Verification of Probabilistic Real-time Systems

David Parker

Abstract—Probabilistic model checking is a formal verification technique for systems that exhibit stochastic behaviour. It has been used to analyse a wide range of systems, including communication protocols, such as Bluetooth and FireWire, randomised security protocols, e.g. for anonymity and contract signing, and many others. This paper gives a short introduction to probabilistic model checking, with a particular focus on systems that incorporate real-time behaviour. We describe the model of probabilistic timed automata (PTAs), which can be used to represent systems with both probabilistic and real-time characteristics. We illustrate how to formally specify quantitative properties of PTAs, and give a brief summary of the techniques and tools that can be used to verify them. Pointers are provided throughout to further reading on this and related topics.

Index Terms—Probabilistic verification, probabilistic model checking, real-time systems, probabilistic timed automata.

1 INTRODUCTION

Model checking [1] is a successful and widely used technique for formal verification. It works by systematically exploring a model of a real-life system, in order to verify that certain correctness properties, typically specified using temporal logic, are satisfied by the model.

In many cases, however, it also necessary to consider quantitative aspects of the system, such as *probabilistic* behaviour or *real-time* constraints. Probability is widely used to model real-life systems. This could be to quantify unreliable or unpredictable behaviour, for example the failure of a system component, or the possibility of message loss across a wireless communication channel. Another common source of probabilistic behaviour is the explicit use of randomisation, for example as a symmetry breaker in back-off schemes for communication protocols such as IEEE 802.11 or Bluetooth. In many of these cases, it is also important to precisely model the timing of the system, for example to quantify the delay associated with the transmission of a message or the response time to a detected failure or a user request.

Probabilistic model checking is a generalisation of model checking to formally verify quantitative properties of such systems. This done by building and analysing a probabilistic model, such as a Markov chain, a Markov decision process, or a probabilistic timed automaton. Properties to be checked against these models are expressed in probabilistic or timed extensions of temporal logics, which allow us to specify requirements such as "the maximum probability of an airbag failing to deploy within 0.02 seconds is at most 10^{-6}".

• *D. Parker is with the School of Computer Science, University of Birmingham, Birmingham, B15 2TT, UK*
E-mail: d.a.parker@cs.bham.ac.uk.

2 PROBABILISTIC TIMED AUTOMATA

Probabilistic model checking techniques have been developed for a wide variety of types of probabilistic models. Some of these models assume a discrete notion of time, others assume it is continuous (dense). Another useful classification is between those that incorporate nondeterministic behaviour (e.g., to model concurrency or control), and those that are instead fully probabilistic. A non-exhaustive list of models in common use, classified along these lines, is as follows:

	Fully probabilistic	**Nondeterministic**
Discrete -time	discrete-time Markov chains	Markov decision processes
		probabilistic automata
Continuous -time	continuous-time Markov chains	**probabilistic timed automata**
		continuous-time Markov decision processes
		interval Markov chains

In this paper, we give a short introduction to *probabilistic timed automata* (PTAs) [2], [3], which permit modelling of probabilistic, nondeterministic and real-time behaviour. For more in-depth coverage of this topic, see for example [4]. For tutorial material on probabilistic checking for other models, see, e.g., [5] for fully probabilistic models, and [6], [7] for discrete-time models.

Probabilistic timed automata are labelled transition systems in which transitions to successor states occur randomly, according to discrete probability distributions, and which incorporate *clocks*: real-valued variables whose values increase simultaneously over time. As in standard timed automata [8], we annotate states and transitions of PTAs with *invariants* (predicates on clocks specifying how long a state can be occupied) and *guards* (predicates on clocks indicating when transitions can occur). Each time a transition between states occurs, one

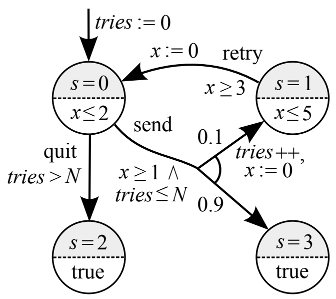

Fig. 1. A simple PTA, with clock x and integer variable $tries$, modelling the attempted message transmission of a message over an unreliable communication channel.

or more clocks may be reset to zero.

Syntactically, invariants and guards are expressed using *clock constraints*. Letting $\mathcal{X}$ denote a set of clocks, the set of all possible clock constraints $CC(\mathcal{X})$ is defined by the following grammar:

$$\chi ::= \mathtt{true} \mid x \leq d \mid c \leq x \mid x{+}c \leq y{+}d \mid \neg\chi \mid \chi \wedge \chi$$

where $x, y \in \mathcal{X}$ and $c, d \in \mathbb{N}$.

