Model checking for probability and time: from theory to practice*

Marta Kwiatkowska
School of Computer Science, University of Birmingham,
Edgbaston, Birmingham B15 2TT, United Kingdom

Abstract

Probability features increasingly often in software and hardware systems: it is used in distributed co-ordination and routing problems, to model fault-tolerance and performance, and to provide adaptive resource management strategies. Probabilistic model checking *is an automatic procedure for establishing if a desired property holds in a probabilistic model, aimed at verifying probabilistic specifications such as "leader election is eventually resolved with probability 1", "the chance of shutdown occurring is at most 0.01%", and "the probability that a message will be delivered within 30ms is at least 0.75". A probabilistic model checker calculates the* probability *of a given temporal logic property being satisfied, as opposed to validity. In contrast to conventional model checkers, which rely on reachability analysis of the underlying transition system graph, probabilistic model checking additionally involves numerical solutions of linear equations and linear programming problems. This paper reports our experience with implementing PRISM (*`www.cs.bham.ac.uk/~dxp/prism/`*), a Probabilistic Symbolic Model Checker, demonstrates its usefulness in analysing real-world probabilistic protocols, and outlines future challenges for this research direction.*

1. Introduction

Probabilistic modelling is widely used in the design and analysis of computer systems, and has been rapidly gaining in importance in recent years. In distributed co-ordination algorithms, electronic coin tossing and randomness are used as symmetry breakers in order to derive efficient algorithms, see e.g. randomised mutual exclusion [55, 54], randomised consensus [4, 16], root contention in IEEE 1394 FireWire, random back-off schemes in IEEE 802.11 and BlueTooth, dynamic routing in telephone networks [34] and self-stabilisation [30]. Probability is also used to quantify unreliable or unpredictable behaviour, for example in fault-tolerant systems, communication protocols and computer networks, where properties such as component failure and packet loss can be described probabilistically. Traditionally, it has also been used as a tool to analyse system performance, where typically queuing theory is applied to obtain estimates of measures such as throughput and mean waiting time.

Probabilistic model checking is an automatic procedure for establishing if a desired property holds in a probabilistic system model. Conventional model checkers input a description of a model, representing 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 (typically variants of *Markov chains*), in the sense that they encode the *probability* of making a transition between states instead of simply the existence of such a transition. A probability space induced on the system behaviours enables the calculation of likelihood of the occurrence of certain events during the execution of the system. This in turn allows one to make *quantitative* statements about the system, in addition to the qualitative statements made by conventional model checking. Probabilities are captured via *probabilistic operators* that extend conventional (timed or untimed) temporal logics, affording the expression of such *probabilistic specifications* as:

- for a randomised leader election algorithm: "leader election is eventually resolved *with probability* 1"
- for a system which can suffer failures: "the chance of shutdown occurring is *at most* 0.01%"
- for a multi-media protocol: "the chance of a video frame being delivered *within* 5ms is *at least* 89%"
- for a wireless communication medium: "the *expected time* for delivering a data packet in the case of a collision is 16ms"
- for a battery-powered device: "in the long-run, the *average* power consumption is 2.1W given the average

*Supported in part by the EPSRC grants GR/N22960 and GR/S11107.

number of requests awaiting service is 0.5".

Probabilistic modelling has its origins in the 1910s, when Erlang developed stochastic capacity planning techniques for telephone exchanges. The now established field of performance evaluation aids the design and engineering process through building a probabilistic model of the system, typically a continuous time Markov chain (CTMC), on which *analytical*, *simulation-based* or *numerical* calculations are performed to obtain the desired quantitative measures [29]. The modelling of uncertainty due to environmental factors has given rise to a different probabilistic model (Markov decision processes), introduced in operations research in the 1950s as a representation for planning and control problems solvable via Bellmann equations and now central to automated planning in AI [13]. Although both these fields offer a range of algorithmic techniques and tools, albeit for different Markovian models, probabilistic model checking combines probabilistic analysis and conventional model checking in a single tool. It provides an alternative to simulation and analytical approaches, with the former often being time consuming and the latter not representing the system at the desired level of detail, while at the same time offering the full benefit of temporal logic model checking.

Since it was first proposed in 1981, model checking has rapidly become established as a mainstream method to analyse and debug protocols and designs, leading to the emergence of academic and commercial model checking tools. The first extension of model checking algorithms to probabilistic systems was proposed in the 1980s [27, 59], originally focussing on *qualitative* probabilistic temporal properties (i.e. those satisfied with probability 1 or 0) but later also introducing *quantitative* properties [18]. However, work on implementation and tools did not begin until recently [25, 28], when the field of model checking matured. Probabilistic model checking draws on conventional model checking, since it relies on reachability analysis of the underlying transition system, but must also entail the calculation of the actual likelihoods through appropriate numerical methods such as those employed in performance analysis tools.

In this paper we give an overview of formalisms and techniques employed in probabilistic model checking as implemented in the Probabilistic Symbolic Model Checker (PRISM) [41, 1]. We begin by introducing four probabilistic models, discrete time Markov chains, Markov decision processes, continuous time Markov chains and probabilistic timed automata. We give the probabilistic specification languages (the logics PCTL, CSL and PTCTL) and outline the corresponding model checking methods that have been implemented in the PRISM tool. Finally, we discuss four real-world examples analysed with PRISM: the Crowds anonymity protocol [56], randomised consensus [43], dynamic power management [51] and IEEE 1394 FireWire root contention [47, 21]. We finish with a statement of challenges and open problems that remain in the area.

2 Notation and Preliminaries

Let Ω be a *sample set*, the set of possible outcomes of an experiment. A subset of Ω is called an *event*. For example, the experiment of tossing a coin infinitely often has as sample set the set of infinite *sequences* of Heads and Tails and as events "the first is Heads", "eventually Tails".

The following serves as useful intuition of the probability space constructions that follow. (Ω, F) is said to be a *sample space* if F is a σ-field of subsets, often built from *basic cylinders/cones* (e.g. all possible extensions of a finite sequence of coin flips) by closing w.r.t. countable unions and complement. $(\Omega, F, \Pr)$ is a *probability space* if $\Pr$ is a probability measure, i.e. $0 \leqslant \Pr[A] \leqslant 1$ for all $A \in F$; $\Pr[\varnothing] = 0$, $\Pr[\Omega] = 1$, and $\Pr[\bigcup_{k=1}^{\infty} A_k] = \Sigma_{k=1}^{\infty} \Pr[A_k]$, A_k disjoint. Thus, the event "eventually Tails" is a disjoint union of cones that have the first occurrence of Tails in the n-th position, and has probability $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ... = 1$.

