Chapter 8

Verification of Real-time Probabilistic Systems

8.1. Introduction

In this chapter, we consider verification techniques for a particular class of real-time systems: those which incorporate *probabilistic* characteristics. This can come from a number of possible sources, for example, unpredictable behaviour resulting from the failure of components of a system, or random choices made according to the execution of a probabilistic protocol. An application domain in which both of these aspects are important is that of probabilistic communication protocols, including for example Bluetooth, IEEE 802.11 Wireless LAN and IEEE 1394 FireWire, which often include potentially faulty communication channels and probabilistic algorithms such as randomised back-off.

Formal verification techniques for systems such as these need to incorporate not only probabilistic characteristics, but also *nondeterminism* (e.g. due to concurrency between asynchronous components or underspecified system parameters) and *real-time* information. For this reason, *probabilistic timed automata* are an ideal modelling formalism in this context. In this chapter, we give an introduction to probabilistic timed automata, how to express properties of these models, and a range of model checking techniques that can be applied to them. We also include an example of their practical application to a case study: the IEEE 1394 FireWire root contention protocol.

The structure of the remainder of this chapter is as follows. In Section 8.2, we give some background material and notation, introduce the probabilistic timed automata

Chapter written by Marta KWIATKOWSKA, Gethin NORMAN, David PARKER and Jeremy SPROSTON.

formalism and then explain how properties of these models can be formalised. In Section 8.3, we present four model checking techniques: the *region graph* method, the *forwards symbolic* and *backwards symbolic* approaches and the *digital clocks* technique. Finally, in Section 8.4, we present the FireWire root contention case study and, using this, summarise the relative performances of the various approaches.

8.2. Probabilistic timed automata

We model real-time probabilistic systems using probabilistic timed automata [JEN 96, KWI 02a], a formalism which features nondeterministic and probabilistic choices between transitions, and also contains constraints on the times at which transitions can or must be taken. The formalism is derived by extending classical timed automata [ALU 94, HEN 94] with discrete probability distributions over edges.

8.2.1. *Preliminaries*

A discrete probability *distribution* over a countable set Q is a function $\mu : Q \to [0,1]$ such that $\sum_{q \in Q} \mu(q) = 1$. Let $\mathsf{Dist}(Q)$ denote the set of all probability distributions over Q. For a distribution μ on Q, let $\mathsf{support}(\mu) = \{q \in Q \mid \mu(q){>}0\}$. For an uncountable set Q' we define $\mathsf{Dist}(Q')$ to be the set of functions $\mu : Q' \to [0,1]$, such that $\mathsf{support}(\mu)$ is a countable set and μ restricted to $\mathsf{support}(\mu)$ is a (discrete) probability distribution. For $q \in Q$, let μ_q be the *point distribution* at q which assigns probability 1 to q.

We use $\mathbb{R}$ to denote the non-negative reals, and $\mathbb{N}$ to denote the naturals. Let $\mathbb{T} \in \{\mathbb{R}, \mathbb{N}\}$ be a *time domain*, and let $\mathcal{X}$ be a finite set of variables called *clocks* which take values from $\mathbb{T}$. A function $v : \mathcal{X} \to \mathbb{T}$ is referred to as a *clock valuation*. The set of all clock valuations is denoted by $\mathbb{T}^{\mathcal{X}}$. Let $\mathbf{0} \in \mathbb{T}^{\mathcal{X}}$ be the clock valuation that assigns 0 to all clocks in $\mathcal{X}$. For any $v \in \mathbb{T}^{\mathcal{X}}$ and $t \in \mathbb{T}$, we use $v{+}t$ to denote the clock valuation defined as $(v{+}t)(x) = v(x){+}t$ 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, and leaving the values of all other clocks unchanged; formally, $v[X{:=}0](x) = 0$ if $x \in X$ and $v[X{:=}0](x) = v(x)$ otherwise.

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

$$\zeta \ ::= \ x \leqslant d \ \mid \ c \leqslant x \ \mid \ x{+}c \leqslant y{+}d \ \mid \ \neg\zeta \ \mid \ \zeta \vee \zeta$$

where $x, y \in \mathcal{X}$ and $c, d \in \mathbb{N}$. As usual, $\zeta_1 \wedge \zeta_2 \equiv \neg(\neg\zeta_1 \vee \neg\zeta_2)$, strict constraints can be written using negation, for example $x{>}2 \equiv \neg(x{\leqslant}2)$, and equality can be written as the conjunction of constraints, for example $x{=}3 \equiv (x{\leqslant}3) \wedge (3{\leqslant}x)$. The clock valuation v *satisfies* the zone ζ, written $v \triangleleft \zeta$, 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.

Intuitively, the semantics of a zone is the set of clock valuations (subset of $\mathbb{T}^{\mathcal{X}}$) which satisfy the zone.

Note that more than one zone may represent the same set of clock valuations (for example, $(x{\leqslant}2)\wedge(y{\leqslant}1)\wedge(x{\leqslant}y{+}2)$ and $(x{\leqslant}2)\wedge(y{\leqslant}1)\wedge(x{\leqslant}y{+}3)$). We henceforth consider only canonical zones, which are zones for which the constraints are as 'tight' as possible. For any valid zone $\zeta \in \mathit{Zones}(\mathcal{X})$, there exists an $O(|\mathcal{X}|^3)$ algorithm to compute the (unique) canonical zone of ζ [DIL 89]. This enables us to use the above syntax for zones interchangeably with semantic, set-theoretic operations.

We require the following classical operations on zones [HEN 94, TRI 98]. For zones ζ, ζ' and set of clocks $X \subseteq \mathcal{X}$:

$$\begin{aligned}
\nearrow_{\zeta'} \zeta &\overset{\text{def}}{=} \big\{v \,|\, \exists t{\geqslant}0.\, \exists v' \triangleleft \zeta.\, \big(v{=}v'{+}t \;\wedge\; \forall t'{\leqslant}t.\,(v'{+}t' \triangleleft \zeta{\vee}\zeta')\big)\big\} \\
\swarrow_{\zeta'} \zeta &\overset{\text{def}}{=} \big\{v \,|\, \exists t{\geqslant}0.\, \big(v{+}t \triangleleft \zeta \;\wedge\; \forall t'{\leqslant}t.\,(v{+}t' \triangleleft \zeta{\vee}\zeta')\big)\big\} \\
[X{:=}0]\zeta &\overset{\text{def}}{=} \{v \,|\, v[X := 0] \in \zeta\} \\
\zeta[X{:=}0] &\overset{\text{def}}{=} \{v[X{:=}0] \,|\, v \in \zeta\}\,.
\end{aligned}$$

Zone $\nearrow_{\zeta'} \zeta$ contains the clock valuations that can be reached from a clock valuation in ζ by letting time pass and remain in ζ' while time elapses. On the other hand, zone $\swarrow_{\zeta'} \zeta$ contains the clock valuations that can, by letting time pass, reach a clock valuation in ζ and remain in ζ' until ζ is reached. Zone $[X{:=}0]\zeta$ contains the clock valuations which result in a clock valuation in ζ when the clocks in X are reset to 0, while $\zeta[X{:=}0]$ contains the clock valuations which are obtained from clock valuations in ζ by resetting the clocks in X to 0.

In addition we will require the concepts of *c-equivalence* and *c-closure* [TRI 98, DAW 98].

DEFINITION 8.1.– *For $c \in \mathbb{N}$, two clock valuations v, v' are described as* c-equivalent *if the following conditions are satisfied:*

– for any $x \in \mathcal{X}$, either $v(x) = v'(x)$, or $v(x){>}c$ and $v'(x){>}c$;

– for any $x, y \in \mathcal{X}$, either $v(x){-}v(y) = v'(x){-}v'(y)$, or $v(x){-}v(y) > c$ and $v'(x){-}v'(y) > c$.

We define the c-closure *of a zone $\zeta \in \mathbb{R}^{\mathcal{X}}$, denoted by $\mathsf{close}(\zeta, c)$, to be the greatest zone $\zeta' \supseteq \zeta$ satisfying the following: for all $v' \in \zeta'$, there exists $v \in \zeta$ such that v and v' are c-equivalent.*

Intuitively, the c-closure of a zone is obtained by removing all of its boundaries that correspond to constraints referring to integers greater than c. Observe that, for a given c, there are only a finite number of c-closed zones.

8.2.2. *Syntax of probabilistic timed automata*

We now define formally probabilistic timed automata, in which, similarly to timed automata [ALU 94, HEN 94], the timing constraints are represented by zones over a finite set of clocks. The difference between timed automata and probabilistic timed automata lies in the edge relation: in timed automata the traversal of an edge corresponds to a single target location and, possibly, the resetting of some clocks; instead probabilistic timed automata allow (discrete) probabilistic choice over target locations and clock resets.

DEFINITION 8.2.– *A* probabilistic timed automaton *is a tuple* $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$ *where:*

– L *is a finite set of* locations *with an* initial location $\bar{l} \in L$*;*

– $\mathcal{X}$ *is a finite set of* clocks;

– Σ *is a finite set of* events;

– $inv : L \to Zones(\mathcal{X})$ *is a function called the* invariant condition;

– $prob \subseteq L \times Zones(\mathcal{X}) \times \Sigma \times \mathsf{Dist}(2^{\mathcal{X}} \times L)$ *is a finite set called the* probabilistic edge relation.

A *state* of a probabilistic timed automaton PTA is a pair $(l, v) \in L \times \mathbb{T}^{\mathcal{X}}$ such that $v \triangleleft inv(l)$. Informally, the behaviour of a probabilistic timed automaton can be understood as follows. The model starts in the initial location $\bar{l}$ with all clocks set to 0, that is, in the state $(\bar{l}, \mathbf{0})$. In this, and any other state (l, v), there is a nondeterministic choice of either (1) making a *discrete transition* or (2) letting *time pass*. In case (1), a discrete transition can be made according to any probabilistic edge $(l, g, \sigma, p) \in prob$ with source location l which is *enabled*, where (l, g, σ, p) is said to be enabled if its zone g is satisfied by the current clock valuation v. Then the probability of moving to the location l' and resetting all of the clocks in X to 0 is given by $p(X, l')$. Case (2) has the effect of increasing the values of all clocks in $\mathcal{X}$ by the amount of time elapsed. We note that, in case (2), the option of letting time pass is available only if the invariant condition $inv(l)$ is satisfied continually while time elapses.

DEFINITION 8.3.– *An* edge *of* PTA *generated by* $(l, g, \sigma, p) \in prob$ *is a tuple of the form* (l, g, σ, p, X, l') *such that* $p(X, l') > 0$*. Let* $\mathsf{edges}(l, g, \sigma, p)$ *be the set of edges generated by* (l, g, σ, p)*, and let* $\mathsf{edges} = \{\mathsf{edges}(l, g, \sigma, p) \mid (l, g, \sigma, p) \in prob\}$.

EXAMPLE 8.1.– *Consider the probabilistic timed automaton modelling a simple probabilistic communication protocol given in Figure 8.1. The nodes represent the locations, namely* di *(sender has data, receiver idle),* si *(sender sent data, receiver idle), and* sr *(sender sent data, receiver received). The automaton starts in location* di *in which data is ready to be sent by the sender (the double border indicates that* di *is the initial location). After between 1 and 2 time units, the sender attempts to send the*

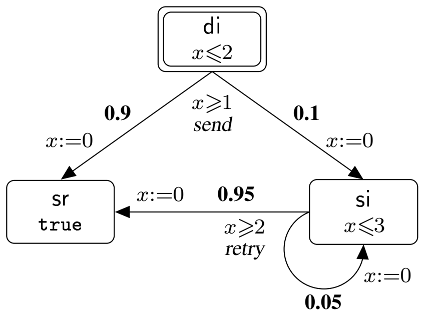


Figure 8.1. *A probabilistic timed automaton modelling a simple communication protocol*

data (event send*) and with probability 0.9 the data is received (location* sr *is reached), or with probability 0.1 the data is lost (location* si *is reached). In* si*, after 2 to 3 time units, the sender will attempt to resend the data (event* resend*), which again can be lost, this time with probability 0.05.*

Note that a timed automaton [ALU 94, HEN 94] corresponds to a probabilistic timed automaton for which every probabilistic edge $(l, g, \sigma, p) \in \mathit{prob}$ is such that $p = \mu_{(X,l')}$ (the point distribution assigning probability 1 to (X, l')) for some $(X, l') \in 2^{\mathcal{X}} \times L$.

We say that a probabilistic timed automaton is *well-formed* if, whenever a probabilistic edge is enabled, the target states resulting from all probabilistic outcomes satisfy the invariant condition. Formally, a probabilistic timed automaton $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, \mathit{inv}, \mathit{prob})$ is said to be well-formed if:

$$\forall (l, g, \sigma, p) \in \mathit{prob}.\ \forall v \in \mathbb{T}^{\mathcal{X}}.\ (v \triangleleft g) \rightarrow \big(\forall (X, l') \in \mathsf{support}(p).\ v[X{:=}0] \triangleleft \mathit{inv}(l')\big)\ .$$

A probabilistic timed automaton can be transformed into a well-formed probabilistic timed automaton by simply replacing the guard g in each probabilistic edge $(l, g, \sigma, p) \in \mathit{prob}$ with

$$\left(\bigwedge_{(X,l') \in \mathsf{support}(p)} [X{:=}0]\mathit{inv}(l') \right) \wedge g\ .$$

For the remainder of the chapter we assume that all probabilistic timed automata are well-formed. Finally, we also assume that the set Σ of events of a probabilistic timed automaton is such that $\Sigma \cap \mathbb{R} = \emptyset$.

8.2.3. *Modelling with probabilistic timed automata*

To aid higher-level modelling, it is often useful to define complex systems as the *parallel composition* of a number of interacting sub-components. The definition of the parallel composition operator $\|$ of probabilistic timed automata uses ideas from the theory of (untimed) probabilistic automata [SEG 95b] and classical timed automata [ALU 94]. Let $\mathsf{PTA}_i = (L_i, \bar{l}_i, \mathcal{X}_i, \Sigma_i, inv_i, prob_i)$ for $i \in \{1, 2\}$ and assume that $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$.

DEFINITION 8.4.– *The* parallel composition *of two probabilistic timed automata* PTA_1 *and* PTA_2 *is the probabilistic timed automaton*

$$\mathsf{PTA}_1 \,\|\, \mathsf{PTA}_2 = (L_1 \times L_2, (\bar{l}_1, \bar{l}_2), \mathcal{X}_1 \cup \mathcal{X}_2, \Sigma_1 \cup \Sigma_2, inv, prob)$$

such that

– $inv(l, l') = inv_1(l) \wedge inv_2(l')$ *for all* $(l, l') \in L_1 \times L_2$;

– $((l_1, l_2), g, \sigma, p) \in prob$ *if and only if one of the following conditions holds:*

1) $\sigma \in \Sigma_1 \backslash \Sigma_2$ *and there exists* $(l_1, g, \sigma, p_1) \in prob_1$ *such that* $p = p_1 \otimes \mu_{(\emptyset, l_2)}$;

2) $\sigma \in \Sigma_2 \backslash \Sigma_1$ *and there exists* $(l_2, g, \sigma, p_2) \in prob_2$ *such that* $p = \mu_{(\emptyset, l_1)} \otimes p_2$;

3) $\sigma \in \Sigma_1 \cap \Sigma_2$ *and there exists* $(l_i, g_i, \sigma, p_i) \in prob_i$ *for* $i = 1, 2$ *such that* $g = g_1 \wedge g_2$ *and* $p = p_1 \otimes p_2$

where for any $l_1 \in L_1$, $l_2 \in L_2$, $X_1 \subseteq \mathcal{X}_1$ *and* $X_2 \subseteq \mathcal{X}_2$:

$$p_1 \otimes p_2(X_1 \cup X_2, (l_1, l_2)) = p_1(X_1, l_1) \cdot p_2(X_2, l_2)\,.$$

In addition (as in the timed automata model-checking tool UPPAAL [LAR 97, BEH 04]), we allow the use of *urgent locations*, which when entered must be left before time can advance. These can be represented in the probabilistic timed automata framework by introducing an auxiliary clock z, setting the invariant condition of the urgent locations to $z \leqslant 0$, and resetting the value of z to 0 on entry to each urgent location. However, rather than introducing an auxiliary clock, it is generally more convenient from a modelling perspective to explicitly denote such locations as urgent.

