PRISM 4.0: Verification of Probabilistic Real-Time Systems

Marta Kwiatkowska[1], Gethin Norman[2], and David Parker[1]

[1] Department of Computer Science,
University of Oxford, Oxford, OX1 3QD, UK
[2] School of Computing Science,
University of Glasgow, Glasgow, G12 8RZ, UK

Abstract. This paper describes a major new release of the PRISM probabilistic model checker, adding, in particular, quantitative verification of (priced) probabilistic timed automata. These model systems exhibiting *probabilistic*, *nondeterministic* and *real-time* characteristics. In many application domains, all three aspects are essential; this includes, for example, embedded controllers in automotive or avionic systems, wireless communication protocols such as Bluetooth or Zigbee, and randomised security protocols. PRISM, which is open-source, also contains several new components that are of independent use. These include: an extensible toolkit for building, verifying and refining abstractions of probabilistic models; an explicit-state probabilistic model checking library; a discrete-event simulation engine for statistical model checking; support for generation of optimal adversaries/strategies; and a benchmark suite.

1 Introduction

This paper describes a major new release, version 4.0, of the PRISM probabilistic model checker. This adds, in particular, formal modelling and analysis capabilities for systems with *probabilistic*, *nondeterministic* and *real-time* characteristics, through support for verification of (priced) *probabilistic timed automata*.

PRISM already provides model checking for several types of probabilistic models: discrete- and continuous-time Markov chains and Markov decision processes, as well as a modelling language in which to express them. The tool has been widely taken up (downloaded more than 20,000 times) and used for *quantitative verification* in a broad spectrum of application domains, from wireless communication protocols to quantum cryptography to systems biology. In many cases, flawed or anomalous behaviour has been identified, from worst-case performance conditions for Bluetooth [2] to behavioural predictions (later validated experimentally) for biological signalling pathways [4].

Increasingly, though, new application domains are dictating the need for quantitative verification techniques and tools for richer classes of models. Embedded systems, such as in multimedia devices or avionic systems, exhibit stochastic behaviour and also operate under constraints on timing and other resources. PRISM 4.0 supports (priced) probabilistic timed automata, a natural model for

G. Gopalakrishnan and S. Qadeer (Eds.): CAV 2011, LNCS 6806, pp. 585–591, 2011.

Time	Nondeterminism	Probabilistic Models
discrete	no	discrete-time Markov chains (DTMCs)
	yes	Markov decision processes (MDPs) probabilistic automata (PAs)
continuous	no	continuous-time Markov chains (CTMCs)
	yes	**probabilistic timed automata (PTAs)** **priced probabilistic timed automata (PPTAs)**

Fig. 1. The types of probabilistic models currently supported by PRISM, classified by modelling of time and the presence of nondeterminism; boldface denotes new additions

such systems. It also incorporates several new components, including engines for *quantitative abstraction-refinement* [9] and *statistical model checking* [14,5]. The components are designed to be extensible and re-usable. As well as improving scalability for existing model types and adding support for (infinite-state) PTAs, they are targeted at facilitating verification of more expressive classes of models such as probabilistic and stochastic hybrid systems.

1.1 Functionality Overview

We begin with a brief overview of the current functionality of the PRISM tool. Items in boldface denote new or improved features in version 4.0, which are described in more detail in the remainder of the paper.

- modelling and construction of many types of probabilistic models (see Fig. 1 for a summary), now including **probabilistic timed automata**; all can be augmented with costs or rewards, in the case of PTAs yielding the model of **priced probabilistic timed automata**;
- model checking of a wide range of quantitative properties, expressed in a language that subsumes the temporal logics PCTL, CSL, LTL and PCTL*, as well as extensions for quantitative specifications and costs/rewards;
- multiple model checking engines, both symbolic (BDD-based) and **explicit-state**; and a variety of probabilistic verification techniques, such as symmetry reduction and **quantitative abstraction refinement**;
- a **discrete-event simulator**, with support for **statistical model checking** methods, including confidence-level approximation and acceptance sampling;
- model import options, e.g. from SBML (systems biology markup language);
- **optimal adversary/strategy** generation for nondeterministic models;
- a GUI, with model editor, simulator and graphing, or command-line tool;
- a **benchmark suite** of probabilistic models and associated properties.