For a finite set S, a *probability distribution* on S is a function $\mu : S \to [0, 1]$ such that $\sum_{t \in S} \mu(t) = 1$.

3 Probabilistic Model Checking

We consider four types of probabilistic models, all variants of Markov chains with *discrete* states; for an approach which deals with *continuous* state spaces see [23]. These are: discrete time Markov chains (which feature probabilistic choice only), Markov decision processes (which feature both non-deterministic as well as probabilistic choice), continuous time Markov chains (which model continuous time, but no nondeterminism) and probabilistic timed automata (which admit continuous time, probabilistic choice and nondeterminism).

3.1 Discrete time Markov chains

A (labelled) *discrete time Markov chain* (DTMC) $\mathcal{D}$ is a tuple $(S, \overline{s}, \mathbf{P}, L)$ where

- S is a finite set of *states*
- $\overline{s}$ is an *initial state*
- $\mathbf{P} : S \times S \to [0, 1]$ is a *probability matrix* such that $\sum_{s' \in S} \mathbf{P}(s, s') = 1$ for all $s \in S$, and
- $L : S \to 2^{AP}$ is a *labelling* with atomic propositions that are true in s.

The matrix component $\mathbf{P}(s, s')$ gives the probability of making a transition from s to s'. A path through a DTMC is a (finite or infinite) sequence of states $\omega = s_0\, s_1\, s_2 \ldots$ with $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $i \geqslant 0$. The probability matrix $\mathbf{P}$ induces a probability space on the set of infinite paths $Path_s$ which start in the state s using the cylinder construction [35] as follows. An observation of a finite path determines a basic event (cylinder). Let $s = s_0$. For $\omega = s_0 s_1 \ldots s_n$, we define the probability measure $\Pr_s^{\text{fin}}$ for the ω-cylinder by putting $\Pr_s^{\text{fin}} = 1$ if ω consists of a single state, and $\Pr_s^{\text{fin}} = \mathbf{P}(s_0, s_1) \cdot \mathbf{P}(s_1, s_2) \cdot \ldots \cdot \mathbf{P}(s_{n-1}, s_n)$ otherwise. This extends to a unique measure $\Pr_s$.

A DTMC admits *probabilistic choice* only, which substantially differs from nondeterminism: the frequency of a probabilistic edge being taken is determined by the distribution, whereas nondeterministic choice is made by the environment which has complete freedom to select any of the alternatives. Note also that there is no notion of real time, though reasoning about discrete time is possible through state variables keeping track of time and 'counting' transition steps.

PCTL model checking over DTMCs. The logic PCTL (Probabilistic CTL) [26] replaces the existential and universal quantification of CTL with the *probabilistic operator* $\mathcal{P}_{\sim p}(\cdot)$ where $p \in [0, 1]$ is a *probability bound* or *threshold*, and $\sim \in \{\leqslant, <, \geqslant, >\}$. The syntax of state formulas ϕ of PCTL is:

$$\phi \quad ::= \quad true \mid a \mid \phi \wedge \phi \mid \neg\phi \mid \mathcal{P}_{\sim p}(\alpha)$$

where α is a path formula (either next state $X\phi$ or until $\phi_1\ U\ \phi_2$). The meaning of the probabilistic operator is:

$$s \models \mathcal{P}_{\sim p}(\alpha) \quad \text{iff} \quad \Pr_s\{\omega \in Path_s \mid \omega \models \alpha\} \sim p$$

with the remaining operators standard. The intuition is that the probability of α-paths is calculated and compared to the probability bound, yielding true or false respectively. Note that while $\mathcal{P}_{>0}(\phi_1\ U\ \phi_2)$ equates to existential quantification, $\mathcal{P}_{\geqslant 1}(\phi_1\ U\ \phi_2)$ is a weaker analogue of universal quantification (just consider the event of eventually tossing Tails which has probability 1).

The PCTL model checking algorithm proceeds as for CTL by induction on ϕ, successively returning the set $Sat(\phi)$ of states satisfying ϕ. The DTMC model is viewed as the matrix $\mathbf{P}$ and $Sat(\phi)$ as a *column vector* $\underline{b}^{\phi} : S \longrightarrow \{0, 1\}$ given by $\underline{b}^{\phi}_s = 1$ if $s \models \phi$ and 0 otherwise. Then $Sat(\mathcal{P}_{\sim p}(X\phi)) = \{s \in S \mid x_s \sim p\}$ where $\underline{x} = \mathbf{P} \cdot \underline{b}^{\phi}$. The corresponding vector $\underline{x}$ for $\phi_1 U \phi_2$ which is compared with the threshold $\sim\ p$ is obtained as a solution of the *linear equation system* in variables $\{x_s \mid s \in S\}$:

$$x_s = \begin{cases} 0 & \text{if } s \in S^{no} \\ 1 & \text{if } s \in S^{yes} \\ \sum_{s' \in S} \mathbf{P}(s, s') \cdot x_{s'} & \text{if } s \in S^{?} \end{cases}$$

where S^{no} (S^{yes}) denote the sets of *all* states that satisfy $\phi_1 U \phi_2$ with probability exactly 0 (1), which can be computed by ordinary fixpoint computation [19], and $S^{?} = S \setminus (S^{no} \cup S^{yes})$. The solution of the linear equation systems can be obtained by any direct method (e.g. Gaussian elimination) or iterative method (e.g. Jacobi, Gauss-Seidel).

The complexity of PCTL model checking for DTMCs is linear in the size of the formula and polynomial in the state space.

3.2 Markov Decision Processes

Markov decision processes [13] (also known as concurrent Markov chains [59, 27]) are a generalisation of discrete time Markov chains that are capable of modelling *asynchronous parallel* composition of probabilistic systems. In a Markov decision process both states and time are discrete.

A (labelled) *Markov decision process* $\mathcal{M}$ is a tuple $(S, \overline{s}, Steps, L)$ where:

- S is a finite set of states
- $\overline{s}$ is an *initial state*
- $Steps$ is a function which assigns to each state $s \in S$ a finite, non-empty set $Steps(s)$ of *probability distributions* on S, and
- $L : S \to 2^{AP}$ is a *labelling* with atomic propositions that are true in s.