8.2.4. *Semantics of probabilistic timed automata*

The semantics of timed automata is generally presented in terms of timed transition systems. To define the semantics of probabilistic timed automata, we employ timed Markov decision processes, which extend timed transition systems with (discrete) probabilistic choice. Timed Markov decision processes can also be seen as an extension of Markov decision processes (see Chapter 9) and a variant of Segala's probabilistic timed automata [SEG 95a].

DEFINITION 8.5.– *A* timed Markov decision process *is a tuple* $\mathsf{TMDP} = (S, \bar{s}, \Sigma, \mathbb{T}, Steps)$ *where:*

– *S is a set of* states*, including an* initial state $\bar{s} \in S$*;*

– *$Steps \subseteq S \times (\Sigma \cup \mathbb{T}) \times \mathsf{Dist}(S)$ is a* probabilistic transition relation*, such that, if $(s, a, \mu) \in Steps$ and $a \in \mathbb{T}$, then μ is a point distribution.*

A *probabilistic transition* $s \xrightarrow{a,\mu} s'$ is performed from a state $s \in S$ by first nondeterministically selecting a pair (a, μ) such that $(s, a, \mu) \in Steps$, and then by making a probabilistic choice of target state s' according to the distribution μ, such that $\mu(s'){>}0$. If $a \in \Sigma$, then the transition is interpreted as corresponding to the execution of an event, whereas if $a \in \mathbb{T}$, the transition is interpreted as corresponding to the elapse of time, the duration of which is a. We require that only event-distribution pairs can be probabilistic; that is, duration-distribution pairs always use a point distribution.

We now proceed to define the formal semantics of probabilistic timed automata. The definition is parametrised by both a time domain $\mathbb{T}$ and a *time increment operator* $\oplus$. A time increment operator is a binary operator which takes a clock valuation $v \in \mathbb{T}^{\mathcal{X}}$ and a time duration $t \in \mathbb{T}$, and returns a clock valuation $v{\oplus}t \in \mathbb{T}^{\mathcal{X}}$ which represents, intuitively, the clock valuation obtained from v after t time units have elapsed. The standard choice of time increment operator $\oplus$ corresponds to addition $+$, where $v{+}t$ is defined as in Section 8.2.1. The semantics of probabilistic timed automata is typically described for the case in which $\mathbb{T} = \mathbb{R}$ and $\oplus$ is $+$. We also consider in Section 8.3.4 an integer semantics in which $\mathbb{T} = \mathbb{N}$ and $\oplus$ is no longer standard addition.

DEFINITION 8.6.– *Let* $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$ *be a probabilistic timed automaton. The* semantics of PTA with respect to time domain $\mathbb{T}$ and time increment $\oplus$ *is the timed Markov decision process* $[\![\mathsf{PTA}]\!]_{\mathbb{T}}^{\oplus} = (S, \bar{s}, \Sigma, \mathbb{T}, Steps)$ *such that:*

– *$S \subseteq L \times \mathbb{T}^{\mathcal{X}}$ where $(l, v) \in S$ if and only if $v \triangleleft inv(l)$;*

– *$\bar{s} = (\bar{l}, \mathbf{0})$;*

– *$((l, v), a, \mu) \in Steps$ if and only if one of the following conditions holds:*

time transitions. *$a{=}t \in \mathbb{T}$ and $\mu = \mu_{(l,v\oplus t)}$ such that $v{\oplus}t' \triangleleft inv(l)$ for all $t' \in [0, t] \cap \mathbb{T}$;*

discrete transitions. *$a{=}\sigma \in \Sigma$ and there exists $(l, g, \sigma, p) \in prob$ such that $v \triangleleft g$ and for any $(l', v') \in S$:*

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

The summation in the definition of discrete transitions is required for the cases in which multiple clock resets result in the same target state.

8.2.5. *Probabilistic reachability and invariance*

In this section, we introduce *probabilistic reachability* and *invariance*, which are standard performance measures for probabilistic systems. Probabilistic reachability refers to the probability with which a certain set of states is reached (for example, states which correspond to the achievement of some goal or error states). On the other hand, probabilistic invariance refers to the probability of remaining in a certain set of states. For algorithms for the computation of such measures for finite state probabilistic systems see, for example [BIA 95, COU 98].

In order to introduce these measures, we first define more precisely the notion of the behaviour of timed Markov decision processes. The behaviour of a timed Markov decision process can be represented in two ways: using paths and adversaries. Formally, a path of a timed Markov decision process is a non-empty finite or infinite sequence of probabilistic transitions

$$\omega = s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} s_2 \xrightarrow{a_2,\mu_2} \cdots .$$

We denote by $\omega(i)$ the $(i{+}1)$th state of ω, by $last(\omega)$ the last state of ω if ω is finite and by $step(\omega, i)$ the event or duration associated with the $(i{+}1)$-th transition (that is, $step(\omega, i) = a_i$). By abuse of notation, we say that a single state s is a path of length 0. The set of infinite paths starting in state s is denoted by $Path(s)$. For any infinite path $\omega = s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} \cdots$, the accumulated duration up to the $(n{+}1)$-th state of ω is defined by:

$$dur(\omega, n{+}1) \stackrel{\text{def}}{=} \sum \{\!| a_i \mid 0 \leqslant i \leqslant n \wedge a_i \in \mathbb{T} |\!\} .$$

In contrast to a path, an adversary (or scheduler) represents a particular resolution of nondeterminism *only*. More precisely, an adversary is a function which chooses an outgoing distribution in the last state of a path. Formally, we have the following definition.

DEFINITION 8.7.– *Let* $\mathsf{TMDP} = (S, \bar{s}, \Sigma, \mathbb{T}, Steps)$ *be a timed Markov decision process. An* adversary A *of* TMDP *is a function mapping every finite path* ω *of* TMDP *to a pair* (a, μ) *such that* $(last(\omega), a, \mu)$ *is an element of Steps. For any state* $s \in S$*, let* $Path^A(s)$ *denote the subset of* $Path(s)$ *which correspond to* A*.*

The behaviour of a timed Markov decision process $\mathsf{TMDP} = (S, \bar{s}, \Sigma, \mathbb{T}, Steps)$ under a given adversary A is purely probabilistic. Using the measure construction of [KEM 76], for a state $s \in S$ and adversary A we can define a probability measure $Prob^A_s$ over the set of paths $Path^A(s)$. For further details see Chapter 9 or [RUT 04].

Models of real-time systems can contain unrealisable behaviours in which time does not exceed a certain bound. Generally, such behaviours are disregarded during the analysis of the model. In the following, we consider only time-divergent adversaries, which are adversaries that guarantee the divergence of time with probability 1. Formally, we say that an infinite path ω is *divergent* if, for any $t \in \mathbb{R}$, there exists $j \in \mathbb{N}$ such that $dur(\omega, j) > t$. An adversary A of a timed Markov decision process TMDP is *divergent* if and only if, for each state s of TMDP, the probability $Prob_s^A$ assigned to the divergent paths of $Path^A(s)$ is 1. Let Adv_{TMDP} be the set of divergent adversaries of TMDP.

We now define probabilistic reachability and invariance at the level of timed Markov decision processes. For a timed Markov decision process $\mathsf{TMDP} = (S, \bar{s}, \Sigma, \mathbb{T}, Steps)$, state $s \in S$, sets $T, I \subseteq S$ of target states and invariant states, and adversary $A \in Adv_{\mathsf{TMDP}}$, let:

$$\begin{aligned} p^A_{\mathsf{TMDP}}(s, \diamond\, T) &\overset{\text{def}}{=} Prob_s^A\{\omega \in Path^A(s) \mid \exists i \in \mathbb{N} \,.\, \omega(i) \in T\} \\ p^A_{\mathsf{TMDP}}(s, \Box\, I) &\overset{\text{def}}{=} Prob_s^A\{\omega \in Path^A(s) \mid \forall i \in \mathbb{N} \,.\, \omega(i) \in I\}\,. \end{aligned}$$

DEFINITION 8.8.– *Let* $\mathsf{TMDP} = (S, \bar{s}, \Sigma, \mathbb{T}, Steps)$ *be a timed Markov decision process. The* maximum and minimum reachability probabilities *of, from a state* $s \in S$, *reaching the set of target states* T *are defined as follows:*

$$\begin{aligned} p^{\max}_{\mathsf{TMDP}}(s, \diamond\, T) &= \sup_{A \in Adv_{\mathsf{TMDP}}} p^A_{\mathsf{TMDP}}(s, \diamond\, T) \\ p^{\min}_{\mathsf{TMDP}}(s, \diamond\, T) &= \inf_{A \in Adv_{\mathsf{TMDP}}} p^A_{\mathsf{TMDP}}(s, \diamond\, T)\,. \end{aligned}$$

The maximum and minimum invariance probabilities *of, from a state* $s \in S$, *remaining in the set of states* I *are defined as follows:*

$$\begin{aligned} p^{\max}_{\mathsf{TMDP}}(s, \Box\, I) &= \sup_{A \in Adv_{\mathsf{TMDP}}} p^A_{\mathsf{TMDP}}(s, \Box\, I) \\ p^{\min}_{\mathsf{TMDP}}(s, \Box\, I) &= \inf_{A \in Adv_{\mathsf{TMDP}}} p^A_{\mathsf{TMDP}}(s, \Box\, I)\,. \end{aligned}$$

The minimum and maximum probabilities of reachability and invariance can be considered as dual properties. More precisely, for any state $s \in S$, sets of states T and I such that $T = S \setminus I$:

$$p^{\max}_{\mathsf{TMDP}}(s, \diamond\, T) = 1 - p^{\min}_{\mathsf{TMDP}}(s, \Box\, I) \;\text{ and }\; p^{\min}_{\mathsf{TMDP}}(s, \diamond\, T) = 1 - p^{\max}_{\mathsf{TMDP}}(s, \Box\, I)\,.$$

Because of this relationship, for the remainder of this chapter we focus on methods for computing probabilistic reachability properties.

In the context of the analysis of probabilistic timed automata, the set of target states is often expressed in terms of a set of target locations $T_L \subseteq L$. More precisely, given a set of target locations T_L, we take $T = \{(l, v) \in S \mid l \in T_L\}$ to be the set of target states and, for simplicity, write $p^{\max}_{[\![\mathsf{PTA}]\!]^{\oplus}_{\mathbb{T}}}(s, \Diamond\, T_L)$ and $p^{\min}_{[\![\mathsf{PTA}]\!]^{\oplus}_{\mathbb{T}}}(s, \Diamond\, T_L)$.

More generally, target states can be expressed by a set of location-zone pairs, rather than a set of locations. Location-zone pairs can be used, for example, to express that a target set of locations must be reached before or after a certain deadline. Using the construction of [KWI 02a], such reachability problems can be reduced to those referring to locations only by modifying syntactically the probabilistic timed automaton of interest.

The following types of properties can be expressed using probabilistic reachability:

Basic probabilistic reachability: the system reaches a certain set of states with a given maximum or minimum probability. For example, "with probability at least 0.999, a data packet is correctly delivered".

Probabilistic time-bounded reachability: the system reaches a certain set of states within a certain time deadline and probability threshold. For example, "with probability 0.01 or less, a data packet is lost within 5 time units".

Probabilistic cost-bounded reachability: the system reaches a certain set of states within a certain cost and probability bound. For example, "with probability 0.75 or greater, a data packet is correctly delivered with at most 4 retransmissions".

Invariance: the system does not leave a certain set of states with a given probability. For example, "with probability 0.875 or greater, the system never aborts".

Bounded response: the system inevitably reaches a certain set of states within a certain time deadline with a given probability. For example, "with probability 0.99 or greater, a data packet will always be delivered within 5 time units".

8.3. Model checking for probabilistic timed automata

In this section, we consider a number of automatic verification methods for probabilistic timed automata. The underlying semantics of a probabilistic timed automaton PTA is generally presented with respect to the time domain $\mathbb{R}$, and hence the resulting semantic timed Markov decision process $[\![\mathsf{PTA}]\!]^{+}_{\mathbb{R}}$ generally has an uncountable number of states and transitions. Therefore, the methods that we present are based on the construction of finite-state Markov decision processes, which are defined so that their analysis can be used to infer reachability probabilities and other performance measures of probabilistic timed automata. We present a different finite-state construction in each of the following four sections.

8.3.1. *The region graph*

Our first method is inspired by the classical region-graph technique for timed automata [ALU 94], in which a finite-state transition system is constructed from a timed automaton. Proofs of the key results can be found in [KWI 02a] where it is shown that the region graph can be used to model check the probabilistic timed temporal logic PTCTL which subsumes the checking of reachability properties.

The construction is based on a finite equivalence relation on the infinite set $\mathbb{R}^{\mathcal{X}}$ of clock valuations. We apply the same equivalence to the set of clock valuations of probabilistic timed automata in order to obtain a probabilistic region graph, which contains sufficient information for the calculation of reachability probabilities. In this section, we assume that probabilistic timed automata are *structurally non-Zeno* [TRI 05]: a probabilistic timed automaton is structurally non-Zeno if, for every sequence X_0, $(l_0, g_0, \sigma_0, p_0), X_1, (l_1, g_1, \sigma, p_1), \ldots, X_n, (l_n, g_n, \sigma, p_n)$, such that $p_i(X_{i+1}, l_{i+1}){>}0$ for $0{\leqslant}i{<}n$, and $p_n(X_0, l_0){>}0$, there exists a clock $x \in \mathcal{X}$ and $0{\leqslant}i, j{\leqslant}n$ such that $x \in X_i$ and $g_j \Rightarrow (x{\geqslant}1)$ (that is, g_j contains a conjunct of the form $x{\geqslant}c$ or $x{>}c$ for some $c{\geqslant}1$).

We recall the definition of the equivalence relation on clock valuations used to define the region graph of (non-probabilistic) timed automata [ALU 94]. For a real number $q \in \mathbb{R}$, let $\lfloor q \rfloor$ be the integer part of q.

DEFINITION 8.9.– *Let $c \in \mathbb{N}$ be a natural number and let $\mathcal{X}$ be a set of clocks. The* clock equivalence (with respect to c) *over $\mathcal{X}$ is defined as the relation $\equiv_c \subseteq \mathbb{R}^{\mathcal{X}} \times \mathbb{R}^{\mathcal{X}}$ over clock valuations, where $v \equiv_c v'$ if and only if:*

– for each clock $x \in \mathcal{X}$, either both $v(x){>}c$ and $v(x'){>}c$ or:

- $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ (clock values have the same integer part)

- $v(x){-}\lfloor v(x) \rfloor = 0$ if and only if $v(x'){-}\lfloor v(x') \rfloor = 0$ (the fractional parts of the clocks' values are either all zero or all positive);

– for each pair of clocks $x, y \in \mathcal{X}$, either $\lfloor v(x){-}v(y) \rfloor = \lfloor v'(x){-}v'(y) \rfloor$ (the integer part of the difference between clocks values is the same) or both $v(x){-}v(y) > c$ and $v'(x){-}v'(y) > c$.