The formal definition of a PTA is then as follows.

Definition 1 (Probabilistic timed automaton)**.** *A probabilistic timed automaton (PTA) is defined by a tuple* $\mathsf{P}{=}(L, \bar{l}, \mathcal{X}, Act, inv, enab, prob, \mathcal{L})$ *where:*

- L *is a finite set of* locations*;*
- $\bar{l} \in L$ *is an* initial location*;*
- $\mathcal{X}$ *is a finite set of* clocks*;*
- Act *is a finite set of actions;*
- $inv : L{\to}CC(\mathcal{X})$ *is the* invariant condition*;*
- $enab : L{\times}Act{\to}CC(\mathcal{X})$ *is the* enabling condition*;*
- $prob : L{\times}Act{\to}Dist(2^{\mathcal{X}}{\times}L)$ *is a (partial)* probabilistic transition function*;*
- $\mathcal{L} : L{\to}2^{AP}$ *is a* labelling function *mapping each location to a set of atomic propositions from a set* AP*.*

The semantics of a PTA is defined as an (infinite-state) Markov decision process whose states take the form $(l, v) \in L \times (\mathbb{R}_{\geq 0})^{\mathcal{X}}$, where l is a location and v is a valuation for all clocks in $\mathcal{X}$. Informally, the behaviour of a PTA can be described as follows. The initial state is $(\bar{l}, \mathbf{0})$, where $\mathbf{0}$ indicates that all clocks have value 0. For each state (l, v), there is a nondeterministic choice between either of the following events:

(i) time elapses, i.e. all clocks increase in value, subject to the invariant $inv(l)$ remaining true;
(ii) an action a is taken, assuming that $prob(l, a)$ is defined and the guard $enab(l, a)$ is satisfied; in this case, $prob(l, a)$ is a distribution over pairs $(X, l') \in 2^{\mathcal{X}} \times L$, which gives the probability of moving to location l' and resetting the clocks in X to zero.