The execution of an MDP proceeds through alternating of non-deterministic and probabilistic choices: when in a particular state, the system *nondeterministically* chooses one of possibly several *probability distributions* over the target states. A *path* ω of $\mathcal{M}$ is a (finite or infinite) sequence of states:

$$s_0 \xrightarrow{\mu_0} s_1 \xrightarrow{\mu_1} s_2 \xrightarrow{\mu_2} \cdots$$

where $s_i \in S$, $\mu_i \in Steps(s_i)$ and $\mu_i(s_{i+1}) > 0$. The set of infinite paths starting in a state s is endowed with a probability space using an analogue of the cylinder construction. Observe that the probability space is undefined if nondeterminism is present. We define an *adversary* (or scheduler) of an MDP $\mathcal{M}$ as a function A mapping every finite path ω of $\mathcal{M}$ to a distribution $A(\omega)$ on S such that $A(\omega)$ is enabled in the last state of ω. The behaviour of the MDP $\mathcal{M}$ under a given adversary can be described as a (usually infinite state) DTMC $\mathbf{P}^A$ whose states are the finite paths of $\mathcal{M}$ and the probability is given by the distributions selected by A: for two finite paths ω and ω', $\mathbf{P}^A(\omega, \omega') = A(\omega)(s)$ if ω' is of the form $\omega \xrightarrow{A(\omega)} s$ and $\mathbf{P}^A(\omega, \omega') = 0$ otherwise. Hence, we can define a probability measure $\Pr_s^A$ over the set of paths $Path_s^A$ of the adversary A.

Probabilistic statements about MDPs typically involve quantification over adversaries, i.e. we aim to compute the

maximum or *minimum* probability that some specified event is observed *for all* possible adversaries (or for a given class, for example the *fair* adversaries). Fairness is necessary in the context of asynchronous parallel to ensure progress for each concurrent component whenever possible, see e.g. [59, 12].

PCTL model checking over MDPs. The logic PCTL (Probabilistic CTL) [14] is defined for MDPs as for DTMCs, the difference being that the semantics is parameterised by a class Adv of adversaries and the probabilistic operator contains explicit universal quantification:

$$s \models_{Adv} \mathcal{P}_{\sim p}(\alpha) \text{ iff } \Pr^A\{\omega \in Path_s^A \mid \omega \models_{Adv} \alpha\} \sim p \text{ for all adversaries } A \in Adv$$

The algorithm for PCTL model checking proceeds by induction on the structure of ϕ as for DTMCs except for probability calculation, which for $\sim \in \{\geqslant, >\}$ reduces to the calculation of the *minimum probability* p_s^{min} (the case of $\sim \in \{\leqslant, <\}$ is dual, via the *maximum* probability):

$$Sat(\mathcal{P}_{\sim p}(\alpha)) \quad = \quad \{s \in S \mid p_s^{min}(\alpha) \sim p\}$$

If $\alpha = X\phi$ this becomes:

$$p_s^{min}(X\phi) \quad = \quad \min_{\mu \in Steps(s)} \big[\textstyle\sum_{s' \in Sat(\phi)} \mu(s')\big]$$

and for $\alpha = \phi_1 \, U \phi_2$ we have:

$$p_s^{min} = \begin{cases} 0 & \text{if } s \in S^{no} \\ 1 & \text{if } s \in S^{yes} \\ \min_{\mu \in Steps(s)} \big[\sum_{s' \in S} \mu(s') \cdot p_{s'}^{min}\big] & \text{if } s \in S^{?} \end{cases}$$

with S^{yes}, S^{no} and $S^{?}$ precomputed in a similar fashion to the DTMC case via conventional fixpoint. The minimum (maximum) probabilities can be approximated iteratively, or alternatively arrived at by solving a linear optimisation problem.

The computation of p_s^{max} remains unchanged for PCTL model checking with fairness [12], and for p_s^{min} reduces to the calculation of p_s^{max} by translating to the *dual problem* [8].

The complexity of PCTL model checking for MDPs is linear in the size of the formula and polynomial in the state space.

PCTL admits only Boolean operators for state formulas; a more expressive logic PCTL* [6, 14] can be formulated, but model checking necessitates an expensive transformation of the underlying model by encoding history information.

3.3 Continuous Time Markov Chains

Discrete time Markov chains and Markov decision processes can model *discrete time* only. Continuous time Markov chains have (finitely many) states that are discrete, a time parameter that ranges over $\mathbb{R}_{\geqslant 0}$, but do not allow nondeterminism. Every transition is subject to an exponentially distributed random delay, and a *race condition* is used to deal with simultaneously enabled transitions.

A (labelled) *continuous time Markov chain* (CTMC) $\mathcal{C}$ is a tuple $(S, \overline{s}, \mathbf{R}, L)$ where

- S is a finite set of *states*
- $\overline{s}$ is an *initial state*
- $\mathbf{R} : S \times S \to \mathbb{R}_{\geqslant 0}$ is the *rate matrix* (note that self-loops are allowed), and
- $L : S \to 2^{AP}$ is a *labelling* with atomic propositions that are true in s.

$E(s) = \sum_{s' \in S} \mathbf{R}(s, s')$ denotes that the probability of taking a transition from s within t time units equals $1 - e^{-E(s) \cdot t}$. In cases where $\mathbf{R}(s, s') > 0$ for more than one state s', a *race* between the outgoing transitions from s exists. That is, the probability $\mathbf{P}(s, s')$ of moving from s to s' in a single step equals the probability that the delay of going from s to s' "finishes before" the delays of any other outgoing transition from s. This is captured by the *embedded* DTMC, given by $emb(\mathcal{C}) = (S, \overline{s}, \mathbf{P}, L)$, where $\mathbf{P}(s, s') = \mathbf{R}(s, s')/E(s)$ if $E(s) > 0$, and $\mathbf{P}(s, s) = 1$ and $\mathbf{P}(s, s') = 0$ for $s \neq s'$ if $E(s) = 0$.

A path in a CTMC is a non-empty sequence $s_0 t_0 s_1 t_1 s_2 \ldots$ where $\mathbf{R}(s_i, s_{i+1}) > 0$ and $t_i \in \mathbb{R}_{>0}$ for all $i \geqslant 0$. The value t_i represents the amount of time spent in the state s_i. Denoting by $Path_s$ the set of all infinite paths starting in state s, we now give the corresponding probability measure $\Pr_s$ [11]. If the states $s_0, \ldots, s_n \in S$ satisfy $\mathbf{R}(s_i, s_{i+1}) > 0$ for all $0 \leqslant i < n$ and $I_0, \ldots, I_{n-1}$ are non-empty intervals in $\mathbb{R}_{\geqslant 0}$, then the *cylinder set* $C(s_0, I_0, \ldots, I_{n-1}, s_n)$ is defined to be the set containing all paths $s_0' t_0 s_1' t_1 s_2' \ldots$ where $s_i = s_i'$ for $i \leqslant n$ and $t_i \in I_i$ for $i < n$. The (unique) probability measure $\Pr_s$ is obtained similarly to the DTMC case by completing the cylinders to the least σ-algebra.