Note that the final point of the definition implies that the ordering on the fractional parts of the clock values of x and y is the same, unless the difference between the clocks is above c. For example, in the case of $\mathcal{X} = \{x_1, x_2, x_3\}$, the clock valuations v and v', where $v(x_1){=}0.1$, $v(x_2){=}0.25$ and $v(x_3){=}0.8$, and $v'(x_1){=}0.5$, $v'(x_2){=}0.9$ and $v'(x_3){=}0.95$, are clock equivalent, but v'', where $v''(x_1){=}0.3$, $v''(x_2){=}0.75$ and $v''(x_3){=}0.6$, is not clock equivalent to v and v'. An example of the partition that clock equivalence induces on a space of valuations of two clocks x_1 and x_2 is shown in Figure 8.2, where c equals 2; each vertical, horizontal or diagonal line segment, open area, and point of intersection of lines is a distinct clock equivalence class. Clock

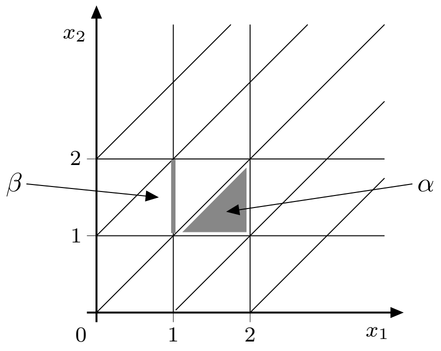


Figure 8.2. *The partition induced by clock equivalence with c=2*

equivalence classes can be interpreted as zones: for example, the clock equivalence class α denoted in Figure 8.2 corresponds to the zone $(1{<}x{<}2) \wedge (1{<}y{<}2) \wedge (x{>}y)$, while the clock equivalence class β corresponds to the zone $(x{=}1) \wedge (1{<}y{<}2)$.

We define the *time successor* of a clock equivalence class α to be the first distinct clock equivalence class reached from α by letting time pass. Formally, the time successor β of α is the clock equivalence class for which $\alpha \neq \beta$ and, for all $v \in \alpha$, there exists $t \in \mathbb{R}$ such that $v + t \in \beta$ and $v + t' \in \alpha \cup \beta$ for all $0 \leqslant t' \leqslant t$. A clock equivalence class α is *unbounded* if, for all $v \in \alpha$ and $t \in \mathbb{R}$, we have $v + t \in \alpha$; in the case of an unbounded clock equivalence class α, we let the time successor of α be α itself. Because clock equivalence classes can be regarded as zones, for a clock equivalence class α and clock set $X \subseteq \mathcal{X}$, we use the notation $\alpha[X := 0]$ to denote the set $\{v[X := 0] \mid v \in \alpha\}$, as for zones. For a zone $\zeta \in \mathit{Zones}(\mathcal{X})$, we write $\alpha \triangleleft \zeta$ to denote that all clock valuations in α satisfy the zone ζ.

For the remainder of this section, assume that the probabilistic timed automaton $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, \mathit{inv}, \mathit{prob})$ is fixed. Let $c^{\mathsf{PTA}}_{\max}$ be the maximum constant against which any clock is compared in the invariant conditions and guards of PTA. A *region* (l, α) of PTA is a pair comprising a location $l \in L$ and a class α of the clock equivalence $\equiv_{c^{\mathsf{PTA}}_{\max}}$. Since the set of clock equivalence classes is finite, and therefore the number of regions is finite, we can define a finite-state Markov decision process, the states of which are regions, and the transitions of which are derived from the time constraints and probabilistic edges of the probabilistic timed automaton.

DEFINITION 8.10.– *The* region graph of $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, \mathit{inv}, \mathit{prob})$ *is a finite-state Markov decision process* $\mathsf{MDP} = (R, \bar{r}, \mathit{Steps}_R)$*, where R is the set of all regions of* PTA*, the initial region $\bar{r}$ is $(\bar{l}, \{\mathbf{0}\})$, and the probabilistic transition relation $\mathit{Steps}_R \subseteq R \times \mathsf{Dist}(R)$ is the smallest relation such that $((l, \alpha), \rho) \in \mathit{Steps}_R$ if either of the following conditions hold:*

time transitions: $\rho = \mu_{(l,\beta)}$, *where* β *is such that* $\beta \lhd inv(l)$, *and* β *is the time successor of* α*;*

discrete transitions: *there exists a probabilistic edge* $(l, g, \sigma, p) \in prob$ *such that* $\alpha \lhd g$ *and, for any region* $(l', \beta) \in R$, *we have:*

$$\rho(l', \beta) = \sum_{\substack{X \subseteq \mathcal{X} \,\&\\ \beta = \alpha[X:=0]}} p(X, l') \ .$$

The following proposition states the correctness of the region graph for calculating minimum and maximum reachability probabilities.

PROPOSITION 8.1.– *For any probabilistic timed automaton* PTA *and corresponding timed Markov decision process* $[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+} = (S, \bar{s}, \Sigma, \mathbb{R}, Steps)$, *if* $\mathsf{MDP} = (R, \bar{r}, Steps_R)$ *is the region graph of* PTA *and* T_R *is a set of target regions, then for any* $s \in S$:

$$p^{\max}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+}}(s, \diamond\, T) = p^{\max}_{\mathsf{MDP}}(r_s, \diamond\, T_R) \ \ \textit{and} \ \ p^{\min}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+}}(s, \diamond\, T) = p^{\min}_{\mathsf{MDP}}(r_s, \diamond\, T_R)$$

where if $s = (l, v)$, *then* $r_s = (l, \alpha) \in R$ *such that* $v \in \alpha$, *and* $T = \{(l, v) \,|\, r_{(l,v)} \in T_R\}$.

EXAMPLE 8.2.– *We now consider the* PTA *in Example 8.1 (see Figure 8.1) and use the region graph and Proposition 8.1 to calculate the minimum and maximum probability of reaching the location* sr *within* 6 *time units. Following the approach of [KWI 02a], we add an additional clock* z *to the automaton and compute the probability of reaching the target set of regions* $T_R = \{(\mathsf{sr}, \alpha) \,|\, \alpha \lhd (z{<}6)\}$. *The Markov decision process representing the region graph has 115 states and 125 transitions. In Figure 8.3 we present a fragment of this Markov decision process. Transitions without probability labels correspond to probability 1. Computing the probability of reaching the target set* T_R *on the region graph, we find that the minimum and maximum probabilities of reaching the location* sr *within* 6 *time units equal 0.995 and 0.99525 respectively.*

8.3.2. *Forwards symbolic approach*

In this section we consider an approach to approximating the maximum reachability probability based on a forward symbolic analysis. This approach allows us to compute an upper bound on the maximum probability of reaching a set of target locations. Before introducing the algorithm, we present a number concepts for representing and manipulating zones and state sets of a probabilistic timed automaton.

In this section and in Section 8.3.3, we assume that probabilistic timed automata allow time to diverge with probability 1 from each state. That is, for each probabilistic timed automaton PTA, we assume that there exists a divergent adversary of $[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+}$.

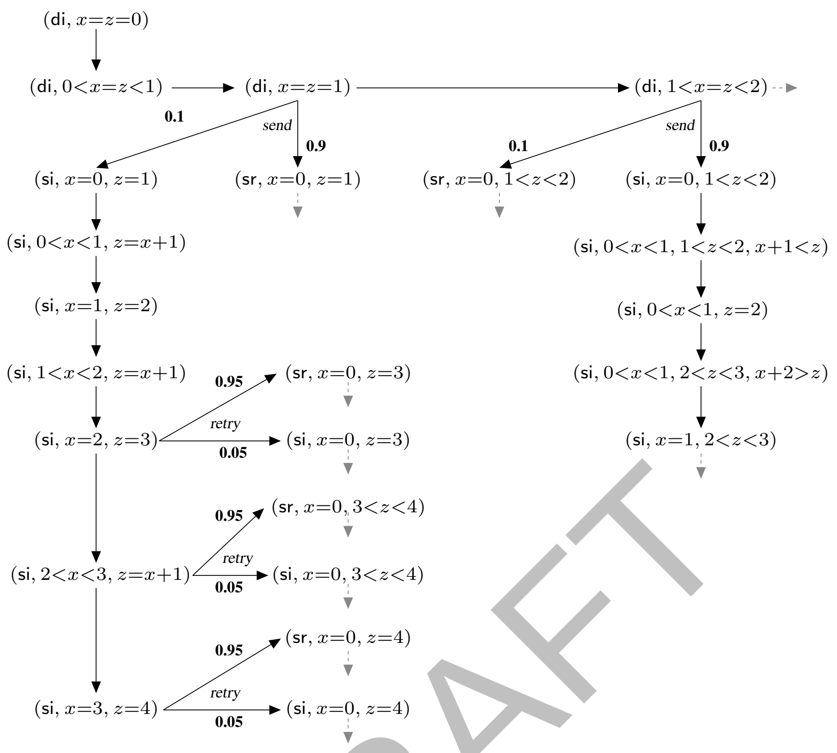


Figure 8.3. *Fragment of the region graph model of Example 8.1*

Proofs of the key results presented in this section are available in [KWI 02a].

8.3.2.1. *Symbolic state operations*

In order to represent symbolically the state sets computed during the analysis, we use, both in this section and Section 8.3.3, the concept of *symbolic state*. A symbolic state is a pair (l, ζ) comprising a location and a zone over the clocks of the probabilistic timed automaton under study. The set of states corresponding to a symbolic state (l, ζ) is $\{(l, v) \mid v \triangleleft \zeta\}$, while the state set corresponding to a set of symbolic states is the union of those corresponding to each individual symbolic state. Symbolic states are denoted by $\mathtt{u}, \mathtt{v}, \mathtt{z}, \ldots$ and sets of symbolic states by $\mathtt{U}, \mathtt{V}, \mathtt{Z}, \ldots$

Consider a probabilistic timed automaton $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$. Our aim of forward exploration through the state space of PTA requires operations to return the successor states of all of the states in a particular set (where the set is represented as a symbolic state). More precisely, we introduce a *discrete successor* operation dpost which, given an edge $e = (l, g, \sigma, p, X, l')$ of PTA and a symbolic state (l, ζ), returns all of the states obtained by traversing e from a state in (l, ζ). Similarly, the *time successor* operation tpost computes, for the symbolic state (l, ζ), the set of states which can be reached from a state in (l, ζ) by letting time elapse. These two operations are then composed to define a generalised successor operation post. For a given symbolic state (l, ζ) and an edge $e = (l, g, \sigma, p, X, l') \in \mathsf{edges}$, the post operation returns the set of states that can be obtained from (l, ζ) by traversing the edge e and then letting time elapse. Note that we also parameterise post by an integer $c \in \mathbb{N}$, and only compute the c-closure of zones obtained by the time successor operation; this is to ensure the termination of the forward algorithm. For a symbolic state (l, ζ), edge $e = (l, g, \sigma, p, X, l')$ and constant $c \in \mathbb{N}$:

$$\begin{aligned}
\mathsf{tpost}(l, \zeta) &\overset{\text{def}}{=} (l, \nearrow_{inv(l)} \zeta) \\
\mathsf{dpost}(e, (l, \zeta)) &\overset{\text{def}}{=} (l', ((\zeta \wedge g)[X := 0]) \wedge inv(l')) \\
\mathsf{close}(l, \zeta, c) &\overset{\text{def}}{=} (l, \mathsf{close}(\zeta, c)) \\
\mathsf{post}[e, c](l, \zeta) &\overset{\text{def}}{=} \mathsf{close}(\mathsf{tpost}(\mathsf{dpost}(e, (l, \zeta))), c)\,.
\end{aligned}$$

8.3.2.2. *Computing maximum reachability probabilities*

An algorithm for generating a finite representation of the state space of a probabilistic timed automaton PTA for a given set of target set T_L of locations is presented in Figure 8.4. Recall from Section 8.3.1 that $c_{\max}^{\mathsf{PTA}}$ denotes the maximum constant against which any clock is compared in the invariant conditions and guards of PTA. The algorithm returns a Markov decision process $\mathsf{MDP} = (\mathsf{Z}, \bar{\mathsf{z}}, Steps)$, called the *forwards zone graph*, and a set of target states $Reached \subseteq \mathsf{Z}$.

The algorithm consists of two distinct computation steps: firstly, the generation of symbolic states Z that encode states of PTA reachable from the initial state with non-zero probability, and secondly, construction of the forwards zone graph generated from Z and the probabilistic edges of PTA. As in the case of similar algorithms in the non-probabilistic context [DAW 96, LAR 97], the algorithm searches forward through a reachable portion of the state space of the system by iterating the transition relation a finite number of times. Any symbolic state (l, ζ) such that $l \in T_L$ (i.e. where the target has been reached) is not explored, but instead added to the set $Reached$. Since, for a given $c \in \mathbb{N}$, there are a finite number of c-closed zones [DAW 98, TRI 98] and the number of locations is finite, the forwards zone graph generated by Figure 8.4 is guaranteed to be finite. This also implies the termination of the algorithm.

Finally, we can then obtain an upper bound on the maximum probability of reaching the target set T_L of locations as the maximum probability of reaching the target set $Reached$ in the forwards zone graph. If $Reached = \emptyset$, then the forward search through the reachable state space has found that no location in T_L is reachable with

```
algorithm ForwReach($T_L$)

1.  $\mathsf{Z} := \emptyset$
2.  $\mathit{Reached} := \emptyset$
3.  $\bar{\mathsf{z}} := \mathsf{close}(\mathsf{tpost}(\bar{l}, \mathbf{0}), c^{\mathsf{PTA}}_{\max})$
4.  $\mathit{Fringe} := \{\bar{\mathsf{z}}\}$
5.  repeat
6.    choose $(l, \zeta) \in \mathit{Fringe}$
7.    $\mathit{Fringe} := \mathit{Fringe} \setminus \{(l, \zeta)\}$
8.    for each $(l, g, \sigma, p, l', X) \in \mathsf{edges}$ do
9.      let $(l', \zeta') := \mathsf{post}[(l, g, \sigma, p, l', X), c^{\mathsf{PTA}}_{\max}](l, \zeta)$
10.     if $\zeta' \neq \emptyset \wedge (l', \zeta') \notin \mathsf{Z} \wedge l' \in T_L$ then
11.       $\mathit{Reached} := \mathit{Reached} \cup \{(l', \zeta')\}$
12.     else if $\zeta' \neq \emptyset \wedge (l', \zeta') \notin \mathsf{Z}$
13.       $\mathit{Fringe} := \mathit{Fringe} \cup \{(l', \zeta')\}$
14.     end if
15.   end for each
16.   $\mathsf{Z} := \mathsf{Z} \cup \{(l, \zeta)\}$
17. until $\mathit{Fringe} = \emptyset$
18. construct $\mathsf{MDP} = (\mathsf{Z}, \bar{\mathsf{z}}, \mathit{Steps}_{\mathsf{Z}})$ where $((l, \zeta), \rho) \in \mathit{Steps}_{\mathsf{Z}}$ if and only if
      there exists $(l, g, \sigma, p) \in \mathit{prob}$ such that
        – $\zeta \cap g \neq \emptyset$
        – for any $(l', \zeta') \in \mathsf{Z}$: $\rho(l', \zeta')$ equals
            $\sum\{\!| p(X, l') \mid \mathsf{post}[(l, g, \sigma, p, l', X), c^{\mathsf{PTA}}_{\max}](l, \zeta) = (l', \zeta') |\!\}$
19. return $(\mathsf{MDP}, \mathit{Reached})$
```

Figure 8.4. *Algorithm* ForwReach(T_L) *for* $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, \mathit{inv}, \mathit{prob})$

positive probability, and therefore we conclude that the maximum probability is 0. The correctness of the algorithm follows from the following proposition.

PROPOSITION 8.2.– *Let* $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, \mathit{inv}, \mathit{prob})$ *be a probabilistic timed automaton and* T_L *be a set of target locations. If the algorithm* ForwReach(T_L) *returns* $\mathsf{MDP} = (\mathsf{Z}, \bar{\mathsf{z}}, \mathit{Steps})$ *and Reached, then:*

$$p^{\max}_{[\![\mathsf{PTA}]\!]^+_{\mathbb{R}}}((\bar{l}, \mathbf{0}), \Diamond\, T_L) \leqslant p^{\max}_{\mathsf{MDP}}(\bar{\mathsf{z}}, \Diamond\, \mathit{Reached})\ .$$

Unfortunately, the probability obtained via this approach may be greater than the probability of reaching the target set via *any* adversary of the probabilistic timed automaton. This is illustrated by the example below.

