PRISM: Probabilistic Symbolic Model Checker*

Marta Kwiatkowska, Gethin Norman, and David Parker

School of Computer Science, University of Birmingham,
Birmingham B15 2TT, United Kingdom
{mzk,gxn,dxp}@cs.bham.ac.uk

Abstract. In this paper we describe PRISM, a tool being developed at the University of Birmingham for the analysis of probabilistic systems. PRISM supports three probabilistic models: discrete-time Markov chains, continuous-time Markov chains and Markov decision processes. Analysis is performed through model checking such systems against specifications written in the probabilistic temporal logics PCTL and CSL. The tool features three model checking engines: one symbolic, using BDDs (binary decision diagrams) and MTBDDs (multi-terminal BDDs); one based on sparse matrices; and one which combines both symbolic and sparse matrix methods. PRISM has been successfully used to analyse probabilistic termination, performance, dependability and quality of service properties for a range of systems, including randomized distributed algorithms [2], polling systems [22], workstation clusters [18] and wireless cell communication [17].

1 Introduction

Probability is widely used in the design and analysis of software and hardware systems: as a means to derive efficient algorithms (e.g. the use of electronic coin flipping and randomness in decision making); as a model for unreliable or unpredictable behaviour (e.g. fault-tolerant systems, computer networks); and as a tool to analyse system performance (e.g. the use of steady-state probabilities in the calculation of throughput and mean waiting time). *Probabilistic model checking* refers to a range of techniques for calculating the likelihood of the occurrence of certain events during the execution of the system, and can be useful to establish properties such as "shutdown occurs with probability 0.01 or smaller" and "the video frame will be delivered within 5ms with probability 0.97 or greater".

In this paper we introduce PRISM, a probabilistic model checking tool being developed at the University of Birmingham. Conventional model checkers input a description of a model, represented as a state transition system, and a specification, typically a formula in some temporal logic, and return "yes" or "no", indicating whether or not the model satisfies the specification. In the case of probabilistic model checking, the models are probabilistic, in the sense that they

* Supported in part by EPSRC grant GR/M04617 and MathFIT studentship for David Parker.

P. Kemper (Eds.), Tools Session of Aachen 2001 International Muliconference on Measurement, Modelling and Evaluation of Computer-Communication Systems, Technical Report 760/2001, pages 7–12, University of Dortmund, 2001.

encode the *probability* of making a transition between states instead of simply the existence of such a transition, and analysis normally entails calculation of the actual likelihoods through appropriate numerical or analytical methods.

2 Probabilistic model checking

A number of probabilistic models exist. The simplest are *discrete-time Markov chains* (DTMCs), which specify the probability $\pi(s, s')$ of making a transition from state s to some target state s', where the probabilities of reaching the target states from a given state must sum up to 1, i.e. $\sum_{s'} \pi(s, s') = 1$. *Markov decision processes* (MDPs) extend DTMCs by allowing both probabilistic and non-deterministic behaviour. Non-determinism enables the modelling of asynchronous parallel composition of probabilistic systems, and permits the under-specification of certain aspects of a system. *Continuous-time Markov chains* (CTMCs), on the other hand, specify the rates $\rho(s, s')$ of making a transition from state s to s', with the interpretation that the probability of moving from s to s' within t time units (for positive real valued t) is $1 - e^{-\rho(s,s')\cdot t}$.

Probabilistic specification formalisms include PCTL [16, 10, 9], a probabilistic extension of the temporal logic CTL applicable in the context of MDPs and DTMCs, and the logic CSL [8], a specification language for CTMCs based on CTL and PCTL.

PCTL allows us to express properties of the form "under any scheduling of processes, the probability that event A occurs is at least p (at most p)". By way of illustration, we consider Pnueli and Zuck's randomised solution to mutual exclusion [26] which gives rise to an MDP. In this algorithm, processes make random choices based on coin tosses to ensure that they can all enter their critical sections eventually (although not simultaneously). We use atomic propositions try_i and $crit_i$ to label states where process i is either trying to enter its critical section or is in it, respectively. Some examples of properties we would wish to verify can be expressed in PCTL as follows:

- $try_1 \rightarrow \mathcal{P}_{\geq 1}[\; true\ \mathcal{U}\ crit_1 \;]$ - "under any scheduling, if process 1 tries to enter its critical section, then it eventually succeeds with probability 1"
- $\mathcal{P}_{<0.5}[\; \neg(crit_2 \vee crit_3)\ \mathcal{U}\ crit_1 \;]$ - "under any scheduling, the probability of process 1 entering its critical section before process 2 or 3 is less than 0.5".

