PRISM 2.0: A Tool for Probabilistic Model Checking

Marta Kwiatkowska Gethin Norman David Parker
School of Computer Science, University of Birmingham, Birmingham, B15 2TT, UK

Abstract

This paper gives a brief overview of version 2.0 of PRISM, a tool for the automatic formal verification of probabilistic systems, and some of the case studies to which it has already been applied.

1. Introduction

The use of probabilistic modelling for the analysis and verification of computer systems is becoming more and more widespread. *Probabilistic model checking* is an automatic, formal method for establishing the correctness, performance and reliability of systems which exhibit random behaviour. This technique involves constructing a probabilistic model, typically from a description in some high-level language, and then establishing whether or not this model satisfies a formal specification of its desired behaviour, usually expressed in temporal logic.

This paper describes the probabilistic model checking tool PRISM. In the following two sections, we give an overview of the capabilities of the tool and highlight the new features of the latest version 2.0. Subsequently, we summarise some of the case studies to which it has been applied and discuss various current and planned improvements and extensions to the tool.

2. Tool Overview

PRISM has been developed at the University of Birmingham. It provides direct support for the analysis of three types of probabilistic models: discrete-time Markov chains (DTMCs), where the behaviour at each discrete time-step is modelled by a discrete probability distribution; continuous-time Markov chains (CTMCs), which allow transitions to occur in real-time, with delays which are exponentially distributed; and Markov decision processes (MDPs), which combine discrete probabilities and nondeterminism for accurate modelling of concurrency. PRISM also provides indirect support (via either digital clocks [3] or KRONOS [2]) for model checking probabilistic timed automata, which include probability, nondeterminism and real-time using clocks. For an overview of all four types of models and the techniques which can be used to analyse them, see e.g. [4].

Models are specified to PRISM using a high-level system description language. Each component of the system is described as a PRISM *module*. The state of a module is determined by a set of finite-ranged variables and its behaviour is given using a guarded-command based notation. Communication between modules is via either global variables or synchronisation over common action labels.

PRISM takes such a model description, constructs the corresponding probabilistic model, a process which includes determining the set of all reachable states of the model. Subsequently, the model can be analysed by verifying that properties, specified in temporal logic, are satisfied by the system. PRISM uses the logic PCTL for properties of DTMCs and MDPs, and the logic CSL for properties of CTMCs. The following examples illustrate the type of properties which can be expressed:

- $\mathcal{P}_{\geq 1}[\lozenge\ terminate]$ – "the algorithm eventually terminates successfully with probability 1"
- $down \implies \mathcal{P}_{>0.75}[\neg fail\ \mathcal{U}^{[1,2]}\ up]$ – "when a shutdown occurs, the probability of system recovery being completed in between 1 and 2 hours without further failures occurring is greater than 0.75"
- $\mathcal{S}_{<0.01}[num\_sensors < min]$ "in the long-run, the probability that an inadequate number of sensors are operational is less than 0.01"

PRISM can be run in two modes: from a command-line or using the graphical user interface. It operates on most Unix-based operating systems. Both binary and source code versions can be freely downloaded from the website [1].

3. New Features in PRISM 2.0

System Description Language: This language has been significantly enhanced. For example, it now includes types (and type checking) and allows probabilities and rates to be defined as expressions, that is, to be dependent on the variables of the model. In addition, the language now supports general process algebra expressions including asyn-