EXAMPLE 8.3.– *Consider the probabilistic timed automaton presented in Figure 8.5 and suppose that target set of interest consists of the single location l. There are only two classes of adversary with a non-zero probability of reaching location l: one in*

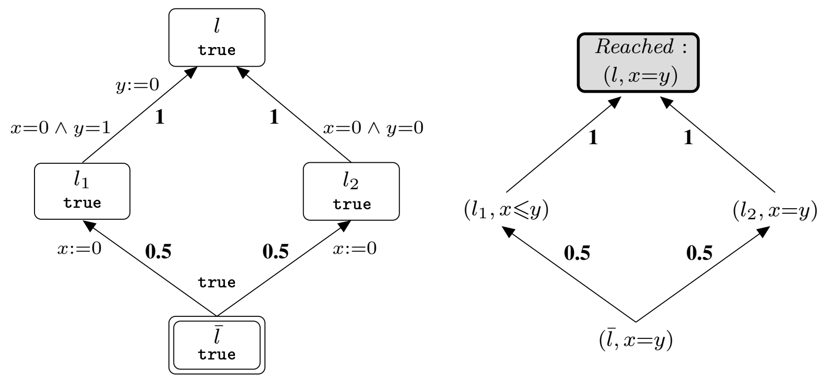


Figure 8.5. *Example demonstrating that* ForwReach *yields only upper bounds*

which the initial state is left when $x{=}y{=}0$, and the other when the initial state is left when $x{=}y{=}1$. In the former case, if the left-hand edge to location l_1 is taken, then the outgoing edge of l_1 can never be taken, and so we must remain in this location; however, if the right-hand edge to l_2 is taken, then the outgoing edge to l can be selected immediately. The case for the selection of the transition of $\bar{l}$ when $x{=}y{=}1$ is symmetric. Therefore, for each of the two classes of adversaries, the probability of reaching the target set $\{l\}$ is 0.5. However, for the forwards zone graph and set $Reached$ generated by ForwReach$(\{l\})$*, as shown in Figure 8.5, the corresponding reachability probability is 1.*

Recall that invariance properties can be stated in terms of reachability properties. The fact that we obtain upper bounds on maximum reachability probabilities implies that we obtain lower bounds on the minimum probability of invariance, and hence the forward reachability method is particularly appropriate for establishing the satisfaction of probabilistic, real-time invariance properties.

One important difference between the algorithm in Figure 8.4 and analogous algorithms in the non-probabilistic, real-time literature, is the fact that the *on-the-fly* property of the latter algorithms is compromised in our context. This property refers to the fact that, if a symbolic state which reaches the target set is computed, then the algorithm can terminate immediately with a "YES" answer to the reachability problem. Such a strategy is insufficient for probabilistic timed automata. For example, consider the case in which we have found a path of symbolic states reaching the target set T_L, and which corresponds to the probability λ. Then it may be possible to find another path to T_L, thus increasing the probability of reaching this target set.

EXAMPLE 8.4.– *Returning to Example 8.1 (see Figure 8.1), we now calculate the maximum probability of reaching the location* sr *within* 6 *time units. Following the*

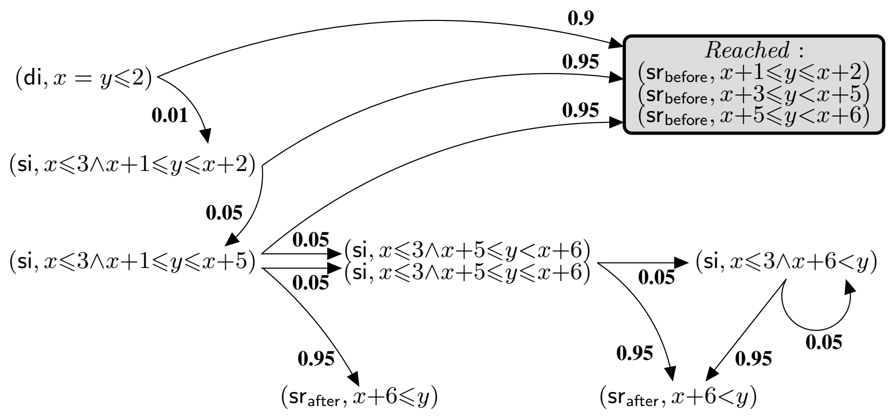

$Reached$:
$(\mathsf{sr}_{\mathsf{before}}, x{+}1{\leqslant}y{\leqslant}x{+}2)$
$(\mathsf{sr}_{\mathsf{before}}, x{+}3{\leqslant}y{<}x{+}5)$
$(\mathsf{sr}_{\mathsf{before}}, x{+}5{\leqslant}y{<}x{+}6)$
$(\mathsf{di}, x = y{\leqslant}2)$
$(\mathsf{si}, x{\leqslant}3{\wedge}x{+}1{\leqslant}y{\leqslant}x{+}2)$
$(\mathsf{si}, x{\leqslant}3{\wedge}x{+}1{\leqslant}y{\leqslant}x{+}5)$
$(\mathsf{si}, x{\leqslant}3{\wedge}x{+}5{\leqslant}y{<}x{+}6)$
$(\mathsf{si}, x{\leqslant}3{\wedge}x{+}5{\leqslant}y{\leqslant}x{+}6)$
$(\mathsf{si}, x{\leqslant}3{\wedge}x{+}6{<}y)$
$(\mathsf{sr}_{\mathsf{after}}, x{+}6{\leqslant}y)$
$(\mathsf{sr}_{\mathsf{after}}, x{+}6{<}y)$
0.9
0.95
0.95
0.01
0.05
0.05
0.05
0.05
0.95
0.95
0.95
0.05

Figure 8.6. *Markov decision process generated by* $\mathsf{ForwReach}(\{\mathsf{sr}_{\mathsf{before}}\})$

approach of [KWI 02a], we add a distinct clock z to the automaton, and split the location sr into locations $\mathsf{sr}_{\mathsf{before}}$ and $\mathsf{sr}_{\mathsf{after}}$. Furthermore, we replace each probabilistic edge which has sr as a target location with two probabilistic edges where sr is replaced with $\mathsf{sr}_{\mathsf{before}}$ or $\mathsf{sr}_{\mathsf{after}}$, and where $z{<}6$ or $z{\geqslant}6$, respectively, is added to the guard. Applying $\mathsf{ForwReach}(\{\mathsf{sr}_{\mathsf{before}}\})$ to this automaton returns the Markov decision process given in Figure 8.6. From Proposition 8.2, it follows that the maximum probability of reaching the location sr within 6 time units is bounded above by 0.99525, corresponding to the maximum probability of reaching Reached from $\bar{\mathsf{z}} = (\mathsf{di}, x{=}y{\leqslant}2)$ in the Markov decision process given in Figure 8.6. In fact, in this case, the computed bound equals the actual probability. Intuitively, this is because there is no conflict between letting time pass on different edges: the maximum probability is obtained by always taking a discrete transition as soon as it is available.

8.3.3. *Backwards symbolic approach*

In this section we consider an approach for computing both the maximum and minimum reachability probability based on a backwards symbolic analysis. Unlike the forwards approach in the previous section, this technique yields exact results. Furthermore, as in the region graph approach, it computes the probabilities for all states of the probabilistic timed automaton rather than for a fixed initial state.

Proofs of the key results, and specialised algorithms for *qualitative* properties (finding states that have probability 0 or 1 of reaching the target set), for which verification can be performed through an analysis of the underlying graph, are available in [KWI 07]. In fact, [KWI 07] presents algorithms for model checking for the probabilistic timed temporal logic PTCTL [KWI 02a], which subsumes reachability properties.

8.3.3.1. *Symbolic state operations*

Consider a probabilistic timed automaton $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$. The backwards exploration through the state space of PTA requires operations to return the predecessors of a set of symbolic states. More precisely, we introduce a *discrete predecessor* operation dpre which, given an edge e of PTA and a set of symbolic states $\mathtt{V}$, returns the set of symbolic states that encodes the set of states which, when edge e is traversed, belong to the set encoded by $\mathtt{V}$. The *time predecessor* operation $\mathsf{tpre}_{\mathtt{U}}$ is parametrised by a set of symbolic states $\mathtt{U}$, and computes, for a set of symbolic states $\mathtt{V}$, the set of symbolic states encoding states that can reach $\mathtt{V}$ by letting time elapse and remaining in $\mathtt{U}$ at all intermediate times.

Formally, we extended the *time predecessor* and *discrete predecessor* functions tpre and dpre of [HEN 94, TRI 98] to probabilistic timed automata as follows. For any sets of symbolic states $\mathtt{U}, \mathtt{V}$ and edge $e = (l, g, \sigma, p, X, l')$:

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

where for any set $\mathtt{U}$ of symbolic states $\zeta^l_{\mathtt{U}} \overset{\text{def}}{=} \bigvee\{\zeta \mid (l, \zeta) \in \mathtt{U}\}$; that is, $\zeta^l_{\mathtt{U}}$ is the zone such that $v \triangleleft \zeta^l_{\mathtt{U}}$ if and only if $(l, v) \in \mathtt{u}$ for some $\mathtt{u} \in \mathtt{U}$.

We also extend clock reset, conjunction and disjunction operations to sets of symbolic states as follows. For any clock x and sets of symbolic states $\mathtt{U}, \mathtt{V}$:

$$\begin{aligned}
x.\mathtt{U} &\overset{\text{def}}{=} \left\{ (l, [\{x\}{:=}0]\zeta^l_{\mathtt{U}}) \;\middle|\; l \in L \right\} \\
\mathtt{U}\wedge\mathtt{V} &\overset{\text{def}}{=} \left\{ (l, \zeta^l_{\mathtt{U}} \wedge \zeta^l_{\mathtt{V}}) \;\middle|\; l \in L \right\} \\
\mathtt{U}\vee\mathtt{V} &\overset{\text{def}}{=} \left\{ (l, \zeta^l_{\mathtt{U}} \vee \zeta^l_{\mathtt{V}}) \;\middle|\; l \in L \right\}.
\end{aligned}$$

Finally, let $[\![\mathtt{false}]\!] \overset{\text{def}}{=} \emptyset$ and $[\![\mathtt{true}]\!] \overset{\text{def}}{=} \{(l, inv(l)) \mid l \in L\}$ be the sets of symbolic states representing the empty and full state sets respectively. In addition, for any $\zeta \in Zones(\mathcal{X})$, let $[\![\zeta]\!] = \{(l, \zeta \wedge inv(l)) \mid l \in L\}$.

8.3.3.2. *Probabilistic until*

The algorithm for computing minimum reachability probabilities (which will be described in Section 8.3.3.4) relies on the computation of (maximum) probabilities for two classes of properties: *probabilistic invariance* (see Section 8.2.5) and *probabilistic until*. Probabilistic until corresponds to the probability of reaching a certain set of target states while remaining in another set of states until the target set is reached. For a timed Markov decision process $\mathsf{TMDP} = (S, \bar{s}, \Sigma, \mathbb{T}, \mathit{Steps})$, state $s \in S$, sets $U, V \subseteq S$ of states, and adversary $A \in \mathit{Adv}_{\mathsf{TMDP}}$:

$$p^A_{\mathsf{TMDP}}(s, U \ \mathcal{U}\ V) \overset{\text{def}}{=} \mathit{Prob}^A_s\{\omega \in \mathit{Path}^A(\bar{s}) \mid \exists i \in \mathbb{N}\,.(\omega(i) \in V \wedge \forall j{<}i\,.\omega(j) \in U)\}\ .$$

Note that probabilistic reachability is a special case of probabilistic until:

$$p^A_{\mathsf{TMDP}}(s, \diamond\, V) = p^A_{\mathsf{TMDP}}(s, S\ \mathcal{U}\ V)\,.$$

DEFINITION 8.11.– *Let* $\mathsf{TMDP} = (S, \bar{s}, \Sigma, \mathbb{T}, \mathit{Steps})$ *be a timed Markov decision process. The* maximum and minimum until probabilities *of, from a state* $s \in S$*, reaching a set of states* V *while remaining in a set of states* U *up until this point are defined as follows:*

$$\begin{aligned} p^{\max}_{\mathsf{TMDP}}(s, U\ \mathcal{U}\ V) &= \sup_{A \in \mathit{Adv}_{\mathsf{TMDP}}} p^A_{\mathsf{TMDP}}(s, U\ \mathcal{U}\ V) \\ p^{\min}_{\mathsf{TMDP}}(s, U\ \mathcal{U}\ V) &= \inf_{A \in \mathit{Adv}_{\mathsf{TMDP}}} p^A_{\mathsf{TMDP}}(s, U\ \mathcal{U}\ V)\,. \end{aligned}$$

8.3.3.3. *Computing maximum reachability probabilities*

In this section we present methods for calculating maximum reachability probabilities. In fact, we present a more general method which can compute maximum until probabilities, as introduced in the previous section.

In the case of computing maximum until probabilities and, in particular maximum reachability probabilities, we use the algorithm MaxU given in Figure 8.7 and adapted from [KWI 01]. The algorithm iteratively applies time predecessor, discrete predecessor and conjunction operations on symbolic states until a fixpoint is reached. The key observation is that to preserve probabilistic branching we must take the conjunctions of symbolic states generated by edges from the same distribution. More precisely, we need to identify the state sets from which *multiple edges* within the support of the *same distribution* of the probabilistic timed automaton can be used to reach previously generated state sets. Upon termination of the fixpoint algorithm, the set of generated symbolic states is used to construct a finite-state Markov decision process which has sufficient information to compute the maximum probability of interest.

We now explain the algorithm $\mathsf{MaxU}(\mathtt{U}, \mathtt{V})$ in more detail. 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 $E_{(l,g,\sigma,p)}$ associated with each probabilistic edge $(l, g, \sigma, p) \in \mathit{prob}$. Lines

```
algorithm MaxU(U, V)

1.  $\mathtt{Z} := \mathsf{tpre}_{\mathtt{U}\vee\mathtt{V}}(\mathtt{V})$
2.  for $(l, g, \sigma, p) \in \mathit{prob}$
3.      $E_{(l,g,\sigma,p)} := \emptyset$
4.  end for
5.  repeat
6.      $\mathtt{Y} := \mathtt{Z}$
7.      for $\mathtt{y} \in \mathtt{Y} \wedge (l, g, \sigma, p) \in \mathit{prob} \wedge e = (l, g, \sigma, p, X, l') \in \mathsf{edges}(l, g, \sigma, p)$
8.          $\mathtt{z} := \mathtt{U} \wedge \mathsf{dpre}(e, \mathsf{tpre}_{\mathtt{U}\vee\mathtt{V}}(\mathtt{y}))$
9.          if $(\mathtt{z} \neq \emptyset) \wedge (\mathtt{z} \not\in \mathsf{tpre}_{\mathtt{U}\vee\mathtt{V}}(\mathtt{V}))$
10.             $\mathtt{Z} := \mathtt{Z} \cup \{\mathtt{z}\}$
11.             $E_{(l,g,\sigma,p)} := E_{(l,g,\sigma,p)} \cup \{(\mathtt{z}, (X, l'), \mathtt{y})\}$
12.             for $(\bar{\mathtt{z}}, (\bar{X}, \bar{l}'), \bar{\mathtt{y}}) \in E_{(l,g,\sigma,p)}$
13.                 if $(\mathtt{z} \wedge \bar{\mathtt{z}} \neq \emptyset) \wedge ((\bar{X}, \bar{l}') \neq (X, l')) \wedge (\mathtt{z} \wedge \bar{\mathtt{z}} \not\in \mathsf{tpre}_{\mathtt{U}\vee\mathtt{V}}(\mathtt{V}))$
14.                     $\mathtt{Z} := \mathtt{Z} \cup \{\mathtt{z} \wedge \bar{\mathtt{z}}\}$
15.                 end if
16.             end for
17.         end if
18.     end for
19. until $\mathtt{Z} = \mathtt{Y}$
20. construct $\mathsf{MDP} = (\mathtt{Z}, \mathit{Steps}_{\mathtt{Z}})$ where $(\mathtt{z}, \rho) \in \mathit{Steps}_{\mathtt{Z}}$ if and only if
        there exists $(l, g, \sigma, p) \in \mathit{prob}$ and $E \subseteq E_{(l,g,\sigma,p)}$ such that
            – $\mathtt{z} \in \{\mathtt{z}' \mid (\mathtt{z}', \mathtt{e}, \mathtt{z}'') \in E\}$
            – $(\mathtt{z}', \mathtt{e}, \mathtt{z}'') \in E \;\Rightarrow\; \mathtt{z}' \supseteq \mathtt{z}$
            – $(\mathtt{z}'_1, \mathtt{e}, \mathtt{z}') \neq (\mathtt{z}'_2, \mathtt{e}', \mathtt{z}'') \in E \;\Rightarrow\; \mathtt{e} \neq \mathtt{e}'$
            – $E$ is maximal
            – $\rho(\mathtt{z}') = \sum\{\!\mid p(X, l') \mid (\mathtt{z}, (X, l'), \mathtt{z}') \in E \mid\!\}\;\; \forall \mathtt{z}' \in \mathtt{Z}$
21. return MDP
```

Figure 8.7. *Algorithm* MaxU(U, V) *for* PTA $= (L, \bar{l}, \mathcal{X}, \Sigma, \mathit{inv}, \mathit{prob})$

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 12–16). The edges of the graph are partitioned into the sets $E_{(l,g,\sigma,p)}$ for $(l, g, \sigma, p) \in \mathit{prob}$, where any $(\mathtt{z}, (X, l'), \mathtt{z}') \in E_{(l,g,\sigma,p)}$ corresponds to a transition from any state in the symbolic state $\mathtt{z}$ to some state in the symbolic state $\mathtt{z}'$ when the outcome (X, l') of the probabilistic edge (l, g, σ, p) is chosen. The graph edges are added in line 11. Line 20 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 Markov decision process MDP. The states of MDP are the symbolic states generated by the previous steps of the algorithm, and the probabilistic transition relation of MDP is constructed by grouping the graph edges generated by the same probabilistic edge of the probabilistic timed automaton under study. The initial state of MDP is irrelevant, and therefore is omitted from the notation.