2 Probabilistic Timed Automata (PTAs)

Probabilistic timed automata (PTAs) [6,12] are finite-state automata enriched with real-valued clocks, in the style of timed automata, and with discrete probabilistic choice, in the style of Markov decision processes (MDPs).

Clocks are real-valued variables, whose values increase simultaneously over time. Predicates over clock variables called *guards* and *invariants* are assigned to transitions and states, respectively, imposing restrictions on when transitions can occur and how long can be spent in a state. For ease of modelling, we can also add finite-ranging *data variables* to a PTA. Transitions between states can reset clocks (to integer values) and update data variables. This is done *probabilistically*: the target state, clock resets and variable updates are specified by a discrete probability distribution. The choice between multiple transitions, as well as the elapse of time (subject to invariant satisfaction) are both *nondeterministic*.

Fig. 2 (left) shows a simple example of a PTA, modelling repeated attempts to transmit a message over an unreliable channel. The system tries to send for between 1 and 2 time-units. With probability 0.1, this fails, in which case a delay of between 3 and 5 time-units elapses before retrying (up to N times).

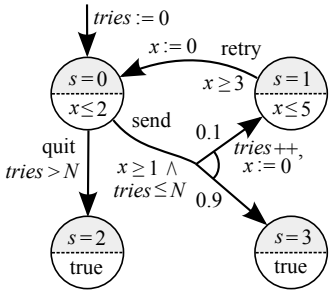
PTAs can be augmented with information about costs incurred or, equivalently, rewards gained (PRISM uses the latter terminology). This model, often known as *priced* PTAs, allows reasoning about a wide range of additional properties, such as energy consumption or resource usage. PRISM supports *linearly* priced PTAs, where costs/rewards are accumulated at a rate proportional to the elapse of time, with the rate depending on the current state (and data variables).

Finally, we mention that PTAs also support *parallel composition* (as for timed automata and MDPs), in which multiple PTAs operate concurrently, synchronising on transitions with matching labels. For precise details, see [11].

Modelling PTAs. PRISM uses a uniform modelling language for all the probabilistic models that it supports, including PTAs. This is a textual language, based on guarded command notation. To support PTAs, PRISM 4.0 adds a new `clock` datatype. Clock variables can appear (as convex expressions) in guards, on the left-hand side of a command, and can be reset, like any other variable, with an update on the right-hand side. A new `invariant` keyword is introduced to allow expression of invariants. Fig. 2 (right) gives a PRISM modelling language description for the example PTA described above. It also shows an example of a PRISM reward structure, labelled "energy", to create a priced PTA which assigns a reward rate of 2.5 when s=0, i.e. during message transmission.

PTA verification techniques. PRISM analyses two main classes of properties for PTAs: (i) the minimum/maximum probability of reaching a target, possibly within a time bound (e.g. "the maximum probability of an airbag failing to deploy within 0.02 seconds"); and (ii) the minimum/maximum expected reward accumulated until a target is reached (e.g. "the maximum expected time for the protocol to terminate"). Two verification methods are implemented:

- *Quantitative abstraction refinement* [9] constructs and analyses a series of probabilistic abstractions, automatically refining at each step to produce more precise results (see also Section 3). By using *stochastic two-player game* abstractions, defined in terms of *zones*, this yields an effective technique for exact verification of probabilistic reachability properties of PTAs [10]. Since experimental results in [10] show that this method generally outperforms all others currently available, this is the default engine in PRISM.




```
pta
const int N;
module transmitter
    // Local variables
    s : [0..3] init 0;
    tries : [0..N+1] init 0;
    x : clock;
    invariant
        (s=0 ⇒ x≤2) & (s=1 ⇒ x≤5)
    endinvariant
    // Guarded commands
    [send] s=0 & x≥1 & tries≤N → 0.9 : (s′=3)
          + 0.1 : (s′=1)&(tries′=tries+1)&(x′=0);
    [retry] s=1 & x≥3 → (s′=0)&(x′=0);
    [quit] s=0 & tries>N → (s′=2);
endmodule
rewards "energy" (s=0) : 2.5; endrewards
```

Fig. 2. Left: A PTA, with clock x and integer variable *tries*, modelling attempted message transmission over an unreliable channel. Right: corresponding PRISM code.