Traditionally, the analysis of (ergodic) CTMCs [58] is based on *transient* (the state of the model at a particular time instant) and *steady-state* (the state of the CTMC in the long run) behaviour. The transient probability $\pi_{s,t}(s')$ is defined as the probability, having started in state s, of being in state s' at time instant t. The steady-state probability $\pi_s(s')$ is defined as $\lim_{t \to \infty} \pi_{s,t}(s')$.

CSL model checking. The logic CSL (Continuous Stochastic Logic) [5, 11] is based on CTL and PCTL; it contains the probabilistic operator of PCTL evaluated with respect to *path-based* measures, as well as a *steady-state* operator.

The syntax for CSL is as follows:

$$\phi ::= true \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \mathcal{S}_{\sim p}(\phi) \mid \mathcal{P}_{\sim p}(\phi\ U^{\leqslant t}\ \phi)$$

where $a \in AP$, $\sim \in \{<, \leqslant, \geqslant, >\}$, $p \in [0,1]$ and $t \in \mathbb{R}_{\geqslant 0}$. The meaning of the steady-state and the probability operator is:

$$\begin{array}{rcl} s \models \mathcal{S}_{\sim p}(\phi) & \text{iff} & \sum_{s' \models \phi} \pi_s(s') \sim p \\ s \models \mathcal{P}_{\sim p}(\phi_1\ U^{\leqslant t}\ \phi_2) & \text{iff} & p_s(\phi_1\ U^{\leqslant t}\ \phi_2) \sim p \end{array}$$

where

$$p_s(\phi_1\ U^{\leqslant t}\ \phi_2) = \Pr_s\{\omega \in Path_s \mid \omega \models \phi_1\ U^{\leqslant t}\ \phi_2\}.$$

To model check the $\mathcal{S}_{\sim p}(\Phi)$ operator, we must compute the steady-state probabilities $\pi_s(s')$ for all states s and s' (this is independent of the initial state s for ergodic CTMCs, and otherwise the computation of bottom strongly connected components must be involved). As is well known [58], steady state probabilities can be computed by solving the linear equation system:

$$\underline{\pi} \cdot \mathbf{Q} = \underline{0} \quad \text{and} \quad \sum_{s' \in S} \pi(s') = 1$$

where $\mathbf{Q}$ is the *generator matrix* of the CTMC given by: $\mathbf{Q}(s,s') = \mathbf{R}(s,s')$ if $s \neq s'$ and $\mathbf{Q}(s,s') = -\sum_{s'' \neq s} \mathbf{R}(s,s'')$ if $s = s'$. This equation can be solved in the transposed form $\mathbf{Q}^T \cdot \underline{\pi}^T = \underline{0}$ using standard methods, direct or iterative.

The case of the bounded until operator $\mathcal{P}_{\sim p}(\phi_1 \mathcal{U}^{\leqslant t}\ \phi_2)$, which requires the probability of reaching a state satisfying ϕ_2 while passing only through states satisfying ϕ_1 within t time units, is more involved. It proceeds by a technique called *uniformisation*, also known as Jensen's method, which transforms the original CTMC to a *uniformised* DTMC with matrix $\mathbf{P}$ (we omit the details, see e.g. [10]), yielding an infinite summation to calculate the vector of transient probabilities $\underline{\pi_{s,t}}$:

$$\underline{\pi_{s,t}} \quad = \quad \underline{\pi_{s,0}} \cdot \sum_{i=0}^{\infty} \gamma_{i,q\cdot t} \cdot \mathbf{P}^i$$

where $\gamma_{i,q\cdot t}$ is the ith Poisson probability with parameter $q \cdot t$, i.e. $\gamma_{i,q\cdot t} = e^{-q\cdot t} \cdot (q \cdot t)^i / i!$. An adaptation of this technique that is more efficient for all states [33] gives the truncated sum:

$$\underline{p(\phi_1 U^{\leqslant t} \phi_2)} \quad = \quad \sum_{i=L_\varepsilon}^{R_\varepsilon} \left(\gamma_{i,q\cdot t} \cdot \tilde{\mathbf{P}}^i \cdot \underline{\phi_2} \right)$$

where $\underline{\phi_2}(s)$ equal to 1 if $s \models \phi_2$ and 0 otherwise, L_ε and R_ε are calculated using the techniques of Fox and Glynn and $\tilde{\mathbf{P}}$ is a modified uniformised DTMC in which states satisfying either $\neg\phi_1 \wedge \neg\phi_2$ or ϕ_2 are made absorbing. The resulting vector of probabilities $\underline{p(\phi_1 U^{\leqslant t} \phi_2)}$ can be approximated iteratively.

The complexity of CSL model checking for CTMCs is linear in the size of the formula, polynomial in the state space, linear in the maximum time bound in the formula, and linear in the largest entity in the generator matrix.

3.4 Probabilistic Timed Automata

Continuous time Markov chains do not allow nondeterminism which often features in real-world distributed protocols, for example random back-off schemes. To derive an appropriate model we extend the formalism of *timed automata* [3] with probabilistic choice. A timed automaton is an automaton extended with *clocks*, positive real valued variables which increase uniformly with time, and whose nodes and edges are labelled with *clock constraints*, e.g. $(x \geqslant 5) \wedge (y < 1)$, respectively called *invariants* and *guards*. The invariants dictate when the automaton may remain in a node, letting time pass, and guards when the corresponding edge can be taken. Transitions are instantaneous, and can involve *clock resets* of the form $x := 0$.

Probabilistic Timed Automata. Let $\mathcal{X}$ be a finite set of $\mathbb{R}_{\geqslant 0}$-valued variables called *clocks*, and let $v \in \mathbb{R}_{\geqslant 0}^{|\mathcal{X}|}$ denote a clock valuation. We adopt the following notation for valuations: $\mathbf{0} \in \mathbb{R}_{\geqslant 0}^{|\mathcal{X}|}$ assigns 0 to all clocks in $\mathcal{X}$, $v + t$ denotes v *incremented* by duration t, and $v[X := 0]$ denotes the clock valuation obtained from v by *resetting* all of the clocks in the set X to 0 and leaving the values of all other clocks unchanged. Probabilistic timed automata in which clocks can be reset according to *continuous probability distributions* are described in [44].