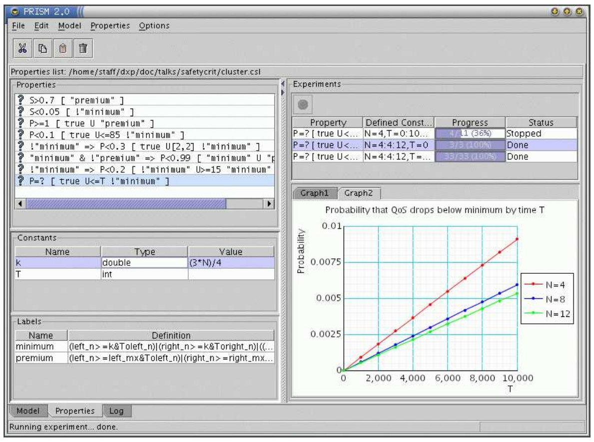

Figure 1. A screenshot of PRISM 2.0 running

chronous and synchronous parallel composition and renaming and hiding of action labels. Using these facilities, PRISM now provides direct support for descriptions in a subset of the PEPA process algebra.

Property Specifications: The tool now supports the full versions of both the logics PCTL and CSL. Furthermore, it now also allows direct querying of probabilities, through properties such as: $\mathcal{P}_{=?}[\Diamond^{\leq T} fault]$ – "what is the probability that a fault has occurred by time T?" In many cases, the most useful way of analysing the behaviour of system or identifying anomalies is to compute a range of such values. PRISM now facilitates this with the idea of *experiments*. The user specifies an experiment (the range of values for constants in the model and/or property in question) and then PRISM automatically verifies each case.

Graphical User Interface: PRISM 2.0 has a completely new graphical user interface (see Figure 1) with features including a built-in text-editor for the PRISM language, easy creation of PCTL/CSL properties and the capability to plot the results of experiments graphically.

Efficiency Improvements: Several additional numerical solution methods have been added to PRISM. Low-level optimisations to the BDD-based implementations of these methods have also been incorporated.

4. Case Studies

PRISM has been successfully applied to a large number of case studies in a wide range of application domains, listed below. Further information for all of these case studies can be obtained from the tool website [1]. PRISM code for many of them is also distributed with the tool itself.

- Analysis of quality of service (QoS) properties of several real-time communication protocols, including IEEE 1394 FireWire root contention, Zeroconf, IEEE 802.3 CSMA/CD and IEEE 802.11 wireless LANs.
- Verification of probabilistic security protocols for anonymity (Crowds protocol, synchronous batching), contract signing, fair exchange and non-repudiation.
- Analysis of randomised distributed algorithms for consensus, Byzantine agreement, leader election, self-stabilisation and mutual exclusion.
- Evaluation of the performance, reliability and dependability of a wide range of systems, including dynamic power management schemes, NAND multiplexing for nanotechnology, controller systems, PC clusters, manufacturing systems and queueing systems.

5. Current and Future Work

Enhancements to PRISM are currently underway in a number of areas. One of the most significant is the development of support for costs and rewards. A simple extension to the PRISM language allows states and transitions of any model to be assigned real-valued costs or rewards. Analysis of the model can then focus not just on the probability of events occurring, but the expected cost or rewards accumulated in doing so. This permits computation of a wide range of useful measures, e.g. expected time, expected power consumption or expected number of failures.

At a higher level, work is also being undertaken to add more powerful model checking techniques to PRISM, for example detection and exploitation of symmetry and compositional verification. Furthermore, work is being carried out to facilitate the specification and verification of mobile applications, in particular ad-hoc network protocols.

Planned improvements to PRISM also include: the addition of parallel, distributed and disk-based engines; the integration of approximation and sampling-based techniques; a simulator for manual or automatic state space exploration; graphical extensions to the PRISM modelling language; and the addition of native support for model checking of probabilistic timed automata.

References

[1] PRISM web site. www.cs.bham.ac.uk/~dxp/prism.

[2] C. Daws, M. Kwiatkowska, and G. Norman. Automatic verification of the IEEE 1394 root contention protocol with KRONOS and PRISM. *International Journal on Software Tools for Technology Transfer (STTT)*, 5(2–3):221–236, 2004.

[3] M. Kwiatkowska, G. Norman, D.Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. In *Proc. FORMATS'03*, volume 2791 of *LNCS*. Springer, 2003.

[4] 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.