Symbolic Model Checking for Probabilistic Timed Automata*

Marta Kwiatkowska[1], Gethin Norman[1], Jeremy Sproston[2], and Fuzhi Wang[1]

[1] School of Computer Science, University of Birmingham,
Birmingham B15 2TT, United Kingdom.
{M.Z.Kwiatkowska,G.Norman,F.Wang}@cs.bham.ac.uk
[2] Dipartimento di Informatica, Università di Torino, 10149 Torino, Italy.
sproston@di.unito.it

Abstract. Probabilistic timed automata are an extension of timed automata with discrete probability distributions, and can be used to model timed randomized protocols or fault-tolerant systems. We present symbolic model-checking algorithms for verifying probabilistic timed automata against properties of PTCTL (Probabilistic Timed Computation Tree Logic). The algorithms operate on zones, which are sets of valuations of the probabilistic timed automaton's clocks, and therefore avoid an explicit construction of the state space. Furthermore, the algorithms are restricted to system behaviours which guarantee the divergence of time with probability 1. We report on a prototype implementation of the algorithms using Difference Bound Matrices, and present the results of its application to the CSMA/CD and FireWire root contention protocol case studies.

1 Introduction

Systems exhibiting both *timed* and *probabilistic* characteristics are widespread, in application contexts as diverse as home entertainment, medicine and business. For example, timing constraints are often vital to the correctness of embedded digital technology, whereas probability exhibits itself commonly in the form of statistical estimates regarding the environment in which a system is embedded. Similarly, protocol designers often exploit the combination of time and probability to design correct, efficient protocols, such as the IEEE 1394 FireWire root contention protocol [13]. The diffusion of such systems has led to methods for obtaining formal correctness guarantees, for instance, adaptations of model checking. *Symbolic model checking* refers to model-checking techniques in which implicit representations – such as BDDs – are used to represent both the transition relation of the system model and the state sets that are computed during the execution of the model-checking algorithm.

* Supported in part by EPSRC grants GR/N22960 and GR/S46727, FORWARD and MIUR-FIRB Perf.
Y. Lakhnech and S. Yovine (Eds.), Joint Conference on Formal Modelling and Analysis of Timed Systems (FORMATS) and Formal Techniques in Real-Time and Fault Tolerant System (FTRTFT), LNCS, 2004, to appear.

In this paper, we consider the modelling formalism of *probabilistic timed automata* [18], an extension of timed automata [3] with discrete probability distributions. This formalism has been shown to be suitable for the description of timed, randomized protocols, such as the backoff strategy of the IEEE 802.11 standard [20], and the link-local address selection protocol of the IPv4 standard [17]. As a requirement specification language for probabilistic timed automata we consider PTCTL (Probabilistic Timed Computation Tree Logic). The logic PTCTL combines the probabilistic threshold operator of the probabilistic temporal logic PCTL [11] with the timing constraints of the timed temporal logic TCTL [1, 12], in order to express properties such as 'with probability at least 0.99, the system elects a leader within 1 second'. Model checking of probabilistic timed automata against PTCTL was shown to be decidable in [18] via an adaptation of the classical region-graph construction [3, 1]. Unfortunately, the region-graph construction (and digital clocks approach [17]) can result in huge state spaces if the maximal constant used in the description of the automaton is large. Instead, the practical success of *symbolic*, *zone*-based techniques for non-probabilistic timed automata [4, 8] suggests that a similar symbolic approach may also be employed for the verification of probabilistic timed automata against PTCTL properties. We answer this hypothesis affirmatively in this paper.

The non-trivial cases of our model-checking algorithm concern PTCTL formulae referring to a temporal modality and a probability threshold. In general, PTCTL properties may have thresholds with arbitrary probability values; we refer to such properties as *quantitative*. Properties which express statements such as 'φ is true with probability below 0.95' impose a bound on the *maximal* probability of φ; analogously, properties such as 'φ is true with probability at least 0.01' impose a bound on the *minimal* probability of φ. In previous work, we presented a zone-based algorithm for the verification of properties referring to maximal probabilities [19]. The aim of this previous algorithm was to construct a finite-state, untimed probabilistic system which has sufficient information to compute the maximum probability using well-established model-checking methods for finite-state probabilistic systems [5]. In this paper, we extend that result by presenting algorithms for probabilistic properties referring to minimal probability of satisfaction, hence permitting verification of arbitrary PTCTL properties.

In order to verify properties of real-world behaviour, it is vital that model-checking algorithms for real-time systems incorporate a notion of *time divergence*. The issue of time divergence is of importance to our algorithms for verifying properties referring to minimal probabilities. For example, to compute the minimum probability of reaching a certain set F, for any state not in F, the probabilistic timed automaton could exhibit behaviour in which the amount of time elapsed converges before F is reached, or even in which no time elapses. Clearly, such behaviours where time does not progress beyond a bound are pathological, and should be disregarded during model checking. We present an algorithm for computing minimum reachability probabilities which considers *only time-divergent behaviour*, based on the non-probabilistic precedent of [12]. The

algorithm is based on computing maximum probabilities for the dual formula while restricting attention to time-divergent behaviours. Because it is possible that a probabilistic timed automaton contains states from which it is impossible for time to diverge with probability 1 (constituting a modelling error), based on [12] we present an algorithm to check for and eliminate such states.

Finally, we report on a prototype implementation of the techniques of this paper using Difference Bound Matrices (DBMs) [10]. We apply this implementation to two case studies: the first concerns the CSMA/CD (Carrier Sense, Multiple Access with Collision Detection) communication protocol [14], whereas the second considers the IEEE 1394 FireWire root contention protocol [13].

The paper proceeds as follows. We review a number of preliminary concepts in Section 2, and in Section 3 we revisit the definitions of probabilistic timed automata and PTCTL. In Section 4, we introduce the model-checking algorithms for PTCTL. Section 5 summarises our prototype implementation of these algorithms. In Section 6, we present the application of the prototype implementation to the case studies, and we conclude the paper in Section 7.

2 Preliminaries

In this section, for the sake of completeness, we recall the definitions of probabilistic and timed probabilistic systems needed to give semantics to probabilistic timed automata. Variants of these were originally introduced in [5, 24, 18].

A (discrete probability) *distribution* over a finite set Q is a function $\mu : Q \to [0,1]$ such that $\sum_{q \in Q} \mu(q) = 1$. For an uncountable set Q', let $\mathsf{Dist}(Q')$ be the set of distributions over finite subsets of Q'. The point distribution μ_q denotes the distribution which assigns probability 1 to q.

Definition 1. *A* probabilistic system, PS, *is a tuple* $(S, \mathit{Steps}, \mathcal{L})$ *where* S *is a set of* states, $\mathit{Steps} \subseteq S \times \mathsf{Dist}(S)$ *is a* probabilistic transition relation, *and* $\mathcal{L} : S \to 2^{AP}$ *is a* labelling function *assigning atomic propositions to states.*

A *probabilistic transition* $s \xrightarrow{\mu} s'$ is made from a state s by nondeterministically selecting a distribution $\mu \in \mathsf{Dist}(S)$ such that $(s, \mu) \in \mathit{Steps}$, and then making a probabilistic choice of target state s' according to μ, such that $\mu(s')>0$.

We consider two ways in which a probabilistic system's computation may be represented. A *path*, representing a particular resolution of both nondeterminism *and* probability, is a non-empty sequence of transitions: $\omega = s_0 \xrightarrow{\mu_0} s_1 \xrightarrow{\mu_1} \cdots$. We denote by $\omega(i)$ the $(i{+}1)$th state of ω and $\mathit{last}(\omega)$ the last state of ω if it is finite. The set of infinite (respectively, finite) paths starting in the state s are denoted by $\mathit{Path}_{\mathit{ful}}(s)$ (respectively, $\mathit{Path}_{\mathit{fin}}(s)$).