States of a PTA are also labelled (by $\mathcal{L}$) with atomic propositions, indicating properties of interest that may be used during model checking. For a precise definition of the semantics of a PTA, see for example [4].

```
pta
const int N;
module transmitter
        // Local variables
        s : [0..3] init 0;
        tries : [0..N+1] init 0;
        x : clock;
        // Invariants
        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. PRISM modelling language description of the example PTA shown in Fig. 1.

Example. Fig. 1 shows a simple example of a PTA modelling repeated attempts to transmit a message over a faulty communication channel. The PTA has 4 locations (denoted $s{=}0, 1, 2, 3$) and a single clock x. There is also an integer variable $tries$, which is used to count the number of transmission attempts so far. For simplicity, we omitted such variables from Definition 1, but we can easily make the PTA of Fig. 1 conform to this definition by expanding it to a larger one with locations of the form $(s, tries)$, rather than just s.

Each location of the PTA is labelled with its invariant (in the lower half of the circle representing it). Transitions, shown as grouped arrows, are annotated with: actions (from the set $Act = \{send, retry, quit\}$), guards (e.g. $x \geq 3$), clock resets (e.g. $x := 0$) and probabilities.

In the system modelled by the PTA, each send happens after at most 2 time-units (because of the invariant $x{\leq}2$ in location $s{=}0$), after which the transmission succeeds with probability 0.9 and fails with probability 0.1. In the latter case, there is a delay of between 3 and 5 time units before transmission is re-attempted (captured by the invariant $x{\leq}5$ in location $s = 1$ and the guard $x{\geq}3$ on its outgoing transition). Once the counter variable $tries$ exceeds a constant N, message transmission is aborted and the PTA moves to location $s{=}2$.

3 Logics and Model Checking

3.1 Property specification

Typically, in probabilistic model checking, properties to be verified against a model are specified in extensions of classical temporal logics such as CTL [9] and LTL [10]. For probabilistic systems, the most common logic is PCTL (Probabilistic Computational Tree Logic) [11], [12], which generalises CTL with a P operator, referring to the probability of an event (specified as a path formula) occurring. There is also an extension of PCTL specifically for PTAs called PTCTL [3] which adds operators from the timed temporal logic TCTL [13].

In [4], a temporal logic for PTAs is proposed that also incorporates several useful features from the property specification language of the PRISM model checker [14]. This includes an $\mathtt{R}$ operator, used to reason about expected rewards (or dually costs), and numerical operators such as $\mathtt{P}_{\mathtt{max}=?}$, which reason directly about probability values, rather than asserting than the probability exceeds some threshold. We omit a full description of the syntax and semantics, and instead give a selection of example properties, taken from [4]:

- $\mathtt{P}_{\geq 0.8}[\mathtt{F}^{\leq k}\ \mathsf{ack}_n]$ – "the probability that the sender has received n acknowledgements within k clock-ticks is at least 0.8";
- $\mathsf{trigger} \rightarrow \mathtt{P}_{<0.0001}[\mathtt{G}^{\leq 20}\ \neg\mathsf{deploy}]$ – "the probability of the airbag failing to deploy within 20 milliseconds of being triggered is strictly less than 0.0001",
- $\mathtt{P}_{\mathtt{max}=?}[\neg\mathsf{sent}\ \mathtt{U}\ \mathsf{fail}]$ – "what is the maximum probability of a failure occurring before message transmission is complete?";
- $\mathtt{R}^{\mathbf{time}}_{\mathtt{max}=?}[\mathtt{F}\ \mathsf{end}]$ – "what is the maximum expected time for the protocol to terminate?";
- $\mathtt{R}^{\mathbf{pwr}}_{<q}[\mathtt{C}^{\leq 60}]$ – "the expected energy consumption during the first 60 seconds is $< q$".

By way of illustration, Fig. 3 shows numerical results obtained using the penultimate property from the list above, applied to a PTA model of the IEEE 1394 FireWire protocol [15]. In particular, an analysis is made of the FireWire root contention protocol, designed to resolve conflicts in a leader election algorithm, used when multiple FireWire devices assemble into a network. The algorithm uses a combination of randomisation and timing constraints to break symmetry: a probabilistic choice is made by each device whether to wait for a 'short' or 'long' delay before resubmitting a message after a collision. Fig. 3 shows the worst-case expected time for the root contention protocol to execute, for a range of different coin biases (i.e., values of the probability of choosing 'short'). Interestingly, this confirms a conjecture from [16] that performance can be optimised by using a biased coin (one that chooses a 'short' delay with probability approximately 0.56).

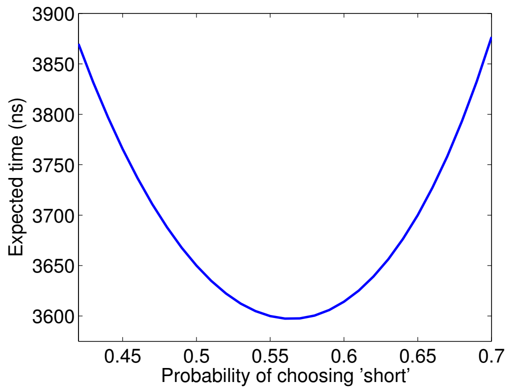

Fig. 3. Worst-case performance (maximum expected completion time) of the FireWire root contention protocol for a range of coin biases.

3.2 Model checking for PTAs

Model checking algorithms for PTAs need to combine techniques from existing algorithms for discrete-time probabilistic systems (notably for Markov decision processes) and for classical (non-probabilistic) timed automata. The challenge is to reduce the problem of verifying PTAs, which are inherently infinite-state models, to an analysis of a finite-state model.

Several different methods have been developed:

- *Digital clocks* [17], which translates a PTA (provided that it does not contain any strict inequalities in clock constraints) into a finite-state MDP, by *digitising* (real-valued) clocks to integer variables. The MDP can then verified using standard model checking techniques [12], [18], [7].