The specification language CSL includes the means to express both transient and steady-state performance measures of CTMCs. Transient properties describe the system at a fixed real-valued time instant t, whereas steady-state properties refer to the behaviour of a system in the "long run". For example, consider a queueing system where the atomic proposition *full* labels states where the queue is full. CSL then allows us to express properties such as:

- $\mathcal{P}_{\leq 0.01}[\; true\ \mathcal{U}^{\leq t}\ full \;]$ - "the probability that the queue becomes full within t time units is less than or equal to 0.01"
- $\mathcal{S}_{\geq 0.98}[\; \neg full \;]$ - "in the long run, the chance that the queue is not full is greater than or equal to 0.98".

3 The Tool

3.1 Functionality

PRISM takes as input a description of a system written in a probabilistic variant of Reactive Modules [1][1]. It first constructs the model from this description and computes the set of reachable states. The model can be a DTMC, an MDP or a CTMC. PRISM accepts specifications in either the logic PCTL or CSL depending on the model type. The tool then performs model checking to determine which states of the system satisfy each specification. For PCTL properties and DTMC or MDP models, PRISM implements the algorithms of [16, 10, 9] (including fairness) and the subsequent improvements of [3]. For CSL and CTMCs, methods based on [7] are used. It is also possible to export the transition matrix of the model, enabling analysis in other applications and visualisation of the model. Fig. 1 shows the structure of the tool and Fig. 2 shows a screen-shot of the tool running.

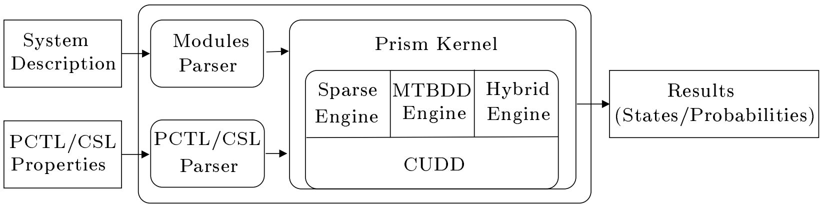

Fig. 1. PRISM System Architecture

In PRISM, model construction and reachability are implemented using MTBDDs and BDDs respectively. It has been shown in [15] that space efficient representations of structured probabilistic models can be constructed using MTBDDs. Reachability analysis using BDDs forms the basis of non-probabilistic symbolic model checking which has proven to be very successful [11, 24].

For both PCTL and CSL, model checking generally reduces to a combination of reachability-based computation (manipulation of sets of states) and the solution of linear equation systems or linear optimisation problems. Again, reachability based computation is performed using BDDs. In the case of numerical computation, however, PRISM supports three different model checking engines. The first is based on symbolic model checking using MTBDDs (multi-terminal BDDs) [13]; more details can be found in [5, 15]. The second uses more conventional data structures for numerical analysis: sparse matrices and full vectors. The latter engine nearly always provides faster numerical computation than its MTBDD counterpart. MTBDDs can, however, sometimes exploit structure in models and represent them far more compactly than a sparse matrix can [15]. In cases where high regularity occurs, we have been able to perform quantitative analysis for models substantially larger than those representable in a sparse matrix form. The third engine can be seen as a hybrid of the other two. It stores