In contrast to a path, an *adversary* represents a particular resolution of nondeterminism *only*. Formally, an adversary A is a function mapping every finite path ω to a distribution μ such that $(\mathit{last}(\omega), \mu) \in \mathit{Steps}$. For any adversary A and state s, we let $\mathit{Path}^A_{\mathit{ful}}(s)$ (respectively, $\mathit{Path}^A_{\mathit{fin}}(s)$) denote the subset of $\mathit{Path}_{\mathit{ful}}(s)$ (respectively, $\mathit{Path}_{\mathit{fin}}(s)$) which corresponds to A and, using classical techniques [15], we can define the probability measure Prob^A_s over $\mathit{Path}^A_{\mathit{ful}}(s)$.

We now consider the definition of timed probabilistic systems.

Definition 2. *A* timed probabilistic system, TPS, *is a tuple* $(S, Steps, \mathcal{L})$ *where:* S *and* $\mathcal{L}$ *are as in Definition 1 and* $Steps \subseteq S \times \mathbb{R} \times \mathsf{Dist}(S)$ *is a* timed probabilistic transition relation, *such that, if* $(s,t,\mu) \in Steps$ *and* $t{>}0$, *then* μ *is a point distribution.*

The component t of a tuple (s,t,μ) is called a *duration*. As for probabilistic systems, we can introduce paths and adversaries for timed probabilistic systems, except transitions are now labelled by duration-distribution pairs and an adversary maps each finite path to a duration-distribution pair.

We restrict attention to *time-divergent adversaries*; a common restriction imposed in real-time systems so that unrealisable behaviour (corresponding to time not advancing beyond a bound) is disregarded during analysis. For any path $\omega = s_0 \xrightarrow{t_0,\mu_0} s_1 \xrightarrow{t_1,\mu_1} \cdots$ of a timed probabilistic system, the duration up to the $n{+}1$th state of ω, denoted $\mathcal{D}_\omega(n{+}1)$, equals $\sum_{i=0}^{n} t_i$, and we say that a path ω is *divergent* if for any $t \in \mathbb{R}$, there exists $j \in \mathbb{N}$ such that $\mathcal{D}_\omega(j){>}t$.

Definition 3. *An adversary* A *of a timed probabilistic system* TPS *is* divergent *if and only if for each state* s *of* TPS *the probability under* $Prob_s^A$ *of the divergent paths of* $Path_{ful}^A(s)$ *is* 1. *Let* Adv_{TPS} *be the set of divergent adversaries of* TPS.

For motivation on why we consider *probabilistic* divergence, as opposed to the stronger notion where an adversary is divergent if and only if all its paths are divergent, see [18]. A restriction we impose on probabilistic timed systems is that of *non-zenoness*, which stipulates that there does not exist a state from which time cannot diverge, as we consider this situation to be a modelling error.

Definition 4. *A probabilistic timed system* TPS *is* non-zeno *if it has at least one divergent adversary.*

3 Probabilistic Timed Automata

In this section we review the definition of probabilistic timed automata [18], a modelling framework for real-time systems exhibiting both nondeterministic and stochastic behaviour. The formalism is derived from classical timed automata [3, 12] extended with discrete probability distributions over edges.

3.1 Clocks and Zones

Let $\mathcal{X}$ be a finite set of variables called *clocks* which take values from the time domain $\mathbb{R}$ (non-negative reals). A point $v \in \mathbb{R}^{\mathcal{X}}$ is referred to as a *clock valuation*. For any clock $x \in \mathcal{X}$, we use $v(x)$ to denote the value v assigns to x. For any $v \in \mathbb{R}^{\mathcal{X}}$ and $t \in \mathbb{R}$, we use $v{+}t$ to denote the clock valuation defined as $v(x){+}t$ for all $x \in \mathcal{X}$. We use $v[X{:=}0]$ to denote the clock valuation obtained from v by resetting all of the clocks in $X \subseteq \mathcal{X}$ to 0.

The set of *zones* of $\mathcal{X}$, written $Zones(\mathcal{X})$, is defined inductively by the syntax:

$$\zeta ::= x \leqslant d \,|c \leqslant x \,|\, x + c \leqslant y + d \,|\, \neg\zeta \,|\, \zeta \vee \zeta$$

where $x, y \in \mathcal{X}$ and $c, d \in \mathbb{N}$. The clock valuation v *satisfies* the zone ζ, written $v \triangleright \zeta$, if and only if ζ resolves to true after substituting each clock $x \in \mathcal{X}$ with the corresponding clock value $v(x)$ from v.

We only consider canonical zones [26] ensuring equality between their syntactic and semantic (subsets of $\mathbb{R}^{\mathcal{X}}$ which satisfy the zone) representations. This enables us to use the above syntax interchangeably with set-theoretic operations. We require the following classical operations on zones [12, 26]. For any zones $\zeta, \zeta' \in Zones(\mathcal{X})$ and subset of clocks $X \subseteq \mathcal{X}$, let:

$$\swarrow_{\zeta'} \zeta \stackrel{\text{def}}{=} \{v \,|\, \exists t \geqslant 0.\, (v + t \triangleright \zeta \ \wedge\ \forall t' \leqslant t.\, (v + t' \triangleright \zeta \vee \zeta'))\}$$
$$[X := 0]\zeta \stackrel{\text{def}}{=} \{v \,|\, v[X := 0] \triangleright \zeta\}\,.$$

3.2 Syntax and Semantics of Probabilistic Timed Automata

Definition 5. *A* probabilistic timed automaton *is a tuple* $(L, \mathcal{X}, inv, prob, \mathcal{L})$ *where:* L *is a finite set of* locations*; the function* $inv : L \to Zones(\mathcal{X})$ *is the* invariant condition*; the finite set* $prob \subseteq L \times Zones(\mathcal{X}) \times \mathsf{Dist}(2^{\mathcal{X}} \times L)$ *is the* probabilistic edge relation*; and* $\mathcal{L} : L \to 2^{AP}$ *is a labelling function assigning atomic propositions to locations.*

A *state* of a probabilistic timed automaton PTA is a pair $(l, v) \in L \times \mathbb{R}^{\mathcal{X}}$ such that $v \triangleright inv(l)$. Informally, the behaviour of a probabilistic timed automaton can be understood as follows. In any state (l, v), there is a nondeterministic choice of either making a *discrete transition* or letting *time pass*. A discrete transition can be made according to any $(l, g, p) \in prob$ with source location l which is *enabled*; that is, zone g is satisfied by the current clock valuation v. Then the probability of moving to the location l' and resetting all clocks in X to 0 is given by $p(X, l')$. The option of letting time pass is available only if the invariant condition $inv(l)$ is satisfied while time elapses.

An *edge* of PTA is a tuple (l, g, p, X, l') such that $(l, g, p) \in prob$ and $p(X, l') > 0$. Let edges denote the set of all edges and $\mathsf{edges}(l, g, p)$ the edges of (l, g, p).

We now give the semantics of probabilistic timed automata defined in terms of timed probabilistic systems.

Definition 6. *Let* $\mathsf{PTA} = (L, \mathcal{X}, inv, prob, \mathcal{L})$ *be a probabilistic timed automaton. The* semantics of PTA *is defined as the timed probabilistic system* $\mathsf{TPS}_{\mathsf{PTA}} = (S, Steps, \mathcal{L}')$ *where:* $S \subseteq L \times \mathbb{R}^{\mathcal{X}}$ *such that* $(l, v) \in S$ *if and only if* $v \triangleright inv(l)$*;* $((l, v), t, \mu) \in Steps$ *if and only if one of the following conditions holds:*

[time transitions] $t \geqslant 0$, $\mu = \mu_{(l, v+t)}$ *and* $v + t' \triangleright inv(l)$ *for all* $0 \leqslant t' \leqslant t$
[discrete transitions] $t = 0$ *and there exists* $(l, g, p) \in prob$ *such that* $v \triangleright g$ *and for any* $(l', v') \in S$*:* $\mu(l', v') = \sum_{X \subseteq \mathcal{X} \wedge v' = v[X:=0]} p(X, l')$.

