PRISM: A Tool for Automatic Verification of Probabilistic Systems*

Andrew Hinton, Marta Kwiatkowska, Gethin Norman, and David Parker

School of Computer Science, University of Birmingham,
Birmingham B15 2TT, United Kingdom
{ug60axh,mzk,gxn,dxp}@cs.bham.ac.uk

Abstract. Probabilistic model checking is an automatic formal verification technique for analysing quantitative properties of systems which exhibit stochastic behaviour. PRISM is a probabilistic model checking tool which has already been successfully deployed in a wide range of application domains, from real-time communication protocols to biological signalling pathways. The tool has recently undergone a significant amount of development. Major additions include facilities to manually explore models, Monte-Carlo discrete-event simulation techniques for approximate model analysis (including support for distributed simulation) and the ability to compute cost- and reward-based measures, e.g. "the expected energy consumption of the system before the first failure occurs". This paper presents an overview of all the main features of PRISM. More information can be found on the website: www.cs.bham.ac.uk/~dxp/prism.

1 Overview

Probabilistic model checking is an automatic formal verification technique for the analysis of systems which exhibit stochastic behaviour. Examples of such systems include well-known communication protocols such as FireWire and Bluetooth, which employ randomisation, and a wide range of computer and communication systems, unpredictable characteristics of which, such as message delays or times to failure, are best represented in a probabilistic fashion. Like traditional model checking, this technique involves constructing, from a description in some high-level formalism, a finite-state model of a real-life system, but additionally including information about the likelihood and timing of transitions between states occurring. From this model, a wide range of quantitative measures of the original system can be automatically computed.

PRISM is a probabilistic model checking tool which has already been used to apply these techniques to a large and diverse set of case studies. In the following sections we describe the types of probabilistic model supported by PRISM and the properties of these models which can be analysed. We then give an overview of the main features of the tool. Finally, we summarise the case studies to which the tool has already been applied and the various resources which are available.

* Supported in part by EPSRC grants GR/S11107 and GR/S46727 and Microsoft Research Cambridge contract MRL 2005-44.

2 PRISM model specification

PRISM has direct support for three types of probabilistic models: *discrete-time Markov chains* (DTMCs), *Markov decision processes* (MDPs) and *continuous-time Markov chains* (CTMCs). In DTMCs, time is modelled as discrete time-steps and the probabilities of transitions occurring are also discrete. They are suitable for analysing systems with simple probabilistic behaviour and no concurrency e.g. synchronous randomised distributed algorithms. MDPs extend DTMCs by permitting a combination of nondeterminism and probability, making them well suited to modelling multiple probabilistic processes executing in parallel or to cases where some parameters of the system or the behaviour of the environment in which it is operating are unknown. CTMCs do not support nondeterminism but model time in a continuous fashion, through the use of the negative exponential distributions, allowing accurate representation of the timing characteristics of e.g. component failures and job arrivals.

PRISM now also allows models to be augmented with *costs* and *rewards*, real values assigned to states and transitions of the model. This permits reasoning about a much wider range of quantitative measures of a system, e.g. "completion time", "energy consumption" or "number of messages lost".

Models are specified using the PRISM modelling language, a simple, state-based language based on the Reactive Modules formalism. Systems are described as the parallel composition of a set of modules. Each module's state is given by a set of finite-ranging variables and its behaviour by a set of probabilistic guarded commands. The language also supports global variables, synchronisation and various process algebraic operations. See the PRISM documentation and example repository at [1] for more information.

3 PRISM property specification

The specification language for properties of the probabilistic models to be analysed in PRISM is based on temporal logic, in particular PCTL and CSL, probabilistic extensions of the logic CTL. The principal operators are P, S and R which refer, respectively, to the probability of an event occurring, the long-run probability of some condition being satisfied and the expected value of the model's costs or rewards. For precise details of the specification language, see the PRISM documentation [1]. For the theoretical background and further references, see [2]. For illustrative purposes, a selection of example properties is shown below:

- $\mathtt{P} \geq 0.9$ [$!repair$ $\mathtt{U} \leq 200$ $done$] - "with probability 0.9 or more, the process will successfully complete within 200 hours and without requiring repairs"
- $\mathtt{P} =?$ [$\mathtt{F} \leq T$ $error$ $\{\mathtt{init}\}\{\mathtt{max}\}$] - "what is the worst-case probability, over all possible initial configurations, that an error has occurred by time T"
- $\mathtt{S} =?$ [$num\_sensors \geq min\_sensors$] - "what is the long-run probability that an acceptable number of sensors are operational?"
- $\mathtt{R} < 3$ [$\mathtt{C} \leq T$] - "the expected number of messages lost during the first T minutes of execution of the communication protocol is less than 3"

– $\mathtt{R}=?\,[\,\mathtt{F}\ \mathit{shutdown}\ \{\mathit{error\_detected}\}\{\mathtt{max}\}\,]$ - "from all situations where an error has been detected, what is the worst-case expected power consumption before the system shuts itself down?"