Let Z be the set of *zones* over $\mathcal{X}$, that is, conjunctions of atomic constraints of the form $x \sim c$ and $x - y \sim c$, with $x, y \in \mathcal{X}$, $\sim \in \{<, \leqslant, \geqslant, >\}$, and $c \in \mathbb{N}$. A clock valuation v *satisfies* the zone ζ, written $v \models \zeta$, if and only if ζ resolves to true after substituting each clock $x \in \mathcal{X}$ with the corresponding clock value $v(x)$.

A *probabilistic timed automaton* (PTA) $\mathcal{T}$ is a tuple $(L, \bar{l}, \mathcal{X}, \Sigma, I, \mathrm{P})$ where:

- L is a finite set of *locations*;
- $\bar{l} \in L$ is the *initial location*;
- Σ is a finite set of *labels*;
- the function $I : L \to \mathrm{Z}$ is the *invariant condition*;
- the finite set $\mathrm{P} \subseteq L \times \mathrm{Z} \times \Sigma \times Dist(2^{\mathcal{X}} \times L)$ is the *probabilistic edge relation*.

An *edge* takes the form of a tuple (l, g, X, l'), where l is its source location, g is its *enabling condition*, X is the set of resetting clocks and l' is the destination location, such that $(l, g, \sigma, p) \in \mathrm{P}$ and $p(X, l') > 0$.

Semantics of PTAs. The semantics of a probabilistic timed automaton is defined in terms of an infinite-state Markov decision process whose states are pairs (l, v), where $l \in L$ and $v \in \mathbb{R}^{|\mathcal{X}|}_{\geqslant 0}$ such that $v \models I(l)$, and whose initial state is $(\bar{l}, \mathbf{0})$. Transitions between states are probabilistic. First, in the current state (l, v) there is a nondeterministic choice of either letting *time pass* as long as the invariant $I(l)$ is satisfied, or making a *discrete probabilistic* transition according to any probabilistic edge in P from l whose enabling condition g is satisfied. Next, if the probabilistic edge (l, g, σ, p) is chosen, then the probability of moving to the location l' and resetting to 0 all clocks in X is given by $p(X, l')$.

Given a probabilistic timed automaton $\mathcal{T} = (L, \bar{l}, \mathcal{X}, \Sigma, I, \mathrm{P})$, the *semantics* of $\mathcal{T}$ is the Act-labelled Markov decision process $[\![\mathcal{T}]\!] = (S, \overline{s}, Act, Steps)$:

States. Let $S \subseteq L \times \mathbb{R}^{|\mathcal{X}|}_{\geqslant 0}$ such that $(l, v) \in S$ if and only if $v \models I(l)$ and $\overline{s} = (\bar{l}, \mathbf{0})$.

Actions. Let $Act = \mathbb{R}_{\geqslant 0} \cup \Sigma$.

Transitions. Let $Steps$ be the least set of probabilistic transitions containing, for each $(l, v) \in S$:

- for each $t \in \mathbb{R}_{\geqslant 0}$, let $(t, \mu) \in Steps(l, v)$ if and only if $\mu(l, v + t) = 1$ and $v + t' \models I(l)$ for all $0 \leqslant t' \leqslant t$.
- for each $(l, g, \sigma, p) \in \mathrm{P}$, let $(\sigma, \mu) \in Steps(l, v)$ if and only if $v \models g$ and for each $(l', v') \in S$:

$$\mu(l', v') = \sum_{X \subseteq \mathcal{X} \,\&\, v' = v[X:=0]} p(X, l') \ .$$

Paths and Probability Space. A *path* of a Markov decision process $[\![\mathcal{T}]\!]$ determined by a PTA is a non-empty finite or infinite sequence of transitions

$$\omega = \overline{s} \xrightarrow{a_0, \mu_0} s_1 \xrightarrow{a_1, \mu_1} s_2 \xrightarrow{a_2, \mu_2} \cdots \ .$$

An *adversary* is a function A mapping every finite path ω to a pair $(a, \mu) \in Act \times Dist(S)$ such that $(last(\omega), a, \mu) \in Steps$. A probability measure $\Pr^A$ over paths $Path^A$ of a given adversary can again be defined following [35] as for Markov decision processes.

Probabilistic real-time specifications. The following is a desirable property for a probabilistic timed automaton: "*with probability 0.98 or greater*, the message is correctly delivered *within* 5 ms, under *any scheduling of nondeterminism*". This can be expressed in temporal logic by:

$$z.[\mathcal{P}_{\geqslant 0.98}(\Diamond(\mathsf{deliver} \wedge z < 5))]$$

where $\Diamond$ is a *temporal modality* (CTL), $z.[\cdot \wedge z < 5]$ is the *reset quantifier* (TCTL) and $\mathcal{P}_{\geqslant 0.98}(\cdot)$ is the *probabilistic operator* (PCTL). The obtained logic is known as PTCTL (Probabilistic Timed CTL) [45] whose syntax:

$$\phi ::= true \mid a \mid \zeta \mid \phi \wedge \phi \mid \neg\phi \mid z.[\phi] \mid \mathcal{P}_{\sim p}(\phi_1 \,\mathcal{U}\, \phi_2)$$

is similar to PCTL except that $z \in \mathcal{Z}$, and ζ are zones over the clocks in $\mathcal{X} \cup \mathcal{Z}$, where $\mathcal{X}$ is the set of clocks of the automaton and $\mathcal{Z}$ is the set of formula clocks such that $\mathcal{X} \cap \mathcal{Z} = \varnothing$. The meaning of CTL and TCTL [3] operators is standard, and the meaning of the probabilistic operators is as for PCTL, though considered with the respect to the induced probability measure on the (infinitakes the form of a finite-state Markov decision processte state) Markov decision process. Care must be taken with *time divergence*; the interested reader is referred to [45] for more details which have been omitted from this presentation.

Model checking for PTAs Since clocks are real-valued, the state space of a probabilistic timed automaton is infinite. However, for conventional timed automata [3] the space of clock valuations can be partitioned into a *finite* set of *symbolic states* called *clock regions*, each containing a finite or infinite number of valuations which satisfy the same TCTL formulas [2]. Combined with the transitions of a timed automaton, a so called *region graph* is obtained which takes the form of a finite-state Markov decision process and therefore admits model checking using well-established methods. In [45] it is shown that the region graph can be adapted for model checking probabilistic timed automata against PTCTL formulas. In this case, the region graph takes the form of a Markov decision process over the regions. Model checking for PTCTL can be reduced to a translation of property to PCTL and invoking the PCTL model checking algorithm on the induced MDP.