Finally, $\mathcal{L}'(l, v) = \mathcal{L}(l)$ *for all* $(l, v) \in S$.

We say that PTA is non-zeno if and only if $\mathsf{TPS}_{\mathsf{PTA}}$ is non-zeno.

3.3 Probabilistic Timed Computation Tree Logic (PTCTL)

We now describe the probabilistic timed logic PTCTL which can be used to specify properties of probabilistic timed automata. As in TCTL [1, 12], PTCTL employs a set of *formula clocks*, $\mathcal{Z}$, disjoint from the clocks $\mathcal{X}$ of the probabilistic timed automaton. Formula clocks are assigned values by a *formula clock valuation* $\mathcal{E} \in \mathbb{R}^{\mathcal{Z}}$. Timing constraints can be expressed using such clocks and the reset quantifier $z.\phi$. As in PCTL [11], PTCTL includes the probabilistic quantifier $\mathcal{P}_{\sim\lambda}[\cdot]$.

Definition 7. *The syntax of* PTCTL *is defined as follows:*

$$\phi ::= a \mid \zeta \mid \neg\phi \mid \phi \vee \phi \mid z.\phi \mid \mathcal{P}_{\sim\lambda}[\phi \,\mathcal{U}\, \phi]$$

where $a \in AP$, $\zeta \in Zones(\mathcal{X} \cup \mathcal{Z})$, $z \in \mathcal{Z}$, $\sim \in \{\leqslant, <, >, \geqslant\}$ *and* $\lambda \in [0, 1]$.

In PTCTL we can express properties such as 'with probability at least 0.95, the system clock x does not exceed 3 before 8 time units elapse', which is represented as the formula $z.\mathcal{P}_{\geqslant 0.95}[(x{\leqslant}3)\ \mathcal{U}\ (z{=}8)]$.

We write $v, \mathcal{E}$ to denote the composite clock valuation in $\mathbb{R}^{\mathcal{X}\cup\mathcal{Z}}$ obtained from $v \in \mathbb{R}^{\mathcal{X}}$ and $\mathcal{E} \in \mathbb{R}^{\mathcal{Z}}$. Given a state and formula clock valuation pair $(l, v), \mathcal{E}$, zone ζ and duration t, by abuse of notation we let $(l, v), \mathcal{E} \triangleright \zeta$ denote $v, \mathcal{E} \triangleright \zeta$, and $(l, v){+}t$ denote $(l, v{+}t)$.

Definition 8. *Let* $\mathsf{TPS} = (S, Steps, \mathcal{L}')$ *be the timed probabilistic system associated with the probabilistic timed automaton* PTA. *For any state* $s \in S$, *formula clock valuation* $\mathcal{E} \in \mathbb{R}^{\mathcal{Z}}$ *and* PTCTL *formula* θ, *the satisfaction relation* $s, \mathcal{E} \models \theta$ *is defined inductively as follows:*

$$\begin{array}{ll}
s, \mathcal{E} \models a & \Leftrightarrow a \in \mathcal{L}'(s) \\
s, \mathcal{E} \models \zeta & \Leftrightarrow s, \mathcal{E} \triangleright \zeta \\
s, \mathcal{E} \models \phi \vee \psi & \Leftrightarrow s, \mathcal{E} \models \phi \text{ or } s, \mathcal{E} \models \psi \\
s, \mathcal{E} \models \neg\phi & \Leftrightarrow s, \mathcal{E} \not\models \phi \\
s, \mathcal{E} \models z.\phi & \Leftrightarrow s, \mathcal{E}[z := 0] \models \phi \\
s, \mathcal{E} \models \mathcal{P}_{\sim\lambda}[\phi \,\mathcal{U}\, \psi] & \Leftrightarrow p^{A}_{s,\mathcal{E}}(\phi \,\mathcal{U}\, \psi) \sim \lambda \text{ for all } A \in Adv_{\mathsf{TPS}}
\end{array}$$

where $p^{A}_{s,\mathcal{E}}(\phi \,\mathcal{U}\, \psi) = Prob^{A}_{s}\{\omega \in Path^{A}_{ful}(s) \mid \omega, \mathcal{E} \models \phi \,\mathcal{U}\, \psi\}$, *and, for any path* $\omega \in Path_{ful}(s)$: $\omega, \mathcal{E} \models \phi \,\mathcal{U}\, \psi$ *if and only if there exists* $i \in \mathbb{N}$ *and* $t \leqslant \mathcal{D}_{\omega}(i{+}1){-}\mathcal{D}_{\omega}(i)$ *such that*

- $\omega(i){+}t, \mathcal{E}{+}\mathcal{D}_{\omega}(i){+}t \models \psi$;
- *if* $t' < t$, *then* $\omega(i){+}t', \mathcal{E}{+}\mathcal{D}_{\omega}(i){+}t' \models \phi \vee \psi$;
- *if* $j < i$ *and* $t' \leqslant \mathcal{D}_{\omega}(j{+}1){-}\mathcal{D}_{\omega}(j)$, *then* $\omega(j){+}t', \mathcal{E}{+}\mathcal{D}_{\omega}(j){+}t' \models \phi \vee \psi$.

In the following sections we will also consider the dual of the sub-formula $\phi \,\mathcal{U}\, \psi$, namely the release formula $\neg\phi \,\mathcal{V}\, \neg\psi$. In the standard manner, we refer to $\phi \,\mathcal{U}\, \psi$ and $\phi \,\mathcal{V}\, \psi$ as *path formulae*, and use the abbreviation $\mathtt{true}\,\mathcal{U}\,\phi = \Diamond\phi$.

algorithm PTCTLModelCheck(PTA, θ)
output: set of symbolic states $[\![\theta]\!]$ **such that**
$[\![a]\!] := \{(l, inv(l)) \mid l \in L \text{ and } l \in \mathcal{L}(a)\}$;
$[\![\zeta]\!] := \{(l, inv(l) \wedge \zeta) \mid l \in L\}$;
$[\![\neg\phi]\!] := \{(l, inv(l) \wedge \neg \bigvee_{(l,\zeta)\in[\![\phi]\!]} \zeta) \mid l \in L\}$;
$[\![\phi \vee \psi]\!] := [\![\phi]\!] \vee [\![\psi]\!]$;
$[\![z.\phi]\!] := \{(l, [\{z\}{:=}0]\zeta) \mid (l, \zeta) \in [\![\phi]\!]\}$;
$[\![\mathcal{P}_{\sim\lambda}[\phi \,\mathcal{U}\, \psi]]\!] := \mathsf{Until}([\![\phi]\!], [\![\psi]\!], \sim\lambda)$;

