *Proc. of 5th TACAS* , Lectures Notes in Computer Science, 1573:193–207, 1999.
4. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.
5. P. D'Argenio, B. Jeannet, H. Jensen and K. Larsen. Reachability analysis of probabilistic systems by successive refinements. *Proc. of the joint PAPM/PROBMIV*, 2001.
6. L. de Alfaro, M. Kwiatkowska, G. Norman, D. Parker, and R. Segala. Symbolic model checking of concurrent probabilistic processes using MTBDDs and the kronecker representation. In *Proc. of 6th TACAS*, Lectures Notes in Computer Science, 1785, 2000.
7. V. Hartonas-Garmhausen, S. Campos, and E. Clarke. Probverus: Probabilistic symbolic model checking. In *5th International AMAST Workshop, ARTS'99, May 1999*, Lecture Notes in Computer Science 1601, 1999.
8. H.Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:512–535, 1994.
9. H. Hermanns, J.-P. Katoen, J. Meyer-Kayser and M. Siegle. A Markov chain model checker. *Proc. of 6th TACAS*, Lectures Notes in Computer Science, 1785:347-362, 2000.
10. W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13-30, 1963.
11. B. Jeannet, P. R. D'Argenio and K. G. Larsen RAPTURE: A tool for verifying Markov Decision Processes. In Proc. of the CONCUR'02. 2002.
12. R.M. Karp, M. Luby and N. Madras. Monte-Carlo algorithms for enumeration and reliability problems. *Journal of Algorithms*, 10:429–448, 1989.
13. M. Kwiatkowska, G. Norman and D. Parker. Probabilistic symbolic model checking with PRISM: a hybrid approach. In Proc. of 8th TACAS, Lecture Notes in Computer Science 2280, 2002.
14. R. Lassaigne and S. Peyronnet. Approximate Verification of Probabilistic Systems. In *Proc. of the 2nd joint PAPM-PROBMIV*, Lecture Notes in Computer Science 2399, 213–214, 2002.
15. D. Monniaux Analyse de programmes probabilistes par interprétation abstraite. *PhD thesis*, Université Paris IX. 2001.
16. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
17. C.H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

18. A. Pnueli and L. Zuck. Verification of multiprocess probabilistic protocols. *Distributed Computing*, pages 1:53–72, 1986.
19. PRISM homepage. http://www.cs.bham.ac.uk/~dxp/prism/.
20. A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.
21. M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. *Proc. 26th Annual Symposium on Foundations of Computer Science*, pages 327–338, 1985.
22. H. L. S. Younes and R. G. Simmons. Probabilistic Verication of Discrete Event Systems using Acceptance Sampling. In Proc. of the 14th International Conference on Computer Aided Verification, Lecture Notes in Computer Science 2404, 223-235. 2002.

RAPPORTS INTERNES AU LRI - ANNEE 2003

N°	Nom	Titre	Nbre de pages	Date parution
1345	FLANDRIN E LI H WEI B	A SUFFICIENT CONDITION FOR PANCYCLABILITY OF GRAPHS	16 PAGES	01/2003
1346	BARTH D BERTHOME P LAFOREST C VIAL S	SOME EULERIAN PARAMETERS ABOUT PERFORMANCES OF A CONVERGENCE ROUTING IN A 2D-MESH NETWORK	30 PAGES	01/2003
1347	FLANDRIN E LI H MARCZYK A WOZNIAK M	A CHVATAL-ERDOS TYPE CONDITION FOR PANCYCLABILITY	12 PAGES	01/2003
1348	AMAR D FLANDRIN E GANCARZEWICZ G WOJDA A P	BIPARTITE GRAPHS WITH EVERY MATCHING IN A CYCLE	26 PAGES	01/2003
1349	FRAIGNIAUD P GAURON P	THE CONTENT-ADDRESSABLE NETWORK D2B	26 PAGES	01/2003
1350	FAIK T SACLE J F	SOME b-CONTINUOUS CLASSES OF GRAPH	14 PAGES	01/2003
1351	FAVARON O HENNING M A	TOTAL DOMINATION IN CLAW-FREE GRAPHS WITH MINIMUM DEGREE TWO	14 PAGES	01/2003
1352	HU Z LI H	WEAK CYCLE PARTITION INVOLVING DEGREE SUM CONDITIONS	14 PAGES	02/2003
1353	JOHNEN C TIXEUIL S	ROUTE PRESERVING STABILIZATION	28 PAGES	03/2003
1354	PETITJEAN E	DESIGNING TIMED TEST CASES FROM REGION GRAPHS	14 PAGES	03/2003
1355	BERTHOME P DIALLO M FERREIRA A	GENERALIZED PARAMETRIC MULTI-TERMINAL FLOW PROBLEM	18 PAGES	03/2003
1356	FAVARON O HENNING M A	PAIRED DOMINATION IN CLAW-FREE CUBIC GRAPHS	16 PAGES	03/2003
1357	JOHNEN C PETIT F TIXEUIL S	AUTO-STABILISATION ET PROTOCOLES RESEAU	26 PAGES	03/2003