– *Digital clocks* [11] performs an automatic model translation to an equivalent finite-state, discrete-time model (with integer-valued clocks)[1] and then uses PRISM's existing MDP model checking techniques. For (non-probabilistic) timed automata, such methods are usually much less efficient than on-the-fly zone-based reachability (as, e.g., in UPPAAL [13]). For PTAs, which lack such on-the-fly methods, the digital clocks approach remains competitive, especially in combination with PRISM's symbolic (BDD-based) implementation. This method also has the widest applicability, supporting both probabilistic reachability properties and expected cumulative rewards for (linearly) priced probabilistic timed automata [11].

Related tools. UPPAAL [13] is the leading verification tool for timed automata. A recent extension, UPPAAL-PRO [15], adds support for PTAs, but currently only analyses maximum probabilistic reachability properties. Fortuna [1] supports the same class but also allows the inclusion of reward-bounds when (linearly) priced PTAs are considered. Another tool aimed at verification of probabilistic real-time systems is `mcpta` [3], which translates a subset of the modelling language Modest into PRISM using *digital clocks* [11]. Finally, MRMC [7], a probabilistic model checker for Markov chains and Markov reward models, has recently also added support for continuous-time MDPs, another model combining nondeterministic, probabilistic and real-time features. For a more detailed list of probabilistic verification tools, including other tools for abstraction refinement (such as RAPTURE, PASS) and statistical model checking, see [18].

3 Other New PRISM Components and Features

Quantitative abstraction refinement toolkit. As described above, PRISM's default PTA verification technique uses *quantitative abstraction refinement* [9]. This can be seen as a quantitative analogue of classical counterexample-guided

[1] Under slight restrictions, e.g. *strict* clock comparisons (such as $x<1$) are not allowed.

abstraction refinement. It provides a fully automatic approach to verification of large or infinite-state probabilistic systems, by iteratively building, analysing and refining increasingly precise probabilistic abstractions. In addition to PTAs [10], the same approach has been applied to verification of probabilistic software (using predicate abstraction and SAT) [8] and to finite-state MDPs [9]. While these implementations all build abstractions of MDPs as stochastic two-player games, the same approach can be used to, for example, build abstractions of Markov chains as Markov decision processes. Quantitative abstraction refinement is implemented in PRISM in the form of an extensible toolkit, with support for multiple model types, a choice of refinement strategies and various configurable optimisations.

Explicit-state probabilistic model checking library. PRISM already features several model checking engines (called "MTBDD", "sparse", and "hybrid"), all either fully or partially *symbolic* (i.e. BDD-based). The tool now incorporates a new, entirely *explicit-state* probabilistic model checking library, implemented in Java and based on sparse matrix data structures. It supports stochastic two-player games, Markov decision processes and discrete- and continuous-time Markov chains. The code is designed to serve as a general purpose library, either for inclusion in other techniques or for prototyping new model checking algorithms. For example, the library is used in the abstraction-refinement toolkit, in which probabilistic models need to be constructed and modified on-the-fly, a task not well-suited to symbolic implementations.

Simulation engine and statistical model checking. Version 4.0 of PRISM incorporates a newly rewritten version of its *discrete-event simulation engine*. This provides efficient random generation of paths through PRISM models, both for the purposes of debugging models and to support so-called *statistical* (or *approximate*) model checking techniques [14,5]. PRISM now offers two types of such analysis. For *quantitative* properties (e.g. `P=?[·]` in PRISM notation), it either generates a confidence interval (based on a given confidence level) or a probabilistic guarantee of correctness, using the Chernoff-Hoeffding bound [5]. For *bounded* properties (e.g. `P<0.1[·]`), it uses *acceptance sampling* [14], implementing Wald's sequential probability ratio test (SPRT). Statistical model checking offers significantly improved scalability, in comparison to conventional probabilistic model checking techniques, and applies to a broader class of models.

Optimal adversary (strategy) generation. PRISM's MDP verification implementation now includes the ability to generate *optimal adversaries* (also known as strategies). This means that, when PRISM computes the minimum or maximum value for a probabilistic reachability (or expected reward) property, it can also generate an adversary (resolution of nondeterminism in the model) that produces it. This can be used to debug or analyse the results of model checking, for example in order to generate probabilistic counterexamples, or to produce an optimal solution for a scheduling problem. Furthermore, by using the digital clocks engine, optimal adversaries can also be generated for PTAs.