Fig. 1. Symbolic PTCTL model-checking algorithm

4 Symbolic PTCTL Model Checking

In this section, we show how a probabilistic timed automaton may be model checked against PTCTL formulae. In order to represent the state sets computed during the model-checking process, we use the concept of *symbolic state*: a symbolic state is a pair (l, ζ) comprising a location and a zone over $\mathcal{X} \cup \mathcal{Z}$. The set of state and formula clock valuation pairs corresponding to a symbolic state (l, ζ) is $\{(l, v), \mathcal{E} \mid v, \mathcal{E} \triangleright \zeta\}$, while the state set corresponding to a set of symbolic states is the union of those corresponding to each individual symbolic state. In the manner standard for model checking, we progress up the parse tree of a PTCTL formula, recursively calling the algorithm PTCTLModelCheck, shown in Figure 1, to compute the set of symbolic states which satisfy each subformula. Handling atomic propositions and Boolean operations is classical, and therefore we only consider only computing $\mathsf{Until}([\![\phi_1]\!], [\![\phi_2]\!], \sim\lambda)$, which arises when we check probabilistically quantified formula. Our technique relies on the comparison of maximum or minimum probabilities with λ, since, from the semantics of PTCTL (Definition 8):

$$\{s, \mathcal{E} \mid s, \mathcal{E} \models \mathcal{P}_{\sim\lambda}[\phi \,\mathcal{U}\, \psi]\} = \begin{cases} \{s, \mathcal{E} \mid p^{\max}_{s,\mathcal{E}}(\phi \,\mathcal{U}\, \psi) \sim \lambda\} \text{ if } \sim \in \{<, \leqslant\} \\ \{s, \mathcal{E} \mid p^{\min}_{s,\mathcal{E}}(\phi \,\mathcal{U}\, \psi) \sim \lambda\} \text{ if } \sim \in \{\geqslant, >\} \end{cases} \quad (1)$$

where for any PTCTL path formula φ:

$$p^{\max}_{s,\mathcal{E}}(\varphi) \stackrel{\text{def}}{=} \sup_{A \in Adv_{\mathsf{TPS}}} p^{A}_{s,\mathcal{E}}(\varphi) \quad \text{and} \quad p^{\min}_{s,\mathcal{E}}(\varphi) \stackrel{\text{def}}{=} \inf_{A \in Adv_{\mathsf{TPS}}} p^{A}_{s,\mathcal{E}}(\varphi)\,.$$

We begin by introducing operations on symbolic states. In Section 4.2, we review algorithms for calculating maximum probabilities, while in Section 4.3 we present new algorithms for calculating minimum probabilities. Proofs of the key results, and specialised algorithms for *qualitative* formulae ($\lambda \in \{0, 1\}$), for which verification can be performed through an analysis of the underlying graph, are available in [22].

4.1 Operations on Symbolic States

In this section we extend the *time predecessor* and *discrete predecessor* functions tpre and dpre of [12, 26] to probabilistic timed automata. For any sets of symbolic

states $\mathtt{U}, \mathtt{V} \subseteq L \times Zones(\mathcal{X} \cup \mathcal{Z})$, clock $x \in \mathcal{X} \cup \mathcal{Z}$ and edge (l, g, p, X, l'):

$$\begin{aligned} x.\mathtt{U} &\stackrel{\text{def}}{=} \{(l, [\{x\}{:=}0]\zeta^l_{\mathtt{U}}) \,|\, l \in L\} \\ \mathsf{tpre}_{\mathtt{U}}(\mathtt{V}) &\stackrel{\text{def}}{=} \{(l, \swarrow_{\zeta^l_{\mathtt{U}} \wedge inv(l)} (\zeta^l_{\mathtt{V}} \wedge inv(l))) \,|\, l \in L\} \\ \mathsf{dpre}((l, g, p, X, l'), \mathtt{U}) &\stackrel{\text{def}}{=} \{(l, g \wedge ([X := 0]\zeta^{l'}_{\mathtt{U}}))\}\,. \end{aligned}$$

where $\zeta^l_{\mathtt{U}} = \bigvee\{\zeta \,|\, (l, \zeta) \in \mathtt{U}\}$, i.e $\zeta^l_{\mathtt{U}}$ is the zone such that $v, \mathcal{E} \triangleright \zeta^l_{\mathtt{U}}$ if and only if $(l, v), \mathcal{E} \in \mathtt{u}$ for some $\mathtt{u} \in \mathtt{U}$. Furthermore, we define the conjunction and disjunction of sets of symbolic states as follows:

$$\mathtt{U} \wedge \mathtt{V} \stackrel{\text{def}}{=} \{(l, \zeta^l_{\mathtt{U}} \wedge \zeta^l_{\mathtt{V}}) \,|\, l \in L\} \text{ and } \mathtt{U} \vee \mathtt{V} \stackrel{\text{def}}{=} \{(l, \zeta^l_{\mathtt{U}} \vee \zeta^l_{\mathtt{V}}) \,|\, l \in L\}\,.$$

Finally, let $[\![\mathtt{false}]\!] = \varnothing$ and $[\![\mathtt{true}]\!] = \{(l, inv(l)) \,|\, l \in L\}$, the sets of symbolic states representing the empty and full state sets respectively.

4.2 Computing Maximum Probabilities

In this section we review the methods for calculating the set of states satisfying a formula of the form $\mathcal{P}_{\lesssim\lambda}[\phi \,\mathcal{U}\, \psi]$ which, from (1), reduces to the computation of $p^{\max}_{s,\mathcal{E}}(\phi \,\mathcal{U}\, \psi)$ for all state and formula clock valuation pairs $s, \mathcal{E}$. Note that, since we consider only non-zeno automata, when calculating these sets we can ignore the restriction to divergent adversaries: letting time diverge can only make reaching ψ less likely. This is similar to verifying (non-probabilistic) non-zeno timed automata against formulae of the form $\phi \,\exists\mathcal{U}\, \psi$ ('there exists a divergent path which satisfies $\phi \,\mathcal{U}\, \psi$') [12].

To compute maximum probabilities, we adopt the algorithm of [19] (see Figure 2). The key observation is that to preserve the probabilistic branching one must take the conjunctions of symbolic states generated by edges from the same distribution. Lines 1–4 deal with the initialisation of $\mathtt{Z}$, which is set equal to the set of time predecessors of $\mathtt{V}$, and the set of edges $E_{(l,g,p)}$ associated with each probabilistic edge $(l, g, p) \in \mathit{prob}$. Lines 5–20 generate a finite-state graph, the nodes of which are symbolic states, obtained by iterating timed and discrete predecessor operations (line 8), and taking conjunctions (lines 12–17). The edges of the graph are partitioned into the sets $E_{(l,g,p)}$ for $(l, g, p) \in \mathit{prob}$, with the intuition that $(\mathtt{z}, (X, l'), \mathtt{z}') \in E_{(l,g,p)}$ corresponds to a transition from any state in $\mathtt{z}$ to some state in $\mathtt{z}'$ when the outcome (X, l') of the probabilistic edge (l, g, p) is chosen. The graph edges are added in lines 11 and 15. The termination of lines 5–20 is guaranteed (see [19]). Line 21 describes the manner in which the probabilistic edges of the probabilistic timed automaton are used in combination with the computed edge sets to construct the probabilistic transition relation *Steps*. Finally, in line 22, model checking is performed on the resulting finite-state probabilistic system PS to obtain $\mathit{MaxProbReach}(\mathtt{z}, \mathsf{tpre}_{\mathtt{U} \vee \mathtt{V}}(\mathtt{V}))$, the maximum probability of reaching $\mathsf{tpre}_{\mathtt{U} \vee \mathtt{V}}(\mathtt{V})$ from $\mathtt{z}$, for each $\mathtt{z} \in \mathtt{Z}$. Note that we write $\mathtt{z} \neq \varnothing$ if and only if $\mathtt{z}$ encodes at least one state and formula clock valuation pair. The following proposition states the correctness of this algorithm.