The region-based model construction is unfortunately expensive, with complexity exponential in the number of clocks and the magnitude of constants featured in the model and formula. Alternative, forwards/backwards reachability techniques have been formulated for conventional timed automata, using coarser partitioning into zones. These have been adapted to probabilistic timed automata; *forwards probabilistic reachability* [45] constructs an MDP over zones by iterating the post operation from the initial state, and yields the *maximum* reachability probability which could be *higher* than that in the original PTA. Exact probability can be obtained with the help of a *backwards probabilistic reachability*, a zone-based symbolic algorithm that iterates the pre operator from the from target set [46]. For a restricted class of probabilistic timed automata, reduction to *integer* semantics enables *exact* maximum and minimum probabilistic reachability calculations [47].

The *continuous* probabilistic timed automata pose a rather more difficult problem. As observed by Alur, the subdivision into regions does not suffice to formulate a model checking approach. In [44] an approach based on *approximate* model checking of such automata and finer subdivision than regions is proposed, but its complexity is prohibitive.

4 Implementation of Probabilistic Model Checking

Symbolic model checking uses Binary-Decision Diagrams (BDDs) to compactly represent the state-transition graph of the model and performs traversal via fixed point calculation [15]. Implemented in the SMV tool [50], symbolic model checking enabled industrial exploitation of verification technology for the first time, particularly in the context of circuit design. Encouraged by the success of symbolic model checking in the combating state-space explosion problem, we embarked on implementation of probabilistic model checking techniques using Multi-Terminal Binary Decision Diagrams (MTBDDs) [17, 7] which are capable of representing probability matrices and support matrix multiplication algorithms.

4.1 Probabilistic Symbolic Model Checking

Traditionally, analysis of probabilistic systems has been carried out using numerical methods, mainly with sparse matrices, analytical methods or simulation. Our approach was to tackle the large state spaces *symbolically*, by an MTBDD-based representation. We have seen that probabilistic model checking techniques can be reduced entirely to a combination of *graph-theoretic traversal*, implementable as BDD fixpoint operations, and an *iterative numerical solution* process for solution of linear equations, linear programming problems and uniformisation which involves matrix-by-vector multiplication. Earlier experiments with direct solution methods were unsuccessful due to the loss of regularity while manipulating the matrix [7]. The first symbolic probabilistic model checking algorithm was introduced in [9] for PCTL over DTMCs, and extended in [22] to MDPs, where also the BDD-based precomputation steps for probability 1/0 were introduced. Based on [10], a CSL model checking algorithm was also implemented. Though the first experiments enabled the compact storage of very large state spaces for MDPs [22], it soon became apparent that the efficiency of numerical solution of steady-state probabilities for CTMCs lagged far behind that for sparse matrices [32]. In [42] a hybrid data structure, a combination of a modified MTBDD representation matrix with a conventional solution method was proposed which can perform better than the sparse matrix techniques, both in terms of space and time; this was further improved in [53] and the memory limitation imposed by the need to store a conventional vector has been tackled with out-of-core techniques, in combination with symbolic matrix representation [36, 37].

4.2 The PRISM Tool

PRISM [41, 1] is a probabilistic symbolic model checker implemented using the CUDD package [57] to obtain BDD/MTBDD-based representation of probabilistic models. PRISM directly supports the DTMC, MDP and CTMC models and the specification languages PCTL and CSL; probabilistic timed automata are currently handled through a pre-processing phase that involves the KRONOS real-time model checker and an implementation of the forward probabilistic reachability [21].

PRISM inputs a description of a system model written in a modelling language based on Reactive Modules with support for the main process-algebraic composition operators, and specifications in the PCTL or CSL syntax. It constructs a symbolic representation of the model from this description and computes the set of reachable and deadlock states. For PCTL model checking over DTMC or MDP models, PRISM implements the algorithms of [26, 14, 12] (including fairness) and the subsequent improvements of [8]. For CSL and CTMCs, methods based on [10, 33], recently extended with random time bounds, cost/rewards and expected time [39, 40], are used. Graph traversal is implemented with BDD fixpoints, but numerical computation can be performed using one of three different model checking engines: symbolic MTBDD-based [9, 22] for both the model and vector; sparse matrix model and full vectors; and hybrid [42, 53] which combines symbolic model and full vector.

PRISM is available for download from [1]. We mention also related tools ETMCC [31] for CTMCs and RAPTURE [20] for MDPs.

5 Case studies

PRISM has been used to build and analyse probabilistic models for a large number of case studies in Birmingham and elsewhere, with encouraging results available at the website [1]. We briefly describe here four illustrative examples, one for each type of model introduced in Sec. 2.

5.1 The Crowds protocol

The Crowds protocol of Reiter and Rubin aims to ensure anonymous Web browsing by routing communications *randomly* within a group of similar users in order to hide them. Even if an eavesdropper observes a message being sent by

a particular user, it can never be sure whether the user is the actual sender, or is simply routing another user's message. Crowds guarantees probable innocence for a single path: the likelihood of the sender being the originator is no greater than to not be the originator.

The protocol model (a DTMC) was analysed in [56], see [1] for the details, and the following disturbing conclusions reached. It was noted that the probability of successful observation of the real sender by the adversary observed grows for any given crowd as the number of paths increases. The second impact of increasing crowd size is the *increased confidence* of the adversary: the bigger the crowd, the more confident the corrupt members are that, once they observed some crowd member more than once, the observed member is the real sender.

An analysis of probabilistic contract signing [52] also detected flaws.

5.2 Randomized consensus

It is well known [24] that there are *no* symmetric solutions to certain distributed systems problems in the presence of failures; using *randomisation* ensures that such solutions exist [55]. In many such algorithms, modelling nondeterminism as well as probabilistic choices is essential, and thus the models are typically Markov decision processes (MDPs). The correctness arguments for such randomised distributed algorithms include both non-probabilistic statements, such as "no two processes are simultaneously in the critical section", as well as probabilistic statements such as "termination occurs with probability 1".

We have analysed two randomised protocols, one for distributed consensus using shared read-write registers that tolerates stopping failures [4] and a Byzantine agreement ABBA (Asynchronous Binary Byzantine Agreement) of [16]. The complexity of these algorithms resulted in large state spaces, and so we have used the Cadence SMV tool that supports data reduction techniques as well as proof methods, in addition to PRISM. The results are described in [43, 38] and [1].