The PRISM benchmark suite. There are a large number of existing PRISM case studies, distributed with the tool, included in publications and on the tool website [16]. These are widely used, for example to evaluate new model checking techniques, or to compare model checking implementations and tools. Unfortunately, there are often several different variants of each model and it is not always easy to locate a particular class of models or properties. The *PRISM benchmark suite* [17] aims to provide a comprehensive source of freely-available benchmarks for probabilistic model checking. It includes a large selection of probabilistic models, of varying types and sizes, and corresponding properties for model checking, grouped by type. External contributions are also welcomed.

Technical details and availability. PRISM is free and open source (GPL). It is coded in a mix of Java and C++, and runs on all major operating systems. It is available for download from `http://www.prismmodelchecker.org/`.

Acknowledgments. The authors are part supported by ERC Advanced Grant VERIWARE, EU-FP7 project CONNECT, DARPA project PRISMATIC and EPSRC grant EP/D07956X. For a full list of PRISM contributors, see [16].

References

1. Berendsen, J., Jansen, D., Vaandrager, F.: Fortuna: Model checking priced probabilistic timed automata. In: Proc. QEST 2010, pp. 273–281 (2010)
2. Duflot, M., Kwiatkowska, M., Norman, G., Parker, D.: A formal analysis of Bluetooth device discovery. STTT 8(6), 621–632 (2006)
3. Hartmanns, A., Hermanns, H.: A Modest approach to checking probabilistic timed automata. In: Proc. QEST 2009, pp. 187–196 (2009)
4. Heath, J., Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O.: Probabilistic model checking of complex biological pathways. TCS 319(3), 239–257 (2008)
5. Hérault, T., Lassaigne, R., Magniette, F., Peyronnet, S.: Approximate probabilistic model checking. In: Steffen, B., Levi, G. (eds.) VMCAI 2004. LNCS, vol. 2937, pp. 73–84. Springer, Heidelberg (2004)
6. Jensen, H.: Model checking probabilistic real time systems. In: Proc. 7th Nordic Workshop on Programming Theory, pp. 247–261 (1996)
7. Katoen, J.P., Hahn, E.M., Hermanns, H., Jansen, D., Zapreev, I.: The ins and outs of the probabilistic model checker MRMC. In: Proc. QEST 2009, pp. 167–176 (2009)
8. Kattenbelt, M., Kwiatkowska, M., Norman, G., Parker, D.: Abstraction refinement for probabilistic software. In: Jones, N.D., Müller-Olm, M. (eds.) VMCAI 2009. LNCS, vol. 5403, pp. 182–197. Springer, Heidelberg (2009)
9. Kattenbelt, M., Kwiatkowska, M., Norman, G., Parker, D.: A game-based abstraction-refinement framework for Markov decision processes. In: FMSD, vol. 36(3) (2010)
10. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic games for verification of probabilistic timed automata. In: Ouaknine, J., Vaandrager, F.W. (eds.) FORMATS 2009. LNCS, vol. 5813, pp. 212–227. Springer, Heidelberg (2009)
11. Kwiatkowska, M., Norman, G., Parker, D., Sproston, J.: Performance analysis of probabilistic timed automata using digital clocks. FMSD 29, 33–78 (2006)

12. Kwiatkowska, M., Norman, G., Segala, R., Sproston, J.: Automatic verification of real-time systems with discrete probability distributions. TCS 282, 101–150 (2002)
13. Larsen, K., Pettersson, P., Yi, W.: UPPAAL in a nutshell. International Journal on Software Tools for Technology Transfer 1(1-2), 134–152 (1997)
14. Younes, H.L.S., Simmons, R.G.: Probabilistic verification of discrete event systems using acceptance sampling. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 223–235. Springer, Heidelberg (2002)
15. http://www.cs.aau.dk/~arild/uppaal-probabilistic/
16. http://www.prismmodelchecker.org/
17. http://www.prismmodelchecker.org/benchmarks/
18. http://www.prismmodelchecker.org/other-tools.php