algorithm $\mathsf{MaxU}(\mathtt{U}, \mathtt{V}, \gtrsim \lambda)$
1. $\mathtt{Z} := \mathsf{tpre}_{\mathtt{U}\vee\mathtt{V}}(\mathtt{V})$
2. **for** $(l,g,p) \in \mathit{prob}$
3. $\quad E_{(l,g,p)} := \varnothing$
4. **end for**
5. **repeat**
6. $\quad \mathtt{Y} := \mathtt{Z}$
7. $\quad$ **for** $\mathtt{y} \in \mathtt{Y} \wedge (l,g,p) \in \mathit{prob} \wedge e = (l,g,p,X,l') \in \mathsf{edges}(l,g,p)$
8. $\quad\quad \mathtt{z} := \mathtt{U} \wedge \mathsf{dpre}(e, \mathsf{tpre}_{\mathtt{U}\vee\mathtt{V}}(\mathtt{y}))$
9. $\quad\quad$ **if** $(\mathtt{z} \neq \varnothing) \wedge (\mathtt{z} \notin \mathsf{tpre}_{\mathtt{U}\vee\mathtt{V}}(\mathtt{V}))$
10. $\quad\quad\quad \mathtt{Z} := \mathtt{Z} \cup \{\mathtt{z}\}$
11. $\quad\quad\quad E_{(l,g,p)} := E_{(l,g,p)} \cup \{(\mathtt{z}, (X,l'), \mathtt{y})\}$
12. $\quad\quad\quad$ **for** $(\bar{\mathtt{z}}, (\bar{X}, \bar{l}'), \bar{\mathtt{y}}) \in E_{(l,g,p)}$
13. $\quad\quad\quad\quad$ **if** $(\mathtt{z} \wedge \bar{\mathtt{z}} \neq \varnothing) \wedge ((\bar{X}, \bar{l}') \neq (X,l')) \wedge (\mathtt{z} \wedge \bar{\mathtt{z}} \notin \mathsf{tpre}_{\mathtt{U}\vee\mathtt{V}}(\mathtt{V}))$
14. $\quad\quad\quad\quad\quad \mathtt{Z} := \mathtt{Z} \cup \{\mathtt{z} \wedge \bar{\mathtt{z}}\}$
15. $\quad\quad\quad\quad\quad E_{(l,g,p)} := E_{(l,g,p)} \cup \{(\mathtt{z} \wedge \bar{\mathtt{z}}, (X,l'), \bar{\mathtt{y}}), (\mathtt{z} \wedge \bar{\mathtt{z}}, (\bar{X}, \bar{l}'), \mathtt{y})\}$
16. $\quad\quad\quad\quad$ **end if**
17. $\quad\quad\quad$ **end for**
18. $\quad\quad$ **end if**
19. $\quad$ **end for**
20. **until** $\mathtt{Z} = \mathtt{Y}$
21. **construct** $\mathsf{PS} = (\mathtt{Z}, \mathit{Steps})$ **where** $(\mathtt{z}, \rho) \in \mathit{Steps}$ **if and only if**
    there exists $(l,g,p) \in \mathit{prob}$ **and** $E \subseteq E_{(l,g,p)}$ **such that**
    $-\ (\,(\mathtt{z}', \mathtt{e}, \mathtt{z}'') \in E \Rightarrow \mathtt{z}' = \mathtt{z}\,) \wedge (\,(\mathtt{z}, \mathtt{e}, \mathtt{z}') \neq (\mathtt{z}, \mathtt{e}', \mathtt{z}'') \in E \Rightarrow \mathtt{e} \neq \mathtt{e}'\,)$
    $-\ \rho(\mathtt{z}') = \sum\{\!|\, p(X,l') \mid (\mathtt{z}, (X,l'), \mathtt{z}') \in E \,|\!\}\ \ \forall \mathtt{z}' \in \mathtt{Z}$
22. **return** $\bigvee\{\mathsf{tpre}_{\mathtt{U}\vee\mathtt{V}}(\mathtt{z}) \mid \mathtt{z} \in \mathtt{Z} \wedge \mathit{MaxProbReach}(\mathtt{z}, \mathsf{tpre}_{\mathtt{U}\vee\mathtt{V}}(\mathtt{V})) \gtrsim \lambda\}$

Fig. 2. Algorithm $\mathsf{MaxU}(\cdot,\cdot,\gtrsim \lambda)$

Proposition 1. *For any probabilistic timed automaton* PTA *and* PTCTL *formula* $\mathcal{P}_{\lesssim\lambda}[\phi\ \mathcal{U}\ \psi]$*, if* $\mathsf{PS} = (\mathtt{Z}, \mathit{Steps})$ *is the probabilistic system generated by* $\mathsf{MaxU}(\llbracket\phi\rrbracket, \llbracket\psi\rrbracket, \gtrsim \lambda)$*, then for any* $s, \mathcal{E} \in S \times \mathbb{R}^{\mathcal{Z}}$*:* $p^{\max}_{s,\mathcal{E}}(\phi\ \mathcal{U}\ \psi){>}0$ *if and only if* $s, \mathcal{E} \in \mathsf{tpre}_{\llbracket\phi\vee\psi\rrbracket}(\mathtt{Z})$*, and if* $p^{\max}_{s,\mathcal{E}}(\phi\ \mathcal{U}\ \psi){>}0$*, then* $p^{\max}_{s,\mathcal{E}}(\phi\ \mathcal{U}\ \psi)$ *equals*

$$\max\left\{ \mathit{MaxProbReach}(\mathtt{z}, \mathsf{tpre}_{\llbracket\phi\vee\psi\rrbracket}\llbracket\psi\rrbracket) \;\middle|\; \mathtt{z} \in \mathtt{Z} \text{ and } s, \mathcal{E} \in \mathsf{tpre}_{\llbracket\phi\vee\psi\rrbracket}(\mathtt{z}) \right\}.$$

Based on the above we set $\mathsf{Until}(\llbracket\phi\rrbracket, \llbracket\psi\rrbracket, \lesssim \lambda) = \llbracket\mathtt{true}\rrbracket \setminus \mathsf{MaxU}(\llbracket\phi\rrbracket, \llbracket\psi\rrbracket, \not\gtrsim \lambda)$.

4.3 Computing Minimum Probabilities

We now consider verifying formulae of the form $\mathcal{P}_{\gtrsim\lambda}[\phi\ \mathcal{U}\ \psi]$ which, using (1), reduces to computing $p^{\min}_{s,\mathcal{E}}(\phi\ \mathcal{U}\ \psi)$ for all state and formula clock valuation pairs $s, \mathcal{E}$. In contrast to the case for maximal probability, the computation of minimal probabilities should be restricted to time-divergent adversaries, in the same way as universally-quantified path formulae of non-probabilistic timed automata should be restricted to time-divergent paths. For example, for any formula clock $z \in \mathcal{Z}$, under divergent adversaries the minimum probability of reaching $z{>}1$ is 1;

algorithm $\mathsf{MaxV}_{\geqslant 1}(c, \mathtt{U}, \mathtt{V})$
$\mathtt{Z}{:=}[\![\mathtt{true}]\!]$
repeat
  $\mathtt{Y}{:=}\mathtt{Z}$
  $\mathtt{Z}{:=}\mathtt{V} \wedge z.\mathsf{MaxU}_{\geqslant 1}(\mathtt{Y}, (\mathtt{U} \wedge \mathtt{Y}) \vee [\![z{>}c]\!])$
until $\mathtt{Z} = \mathtt{Y}$
return $\mathtt{Z}$

algorithm NonZeno
$\mathtt{Z}{:=}[\![\mathtt{true}]\!]$
repeat
  $\mathtt{Y}{:=}\mathtt{Z}$
  $\mathtt{Z}{:=}z.\mathsf{MaxU}_{\geqslant 1}([\![\mathtt{true}]\!], \mathtt{Y} \wedge [\![z{=}1]\!])$
until $\mathtt{Z} = \mathtt{Y}$
return $\mathtt{Z}_0$

Fig. 3. $\mathsf{MaxV}_{\geqslant 1}(c, \cdot, \cdot)$ and NonZeno algorithms

however, if we remove the restriction to time divergent adversaries the minimum probability is 0 (by letting time converge before z exceeds 1).