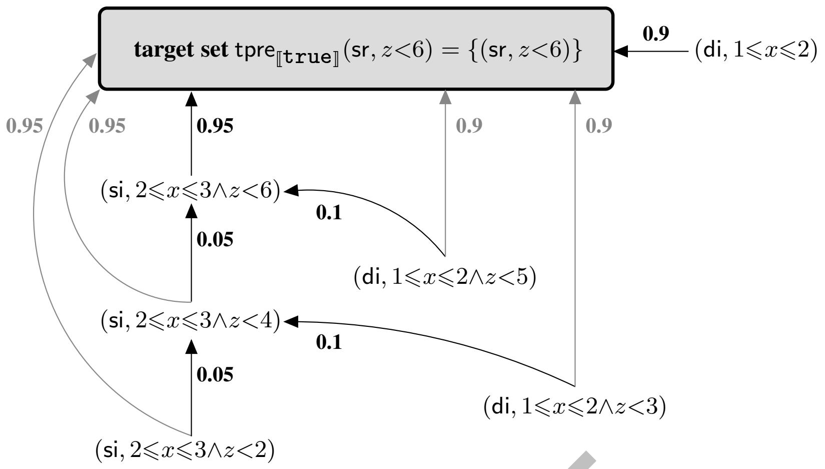

target set $\mathsf{tpre}_{[\![\mathtt{true}]\!]}(\mathsf{sr}, z<6) = \{(\mathsf{sr}, z<6)\}$
0.9
$(\mathsf{di}, 1\leqslant x\leqslant 2)$
0.95
0.95
0.95
0.9
0.9
$(\mathsf{si}, 2\leqslant x\leqslant 3\wedge z<6)$
0.1
0.05
$(\mathsf{di}, 1\leqslant x\leqslant 2\wedge z<5)$
$(\mathsf{si}, 2\leqslant x\leqslant 3\wedge z<4)$
0.1
0.05
$(\mathsf{di}, 1\leqslant x\leqslant 2\wedge z<3)$
$(\mathsf{si}, 2\leqslant x\leqslant 3\wedge z<2)$

Figure 8.8. *Markov decision process generated by* $\mathsf{MaxU}([\![\mathtt{true}]\!], (\mathsf{sr}, z<6))$

The following proposition states the correctness of our algorithm for calculating maximum until probabilities.

PROPOSITION 8.3.– *For any probabilistic timed automaton* PTA*, corresponding timed Markov decision process* $[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+} = (S, \bar{s}, \Sigma, \mathbb{R}, \mathit{Steps})$ *and sets of symbolic states* $\mathtt{U}$ *and* $\mathtt{V}$*, if* $\mathsf{MDP} = (\mathtt{Z}, \mathit{Steps}_{\mathtt{Z}})$ *is the Markov decision process generated by* $\mathsf{MaxU}(\mathtt{U}, \mathtt{V})$*, then for any* $s \in S$*:*

– $p^{\max}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+}}(s, \mathtt{U}\ \mathcal{U}\ \mathtt{V})>0$ *if and only if* $s \in \mathsf{tpre}_{\mathtt{U}\vee\mathtt{V}}(\mathtt{Z})$*;*

– *if* $p^{\max}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+}}(s, \mathtt{U}\ \mathcal{U}\ \mathtt{V})>0$*, then*

$$p^{\max}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+}}(s, \mathtt{U}\ \mathcal{U}\ \mathtt{V}) = \max\{p^{\max}_{\mathsf{MDP}}(\mathtt{z}, \Diamond\ \mathsf{tpre}_{\mathtt{U}\vee\mathtt{V}}(\mathtt{V}))\ |\ \mathtt{z}\in\mathtt{Z}\wedge s\in\mathsf{tpre}_{\mathtt{U}\vee\mathtt{V}}(\mathtt{z})\}\ .$$

EXAMPLE 8.5.– *We now return to the probabilistic timed automaton in Example 8.1 (see Figure 8.1) and repeat the calculation from Example 8.4, i.e. the maximum probability of reaching the location* sr *within* 6 *time units. We therefore add a distinct clocks* z *to the automaton and consider the probability of reaching the symbolic state* $(\mathsf{sr}, z<6)$*. Applying* $\mathsf{MaxU}([\![\mathtt{true}]\!], \{(\mathsf{sr}, z<6)\})$ *returns the Markov decision process given in Figure 8.8 where the darker arrows correspond to those edges generated by time and discrete predecessor operations (line 11 of Figure 8.7) and the lighter arrows are those generated in the construction of the Markov decision process (line 20 of Figure 8.7). From Proposition 8.3 it follows that, starting from the initial state, the maximum probability of reaching location* sr *within* 6 *time units is 0.99525, corresponding to the maximum probability of* $(\mathsf{di}, 1\leqslant x\leqslant 2 \wedge z<3)$ *reaching the target set in the Markov decision process given in Figure 8.8.*

8.3.3.4. *Computing minimum reachability probabilities*

As in the cases of (non-probabilistic) timed automata and (finite-state) Markov decision processes with fairness constraints, when considering properties which have universal quantification over paths or require the computation of minimum probabilities, standard algorithms can no longer be applied. For example, under divergent adversaries the minimum probability of letting one time unit elapse is 1; however, if we remove the restriction to time-divergent adversaries, then this minimum probability becomes 0.

The results presented below are an extension of the results for (non-probabilistic) timed automata [HEN 94] which show that verifying $\exists\Box\ I$ ("there exists a divergent path for which all states along the paths are in the set I") reduces to computing the greatest fixpoint and, for certain sets of states U and V, computing the set of states satisfying $U\ \exists\mathcal{U}\ V$ ("there exists a divergent path for which a state in V is reached and it remains in the state U until a state in V is reached"). More precisely, for any set of symbolic states $\mathtt{I}$, the set of states that satisfy $\exists\Box\ \mathtt{I}$ reduces to computing the greatest fixpoint:

$$gfp\mathtt{X}.\big(\mathtt{I} \wedge z.(\mathtt{X}\ \exists\mathcal{U}\ [\![z{>}c]\!])\big) \tag{8.1}$$

for any $c(>0) \in \mathbb{N}$ and clock z that does not appear in any of the guards, invariants or resets of the timed automata under study. An important point is that the expression requires that more than c time units elapse repeatedly.

Recall, from the duality between reachability and invariance, for any state s of $[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+}$:

$$p_s^{\min}(\Diamond\, T) = 1 - p_s^{\max}(\Box\ (L \setminus T))\,,$$

and hence, to calculate the minimum probability of reaching a set of target locations, it suffices to calculate the maximum probability of remaining in the states not in the target set, i.e. a maximum invariance probability. Note that, although we have reduced the problem to that of calculating a maximum probability, we cannot ignore time divergence when calculating such probabilities. For example, returning to the probabilistic timed automaton in Example 8.1 and the invariance set $\{(\mathsf{di}, 0{\leqslant}x{\leqslant}3)\}$, under all adversaries the maximum probability of remaining in this set is 1; however, as we cannot let more that 3 time units pass in di, under divergent adversaries this probability is 0.

Proposition 8.4 shows that we can reduce the computation of the maximum probability of invariance to that of computing the maximum probability of until within which a *qualitative* invariance is nested. As issues of time divergence are irrelevant to the computation of the maximum until probabilities, the proposition allows us to focus our attention on incorporating time divergence when finding the states which have maximum invariance probability 1.

```
algorithm $\mathsf{MaxI}_{\geqslant 1}(c, \mathtt{I})$

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

Figure 8.9. *Algorithm* $\mathsf{MaxI}_{\geqslant 1}(c, \mathtt{I})$ *for* $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$

PROPOSITION 8.4.– *For any probabilistic timed automaton* PTA*, corresponding timed Markov decision process* $[\![\mathsf{PTA}]\!]^+_{\mathbb{R}} = (S, \bar{s}, \Sigma, \mathbb{R}, Steps)$*, state* $s \in S$ *and set of symbolic states* $\mathtt{I}$*:*

$$p^{\max}_{[\![\mathsf{PTA}]\!]^+_{\mathbb{R}}}(s, \Box\, \mathtt{I}) = p^{\max}_{[\![\mathsf{PTA}]\!]^+_{\mathbb{R}}}(s, \mathtt{I}\ \mathcal{U}\ [\![\Box\, \mathtt{I}]\!]_{=1})$$

where $[\![\Box\, \mathtt{I}]\!]_{=1} = \{s' \mid s' \in S \wedge p^{\max}_{[\![\mathsf{PTA}]\!]^+_{\mathbb{R}}}(s', \Box\, \mathtt{I}) = 1\}$.

Since we have already introduced an algorithm for calculating maximum until probabilities, it remains to consider a method for calculating the set $[\![\Box\, \mathtt{I}]\!]_{=1}$. Based on (8.1) we obtain the following proposition.

PROPOSITION 8.5.– *For any probabilistic timed automaton* PTA*, corresponding timed Markov decision process* $[\![\mathsf{PTA}]\!]^+_{\mathbb{R}} = (S, \bar{s}, \Sigma, \mathbb{R}, Steps)$*, set of symbolic states* $\mathtt{I}$*, constant* $c(>0) \in \mathbb{N}$*, if* $z \in \mathcal{X}$ *does not appear in any of the guards or invariant conditions of* PTA*, then the set* $[\![\Box\, \mathtt{I}]\!]_{=1}$ *is given by the greatest fixpoint*

$$gfp\mathtt{X}.\big(\mathtt{I} \wedge z.[\![\, \mathtt{X}\ \mathcal{U}\ [\![z{>}c]\!]\,]\!]_{=1}\big)$$

where $[\![\mathtt{U}\ \mathcal{U}\ \mathtt{V}]\!]_{=1} = \{s \mid s \in S \wedge p^{\max}_{[\![\mathsf{PTA}]\!]^+_{\mathbb{R}}}(s, \mathtt{U}\ \mathcal{U}\ \mathtt{V}) = 1\}$.

The algorithm $\mathsf{MaxI}_{\geqslant 1}(c, \mathtt{I})$ for calculating the set $[\![\Box\, \mathtt{I}]\!]_{=1}$ follows from Proposition 8.5 and is given in Figure 8.9. The algorithm calls $\mathsf{MaxU}_{\geqslant 1}(\mathtt{U}, \mathtt{V})$, given in Figure 8.10, which computes the set of states $[\![\mathtt{U}\ \mathcal{U}\ \mathtt{V}]\!]_{=1}$ and is based on a similar algorithm for finite-state Markov decision processes [ALF 97].

EXAMPLE 8.6.– *We now return to the probabilistic timed automaton in Example 8.1 (see Figure 8.1) and compute the* minimum *probability of a message being correctly delivered before 6 time units have elapsed. This is achieved by adding a clock* z *to the probabilistic timed automaton and computing the maximum probability of remaining in the set of symbolic states*

$$\mathtt{I} = [\![\mathtt{true}]\!] \setminus \{(\mathsf{sr}, z{<}6)\} = \{(\mathsf{di}, 0{\leqslant}x{\leqslant}3), (\mathsf{si}, 0{\leqslant}x{\leqslant}2), (\mathsf{sr}, z{\geqslant}6)\}\,,$$

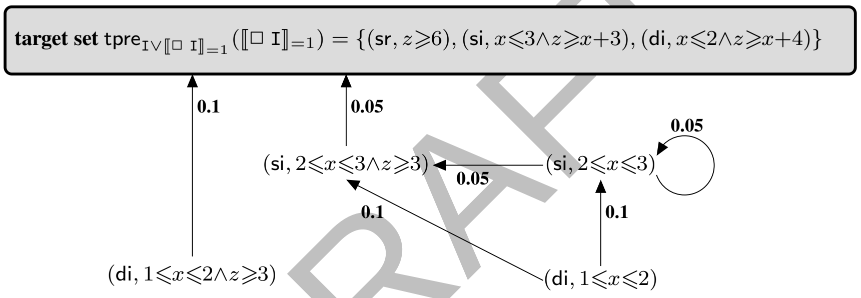
```
algorithm $\mathsf{MaxU}_{\geqslant 1}(\mathtt{U},\mathtt{V})$

$\mathtt{Z}_0 := [\![\mathtt{true}]\!]$
repeat
    $\mathtt{Y}_0 := \mathtt{Z}_0$
    $\mathtt{Z}_1 := [\![\mathtt{false}]\!]$
    repeat
        $\mathtt{Y}_1 := \mathtt{Z}_1$
        $\mathtt{Z}_1 := \mathtt{V} \vee (\mathtt{U} \wedge \mathsf{dpre1}(\mathtt{Y}_0, \mathtt{Y}_1))$
        $\mathtt{Z}_1 := \mathtt{Z}_1 \vee \mathsf{tpre}_{\mathtt{U}\vee\mathtt{V}}(\mathtt{Y}_0 \wedge \mathtt{Y}_1)$
    until $\mathtt{Z}_1 = \mathtt{Y}_1$
    $\mathtt{Z}_0 := \mathtt{Z}_1$
until $\mathtt{Z}_0 = \mathtt{Y}_0$
return $\mathtt{Z}_0$
```