[1] For further information on the language, see `www.cs.bham.ac.uk/~dxp/prism`

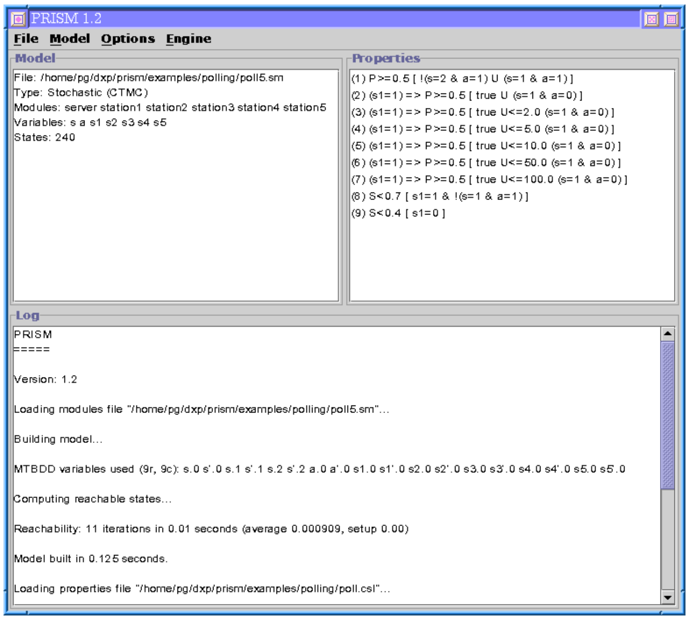

Fig. 2. The PRISM User Interface

models in a MTBDD-like structure which is adapted so that numerical computation can be carried out in combination with a full vector. This hybrid approach is faster than MTBDDs and can handle larger systems than sparse matrices.

3.2 Implementation

PRISM is implemented using a combination of Java and C++. All high-level parts of the tool, such as the user interface and parsers are written in Java. Low-level code and libraries are mostly in C++. For BDDs and MTBDDs, PRISM uses the CUDD package [28], which is written in C.

4 Results

We have used PRISM to build and analyse probabilistic models for a number of case studies. For MDP models, we have considered several randomised distributed algorithms, including the randomised mutual exclusion protocols of [27, 26] and the randomised consensus protocol of [2]. In the latter case, we were able to verify quantitative PCTL properties for MDPs with up to 10^{10} states using the MTBDD engine [23]. We have also considered a number of CTMC models. These include a cyclic polling system [22], a tandem queueing network [21], a kanban flexible manufacturing system [12], a workstation cluster [18] and a cell of a wireless communication network [17]. For example, in the workstation cluster case study, we have used the hybrid engine in PRISM to verify the property "the chance that the quality of service drops below minimum quality within 85 time units is less than 10%" for systems of up to 9 million states. Fig. 3 below includes statistics for some of the case studies mentioned above.

model	number of states	model construction time (in *sec*)	model size (KB)	number of iterations	time per iteration (in *sec*)		
					MTBDD engine	Sparse engine	Hybrid engine
consensus protocol	4.3×10^8	13.2	106	181,791	6.0	-	-
	1.0×10^{10}	16.0	170	85,641	11.5	-	-
workstation cluster	2.3×10^6	33.6	1878	2570	-	-	11.3
	9.4×10^6	151.2	3908	2630	-	-	44.5
polling system	$73,728$	0.4	36	584	1.29	0.17	0.25
	$159,744$	0.8	42	584	3.03	0.32	0.55

Fig. 3. Statistics for model checking with PRISM

Further information about these examples, additional case studies and the tool itself can be found on the PRISM web site at `www.cs.bham.ac.uk/~dxp/prism`.

5 Conclusions and Future Work

We have introduced PRISM, a tool to build and analyse probabilistic systems which supports three types of models (DTMCs, MDPs and CTMCs) and two probabilistic logics (PCTL and CSL). Several DTMC and CTMC analysis tools are available, for example MARCA [29] and TIPPtool [19], which do not allow logic specifications and instead support steady-state and transient analysis. Of the two probabilistic model checking tools that we are aware of, ProbVerus [4] only supports DTMCs and a subset of PCTL, whereas $\mathsf{E} \vdash \mathsf{MC}^2$ [20] only supports the model checking of CTMCs using CSL specifications. PRISM is the only model checking tool which allows the quantitative model checking of MDPs.

The development of PRISM is an ongoing activity. In the near future we plan to consider extensions of PCTL for expressing expected time and long run average properties [14] and of CSL to include rewards [6]. We are also in the process of making efficiency improvements to the tool, in particular to the hybrid engine. Details of this work will be available in [25].