The techniques we introduce here are based on the result of [12] that verifying $\phi\ \forall\mathcal{U}\ \psi$ ('all divergent paths satisfy $\phi\ \mathcal{U}\ \psi$') reduces to computing the fixpoint:

$$\mu X.(\psi \vee \neg z.((\neg X)\ \exists\mathcal{U}\ (\neg(\phi \vee X) \vee (z{>}c))) \tag{2}$$

for any $c(>0) \in \mathbb{N}$. The important point is that the universal quantification over paths has been replaced by an existential quantification, which, together with the constraint $z{>}c$, allows one to ignore the restriction to time divergence in the verification procedure. Using the duality $\phi\ \mathcal{U}\ \psi \equiv \neg(\neg\phi\ \mathcal{V}\ \neg\psi)$, we have, for any state s of $\mathsf{TPS}_{\mathsf{PTA}}$ and formula clock valuation $\mathcal{E}$:

$$p^{\min}_{s,\mathcal{E}}(\phi\ \mathcal{U}\ \psi) = 1 - p^{\max}_{s,\mathcal{E}}(\neg\phi\ \mathcal{V}\ \neg\psi)\,,$$

and hence, to verify $\mathcal{P}_{\gtrsim\lambda}[\phi\ \mathcal{U}\ \psi]$, it suffices to calculate $p^{\max}_{s,\mathcal{E}}(\neg\phi\ \mathcal{V}\ \neg\psi)$ for all state and formula clock valuation pairs. Now, although we have reduced the problem to calculating a maximum probability, we cannot ignore time divergence when calculating such probabilities. For example, consider the formula $\mathtt{false}\ \mathcal{V}\ \phi$, meaning 'always ϕ' ($\Box\phi$): the probability of this formula is 1 within states satisfying ϕ by always taking time transitions with duration 0.

Proposition 2 below shows that we can reduce the computation of the maximum probability of satisfying a release formula to that of computing the maximum probability of an until operator within which a *qualitative* release formula is nested. As issues of time divergence are irrelevant to the computation of the maximum probabilities of until formulae, the proposition permits us to focus our attention on incorporating time divergence within the verification of the qualitative release formula.

Proposition 2. *For any timed probabilistic system* $\mathsf{TPS}_{\mathsf{PTA}} = (S, \mathit{Steps}, \mathcal{L}')$, $s \in S$, *formula clock valuation* $\mathcal{E} \in \mathbb{R}^{\mathcal{Z}}$ *and PTCTL formulae* ϕ, ψ:

$$p^{\max}_{s,\mathcal{E}}(\phi\ \mathcal{V}\ \psi) = p^{\max}_{s,\mathcal{E}}(\psi\ \mathcal{U}\ \neg\mathcal{P}_{<1}[\phi\ \mathcal{V}\ \psi])\,.$$

Since we have already introduced an algorithm for calculating the maximum probability of satisfying until formulae, it remains to consider a method for calculating the set of states satisfying $\neg\mathcal{P}_{<1}[\phi\ \mathcal{V}\ \psi]$; that is, $\{s, \mathcal{E} \mid p^{\max}_{s,\mathcal{E}}(\phi\ \mathcal{V}\ \psi) \geqslant 1\}$. Based on (2) we obtain the following proposition.

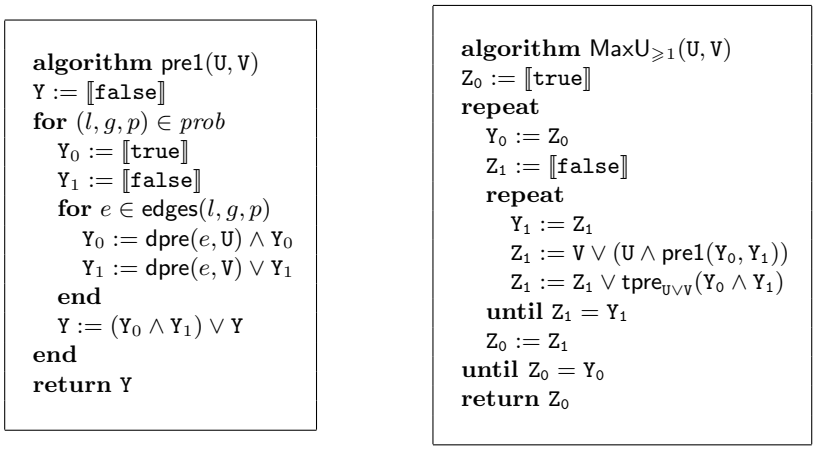

Fig. 4. $\mathsf{MaxU}_{\geqslant 1}(\cdot,\cdot)$ algorithm

Proposition 3. *For any positive $c \in \mathbb{N}$ and PTCTL formulae ϕ, ψ, if $z \in \mathcal{Z}$ does not appear in either ϕ or ψ, then the set $\{s, \mathcal{E} \mid p^{\max}_{s,\mathcal{E}}(\phi \ \mathcal{V} \ \psi) \geqslant 1\}$ is given by the fixpoint $\nu X.(\psi \wedge z.\neg\mathcal{P}_{<1}[X \ \mathcal{U} \ ((X \wedge \phi) \vee z{>}c)])$.*

The algorithm $\mathsf{MaxV}_{\geqslant 1}$ for calculating the set $\{s, \mathcal{E} \mid p^{\max}_{s,\mathcal{E}}(\phi \ \mathcal{V} \ \psi) \geqslant 1\}$ follows from Proposition 3 and is given in Figure 3. The algorithm calls $\mathsf{MaxU}_{\geqslant 1}([\![\phi]\!], [\![\psi]\!])$, given in Figure 4, which computes the set of states $\{s, \mathcal{E} \mid p^{\max}_{s,\mathcal{E}}(\phi \ \mathcal{U} \ \psi) \geqslant 1\}$ and is based on a similar algorithm for finite-state probabilistic systems [9]. Putting this together, and letting $\bar{\geqslant} \, = >$ and $\bar{>} \, = \geqslant$, we set

$$\mathsf{Until}([\![\phi]\!], [\![\psi]\!], \gtrsim \lambda) = [\![\mathtt{true}]\!] \setminus \mathsf{MaxU}([\![\neg\psi]\!], \mathsf{MaxV}_{\geqslant 1}(c, [\![\neg\phi]\!], [\![\neg\psi]\!]), \bar{\gtrsim} 1{-}\lambda) \, .$$

4.4 Checking Non-Zenoness

In the non-probabilistic case [12] checking non-zenoness corresponds to finding the greatest fixpoint $\nu X.(z.(\exists\Diamond \ ((z{=}1){\wedge}X)))$. For probabilistic timed automata, we can replace $\exists$ with $\neg\mathcal{P}_{<1}[\cdot]$, i.e replace 'there exists a path that reaches $(z{=}1){\wedge}X$' with 'there exists an adversary which reaches $(z{=}1) \wedge X$ with probability 1'. Following this approach, the algorithm for calculating the set of non-zeno states is given in Figure 3. Formally, we have the following proposition.

Proposition 4. *A probabilistic timed automaton* PTA *is non-zeno if and only if $\{(l, inv(l) \mid l \in L\}$ equals the fixpoint $\nu X.(z.\neg\mathcal{P}_{<1}[\Diamond((z{=}1) \wedge X)])$.*

Similarly to [12], the algorithm can be used to convert a 'zeno' probabilistic timed automaton into a non-zeno automaton by strengthening invariants.

5 Implementation

In this section we briefly summarise our prototype implementation of the model-checking algorithms given in Section 4. It is important to note that the aim of

our implementation is to validate the algorithms presented for model checking probabilistic timed automata against PTCTL, rather than to devise an efficient implementation; the latter will be the subject of future work. Note that, to perform the final step of the algorithm MaxU (line 22 of Figure 2), that is compute maximum reachability probabilities on a finite-state probabilistic system, we export the problem to the probabilistic symbolic model checker PRISM [16, 23].

The main step in the implementation of our techniques is the representation of (sets of) symbolic states and the operations required on them; more precisely, since a symbolic state is a pair (l, ζ) where $l \in L$ and ζ is a zone, we require a method for representing zones and performing operations on zones.