Note that is possible to either determine whether a probability or expected quantity satisfies a given bound or obtain the actual value. In the latter case, it is often beneficial to compute a range of values in order to identify trends or anomalies. The ability to examine worst-case (or best-case) scenarios, as illustrated in the examples above, is also very powerful.

4 The PRISM tool

The core functionality of PRISM, namely constructing a probabilistic model, and then evaluating the result of one or more corresponding properties, is available from either a command-line or a graphical user interface. The latter includes editors for the PRISM modelling and property specification languages. It also facilitates generation of series of quantitative results and plotting of graphs to visualise them. A recent addition is the ability to view specific traces of model execution for the purposes of debugging or sanity checks. These are generated either by manual exploration or automatically in probabilistic fashion. Figure 1 shows screenshots of some of this functionality in operation.

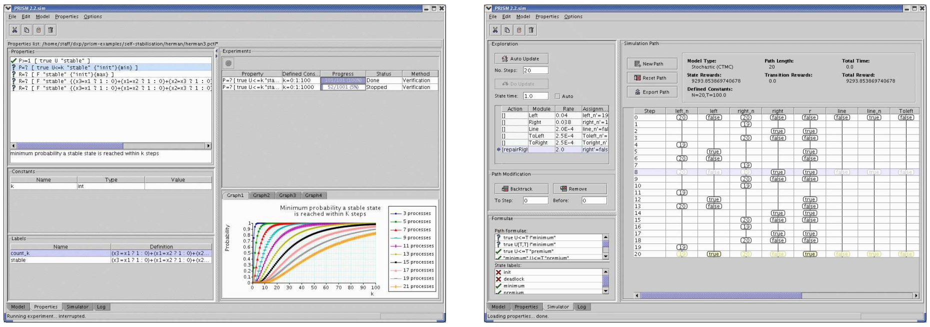

Fig. 1. Screenshots of PRISM running. Left: graphical visualisation of quantitative model checking results. Right: manual exploration of model traces

PRISM incorporates a range of model analysis techniques. These include qualitative methods, such as graph-based algorithms for reachability, and quantitative methods for numerical computation of probabilities and expected cost or reward values. For the latter, multiple implementations are provided. In particular, this includes state-of-the-art symbolic approaches which use data structures based on binary decision diagrams (BDDs) to exploit model structure and regularity.

The most recent addition is support for approximate numerical computation using Monte-Carlo methods and discrete event simulation. PRISM can generate multiple executions through a model based on a faithful simulation of its probabilistic and timing characteristics. These samples are then used to compute approximate quantitative results. Since this approach avoids the (costly) construction of the full probabilistic model, working instead with the PRISM language description, it is potentially applicable to much larger models than the alternative numerical solution approach. Furthermore, samples can be generated independently, so it is possible to distribute the simulation process over multiple computers. The PRISM user interface includes a tool to manage this process.

Connections to other tools and formalisms
To allow connections with external tools, PRISM allows the export of a model's transition matrix and state space in a variety of formats: either plain text or tailored for specific tools, including Matlab, ETMCC and MRMC. Models specified in alternative formalisms can also be imported via translation. PRISM already has native support for a subset of the stochastic process algebra PEPA and others are underway. It is now also possible to import the transition matrix and state space of a model directly in a simple textual format.

Examples and case studies
PRISM has been successfully applied to a large number of case studies from a wide array of application areas, on several occasions resulting in the identification of interesting or anomalous behaviour. The website [1] provides details of over thirty case studies, developed both by members of the PRISM team and external research groups, including links to the corresponding publications and source code. Examples include analysis of the performance, reliability or correctness of:

- real-time communication protocols, including IEEE 1394 FireWire, Bluetooth, Zeroconf, IEEE 802.3 CSMA/CD and IEEE 802.11 wireless LANs;
- probabilistic security protocols for anonymity (Crowds protocol, synchronous batching), contract signing, fair exchange and non-repudiation;
- randomised distributed algorithms for leader election, consensus, Byzantine agreement, self-stabilisation and mutual exclusion;
- dynamic power management and voltage scaling schemes;
- biological signalling pathways.

Tool availability and resources
PRISM is a free and open source tool, distributed under the GNU General Public License (GPL), and now supports most major operating systems: Linux, Solaris, Windows and Mac OS X. Ports have also been developed for 64-bit architectures. The PRISM website [1] contains a wealth of further information and resources, including related publications, the tool source code and binaries, user manual and a large repository of illustrative example models.

References

[1] PRISM web site. www.cs.bham.ac.uk/~dxp/prism.
[2] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems,* P. Panangaden and F. van Breugel (eds.), volume 23 of *CRM Monograph Series.* AMS, 2004.