```
algorithm $\mathsf{dpre1}(\mathtt{U},\mathtt{V})$

$\mathtt{Y} := [\![\mathtt{false}]\!]$
for $(l, g, \sigma, p) \in prob$
    $\mathtt{Y}_0 := [\![\mathtt{true}]\!]$
    $\mathtt{Y}_1 := [\![\mathtt{false}]\!]$
    for $e \in \mathsf{edges}(l, g, p)$
        $\mathtt{Y}_0 := \mathsf{dpre}(e, \mathtt{U}) \wedge \mathtt{Y}_0$
        $\mathtt{Y}_1 := \mathsf{dpre}(e, \mathtt{V}) \vee \mathtt{Y}_1$
    end
    $\mathtt{Y} := (\mathtt{Y}_0 \wedge \mathtt{Y}_1) \vee \mathtt{Y}$
end
return $\mathtt{Y}$
```

Figure 8.10. *Algorithm* $\mathsf{MaxU}_{\geqslant 1}(\mathtt{U},\mathtt{V})$ *for* $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$



target set $\mathsf{tpre}_{\mathtt{I}\vee[\![\Box\ \mathtt{I}]\!]_{=1}}([\![\Box\ \mathtt{I}]\!]_{=1}) = \{(\mathsf{sr}, z{\geqslant}6), (\mathsf{si}, x{\leqslant}3{\wedge}z{\geqslant}x{+}3), (\mathsf{di}, x{\leqslant}2{\wedge}z{\geqslant}x{+}4)\}$
0.1
0.05
0.05
$(\mathsf{si}, 2{\leqslant}x{\leqslant}3{\wedge}z{\geqslant}3)$
0.05
$(\mathsf{si}, 2{\leqslant}x{\leqslant}3)$
0.1
0.1
$(\mathsf{di}, 1{\leqslant}x{\leqslant}2{\wedge}z{\geqslant}3)$
$(\mathsf{di}, 1{\leqslant}x{\leqslant}2)$

Figure 8.11. *Markov decision process generated by* $\mathsf{MaxU}(\mathtt{I}, [\![\Box\ \mathtt{I}]\!]_{=1})$

i.e. the states where either the message has not been delivered or clock z is greater than or equal to 6. Using Proposition 8.4 we have $p^{\max}_s(\Box\ \mathtt{I}) = p^{\max}_s(\mathtt{I}\,\mathcal{U}\,[\![\Box\ \mathtt{I}]\!]_{=1})$, and hence we first find $[\![\Box\ \mathtt{I}]\!]_{=1}$ (the set of states for which the maximum probability of remaining in $\mathtt{I}$ is 1). Using Proposition 8.5 this set is given by $\mathsf{MaxI}_{\geqslant 1}(c, \mathtt{I})$ *which returns:*

$$[\![\Box\ \mathtt{I}]\!]_{=1} = \{(\mathsf{sr}, z{\geqslant}6), (\mathsf{si}, x{\leqslant}3 \wedge z{\geqslant}x{+}3), (\mathsf{di}, x{\leqslant}2 \wedge z{\geqslant}x{+}4)\}\,.$$

Next, applying $\mathsf{MaxU}(\mathtt{I}, [\![\Box\ \mathtt{I}]\!]_{=1})$ *returns the Markov decision process given in Figure 8.11. As* $(\mathsf{di}, 1{\leqslant}x{\leqslant}2)$ *is the only symbolic state in Figure 8.11 for which the time predecessor set includes* $(\mathsf{di}, x{=}0{\wedge}z{=}0)$*, using Proposition 8.3, from the initial state, the maximum probability of remaining in* $\mathtt{I}$ *until a state in* $[\![\Box\ \mathtt{I}]\!]_{=1}$ *is reached equals the maximum probability of* $(\mathsf{di}, 1{\leqslant}x{\leqslant}2)$ *reaching* $\mathsf{tpre}_{\mathtt{I}\vee[\![\Box\ \mathtt{I}]\!]_{=1}}([\![\Box\ \mathtt{I}]\!]_{=1})$*, and hence equals 0.005.*

Finally, using Proposition 8.4 and the duality between probabilistic reachability of invariance, starting from di *with* x *equal to 0, the minimum probability of correctly delivering before 6 time units have elapsed equals* $1 - 0.005 = 0.995$.

8.3.4. *Digital clocks*

In this section we present the *integral semantics* (or "digital clocks") model where the time domain is the natural numbers as opposed to the reals. This leads to a finite-state Markov decision process which can therefore be model checked directly by employing the efficient symbolic methods in tools such as PRISM [HIN 06, PRI]. This approach is based on the work of [HEN 92] which studies the question of when real-time properties of timed automata can be verified using only integral durations (digital clocks), and shows that such a reduction is possible for a large class of systems and properties, including time-bounded invariance and response. However, before we discuss the integral semantics, we introduce an additional measure for probabilistic timed automata, *expected reachability*, since this measure is, along with probabilistic reachability, preserved under certain restrictions by the integral semantics.

In this section, we assume that probabilistic timed automata are *structurally non-Zeno* (see Section 8.3.1) and also *closed* and *diagonal-free*, that is, automata whose zones do not compare the values of clocks with each another or contain strict comparisons with constants. The correctness of the results presented in this section can be found in [KWI 06].

8.3.4.1. *Expected reachability*

Expected reachability is defined with respect to a set of target states and a cost function mapping state-event and state-duration pairs to real values (the cost of performing an event or letting a certain amount of time pass in the corresponding state, respectively). This measure corresponds to the expected cost (with respect to the cost function) of reaching the target set. More formally, for a timed Markov decision process $\mathsf{TMDP} = (S, \bar{s}, \Sigma, \mathbb{T}, \mathit{Steps})$, cost function $\mathcal{C} : S \times (\Sigma \cup \mathbb{T}) \to \mathbb{R}$, state $s \in S$, set $T \subseteq S$ of target states, and adversary $A \in \mathit{Adv}_{\mathsf{TMDP}}$, let $e^A_{\mathsf{TMDP}}(s, \mathit{cost}(\mathcal{C}, T))$ denote the usual expectation of the function $\mathit{cost}(\mathcal{C}, T)$ (which returns, for a given path $\omega \in \mathit{Path}(s)$, the total cost accumulated until a state in T is reached along ω) with respect to the measure Prob^A_s over $\mathit{Path}^A(s)$. That is:

$$e^A_{\mathsf{TMDP}}(s, \mathit{cost}(\mathcal{C}, T)) = \int_{\omega \in \mathit{Path}^A(s)} \mathit{cost}(\mathcal{C}, T)(\omega)\, d\mathit{Prob}^A_s$$

where for any $\omega \in \mathit{Path}^A(s)$:

$$\mathit{cost}(\mathcal{C}, T)(\omega) \overset{\text{def}}{=} \sum_{i=1}^{\min\{j \,|\, \omega(j) \in T\}} \mathcal{C}(\omega(i-1), \mathit{step}(\omega, i-1))$$

if there exists $j \in \mathbb{N}$ such that $\omega(j) \in T$, and $cost(\mathcal{C}, T)(\omega) \overset{\text{def}}{=} \infty$ otherwise.

Since cost of a path which does not reach T is set to ∞, even though the total cost of the path may not be infinite, the expected cost of reaching T from s is finite if and only if a state in T is reached from s with probability 1. *Expected time reachability* (the expected time with which a given set of states can be reached) is a special case of expected reachability, corresponding to the case when $\mathcal{C}(s, \sigma) = 0$ for all $s \in S$ and $\sigma \in \Sigma$ and $\mathcal{C}(s, t) = t$ for all $s \in S$ and $t \in \mathbb{T}$.

DEFINITION 8.12.– *Let* $\mathsf{TMDP} = (S, \bar{s}, \Sigma, \mathbb{T}, Steps)$ *be a timed Markov decision process. The* maximum and minimum expected costs *of, from a state* $s \in S$*, reaching a set of states* T *under the cost function* $\mathcal{C}$ *are defined as follows:*

$$\begin{aligned} e^{\max}_{\mathsf{TMDP}}(s, \mathcal{C}, \Diamond\, T) &= \sup_{A \in Adv_{\mathsf{TMDP}}} e^{A}_{\mathsf{TMDP}}(s, cost(\mathcal{C}, T)) \\ e^{\min}_{\mathsf{TMDP}}(s, \mathcal{C}, \Diamond\, T) &= \inf_{A \in Adv_{\mathsf{TMDP}}} e^{A}_{\mathsf{TMDP}}(s, cost(\mathcal{C}, T))\,. \end{aligned}$$

Calculating expected reachability for finite-state Markov decision processes is equivalent to the *stochastic shortest path problem*; see for example [BER 91, ALF 99].

In practice, cost functions are defined not at the level of timed Markov decision processes, but in terms of probabilistic timed automata. At this level, cost functions can be defined using a pair (c_Σ, r), where $c_\Sigma : L \times \Sigma \to \mathbb{R}$ is a function assigning the cost, in each location, of executing each event in Σ, and $r : L \to \mathbb{R}$ is a function assigning to each location the rate at which costs are accumulated as time passes in that location. The associated cost function $\mathcal{C}_{c_\Sigma, r}$ is defined as follows, for each $(l, v) \in L \times \mathbb{T}^{\mathcal{X}}$ and $a \in \Sigma \cup \mathbb{T}$:

$$\mathcal{C}_{c_\Sigma, r}((l, v), a) \overset{\text{def}}{=} \begin{cases} c_\Sigma(l, a) & \text{if } a \in \Sigma \\ a \cdot r(l) & \text{otherwise.} \end{cases}$$

A probabilistic timed automaton equipped with a pair (c_Σ, r) is a probabilistic extension of priced timed automata (also known as weighted timed automata) [BEH 01, ALU 04].

Expected time reachability allows us to express, for example, “the expected time until a data packet is delivered is at most 20 ms” and “the expected time until a packet collision occurs is at least 100 seconds”. In general, expected reachability allows us to compute the maximum and an minimum values for measures including: “the expected number of retransmissions before the message is correctly delivered”, “the expected number of packets sent before failure” and “the expected number of lost messages within the first 200 seconds". For example, in the case of the last property, to calculate this measure, we would first need to modify the probabilistic timed automaton under study by adding a distinct clock and location such that, from all locations, once this clock has reached 200 seconds the only transition is to this new location. The set of target states would then be the set containing only this new location and the cost function would equal 0 on all time transitions and events except the event(s) corresponding to a message being lost, whose cost would be set to 1.

In addition, using cost functions of the form $\mathcal{C}_{c_\Sigma,r}$, we can consider performance measures such as:

– the expected time the channel is free before N messages are sent (by setting $r(l)$ to be 1 if location l corresponds to a state in which the channel is free, and 0 otherwise);

– the expected time a sender spends waiting for an acknowledgement (by setting $r(l)$ to be 1 if location l corresponds to a state in which the sender is waiting for an acknowledgement, and 0 otherwise);

– the expected energy consumption within the first $T(\in \mathbb{N})$ seconds (by setting $r(l)$ to the power usage (Watts) of location $l \in L$ and $c_\Sigma(l,\sigma)$ to be the energy consumption associated with performing the event σ in location l).

8.3.4.2. *Integral semantics*

Recall that the semantics of a probabilistic timed automaton given in Section 8.2.4 is parameterised by both the time domain $\mathbb{T}$ and time increment $\oplus$. In this section we set the time domain $\mathbb{T}$ equal to $\mathbb{N}$, we let the time increment operator $\oplus$ equal $\oplus_\mathbb{N}$ which is defined below, and refer to $[\![\mathsf{PTA}]\!]_\mathbb{N}^{\oplus_\mathbb{N}}$ as the integral semantics of PTA. To define $\oplus_\mathbb{N}$, first, for any $x \in \mathcal{X}$, let c_x denote the greatest constant the clock x is compared to in the clock constraints of PTA. If the value of the clock x exceeds c_x, then its exact value is not relevant when deciding which probabilistic edges are enabled. This means that $c_x{+}1$ is the maximum value of clock x that needs to be represented, because we can interpret this value as corresponding to all clock values greater than c_x, which leads us to the following definition of $\oplus_\mathbb{N}$. For any clock valuation $v \in \mathbb{N}^\mathcal{X}$ and time duration $t \in \mathbb{N}$, let $v \oplus_\mathbb{N} t$ be the clock valuation of $\mathcal{X}$ which assigns the value $\min\{v(x){+}t, c_x{+}1\}$ to all clocks $x \in \mathcal{X}$ (although the operator $\oplus_\mathbb{N}$ is dependent on PTA, we omit the sub- or superscript indicating this for clarity).

The definition of integral semantics for probabilistic timed automata is a generalisation of the analogous definition for the classical model in [BEY 01]. The fact that the integral semantics of a probabilistic timed automaton is finite can be derived from the definitions.

The results below demonstrate that digital clocks are sufficient for calculating probabilistic reachability and expected reachability properties of probabilistic timed automata.

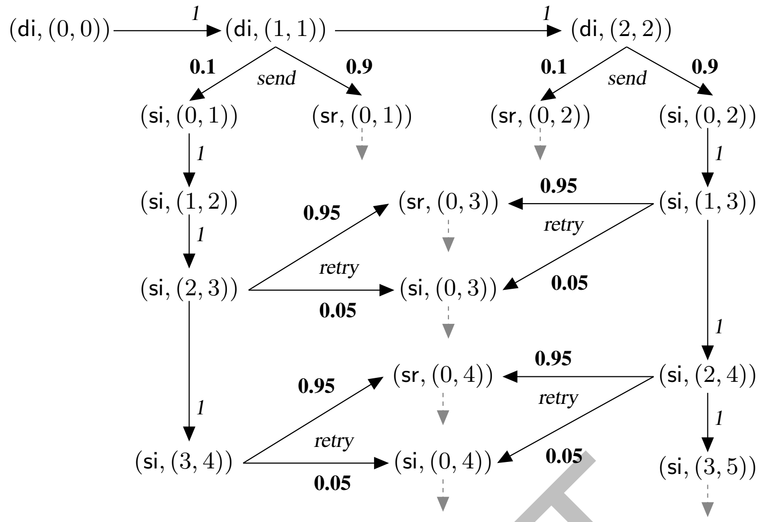


Figure 8.12. *Fragment of the integral semantic model of Example 8.1*

PROPOSITION 8.6.– *For any probabilistic timed automaton* PTA *and target set of symbolic states* T *in which all zones are closed and diagonal free:*

$$p^{\max}_{[\![\mathsf{PTA}]\!]^+_{\mathbb{R}}}((\bar{l},\mathbf{0}),\Diamond\,\mathtt{T}) = p^{\max}_{[\![\mathsf{PTA}]\!]^{\oplus_{\mathbb{N}}}_{\mathbb{N}}}((\bar{l},\mathbf{0}),\Diamond\,\mathtt{T})$$
$$p^{\min}_{[\![\mathsf{PTA}]\!]^+_{\mathbb{R}}}((\bar{l},\mathbf{0}),\Diamond\,\mathtt{T}) = p^{\min}_{[\![\mathsf{PTA}]\!]^{\oplus_{\mathbb{N}}}_{\mathbb{N}}}((\bar{l},\mathbf{0}),\Diamond\,\mathtt{T})\ .$$

Furthermore, for any (non-negative) cost function $\mathcal{C}_{c_\Sigma,r}$ with rational coefficients, if all probability values appearing in PTA *are rational, then:*

$$e^{\max}_{[\![\mathsf{PTA}]\!]^+_{\mathbb{R}}}((\bar{l},\mathbf{0}),\mathcal{C}_{c_\Sigma,r},\Diamond\,\mathtt{T}) = e^{\max}_{[\![\mathsf{PTA}]\!]^{\oplus_{\mathbb{N}}}_{\mathbb{N}}}((\bar{l},\mathbf{0}),\mathcal{C}_{c_\Sigma,r},\Diamond\,\mathtt{T})$$
$$e^{\min}_{[\![\mathsf{PTA}]\!]^+_{\mathbb{R}}}((\bar{l},\mathbf{0}),\mathcal{C}_{c_\Sigma,r},\Diamond\,\mathtt{T}) = e^{\min}_{[\![\mathsf{PTA}]\!]^{\oplus_{\mathbb{N}}}_{\mathbb{N}}}((\bar{l},\mathbf{0}),\mathcal{C}_{c_\Sigma,r},\Diamond\,\mathtt{T})\ .$$