5.3 Dynamic power management

Power management is receiving much attention due to an increasing trend in the usage of portable, mobile and hand-held electronic devices. These devices usually run on batteries and any savings in power usage translate to extended battery life. Dynamic Power Management (DPM) refers to system-level strategies that attempt to make power mode changing decisions based on the information about their usage pattern available at runtime. The objective is to minimize power consumption, while minimizing the effect on performance. *Stochastic optimum control* schemes are being applied, alongside the more conventional predictive schemes.

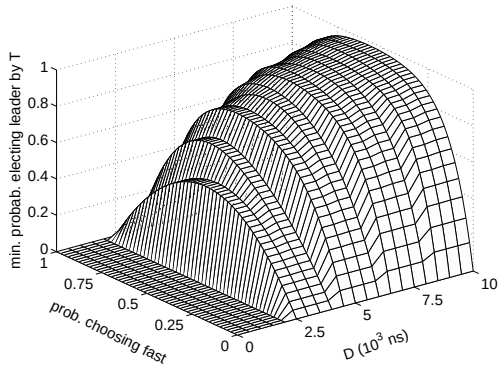

Figure 1. FireWire with a biased coin.

We have analysed two portable disks, Fujitsu and IBM, based on real measurement data [51]. Since quantities such as power consumption feature in the model, we have used continuous and discrete time Markov chains, both with rewards/costs, and optimisation. This has enabled us to analyse trade offs between average power consumption versus the average number of queued requests, confirming that PRISM is a useful tool for analysing resource management schemes.

5.4 IEEE 1394 FireWire Root Contention

This case study concerns the Tree Identify Protocol of the IEEE 1394 High Performance Serial Bus (called "FireWire"), an efficient and widely used multimedia protocol. It has a scalable architecture and is "hot-pluggable", meaning that at any time devices can easily be added or removed from the network.

The tree identify process of IEEE 1394 is a randomised leader election protocol designed for connected acyclic network topologies. It executes after a bus reset in the network, i.e. when a node is added or removed from the network, and returns a leader (root) to act as the manager of the bus for subsequent phases of IEEE 1394. The nodes nodes communicate "be my parent" messages, starting from the leaf nodes. If two nodes contend the leadership (root contention), the contenders exchange additional messages and involve *time delays* and *electronic coin tosses*. The model is thus a probabilistic timed automaton (PTA).

We have analysed the probabilistic aspects of the protocol, such as termination with probability 1, expected time to contention resolution and soft deadlines (the probability of resolution within a given time period) using the KRONOS-PRISM connection [21] and integer semantics [47]. We

have automatically confirmed an unusual property conjectured by Stoelinga that the time to resolution varies if a coin is biased [21], shown in Figure 1.

Probabilistic timed automata arise also in random backoff schemes of physical layer protocols, as in e.g. IEEE 802.11 Wireless LAN MAC protocol analysed in [48].

6 Conclusion and Future Challenges

In this paper we have described our experience with implementing probabilistic model checking techniques within PRISM, a Probabilistic Symbolic Model Checker [41, 1], and described its application to real-world case studies. So far we have concentrated our efforts on efficient implementation of the techniques, but not explored any data reduction or compositionality and proof methods; these are currently being investigated, as are parallel out-of-core solution methods, extension with real-time clocks, sampling-based techniques derived from [60] and Monte Carlo approximation for PCTL [49]. Further applications of PRISM to model and analyse quantum cryptographic protocols, mobile ad hoc networks and nano-circuits are underway.

Acknowledgements: These achievements would not have been possible without the many collaborators who contributed in various ways, including encouragement and finding errors. The following deserve specific mention: Dave Parker, Rashid Mehmood and Stephen Gilmore (PRISM tool development), Gethin Norman (theory and case studies), Jeremy Sproston, Roberto Segala and Conrado Daws (probabilistic timed automata), Christel Baier, Luca de Alfaro and Sylvain Peyronnet (PCTL algorithms), Markus Siegle and Holger Hermanns (MTBDDs), Joost-Pieter Katoen, António Pacheco and Håkan Younes (CSL algorithms), and Vitaly Shmatikov and Sandeep Shukla (contribution to case studies).

References

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

[2] R. Alur, C. Courcoubetis, and D. Dill. Model checking in dense real time. *Information and Computation*, 104(1):2–34, 1993.

[3] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[4] J. Aspnes and M. Herlihy. Fast randomized consensus using shared memory. *J. of Algorithms*, 15(1):441–460, 1990.

[5] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Verifying continuous time Markov chains. In *Proc. CAV'96*, volume 1102 of *LNCS*, pages 269–276. Springer, 1996.

[6] A. Aziz, V. Singhal, F. Balarin, R. Brayton, and A. Sangiovanni-Vincentelli. It usually works: The temporal logic of stochastic systems. In *Proc. CAV'95*, volume 939 of *LNCS*, pages 155–165. Springer, 1995.

[7] I. Bahar, E. Frohm, C. Gaona, G. Hachtel, E.Macii, A. Pardo, and F. Somenzi. Algebraic decision diagrams and their applications. In *Proc. ICCAD'93*, 1993.

[8] C. Baier. On algorithmic verification methods for probabilistic systems. Habilitation thesis, Universität Mannheim, 1998.

[9] C. Baier, E. Clarke, V. Hartonas-Garmhausen, M. Kwiatkowska, and M. Ryan. Symbolic model checking for probabilistic processes. In *Proc. ICALP'97*, volume 1256 of *LNCS*, pages 430–440. Springer, 1997.

[10] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model checking continuous-time Markov chains by transient analysis. In *Proc. CAV'00*, volume 1855 of *LNCS*, pages 358–372. Springer, 2000.

[11] C. Baier, J.-P. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In *Proc. CONCUR'99*, volume 1664 of *LNCS*, 1999.

[12] C. Baier and M. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11(3):125–155, 1998.

[13] D. Bertsekas. *Dynamic Programming and Optimal Control*, Volumes 1 and 2. Athena Scientific, 1995.

[14] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proc. FST&TCS*, volume 1026 of *LNCS*, 1995.

[15] 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*, pages 428–439. IEEE Computer Society Press, 1990.

[16] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in constantipole: practical asynchronous Byzantine agreement using cryptography. In *Proc. PODC'00*, 2000.

[17] E. Clarke, M. Fujita, P. McGeer, K. McMillan, J. Yang, and X. Zhao. Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. In *Proc. International Workshop on Logic Synthesis (IWLS'93)*, 1993.

[18] C. Courcoubetis and M. Yannakakis. Verifying temporal properties of finite state probabilistic programs. In *Proc. FOCS'88*, pages 338–345, 1988.