References

1. R. Alur and T. Henzinger. Reactive modules. In *Proc. LICS'96*, 1996.
2. J. Aspnes and M. Herlihy. Fast Randomized Consensus using Shared Memory. *Journal of Algorithms*, 15(1), 1990.
3. C. Baier. On algorithmic verification methods for probabilistic systems. Universität Mannheim, 1998.
4. C. Baier, E. Clarke, and V. Hartonas-Garmhausen. On the semantic foundations of Probabilistic VERUS. In *Proc. PROBMIV '98*, volume 21 of *ENTCS*, 1998.
5. C. Baier, E. Clarke, V. Hartonas-Garmhausen, M. Kwiatkowska, and M. Ryan. Symbolic model checking for probabilistic processes. In *Proc. ICALP'97*, 1997.

6. C. Baier, B. Haverkort, H. Hermanns, and J. Katoen. On the logical characterisation of performability properties. In *Proc. ICALP 2000*, 2000.
7. C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model checking continuous-time Markov chains by transient analysis. In *CAV 2000*, 2000.
8. C. Baier, J.-P. Katoen, and H. Hermanns. Approximative symbolic model checking of continuous-time Markov chains. In *Proc. CONCUR'99*, 1999.
9. C. Baier and M. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11(3), 1998.
10. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proc. FST & TCS*, 1995.
11. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proc. LICS'90*, 1990.
12. G. Ciardo and M. Tilgner. On the use of Kronecker operators for the solution of generalized stocastic Petri nets. ICASE Report 96-35, Institute for Computer Applications in Science and Engineering, 1996.
13. E. Clarke, M. Fujita, P. McGeer, J. Yang, and X. Zhao. Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. In *Proc. IWLS'93*, 1993.
14. L. de Alfaro. How to specify and verify the long-run average behavior of probabilistic systems. In *Proc. LICS'98*, 1998.
15. 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. TACAS 2000*, volume 1785 of *LNCS*, 2000.
16. H. Hansson and B. Jonsson. A logic for reasoning about time and probability. *Formal Aspects of Computing*, 6, 1994.
17. G. Haring, R. Marie, R. Puigjaner, and K. Trivedi. Loss formulae and their application to optimization for cellular networks. In *IEEE Trans. on Vehicular Technology*, 2000.
18. B. Haverkort, H. Hermanns, and J.-P. Katoen. On the use of model checking techniques for dependability evaluation. In *Proc. 19th IEEE Symposium on Reliable Distributed Systems*, 2000.
19. H. Hermanns, U. Herzog, U. Klehmet, V. Mertsiotakis, and M. Siegle. Compositional performance modelling with the TIPPtool. *Perf. Eval.*, 39(1-4), 2000.
20. H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. A Markov Chain Model Checker. In *Proc. TACAS 2000*, volume 1785 of *LNCS*, 2000.
21. H. Hermanns, J. Meyer-Kayser, and M. Siegle. Multi terminal binary decision diagrams to represent and analyse continuous time Markov chains. In *Proc. NSMC'99*, 1999.
22. O. Ibe and K. Trivedi. Stochastic Petri net models of polling systems. *IEEE Journal on Selected Areas in Communications*, 8(9), 1990.
23. M. Kwiatkowska, G. Norman, and R. Segala. Automated verification of a randomized distributed consensus protocol using Cadence SMV and PRISM. In *Proc. CAV'01*, 2001. To appear.
24. K. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.
25. D. Parker. *Implementation of symbolic model checking for probabilistic system*. PhD thesis, University of Birmingham, 2001. To appear.
26. A. Pnueli and L. Zuck. Verification of multiprocess probabilistic protocols. *Distributed Computing*, 1(1), 1986.
27. M. Rabin. N-process mutual exclusion with bounded waiting by $4\log_2 N$-valued shared variable. *Journal of Computer and System Sciences*, 25(1), 1982.

28. F. Somenzi. CUDD: CU Decision Diagram package. Public software, Colorado University, Boulder, 1997.
29. W. Stewart. MARCA: Marcov chain analyzer. a software package for Markov modelling. In *Proc. NSMC'91*, 1991.