- *Backwards reachability* [19], which traverses the state space of a PTA backwards using symbolic (e.g., zone-based) data structures, again resulting in a finite-state MDP than can be analysed in standard fashion. Several useful optimisations to improve the efficiency of this technique can be found in [20].
- *Abstraction refinement with stochastic games* [21], which applies the quantitative abstraction refinement approach of [22] to iteratively construct increasingly precise abstractions of a PTA (represented as stochastic two-player games), finally resulting in exact results for the PTA model checking problem.

See the tutorial paper [4] for more information, or the references cited above for the full details.

3.3 Tool support for PTAs

There are now several software tools that can be used for probabilistic model checking of PTAs. One is the probabilistic model checker PRISM [14], which has support for many different types of probabilistic model, including Markov chains, Markov decision processes and PTAs. Models are specified using PRISM's modelling language, which is a textual language based on guarded commands. Fig. 2 shows the PRISM model description for the example PTA from Fig. 1. Properties are specified in temporal logic, as in the examples given earlier.

Other tools include: `mcpta` [23], part of the Modest [24] Toolset, which uses the "digital clocks" method and a connection to PRISM to verify a subset of the Modest modelling language; Fortuna [20], a tool that specialises in reward-based properties of PTAs; and UPPAAL PRO, a probabilistic extension of the popular timed automaton verifier UPPAAL [25].

References

[1] E. Clarke, O. Grumberg, and D. Peled, *Model Checking*. The MIT Press, 2000.

[2] H. Jensen, "Model checking probabilistic real time systems," in *Proc. 7th Nordic Workshop on Programming Theory*, 1996, pp. 247–261.

[3] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston, "Automatic verification of real-time systems with discrete probability distributions," *Theoretical Computer Science*, vol. 282, pp. 101–150, 2002.

[4] G. Norman, D. Parker, and J. Sproston, "Model checking for probabilistic timed automata," *Formal Methods in System Design*, 2012.

[5] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic model checking," in *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, ser. LNCS (Tutorial Volume), M. Bernardo and J. Hillston, Eds., vol. 4486. Springer, 2007, pp. 220–270.

[6] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[7] V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker, "Automated verification techniques for probabilistic systems," in *Formal Methods for Eternal Networked Software Systems (SFM'11)*, ser. LNCS, M. Bernardo and V. Issarny, Eds., vol. 6659. Springer, 2011, pp. 53–113.

[8] R. Alur and D. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

[9] E. Clarke and A. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic," in *Proc. Workshop on Logic of Programs*, ser. LNCS, vol. 131. Springer, 1981.

[10] A. Pnueli, "The temporal logic of programs," in *Proc. 18th Annual Symposium on Foundations of Computer Science (FOCS'77)*. IEEE Computer Society Press, 1977, pp. 46–57.

[11] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal Aspects of Computing*, vol. 6, no. 5, pp. 512–535, 1994.

[12] A. Bianco and L. de Alfaro, "Model checking of probabilistic and nondeterministic systems," in *Proc. 15th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'95)*, ser. LNCS, P. Thiagarajan, Ed., vol. 1026. Springer, 1995, pp. 499–513.

[13] T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, "Symbolic model checking for real-time systems," *Information and Computation*, vol. 111, no. 2, pp. 193–244, 1994.

[14] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 585–591.

[15] M. Kwiatkowska, G. Norman, and J. Sproston, "Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol," *Formal Aspects of Computing*, vol. 14, no. 3, pp. 295–318, 2003.

[16] M. Stoelinga, "Alea jacta est: Verification of probabilistic, real-time and parametric systems," Ph.D. dissertation, University of Nijmegen, 2002.

[17] M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston, "Performance analysis of probabilistic timed automata using digital clocks," *Formal Methods in System Design*, vol. 29, pp. 33–78, 2006.

[18] L. de Alfaro, "Formal verification of probabilistic systems," Ph.D. dissertation, Stanford University, 1997.

[19] M. Kwiatkowska, G. Norman, J. Sproston, and F. Wang, "Symbolic model checking for probabilistic timed automata," *Information and Computation*, vol. 205, no. 7, pp. 1027–1077, 2007.

[20] J. Berendsen, D. Jansen, and F. Vaandrager, "Fortuna: Model checking priced probabilistic timed automata," in *Proc. 7th International Conference on Quantitative Evaluation of SysTems (QEST'10)*, 2010, pp. 273–281.

[21] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic games for verification of probabilistic timed automata," in *Proc. 7th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'09)*, ser. LNCS, J. Ouaknine and F. Vaandrager, Eds., vol. 5813. Springer, 2009, pp. 212–227.

[22] M. Kattenbelt, M. Kwiatkowska, G. Norman, and D. Parker, "A game-based abstraction-refinement framework for Markov decision processes," *Formal Methods in System Design*, vol. 36, no. 3, pp. 246–280, 2010.

[23] A. Hartmanns and H. Hermanns, "A modest approach to checking probabilistic timed automata," in *Proc. 6th International Conference on Quantitative Evaluation of Systems (QEST'09)*, 2009, to appear.

[24] H. Bohnenkamp, P. D'Argenio, H. Hermanns, and J.-P. Katoen, "Modest: A compositional modeling formalism for hard and softly timed systems," *IEEE Trans. Software Engineering*, vol. 32, no. 10, pp. 812–830, 2006.

[25] K. Larsen, P. Pettersson, and W. Yi, "UPPAAL in a nutshell," *International Journal on Software Tools for Technology Transfer*, vol. 1, no. 1-2, pp. 134–152, 1997.