# APMC 3.0: Approximate Verification of Discrete and Continuous Time Markov Chains

Thomas Hérault
LRI - U. Paris XI

Richard Lassaigne
Equipe de Logique - U. Paris VII

Sylvain Peyronnet
LRDE/EPITA

***Abstract*— In this paper, we give a brief overview of APMC (Approximate Probabilistic Model Checker). APMC is a model checker that implements approximate probabilistic verification of probabilistic systems. It is based on Monte-Carlo method and the theory of randomized approximation schemes and allows to verify extremly large models without explicitely representing the global transition system. To avoid the state-space explosion phenomenon, APMC gives an accurate approximation of the satisfaction probability of the property instead of the exact value, but using only a very small amount of memory. The version of APMC we present in this paper can now handle efficiently both discrete and continuous time probabilistic systems.**

## I. Introduction

In the last years, general methods have been presented for the model checking of fully probabilistic systems. Most of these methods reduced a probabilistic statement to the resolution of a linear system on the state space. However, due to the state space explosion phenomenon, the representation of the transition matrix can be so large that the verification becomes intractable. To overcome this, symbolic and numerical methods have been introduced in tools such as PRISM [2]. In this paper, we present a tool that implements a completely different solution. Our tool APMC (Approximate Probabilistic Model Checker) uses a randomized algorithm [4] to approximate the probability that a temporal formula is true, by using sampling of execution paths of the system [3]. APMC uses a distributed computation model to distribute path generation and formula verification on a cluster of workstations. The implementation of the tool started in 2003 and was originally done using C programming language together with lex and yacc. APMC was rewritten recently in Java for its version 3.0.

## II. Approximate verification

In this section we introduce our verification framework.

### Model and property

From the theoretical point of view, the method of APMC can handle any probabilistic systems that support execution path generation. However, we made the choice for APMC to use the same input language as PRISM [6] which is a variant of the Reactive Modules language. Its means that APMC handles modular description of either DTMCs and CTMCs (Discrete and Continuous Markov Chains).

The specification language that can be used are probabilistic LTL and PCTL. Using APMC we can compute, approximately, the probability of a temporal property over the probabilistic system. In the following, for the sake of clarity, we consider only bounded temporal properties. For more informations about the algorithm for general temporal properties, one can read [5].

### Verification

In order to estimate the probability $p$ of a bounded property $\psi$ with a simple randomized algorithm, we generate random paths in the probabilistic space underlying the system structure of depth $k$ and compute a random variable $A/N$ which estimates $Prob_k[\psi]$. Our approximation is $\varepsilon$-good with confidence $(1-\delta)$ after a number of samples polynomial in $\frac{1}{\varepsilon}$ and $\log\frac{1}{\delta}$. It means that the output value of our algorithm is in the interval $[p-\varepsilon, p+\varepsilon]$. The main advantage is that, in order to design a path generator, we only need to simulate the behavior of the system and store the values of all variable at each computation step. For that purpose, we use the $diagram$, which is a succinct representation such as the description of the system in the PRISM input language.

Our algorithm is the following:

```
Generic approximation algorithm $\mathcal{GAA}$
Input: $diagram, k, \psi, \varepsilon, \delta$
Output: approximation of $Prob_k[\psi]$
$N := \ln(\frac{2}{\delta})/2\varepsilon^2$
$A := 0$
For $i = 1$ to $N$ do
    $A := A +$ Random Path$(diagram, k, \psi)$
Return $A/N$
```

where **Random Path** is:

```
Random Path
Input: $diagram, k, \psi$
Output: samples a path $\pi$ of length $k$ and check formula $\psi$ on $\pi$
  1) Generate a random path $\pi$ of length $k$ (with the diagram)
  2) If $\psi$ is true on $\pi$ then return $1$ else $0$
```

The approximation algorithm we use consists in generating $O(\frac{1}{\varepsilon^2}.\log\frac{1}{\delta})$ execution paths, verifying the formula $\psi$ on each path and computing the fraction of satisfying paths, that is an $\varepsilon$-approximation of $Prob_k[\psi]$.