Difference Bound Matrices (DBMs) [10] are a well known data-structure for the representation of convex zones and are used in the model checkers UPPAAL [4] and KRONOS [8]. As the operations required by our algorithm can introduce non-convexity, we also represent non-convex zones. Following the approach presented in [26], we represent non-convex zones by lists of DBMs, that is, we represent a non-convex zone ζ by a list of convex zones $\zeta_1, \ldots, \zeta_n$ such that $\zeta = \zeta_1 \cup \cdots \cup \zeta_n$. It thus follows that a symbolic state can be represented by a location and a list of DBMs. This representation is clearly not canonical: there are many ways of decomposing a non-convex zone into a set of convex zones. However, [26] presents algorithms (used by KRONOS [6]) for the operations we require when zones are represented as lists of DBMs. Based on [26], we have implemented, in Java, a prototype DBM package and the operations on lists of DBMs required by our model-checking algorithms. Note that the equality checking performed by the algorithms $\mathsf{MaxV}_{\geqslant 1}$, $\mathsf{MaxU}_{\geqslant 1}$ and NonZeno reduces to an inclusion test based on whether a least or greatest fixpoint is being performed.

6 Case Studies

In this section we report on the results of our prototype implementation applied to two case studies: the CSMA/CD communication protocol [14], and the IEEE1394 FireWire root contention protocol [13]. Due to space limitations, we include the results for the generation of the finite-state probabilistic system, and not the verification of this system which is performed by PRISM, and is therefore standard; further details are available from the PRISM web page [23].

We test both the maximum and minimum probability algorithms, in each case confirming the results with those obtained using the digital clocks approach in PRISM [17, 21] and, when possible, the forward reachability approach [18, 7]. This comparison is feasible since the models are 'closed' and 'diagonal-free' (they do not feature either strict inequalities or comparisons between clocks), but our algorithms are applicable to general probabilistic timed automata. When calculating minimum probabilities of deadline properties, for comparison we also use the alternative method introduced in [21], as explained by the following remark.

Remark 1. We observe that certain deadline properties referring to minimum probability can be expressed in terms of properties referring to maximum prob-

Table 1. Statistics for $\mathsf{MaxV}_{\geqslant 1}$ as c varies, when verifying the CSMA/CD protocol

c	$\mathcal{P}_{\geqslant 1}[\Diamond\ \mathsf{done}]$			$z.\mathcal{P}_{\geqslant \lambda}[\Diamond(\mathsf{done} \wedge z \leqslant 2000)]$		
	time (sec)	$\mathsf{MaxV}_{\geqslant 1}$ iters	$\mathsf{MaxU}_{\geqslant 1}$ iters	time (sec)	$\mathsf{MaxV}_{\geqslant 1}$ iters	$\mathsf{MaxU}_{\geqslant 1}$ iters
10	119	99	583	56.4	15	97
26	56.2	40	261	31.3	9	83
50	37.6	22	156	24.0	6	65
100	136	13	120	720	8	94
808	11,240	4	89	23,276	5	115

Table 2. Model sizes (and generation times in seconds) for CSMA/CD protocol

D	$z.\mathcal{P}_{\sim \lambda}[\Diamond(\mathsf{done} \wedge z \leqslant D)]$				$\mathcal{P}_{\leqslant \lambda}[\Diamond \mathsf{exceeded}]$		digital
(μs)	maximum [$\sim = \leqslant$]		minimum [$\sim = \geqslant$]				clocks [20]
1600	431	(83.8)	451	(58.2)	362	(18.6)	4,501,705
2000	725	(125)	691	(75.6)	562	(29.4)	6,570,692
2400	997	(153)	1,075	(151)	882	(77.2)	8,654,692
2800	1,263	(205)	1,435	(284)	1,182	(158)	10,738,692

ability. Consider a property $z.\mathcal{P}_{\geqslant \lambda}[\Diamond(\phi \wedge z \leqslant D)]$ and assume that ϕ is reachable with probability 1 for all adversaries and states. We adjust the model so that states in which ϕ is true are forced to make a transition to a sink-location; furthermore, we allow the model to make a transition to a different, "deadline exceeded" sink-location, denoted exceeded, as soon as the value of the clock z exceeds D [21] (provided that we are not in a state satisfying ϕ). Because this location is a sink, ϕ cannot become true after it is entered. Then, given any adversary A, state s and formula clock valuation $\mathcal{E}$, we have that $p^A_{s,\mathcal{E}}(\Diamond(\phi \wedge z \leqslant D)) = 1 - p^A_{s,\mathcal{E}}(\Diamond\ \mathsf{exceeded})$. Hence, we are able to reduce the computation of a minimum probability to a maximum probability.

CSMA/CD protocol The CSMA/CD (Carrier Sense, Multiple Access with Collision Detection) protocol is designed for networks with a single channel and specifies the behaviour of stations with the aim of minimising simultaneous use of the channel (data collision). The basic structure of the protocol is as follows: when a station has data to send, it listens to the medium, after which, if the medium was free (no one transmitting), the station starts to send its data. On the other hand, if the medium was sensed busy, the station waits a random amount of time and then repeats this process. For the case study we have supposed that the random choice is a uniform choice between two delays. Further details are available from the PRISM web page [23].

The first property we check is that the minimum probability that both stations correctly deliver their packets is 1; that is, we verify $\mathcal{P}_{\geqslant 1}[\Diamond\ \mathsf{done}]$. The algorithm $\mathsf{MaxV}_{\geqslant 1}$ returns no symbolic states, which implies that the minimum probability is 1. In Table 1 we give the model-checking statistics for $\mathsf{MaxV}_{\geqslant 1}$ as we vary the parameter c (26 and 808 are the smallest and largest non-zero constants appearing in the model). As in the non-probabilistic case [27], further investigations and case studies are needed to establish if there is any way of finding a 'good' choice for the parameter c in advance.

Table 3. Statistics for $\mathsf{MaxV}_{\geqslant 1}$ as c varies, when verifying $\mathtt{I}_1^{\mathtt{p}}$

c	$\mathcal{P}_{\geqslant 1}[\Diamond\ \mathsf{elect}]$			$z.\mathcal{P}_{\geqslant \lambda}[\Diamond(\mathsf{elect} \wedge z \leqslant 10000)]$		
	time (sec)	$\mathsf{MaxV}_{\geqslant 1}$ iters	$\mathsf{MaxU}_{\geqslant 1}$ iters	time (sec)	$\mathsf{MaxV}_{\geqslant 1}$ iters	$\mathsf{MaxU}_{\geqslant 1}$ iters
10	21.26	372	1,492	13.3	85	373
100	2.63	39	162	3.96	18	95
360	1.22	13	67	1.84	7	55
1,670	0.646	5	30	1.76	5	46
5,000	0.479	3	24	1.07	3	34
10,000	0.571	3	31	1.19	3	41

Table 4. Model sizes and (generation times in seconds) when verifying $\mathtt{I}_1^{\mathtt{p}}$

D (10^3ns)	$z.\mathcal{P}_{\geqslant \lambda}[\Diamond(\mathsf{elect} \wedge z \leqslant D)]$		$\mathcal{P}_{\leqslant \lambda}[\Diamond \mathsf{exceeded}]$		forwards [7]	digital clocks [20]
2	15	(1.09)	25	(0.197)	53 (0.00)	68,056
4	25	(1.20)	47	(0.280)	131 (0.00)	220,565
8	81	(1.38)	126	(0.615)	372 (0.02)	530,965
10	126	(1.65)	183	(1.09)	526 (0.03)	686,165
20	528	(12.2)	643	(19.3)	1,876 (0.09)	1,462,165
40	2,168	(886)	2,395	(1,333)	7,034 (0.46)	3,014,165

The remaining properties we consider are the maximum and minimum probabilities that both stations deliver their packets by time D, that is, the property $z.\mathcal{P}_{\sim \lambda}[\Diamond(\mathsf{done} \wedge z \leqslant D)]$. Using the observation given in Remark 1 and the results for $\mathcal{P}_{\geqslant 1}[\Diamond\ \mathsf{done}]$, we also compute the minimum probabilities via a translation into a computation of maximum probabilities. In Table 2 we have presented the model sizes (and generation times) of the finite-state probabilistic system generated by our implementation and, for comparison, the size of the model constructed using the digital clocks approach [17, 21] (there are no generation times in this case as the digital semantics leads directly to a finite-state system). The results show a significant decrease in the model size when compared to the digital clocks approach. Comparing the results for $z.\mathcal{P}_{\geqslant \lambda}[\Diamond(\mathsf{done} \wedge z \leqslant D)]$ and $\mathcal{P}_{\leqslant \lambda}[\Diamond \mathsf{exceeded}]$, we see that using Remark 1 can decrease both the states and generation time. Table 1 includes the model-checking statistics for the $\mathsf{MaxV}_{\geqslant 1}$ algorithm when verifying $z.\mathcal{P}_{\sim \lambda}[\Diamond(\mathsf{done} \wedge z \leqslant 2000)]$.