EXAMPLE 8.7.– *We return once more to the* PTA *in Example 8.1 (see Figure 8.1) and consider the integer semantic model of this automaton. Similarly to the cases before, since we are interested in the probability of reaching the location* sr *within* 6 *time units, we add an additional clock z to the automaton. Note that, since all zones must be closed (and diagonal-free), in this case we cannot consider a strict bound on the clock z, i.e. we can consider the target set* (sr, $z \leqslant 6$) *but not the target set* (sr, $z<6$)*. The Markov decision process obtained through the integer semantics has 32 states and*

42 transitions. In Figure 8.12 we present a fragment of this Markov decision process in which a clock valuation v is written as a pair (r_1, r_2) rather than $v(x) = r_1$ and $v(z) = r_2$. We find that the minimum and maximum probabilities equal 0.995 and 0.9975 respectively. The fact that maximum probability differs from the previous cases is due to the fact that the target set now includes the case when the clock z equals 6.

8.4. Case study: the IEEE FireWire root contention protocol

We illustrate the practical applicability of the techniques presented in this chapter with a real-life case study: the IEEE FireWire root contention protocol [IEE 95], which uses both randomisation and timing delays to determine a leader among two contending processes. This section is based on results presented in [KWI 03, DAW 04, KWI 06].

8.4.1. *Overview*

The IEEE 1394 High Performance Serial Bus is used to transport digital video and audio signals within a network of multimedia systems and devices, such as PCs, laptops and camcorders. It has a scalable architecture, and it is hot-pluggable, meaning that devices can be added to or removed from the network at any time, supports both isochronous and asynchronous communication and allows quick, reliable and inexpensive data transfer. It is currently one of the standard protocols for interconnecting multimedia equipment. The standard comprises various protocols. The one that we consider here is a leader election protocol called the *root contention protocol*.

The root contention protocol takes place after a bus reset in the network, i.e. when a node (device or peripheral) is added to or removed from the network. After a bus reset, all nodes in the network have equal status and know only to which nodes they are directly connected, so a leader must then be chosen. The aim of this protocol is to check whether the network topology is a tree and, if so, to construct a spanning tree over the network whose root is the leader elected by the protocol.

In order to elect a leader, nodes exchange “be my parent” requests with its neighbours. However, *contention* may arise when two nodes simultaneously send such requests to each other. The solution adopted by the standard to overcome this conflict, called *root contention*, is both probabilistic and timed: each node will flip a coin in order to decide whether to wait for a short or for a long time before sending a request.

8.4.2. *Probabilistic timed automata model*

The model comprises four components: two contending nodes and two connecting wires. Figure 8.13 shows the probabilistic timed automaton $\mathtt{Node}_i^{\mathtt{P}}$ for a node.

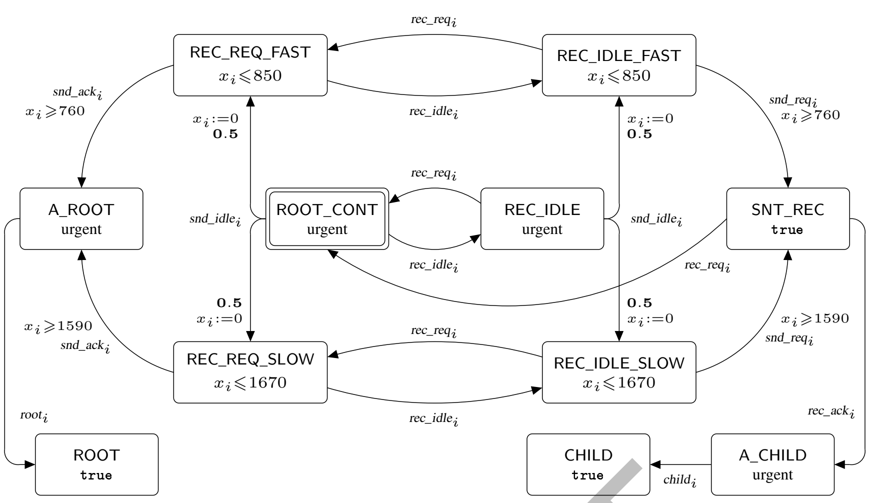
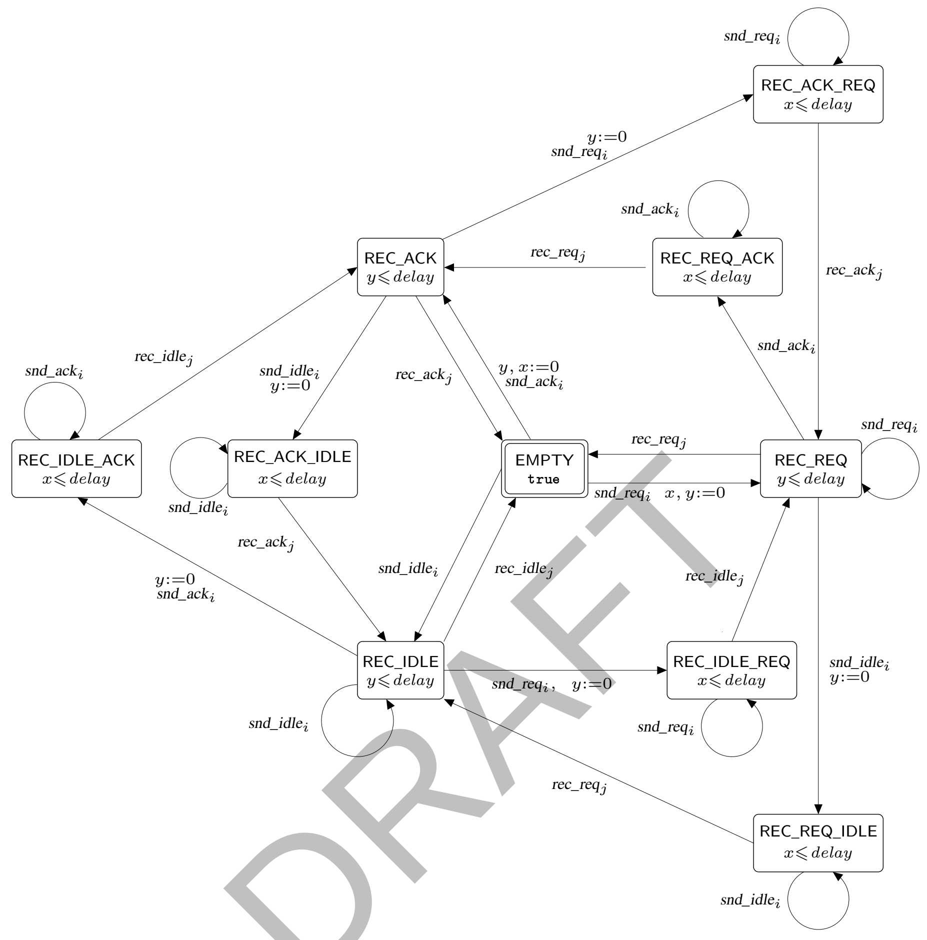


Figure 8.13. *The probabilistic timed automaton* $\mathtt{Node}_i^{\mathtt{P}}$

This model is a probabilistic extension of the timed automaton model presented in [SIM 01].

The behaviour of the model commences in location ROOT_CONT, which models the situation in which the node has detected root contention. Because this location is urgent, $\mathtt{Node}_i^{\mathtt{P}}$ is forced to select an outgoing probabilistic edge instantly. Consider the bifurcating probabilistic edge labelled by *snd_idle*$_i$, which corresponds to the node flipping a coin in order to determine whether it should wait for a short or long time. The *snd_idle*$_i$ event is sent by $\mathtt{Node}_i^{\mathtt{P}}$ to its communication medium, referring to a transmission of an idle signal across the node's wire to the other node. In both of the locations REC_REQ_FAST and REC_REQ_SLOW which may be reached after taking the probabilistic edge, the passage of time may mean that the value of clock x_i can reach a value enabling the edges, which means that an acknowledgement is sent (event *snd_ack*$_i$), and the node then declares itself to be leader (the event *root*$_i$ which labels the subsequent left-pointing edge to the location ROOT). In contrast, in the locations REC_REQ_FAST and REC_REQ_SLOW, if the node receives an idle signal from the other contending node (event *rec_idle*$_i$) before sending an acknowledgement, it is forced to move to the right to REC_IDLE_FAST or REC_IDLE_SLOW, respectively.

In this case, after a certain amount of time elapses, $\mathtt{Node}_i^{\mathtt{P}}$ can issue a request to the other node to be its parent by sending the event *snd_req*$_i$ to its wire. If the node then subsequently detects a parent request from the other node (event *rec_req*$_i$), it returns to the location ROOT_CONT, and restarts the root contention process. If, on the other hand, the node detects an acknowledgement from the other node (event *rec_ack*$_i$), it proceeds to declare itself as the child by sending a *child*$_i$ event.



REC_ACK_REQ $x \leqslant delay$
REC_ACK $y \leqslant delay$
REC_REQ_ACK $x \leqslant delay$
REC_IDLE_ACK $x \leqslant delay$
REC_ACK_IDLE $x \leqslant delay$
EMPTY true
REC_REQ $y \leqslant delay$
REC_IDLE $y \leqslant delay$
REC_IDLE_REQ $x \leqslant delay$
REC_REQ_IDLE $x \leqslant delay$
snd_req_i
$y{:=}0$ snd_req_i
snd_ack_i
rec_req_j
rec_ack_j
rec_idle_j
snd_idle_i $y{:=}0$
$y, x{:=}0$ snd_ack_i
snd_req_i $x, y{:=}0$
snd_idle_i
$y{:=}0$ snd_ack_i
snd_req_i, $y{:=}0$

Figure 8.14. *The probabilistic timed automaton* $\texttt{Wire}_i$

The communication medium between the nodes comprises two wires modelled as two-place buffers, along which signals are driven continuously. These are represented by the timed automata $\texttt{Wire}_1$ and $\texttt{Wire}_2$. Figure 8.14 illustrates the general case, $\texttt{Wire}_i$. This model is the classical (non-probabilistic) timed automaton taken from [SIM 01], with the interpretation that all transitions are made with probability 1. The parallel composition

$$\texttt{Impl}^{\texttt{P}} = \texttt{Node}_1^{\texttt{P}} \parallel \texttt{Wire}_1 \parallel \texttt{Wire}_2 \parallel \texttt{Node}_2^{\texttt{P}}$$

of the resulting probabilistic timed automata is then constructed using Definition 8.4.

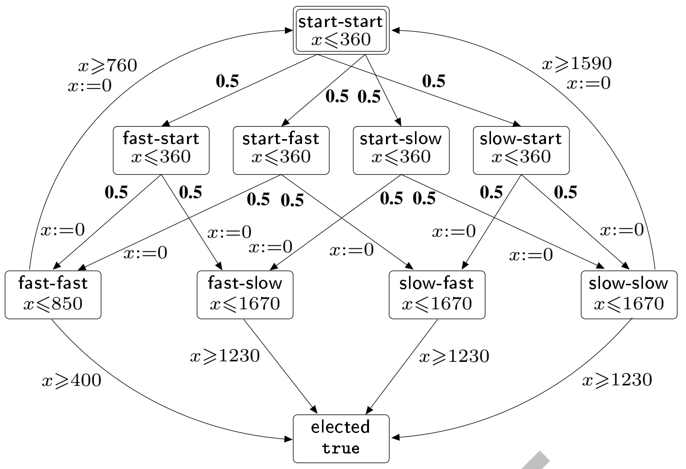


Figure 8.15. *The probabilistic timed automaton* $\mathtt{I}_1^{\mathtt{p}}$

D (10^3ns)	Region graph	Forwards	Backwards	Digital clocks
2	423,016	53 (0.00)	15 (1.39)	68,056
4	1,395,390	131 (0.00)	25 (1.55)	220,565
8	3,375,390	372 (0.02)	81 (1.35)	530,965
10	4,365,390	526 (0.03)	126 (1.61)	686,165
20	9,315,390	1,876 (0.09)	528 (11.0)	1,462,165
30	14,265,390	4,049 (0.20)	1,206 (11.2)	2,238,165
40	19,215,390	7,034 (0.46)	2,168 (1,030)	3,014,165
50	24,165,390	10,865 (1.23)	3,426 (6,464)	3,790,165
60	29,115,390	15,511 (2.74)	4,964 (26,995)	4,566,165
∞	1,542	-	0 (0.55)	776

Table 8.1. *Model sizes (and generation times in seconds) for* $\mathtt{I}_1^{\mathtt{p}}$

We also study the abstract probabilistic timed automaton $\mathtt{I}_1^{\mathtt{p}}$ of the root contention protocol given in Figure 8.15. It is a probabilistic extension of the classical timed automaton $\mathtt{I}_1$ of [SIM 01], where each instance of bifurcating edges corresponds to a coin being flipped. For example, in the initial location start-start, there is a non-deterministic choice corresponding to node 1 (respectively node 2) starting the root contention protocol and flipping its coin, leading with probability 0.5 to each of slow-start and fast-start (respectively start-slow and start-fast). To simplify the presentation, the events of the probabilistic edges of $\mathtt{I}_1^{\mathtt{p}}$ have been omitted.

D (10^3ns)	Forwards	Backwards	Digital clocks
2	965 (0.08)	551 (447)	6,719,773
4	2,599 (0.94)	2,220 (3,854)	44,366,235
6	4,337 (1.64)	4,264 (14,954)	86,813,479
8	7,831 (2.93)	8,075 (100,893)	129,267,079
10	11,119 (4.27)	13,428 (608,392)	171,720,679
20	41,017 (18.7)	-	383,988,679
30	89,283 (56.1)	-	596,256,679
40	155,675 (129)	-	808,524,679
∞	-	0 (97.2)	212,268

Table 8.2. *Model sizes (and generation times in seconds) for* `Impl`$^{\texttt{P}}$

D	Region graph		Forwards		Backwards		Digital clocks	
(10^3ns)	const.	m/c	const.	m/c	const.	m/c	const.	m/c
2	14.3	5.90	0.08	0.02	0.08	0.03	2.90	0.53
4	29.2	45.2	0.10	0.03	0.07	0.02	8.67	3.81
8	72.4	215	0.15	0.03	0.30	0.03	31.4	22.0
10	100	332	0.16	0.04	0.59	0.03	49.0	34.1
20	314	1,140	0.70	0.10	11.5	0.05	183	134
30	898	2,535	1.83	0.16	44.7	0.12	406	280
40	1,186	2,786	4.66	0.29	268	0.29	746	312
50	1,906	2,861	11.0	0.50	448	0.53	1,191	302
60	2,666	2,938	19.6	0.71	1,125	0.73	1,754	300
∞	0.61	0.33	-	-	0	0	0.41	0.11

Table 8.3. *Model construction and model checking times in seconds for* $\texttt{I}_1^{\texttt{P}}$

8.4.3. *Model checking statistics*

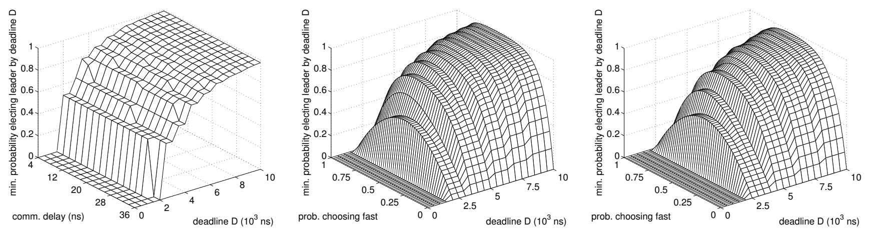
In this section we investigate the state spaces and timing statistics of the different approaches presented in this chapter when applied to the root contention protocol. The statistics were obtained when setting the maximum transmission delay equal to 360 nanoseconds and calculating the minimum probability of electing a leader by a deadline D and the minimum probability of eventually electing a leader (i.e. when $D{=}\infty$). The generation times for the symbolic backwards approach are for the prototype implementation of [KWI 07] and in the case of the forwards algorithm for the implementation based on KRONOS [DAW 96] presented in [DAW 04]. To apply the forwards algorithm, which can only compute upper bounds on maximum reachability probabilities, the model transformations of `Impl`$^{\texttt{P}}$ and $\texttt{I}_1^{\texttt{P}}$ presented in [KWI 03] were employed. Since no model transformation is known to reduce the computation of the minimum probability of eventually electing a leader to a maximum reachability probability, the forwards approach has not been applied to this case.

D	Forwards		Backwards		Digital clocks	
(10^3ns)	const.	m/c	const.	m/c	const.	m/c
2	3.02	0.08	10.8	0.04	264	2.49
4	8.98	0.11	157	0.08	1,627	300
6	14.1	0.15	646	0.09	2,967	13,812
8	41.3	0.23	2,065	0.15	5,690	26,583
10	79.5	0.34	5,081	0.30	7,930	34,885
20	1,303	1.11	-	-	16,748	104,112
30	6,893	4.54	-	-	26,969	141,785
40	25,417	8.38	-	-	32,325	199,102
∞	-	-	0	0	23.0	68.9

Table 8.4. *Model construction and model checking times in seconds for* $\texttt{Impl}^{\texttt{P}}$

Tables 8.1 and 8.2 present the state spaces for all techniques and generation times for the backwards and forward algorithms. The results demonstrate that the region graph and digital clocks models have very large state spaces and that the region graph becomes prohibitively large very quickly (limiting its applicability to only the abstract version of the contention protocol $\texttt{I}_1^{\texttt{P}}$). The results also show that the forwards and backwards methods lead to models of similar sizes and that the generation times are much larger for the backward algorithms. This difference in times can be attributed to the fact that the backwards technique requires more complex operations on zones. This difference is increased by the fact that the forwards implementation employs the established tool KRONOS. For the case $D{=}\infty$, note that the backwards algorithm generates an empty state space. This is because after the first step (calculating $[\![\Box\ \texttt{I}]\!]_{=1}$) there is no further work to be done.

Tables 8.3 and 8.4 report on the construction and model checking times for the (finite-state) Markov decision processes when using the PRISM tool [HIN 06, PRI]. Due to the size of the state spaces for the region graph and digital clocks models, the construction and model checking times are larger in these cases. Although the digital clocks approach generates much larger state spaces, we can see that this does not have a drastic effect on the model checking times. This is due to regularity in the model which is exploited in the symbolic data structures employed by PRISM. The fact that construction times for the forwards models are much faster than those obtained for the backwards models is due to the forwards implementation optimising the PRISM input [DAW 04]. Finally we note that, due to similarity between the model sizes, there is no significant difference between the model checking times of the forwards and backwards models.



(a) Communication delay (b) Biased coin (short wire) (c) Biased coin (long wire)