## III. Architecture and Implementation

APMC includes two independent components: 1) the compiler, 2) the deployer (see figure 1). The APMC compiler takes the model description written with the PRISM language (a

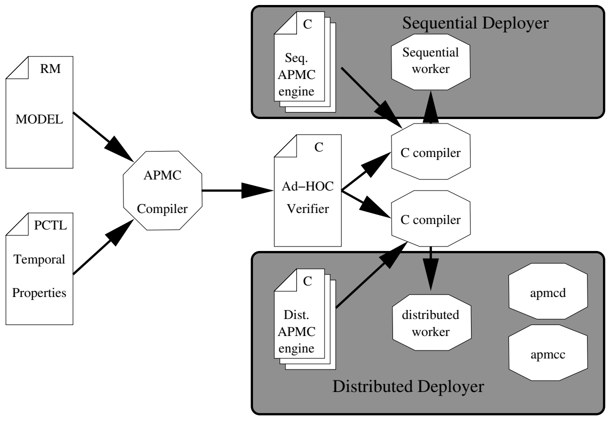

Fig. 1. APMC compilation cycle

variant of Reactive Modules), and a list of temporal properties to check on this model. It produces an ad-hoc verifier for this set of properties over the given model. The output of the compiler is in fact a set of functions in ANSI C suitable for verifying the properties on the model. This file lacks a main function and an engine to produce the verification.

Providing the engine and the missing functions for the ad-hoc verifier is the goal of the deployer. There exists many versions of the deployer and we introduce two of them here. The first one is a simple, sequential one. It produces a stand-alone binary which takes only three parameters: the approximation parameter, the confidence and the paths length. It then runs the simulation and outputs the approximated probabilities for each of the temporal formulas.

The other deployer provides the working program suitable for a distributed verification inside a LAN. This distributed deployment strategy runs in parallel these components on all the participating nodes and provides the same result with a linear acceleration.

APMC-3.0 has been completely rewritten to include all the features of the PRISM language. The compiler uses the JAVA parser of PRISM and is itself written in JAVA. Reactive modules structures are translated in a set of independent guarded rules (if the compiler is configured to compute the synchronizations before generating the ad-hoc verifier), or synchronized guarded rules (if the compiler is configured to use less memory). These guarded rules are translated in a set of ANSI C functions and appended to the output file. An engine designed to produce an execution path of bounded length is then added to the output. Then, the properties are translated in ANSI C functions and a higher level engine to compute the truth value of each of the property on a given path finishes the output of the compiler.

The sequential deployer is written in ANSI C and simply computes the number of paths to generate for a given confidence and approximation ratio, then iteratively generates a path of the given length, verifies it against all the formulas and updates the corresponding probabilities.

The distributed deployer is written in ANSI C and SUN RPCs. It includes the main worker engine and two more components (see figure 2). *apmcd* is the APMC daemon and runs on all the participating nodes (it is generic for any verification, as *apmcc*, the client). It is the server of the set of Remote Procedure Calls used by the ad-hoc worker engine to communicate the computations from each of the ad-hoc verifiers. *apmcd* is also client and server for others *apmcd*. They are registered one with the others in order to build a spanning tree of the local area network. Then, *apmcc*, the client for the *apmcd* is used to let the *apmcd* launch a worker (or more) on each of the nodes running *apmcd*. The workers issue Remote Procedure Calls to their *apmcd* in order to register their computations. Regularly, each *apmcd* calls remote procedures of their parent *apmcd* to communicate the sum of their worker and their children. The root *apmcd* computes the sum of all works done and regularly, *apmcc* calls a procedure on this *apmcd* to update the probabilities counts.

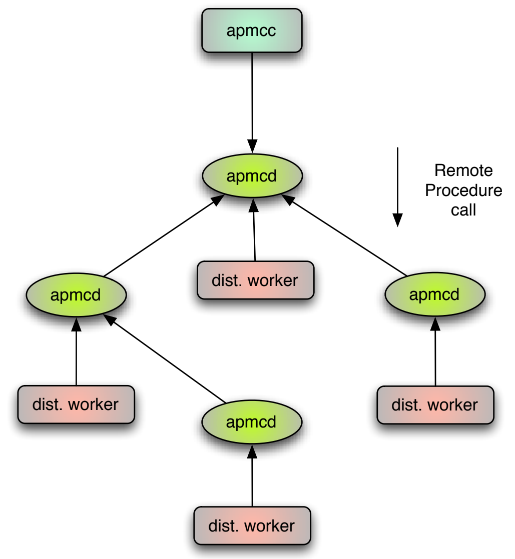

Fig. 2. Distributed Deployer Architecture

Measurements demonstrated that this distribution scheme is scalable and provides a linear acceleration.