FireWire root contention protocol We consider the abstract probabilistic timed automaton model $\mathtt{I}_1^{\mathtt{p}}$, which is a probabilistic extension of the classical timed automaton $\mathtt{I}_1$ of [25], as studied in [7, 21]. The timing constraints are derived from those given in the standard when the communication delay is 360ns. The properties we consider concern the minimum probability to elect a leader with and without a deadline, that is, the properties $\mathcal{P}_{\geqslant \lambda}[\Diamond\ \mathsf{elect}]$ and $z.\mathcal{P}_{\geqslant \lambda}[\Diamond(\mathsf{elect} \wedge z \leqslant D)]$.

When verifying $\mathcal{P}_{\geqslant \lambda}[\Diamond\ \mathsf{elect}]$, the algorithm $\mathsf{MaxV}_{\geqslant 1}$ returns no symbolic states, which implies that the minimum probability is 1. In Table 3 we give the model-checking statistics for the $\mathsf{MaxV}_{\geqslant 1}$ algorithm as the value of c changes (360 and 1670 are the smallest and largest non-zero constants appearing in the model). In Table 4 we have reported on the size and generation times in sec-

onds when verifying $z.\mathcal{P}_{\geqslant\lambda}[\Diamond(\mathsf{elect} \wedge z \leqslant D)]$ for a range of deadlines. As for the CSMA/CD case study, we can use Remark 1 and instead verify $\mathcal{P}_{\leqslant\lambda}[\Diamond\mathsf{exceeded}]$ on a modified model. Additionally, in Table 4 we include the results obtained when applying the forwards approach [18, 7] and using digital clocks [21]. Note that the approach of [18, 7] cannot be used to calculate the minimum probability of eventually electing a leader. The results show that the use of the algorithms presented in this paper leads to a smaller state space than the other approaches. The generation times for our prototype implementation are considerably greater than those obtained with the forwards approach as these are generated with the optimized tool KRONOS. Comparing the results obtained when verifying $z.\mathcal{P}_{\geqslant\lambda}[\Diamond(\mathsf{elect} \wedge z \leqslant D)]$ and $\mathcal{P}_{\leqslant\lambda}[\Diamond\mathsf{exceeded}]$, we see that the direct approach leads to a smaller state space and, for large deadlines, is faster than the approach based on Remark 1.

7 Conclusions

We have presented the theoretical foundations for the symbolic model checking of probabilistic timed automata and PTCTL and validated them through a prototype implementation using DBMs. For quantitative formulae, our algorithm is expensive, as, in the worst case, the MaxU algorithm constructs the powerset of the region graph, which itself is exponential in the largest constant used in zones and the number of clocks. However, for the case studies considered, we observe much smaller state spaces than this upper bound, which confirms that the algorithms are feasible in practice. Note that we do not construct a partition of the state space (as in [2], for example), but rather a (property dependent) set of overlapping symbolic states to avoid potentially expensive disjunction operations on zones within MaxU.

Future work will address the efficient symbolic implementation of the presented algorithms, adaptations to probabilistic polyhedral hybrid automata and symbolic probabilistic systems [19], and a comparison of our approach with state partitioning techniques, for example [2], extended to the probabilistic setting.

References

1. Alur, R., Courcoubetis, C., and Dill, D. Model checking in dense real time. *Information and Computation*, 104(1):2–34, 1993.
2. Alur, R., Courcoubetis, C., Dill, D. L., Halbwachs, N., and Wong-Toi, H. Minimization of timed transition systems. In *Proc. CONCUR'92*, volume 630 of *LNCS*. Springer, 1992.
3. Alur, R. and Dill, D. L. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
4. Behrmann, G., David, A., Larsen, K., Möller, O., Pettersson, P., and Yi, W. UPPAAL - present and future. In *Proc. CDC'01*. IEEE, 2001.
5. Bianco, A. and de Alfaro, L. Model checking of probabilistic and nondeterministic systems. In *Proc. FST&TCS'95*, volume 1026 of *LNCS*, pages 499–513. Springer, 1995.

6. Daws, C. Private communication. 2004.
7. Daws, C., Kwiatkowska, M., and Norman, G. Automatic verification of the IEEE 1394 root contention protocol with KRONOS and PRISM. *International Journal on Software Tools for Technology Transfer*, 5(2–3):221–236, 2004.
8. Daws, C., Olivero, A., Tripakis, S., and Yovine, S. The tool KRONOS. In *Proc. Hybrid Systems III*, volume 1066 of *LNCS*, pages 208–219. Springer, 1996.
9. de Alfaro, L. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.
10. Dill, D. Timing assumptions and verification of finite-state concurrent systems. In *Proc. Automatic Verification Methods for Finite State Systems*, volume 407 of *LNCS*, pages 197–212. Springer, 1990.
11. Hansson, H. and Jonsson, B. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(4):512–535, 1994.
12. Henzinger, T., Nicollin, X., Sifakis, J., and Yovine, S. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.
13. IEEE 1394-1995. High Performance Serial Bus Standard. 1995.
14. IEEE 802.3-2002. Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Standard. 2002.
15. Kemeny, J., Snell, J., and Knapp, A. *Denumerable Markov Chains*. Springer, 1976.
16. Kwiatkowska, M., Norman, G., and Parker, D. PRISM: Probabilistic symbolic model checker. In *Proc. TOOLS'02*, volume 2324 of *LNCS*, pages 200–204. Springer, 2002.
17. Kwiatkowska, M., Norman, G., Parker, D., and Sproston, J. Performance analysis of probabilistic timed automata using digital clocks. In *Proc. FORMATS'03*, volume 2791 of *LNCS*, pages 105–120. Springer, 2003.
18. Kwiatkowska, M., Norman, G., Segala, R., and Sproston, J. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 282:101–150, 2002.
19. Kwiatkowska, M., Norman, G., and Sproston, J. Symbolic computation of maximal probabilistic reachability. In *Proc. CONCUR '01*, volume 2154 of *LNCS*, pages 169–183. Springer, 2001.
20. Kwiatkowska, M., Norman, G., and Sproston, J. 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.
21. Kwiatkowska, M., Norman, G., and Sproston, J. Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol. *Formal Aspects of Computing*, 14:295–318, 2003.
22. Kwiatkowska, M., Norman, G., and Sproston, J. Symbolic model checking for probabilistic timed automata. Technical Report CSR-03-10, School of Computer Science, University of Birmingham, 2003.
23. PRISM Web site. www.cs.bham.ac.uk/~dxp/prism.
24. Segala, R. *Modelling and Verification of Randomized Distributed Real Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.
25. Simons, D. and Stoelinga, M. Mechanical verification of the IEEE 1394a root contention protocol using Uppaal2k. *Springer International Journal of Software Tools for Technology Transfer*, 3(4):469–485, 2001.
26. Tripakis, S. *L'Analyse Formelle des Systèmes Temporisés en Pratique*. PhD thesis, Université Joseph Fourier, 1998.
27. Wang, F., Hwang, G.-D., and Yu, F. TCTL inevitability analysis of dense-time systems. In *Proc. CIAA'03*, volume 2759 of *LNCS*, pages 176–187. Springer, 2003.