[19] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.

[20] P. D'Argenio, B. Jeannet, H. Jensen, and K. Larsen. Reachability analysis of probabilistic systems by successive refinements. In *Proc. PAPM/PROBMIV'01*, volume 2165 of *LNCS*, pages 39–56. Springer, 2001.

[21] C. Daws, M. Kwiatkowska, and G. Norman. Automatic verification of the IEEE 1394 root contention protocol with KRONOS and PRISM. In *Proc. FMICS'02*, volume 66.2 of *ENTCS*. Elsevier, 2002.

[22] 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'00*, volume 1785 of *LNCS*, 2000.

[23] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Approximating labeled Markov processes. In *Proc. LICS 2000*, pages 95–106. IEEE Computer Society Press, 2000.

[24] M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed commit with one faulty process. *Journal of the ACM*, 32(5):374–382, 1985.

[25] H. Hansson. *Time and Probability in Formal Design of Distributed Systems*. Elsevier, 1994.

[26] H. Hansson and B. Jonsson. A logic for reasoning about time and probability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[27] S. Hart, M. Sharir, and A. Pnueli. Termination of probabilistic concurrent programs. *ACM Transactions on Programming Languages and Systems*, 5(3):356–380, 1983.

[28] V. Hartonas-Garmhausen, S. Campos, and E. Clarke. ProbVerus: Probabilistic symbolic model checking. In *Proc. ARTS'99*, volume 1601 of *LNCS*, pages 96–110, 1999.

[29] B. Haverkort. *Performance of Computer-Communication Systems: A Model-Based Approach*. Wiley & Sons, 1998.

[30] T. Herman. Probabilistic self-stabilization. *Information Processing Letters*, 35(2):63–67, 1990.

[31] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. A Markov chain model checker. In *Proc. TACAS'00*, volume 1785 of *LNCS*, pages 347–362. Springer, 2000.

[32] H. Hermanns, M. Kwiatkowska, G. Norman, D. Parker, and M. Siegle. On the use of MTBDDs for performability analysis and verification of stochastic systems. *Journal of Logic and Algebraic Programming*, 2002. To appear.

[33] J.-P. Katoen, M. Kwiatkowska, G. Norman, and D. Parker. Faster and symbolic CTMC model checking. In *Proc. PAPM/PROBMIV'01*, volume 2165 of *LNCS*, 2001.

[34] F. Kelly. Modelling communication networks, present and future. *Philosophical Transactions of the Royal Society of London*, 354(1707):437–463, 1996.

[35] J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov Chains*. D. Van Nostrand Company, 1966.

[36] M. Kwiatkowska and R. Mehmood. Out-of-core solution of large linear systems of equations arising from stochastic modelling. In *Proc. PAPM/PROBMIV'02*, volume 2399 of *LNCS*, pages 135–151. Springer, 2002.

[37] M. Kwiatkowska, R. Mehmood, G. Norman, and D. Parker. A symbolic out-of-core solution method for Markov models. In *Proc. PDMC'02*, volume 68.4 of *ENTCS*. Elsevier, 2002.

[38] M. Kwiatkowska and G. Norman. Verifying randomized byzantine agreement. In *Proc. FORTE'02*, volume 2529 of *LNCS*, pages 194–209. Springer, 2002.

[39] M. Kwiatkowska, G. Norman, and A. Pacheco. Model checking CSL until formulae with random time bounds. In *Proc. PAPM/PROBMIV'02*. Springer, 2002.

[40] M. Kwiatkowska, G. Norman, and A. Pacheco. Model checking expected time and expected reward formulae with random time bounds. In *Proc. 2nd Euro-Japanese Workshop on Stochastic Risk Modelling for Finance, Insurance, Production and Reliability*, 2002. To appear.

[41] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In *Proc. TOOLS'02*, volume 2324 of *LNCS*, pages 200–204. Springer, 2002.

[42] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. In *Proc. TACAS'02*, volume 2280 of *LNCS*, 2002.

[43] M. Kwiatkowska, G. Norman, and R. Segala. Automated verification of a randomized distributed consensus protocol using Cadence SMV and PRISM. In *Proc. CAV'01*, volume 2102 of *LNCS*, pages 194–206. Springer, 2001.

[44] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Verifying quantitative properties of continuous probabilistic timed automata. In *Proc. CONCUR'00*, volume 1877 of *LNCS*, pages 123–137. Springer, 2000.

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

[46] M. Kwiatkowska, G. Norman, and J. Sproston. Symbolic computation of maximal probabilistic reachability. In *Proc. CONCUR'01*, volume 2154 of *LNCS*, 2001.

[47] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol. *Special Issue of Formal Aspects of Computing*, 2002. To appear.

[48] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In *Proc. PAPM/PROBMIV'02*, volume 2399 of *LNCS*, pages 169–187. Springer, 2002.

[49] R. Lassaigne and S. Peyronnet. Approximate verification of probabilistic systems. In H. Hermanns and R. Segala, editors, *Proc. PAPM/PROBMIV'02*, volume 2399 of *LNCS*, pages 213–214. Springer, 2002.

[50] K. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[51] G. Norman, D. Parker, M. Kwiatkowska, S. Shukla, and R. Gupta. Formal analysis and validation of continuous time Markov chain based system level power management strategies. In *Proc. HLDVT'02*, pages 45–50, 2002.

[52] G. Norman and V. Shmatikov. Analysis of probabilistic contract signing. In *Proc. BCS-FACS Formal Aspects of Security (FASec'02)*, LNCS. Springer, 2002.

[53] D. Parker. *Implementation of Symbolic Model Checking for Probabilistic Systems*. PhD thesis, University of Birmingham, 2002.

[54] A. Pnueli and L. Zuck. Verification of multiprocess probabilistic protocols. *Distributed Computing*, 1(1):53–72, 1986.

[55] 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):66–75, 1982.

[56] V. Shmatikov. Probabilistic analysis of anonymity. In *Proc. CSFW'02*, pages 119–128, 2002.

[57] F. Somenzi. CUDD: Colorado University decision diagram package. Public software, Colorado Univeristy, Boulder, http://vlsi.colorado.edu/~fabio/, 1997.

[58] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton, 1994.

[59] M. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Proc. FOCS'85*, pages 327–338. IEEE Computer Society Press, 1985.

[60] H. Younes and R. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In E. Brinksma and K. Larsen, editors, *Proc. (CAV'02)*, volume 2404 of *LNCS*, pages 223–235. Springer, 2002.