## References

[1] APMC homepage. http://apmc.berbiqui.org

[2] 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 TACAS*, LNCS, 1785, 2000.

[3] T. Herault, R. Lassaigne, F. Magniette and S. Peyronnet. Approximate Probabilistic Model Checking. In *proc. of the 5th Verification, Model Checking and Abstract Int erpretation (VMCAI 2004)*, Venice, Italy, LNCS 2937, pages 73-84. 2004.

[4] R.M. Karp and M. Luby. Monte-Carlo algorithms for enumeration and reliability problems. *In Proceedings of the 24th FOCS*, 56–64, 1983.

[5] R. Lassaigne and S. Peyronnet. Probabilistic verification and approximation. In Proc. of Wollic 05. Electr. Notes Theor. Comput. Sci. 143: 101-114 (2006).

[6] PRISM homepage. http://cs.bham.ac.uk/ dxp/prism

## IV. More information for reviewers

### A. Basic facts about APMC

APMC is still under development and is freely available under GPL license. The team is composed (May 2006) of :

- Thomas Hérault - LRI - University Paris-Sud
- Richard Lassaigne - Equipe de Logique - University Paris 7
- Sylvain Peyronnet - LRDE/EPITA
- 7 students (ranging from PhD to last year of Bachelor in CS)

APMC is in use in several universities/compagny (20+), including :

- Universities Paris 7, Paris-Sud, Paris XII
- EPITA
- State University of NY at Stony Brook
- University of Birmingham
- France Telecom
- CRIL technology

APMC was downloaded from other places (40+), mainly by individuals.

### B. Paper about/using APMC

All papers by the APMC team are available on APMC website. Here is a list of this papers:

- Probabilistic verification of sensor networks. Akim Demaille, Thomas Herault and Sylvain Peyronnet. RIVF 2006.
- Distribution, approximation and probabilistic model checking. G. Guirado, T. Hrault, R. Lassaigne and S. Peyronnet. In Proc. of the 4th Parallel and Distributed Methods in Verification (PDMC 05), Lisboa, Portugal., Electronic Notes in Theor. Comp. Sci., 2005. To appear.
- Probabilistic verification and approximation. R. Lassaigne and S. Peyronnet. In Proc. of the 12th Workshop on Logic, Language, Information and Computation (Wollic 05). Florianapolis, Brazil, July 2005, Electronic Notes in Theor. Comp. Sci., 2005. To appear.
- Verification of the CSMA/CD protocol using PRISM and APMC. M. Duflot, L. Fribourg, Th. Hrault, R. Lassaigne, F. Magniette, S. Messika, S. Peyronnet, and C. Picaronny. In Proc. 4th Int. Workshop on Automated Verification of Critical Systems (AVoCS 2004), London, UK, Sep. 2004, Electronic Notes in Theor. Comp. Sci., 2004. To appear.
- Approximate Probabilistic Model Checking. T. Hrault, R. Lassaigne, F. Magniette and S. Peyronnet. In proceedings of the 5th Verification, Model Checking and Abstract Interpretation (VMCAI 2004), Venice, Italy, LNCS 2937, pages 73-84. 2004.
- Model checking et vrification probabiliste. S. Peyronnet. PhD Thesis, University of Paris XI, 2003.
- Approximate verification of probabilistic systems. R. Lassaigne and S. Peyronnet. In proceedings of 2nd joint Process Algebra and Performance Modelling and Probabilistic Methods in Verification (PAPM-PROBMIV 2002), LNCS 2399, pages 213-214. 2002.

### C. Case studies

Case studies have been done and published by the team (see papers), but also by students (verification of an atomic broadcast protocol, verification of biological processes) and by externals (mainly communication protocols). The total number of case studies is around 15.