Figure 8.16. *Minimum probability of electing a leader by deadline D*

8.4.4. *Performance analysis*

We now present an analysis of the performance of the FireWire root contention protocol based on the application of the techniques described in the previous section. Firstly, Figure 8.16(a) presents the minimum probability of electing a leader within a deadline as the communication delay (wire length) between the nodes varies. As expected, as the communication delay between the nodes increases, the probability of electing the root before a deadline decreases.

In the remaining analysis, we will consider two cases for the maximum transmission delay along the wires (the constant $delay$ in Figure 8.14): 360 nanoseconds (ns) and 30 ns. This models the distance between the two nodes, i.e. the length of the connecting wires. A delay of 360 ns represents the assumption that the nodes are separated by a distance close to the maximum required for the correctness of the protocol (from the analysis of [SIM 01]). A delay of 30 ns corresponds more closely to the maximum separation of nodes specified in the actual IEEE standard; this value is 22.725 ns, and therefore our figure of 30 ns is an over-approximation. This is for efficiency reasons: it allows us to use a time granularity of 10 ns when we consider probabilistic model checking using digital clocks. In the following paragraphs, we will refer to the two cases (360 ns and 30 ns) as “long wire” and “short wire”, respectively.

Figures 8.16(b), 8.16(c) and 8.17 report on the effect of using a biased coin with respect to the minimum probability of electing a leader within a deadline and the maximum expected time to elect a leader, respectively. Note that we assume that the nodes in root contention use coins of the same bias. Although it is possible to improve the performance of the protocol by assuming that the nodes’ coins have different biases, i.e. one coin is biased towards fast and the other towards slow, this is not feasible in practice since each node follows the same procedure and is not known in advance which nodes of the network will take part in the root contention protocol. Furthermore, assuming that two nodes enter the contention protocol, to decide which node should flip which sort of coin is equivalent to electing a leader.

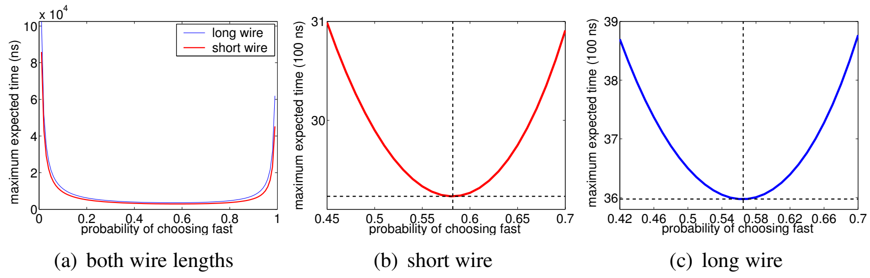


Figure 8.17. *Maximum expected time to elect a leader*

The results demonstrate that the (timing) performance of the root contention protocol can be improved using a biased coin which has a higher probability of flipping "fast". The idea behind this result is that, although the use of such a biased coin decreases the likelihood of the nodes flipping different values, when nodes flip the same values there is a greater chance that less time passes before they flip again (i.e. when both flip "fast") [STO 02]. There is a compromise here, because as the coin becomes more biased towards "fast", the probability of the nodes actually flipping different values (which is required for a leader to be elected) decreases even though the delay between coin flips will on average decrease. The results in Figure 8.17 further demonstrate that, for a shorter wire length, there is a greater advantage when using a biased coin (for the short wire length the minimum occurs for a coin which flips "fast" with a probability near 0.58 while for the long wire length the minimum is near 0.56). The reasoning behind this result is that for the short wire length more time is saved when both nodes flip fast than for a longer wire length, since the time required when both nodes flip fast is dependent on a constant delay given by the protocol specification plus a delay dependent on the wire length.

8.5. Conclusions

In this chapter we gave an introduction to the probabilistic timed automata formalism, which is suitable for the modelling and analysis of systems exhibiting both real-time and probabilistic characteristics. We described four different techniques for model checking probabilistic timed automata. In brief these can be summarised as follows:

– The *region graph* approach is applicable to a large class of probabilistic timed automata and full PTCTL, but its application can be prohibitively expensive.

– The *forwards reachability* approach is applicable to general probabilistic timed automata, but is restricted to computing upper bounds on maximum reachability probabilities.

– The *backwards reachability* approach is applicable to general probabilistic timed automata and full PTCTL, but is more expensive than the forwards approach and, at the time of writing, cannot be applied to expected reachability.

– The *digital clocks* approach is applicable to closed and diagonal free probabilistic timed automata; it can be used to compute probabilistic and expected reachability measures, but cannot be used to verify full PTCTL.

This chapter also demonstrated the applicability of these methods to the analysis of the IEEE FireWire root contention protocol. Additional case studies using these methods include Bluetooth Device Discovery [DUF 06], IEEE 802.11 Wireless LAN [KWI 02b], IPV4 Zeroconf [KWI 06], IEEE 802.15.4 CSMA-CA (ZigBee) [FRU 06] and IEEE 802.3 CSMA/CD [KWI 07]. See also [PRI] for further details.

8.6. Bibliography

[ALF 97] DE ALFARO L., Formal verification of probabilistic systems, PhD thesis, Stanford University, 1997.

[ALF 99] DE ALFARO L., "Computing minimum and maximum reachability times in probabilistic systems", BAETEN J., MAUW S., Eds., *Proc. 10th Int. Conf. Concurrency Theory (CONCUR'99)*, vol. 1664 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 66–81, 1999.

[ALU 94] ALUR R., DILL D., "A theory of timed automata", *Theoretical Computer Science*, vol. 126, num. 2, p. 183–235, 1994.

[ALU 04] ALUR R., LA TORRE S., PAPPAS G., "Optimal paths in weighted timed automata", *Theoretical Computer Science*, vol. 318, num. 3, p. 297–322, 2004.

[BEH 01] BEHRMANN G., FEHNKER A., HUNE T., LARSEN K., PETTERSSON P., ROMIJN J., VAANDRAGER F., "Minimum-cost reachability for linearly priced timed automata", BENEDETTO M. D., SANGIOVANNI-VINCENTELLI A., Eds., *Proc. 4th Int. Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, vol. 2034 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 147–162, 2001.

[BEH 04] BEHRMANN G., DAVID A., LARSEN K. G., "A tutorial on UPPAAL", BERNARDO M., CORRADINI F., Eds., *Formal Methods for the Design of Real-Time Systems: 4th Int. School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, vol. 3185 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 200–236, 2004.

[BER 91] BERTSEKAS D., TSITSIKLIS J., "An analysis of stochastic shortest path problems", *Mathematics of Operations Research*, vol. 16, num. 3, p. 580–595, 1991.

[BEY 01] BEYER D., "Improvements in BDD-based reachability analysis of timed automata", OLIVEIRA J., ZAVE P., Eds., *Proc. Symp. Formal Methods Europe (FME'01)*, vol. 2021 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 318–343, 2001.

[BIA 95] BIANCO A., DE ALFARO L., "Model checking of probabilistic and nondeterministic systems", THIAGARAJAN P., Ed., *Proc. 15th Conf. Foundations of Software Technology and Theoretical Computer Science*, vol. 1026 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 499–513, 1995.

[COU 98] COURCOUBETIS C., YANNAKAKIS M., "Markov decision processes and regular events", *IEEE Transactions on Automatic Control*, vol. 43, num. 10, p. 1399–1418, 1998.

[DAW 96] DAWS C., OLIVERO A., TRIPAKIS S., YOVINE S., "The tool KRONOS", ALUR R., HENZINGER T., SONTAG E., Eds., *Hybrid Systems III, Verification and Control*, vol. 1066 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 208–219, 1996.

[DAW 98] DAWS C., TRIPAKIS S., "Model checking of real-time reachability properties using abstractions", STEFFEN B., Ed., *Proc. 4th Int. Conf. Tools and Algorithms for Construction and Analysis of Systems (TACAS'98)*, vol. 1384 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 313–329, 1998.

[DAW 04] DAWS C., KWIATKOWSKA M., NORMAN G., "Automatic verification of the IEEE 1394 root contention protocol with KRONOS and PRISM", *Int. Journal on Software Tools for Technology Transfer*, vol. 5, num. 2–3, p. 221–236, 2004.

[DIL 89] DILL D., "Timing assumptions and verification of finite-state concurrent system", SIFAKIS J., Ed., *Proc. Int. Workshop Automatic Verification Methods for Finite State Systems*, vol. 407 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 197–212, 1989.

[DUF 06] DUFLOT M., KWIATKOWSKA M., NORMAN G., PARKER D., "A formal analysis of Bluetooth device discovery", *Int. Journal on Software Tools for Technology Transfer*, vol. 8, num. 6, p. 621–632, 2006.

[FRU 06] FRUTH M., "Probabilistic model checking of contention resolution in the IEEE 802.15.4 low-rate wireless personal area network protocol", *Proc. 2nd Int. Symp. Leveraging Applications of Formal Methods, Verification and Validation (ISOLA'06)*, 2006.

[HEN 92] HENZINGER T., MANNA Z., PNUELI A., "What good are digital clocks?", KUICH W., Ed., *Proc. 19th Int. Colloquium on Automata, Languages and Programming (ICALP'92)*, vol. 623 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 545–558, 1992.

[HEN 94] HENZINGER T., NICOLLIN X., SIFAKIS J., YOVINE S., "Symbolic model checking for real-time systems", *Information and Computation*, vol. 111, num. 2, p. 193–244, 1994.

[HIN 06] HINTON A., KWIATKOWSKA M., NORMAN G., PARKER D., "PRISM: A tool for automatic verification of probabilistic systems", HERMANNS H., PALSBERG J., Eds., *Proc. 12th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, vol. 3920 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 441-444, 2006.

[IEE 95] IEEE 1394-1995, High Performance Serial Bus Standard, 1995.

[JEN 96] JENSEN H., "Model checking probabilistic real time systems", BJERNER B., LARSSON M., NORDSTRÖM B., Eds., *Proc. 7th Nordic Workshop on Programming Theory*, Report 86, Chalmers University of Technology, p. 247–261, 1996.

[KEM 76] KEMENY J., SNELL J., KNAPP A., *Denumerable Markov Chains*, Springer-Verlag, 2nd edition, 1976.

[KWI 01] KWIATKOWSKA M., NORMAN G., SPROSTON J., "Symbolic computation of maximal probabilistic reachability", LARSEN K., NIELSEN M., Eds., *Proc. 13th Int. Conf. Concurrency Theory (CONCUR'01)*, vol. 2154 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 169–183, 2001.

[KWI 02a] KWIATKOWSKA M., NORMAN G., SEGALA R., SPROSTON J., "Automatic verification of real-time systems with discrete probability distributions", *Theoretical Computer Science*, vol. 282, p. 101–150, 2002.

[KWI 02b] KWIATKOWSKA M., NORMAN G., SPROSTON J., "Probabilistic model checking of the IEEE 802.11 wireless local area network protocol", HERMANNS H., SEGALA R., Eds., *Proc. 2nd Joint Int. Workshop Process Algebra and Probabi listic Methods, Performance Modeling and Verification (PAPM/PROBMIV'02)*, vol. 2399 of *LNCS*, Springer, p. 169–187, 2002.

[KWI 03] KWIATKOWSKA M., NORMAN G., SPROSTON J., "Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol", *Formal Aspects of Computing*, vol. 14, p. 295–318, 2003.

[KWI 06] KWIATKOWSKA M., NORMAN G., PARKER D., SPROSTON J., "Performance analysis of probabilistic timed automata using digital clocks", *Formal Methods in System Design*, vol. 29, p. 33–78, 2006.

[KWI 07] KWIATKOWSKA M., NORMAN G., SPROSTON J., WANG F., "Symbolic model checking for probabilistic timed automata", *Information and Computation*, vol. 205, num. 7, 2007.

[LAR 97] LARSEN K., PETTERSSON P., YI W., "UPPAAL in a nutshell", *Int. Journal on Software Tools for Technology Transfer*, vol. 1, num. 1-2, p. 134–152, 1997.

[PRI] PRISM web page, www.theprismmodelchecker.org/.

[RUT 04] RUTTEN J., KWIATKOWSKA M., NORMAN G., PARKER D., *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems,* P. Panangaden and F. van Breugel (eds.), vol. 23 of *CRM Monograph Series*, American Mathematical Society, 2004.

[SEG 95a] SEGALA R., Modelling and verification of randomized distributed real-time systems, PhD thesis, Massachusetts Institute of Technology, 1995.

[SEG 95b] SEGALA R., LYNCH N. A., "Probabilistic simulations for probabilistic processes", *Nordic Journal of Computing*, vol. 2, num. 2, p. 250–273, 1995.

[SIM 01] SIMONS D., STOELINGA M. I. A., "Mechanical verification of the IEEE 1394a root contention protocol using UPPAAL2k", *Int. Journal on Software Tools for Technology Transfer*, vol. 3, num. 4, p. 469–485, Springer-Verlag, 2001.

[STO 02] STOELINGA M., Alea jacta est: verification of probabilistic, real-time and parametric systems, PhD thesis, University of Nijmegen, The Netherlands, 2002.

[TRI 98] TRIPAKIS S., The formal analysis of timed systems in practice, PhD thesis, University of Joseph Fourier, 1998.

[TRI 05] TRIPAKIS S., YOVINE S., BOUAJJANI A., “Checking timed Büchi automata emptiness efficently”, *Formal Methods in System Design*, vol. 26, num. 3, p. 267–292, 2005.