Model Checking for Probabilistic Timed Systems *

Jeremy Sproston

Dipartimento di Informatica, Università di Torino, 10149 Torino, Italy

Abstract. Application areas such as multimedia equipment, communication protocols and networks often feature systems which exhibit both probabilistic and timed behaviour. In this paper, we consider analysis of such probabilistic timed systems using the technique of model checking, in which it is verified automatically whether a system satisfies a certain desired property. In order to describe formally probabilistic timed systems, we consider probabilistic extensions of timed automata, such as real-time probabilistic processes, probabilistic timed automata and continuous probabilistic timed automata, the underlying semantics of each of which is an infinite-state structure. For each formalism, we consider how the well-known region equivalence relation can be used to reduce the infinite state-space model into a finite-state system, which can then be used for model checking.

1 Introduction

The increasing reliance on complex computer systems in diverse fields such as business, transport and medicine has led to increased interest in obtaining formal guarantees of system correctness. In particular, *model checking* is an automatic method for guaranteeing that a mathematical model of a system satisfies a formula representing a desired property [16]. Many real-life systems, such as multimedia equipment, communication protocols, networks and fault-tolerant systems, exhibit *probabilistic* behaviour, leading to the study of *probabilistic model checking* of probabilistic and stochastic models [48, 25, 18, 11, 9]. Similarly, it is common to observe complex *real-time* behaviour in such systems. Model checking of timed systems has been studied extensively in, for example, [4, 27, 50, 39, 12, 53], with much attention given to the formalism of *timed automata* [5]. In this paper, we present an overview of model checking techniques for probabilistic extensions of timed automata. Note that our aim is not to explore all methods for model checking probabilistic timed models; model checking has also been developed for stochastic processes such as Discrete-Time Markov Chains [25], Continuous-Time Markov Chains [8], and Semi-Markov Chains [41].

A timed automaton consists of a finite graph equipped with a finite set of real-valued variables called *clocks* which increase at the same rate as real-time. Changes in the values of the clocks can enable or disable edges of the graph;

* Supported in part by the EU within the DepAuDE project IST-2001-25434.

furthermore, an edge traversal may result in a clock being reset to 0. In general, we consider the verification of a timed automaton model against a timed property, for instance "a request will always be followed by a response within 5 milliseconds". Such timed properties can be expressed in timed extensions of temporal logic such as TCTL [4,27]. A problem is that the underlying semantic model of a timed automaton is infinite-state, as clocks, which characterize partially each state, are real-valued. Recall that a fundamental requirement of traditional model checking, as exemplified by the seminal work of [15,44], is that the state space of the labelled transition system is of *finite* cardinality. This restriction implies that the fixpoint computations of the model-checking algorithm, which compute progressively sets of states which satisfy given temporal logic formulae, will terminate, because there is a finite number of state sets that can be computed, implying in turn that the entire model-checking algorithm will terminate. A key notion for obtaining decidability and complexity results for timed automata model checking is *clock equivalence* [5,4]. This equivalence relation guarantees that the future behaviour of the system from two clock-equivalent states will be sufficiently similar from the point of view of timed properties. More precisely, the same TCTL properties are satisfied by clock equivalent states. Furthermore, the fact that this equivalence has a finite number of classes for any timed automaton means that we are able to obtain a finite state space on which TCTL model checking can be performed. After a section of preliminaries, we review timed automata and their clock-equivalence-based model-checking methods in Section 3.

This paper surveys the following probabilistic extensions of timed automata. In Section 4, we consider *real-time probabilistic processes* [2,3]. This formalism, which is similar to generalized semi-Markov processes [49,24], resets clock values not to 0, as in the case of timed automata, but to negative values drawn from continuous density functions. Clocks then trigger edge traversals in the graph of the model when their value reaches 0. We focus on *qualitative* ("almost-sure") probabilistic timed properties, which refer to the satisfaction of a property with probability strictly greater than 0 or with probability 1 only. For example, we can express properties such as "a request will be followed by a response within 5 milliseconds with probability 1". Clock equivalence can be used to obtain a finite-state system which represents the behaviour of a real-time probabilistic process, which can then be subject to traditional graph-theoretic model-checking methods.

The formalism of *probabilistic timed automata* [30,35] is considered in Section 5. Probabilistic timed automata extend both timed automata and *Markov decision processes* [23,43], which are stochastic processes in which both nondeterministic behaviour and probabilistic behaviour coexist. Therefore, probabilistic timed automata are timed automata for which probabilistic choice may be made over certain edges of the graph. This formalism has proved particularly useful for the study of timed randomized algorithms, such as those used to obtain the election of a leader in the IEEE1394 FireWire standard [38] and those used in "backoff" rules in communication protocols [37]. Probabilistic timed automata

may be verified against *quantitative* probabilistic timed properties, which refer to the satisfaction of a property with a certain probability. An example of such a property is "a request will be followed by a response within 5 milliseconds with probability 0.99 or greater". Model-checking methods for probabilistic timed automata can use clock equivalence to obtain a finite-state Markov decision process, which can subsequently be subject to probabilistic-model-checking methods [11, 9], such as those implemented in the tool PRISM [32].

Finally, *continuous probabilistic timed automata* [34] feature a combination of aspects of real-time probabilistic processes and probabilistic timed automata, and are considered in Section 6. This formalism exhibits both nondeterministic choice and probabilistic choice, like probabilistic timed automata, and has the ability to reset clocks according to continuous density functions, like real-time probabilistic processes. We consider model checking quantitative probabilistic timed properties of continuous probabilistic timed automata. The application of clock equivalence yields a finite-state Markov decision process which *conservatively approximates* the probability with which a property is satisfied, in the sense that we can obtain an interval of probabilities within which the true probability is contained.

For surveys considering timed automata and real-time probabilistic processes in more depth, we refer the reader to [53, 1] and [17], respectively.

2 Clocks

The models studied in this paper are defined as operating in *continuous real-time*. That is, system transitions can occur at any point within the continuum defined by the set of reals. In order to analyze the behaviour of the models, we use the notion of multidimensional real-valued space, the points in which characterize partially the particular system configurations at a certain point in time. Let $\mathcal{X}$ be a finite set of real-valued variables called *clocks*, the values of which are interpreted as increasing at the same rate as real-time. Using $\mathbb{R}$ to denote the set of real numbers, we refer to a point $val \in \mathbb{R}^{|\mathcal{X}|}$ within $|\mathcal{X}|$-dimensional, real-valued space as a *clock valuation*. [1] For a clock $x \in \mathcal{X}$ and a clock valuation val, we use val_x to denote the value of x with regard to val (formally, the value val_x is the projection of val on the x-axis). Let $\mathbf{0} \in \mathbb{R}^{|\mathcal{X}|}$ be the clock valuation which assigns 0 to all clocks in $\mathcal{X}$. Denoting by $\mathbb{R}_{\geq 0}$ the set of non-negative real numbers, then for any clock valuation $val \in \mathbb{R}^{|\mathcal{X}|}$ and value $\delta \in \mathbb{R}_{\geq 0}$, we use $val + \delta$ to denote the clock valuation obtained from the values $(val + \delta)_x = val_x + \delta$ for all clocks $x \in \mathcal{X}$. Finally, for a clock set $X \subseteq \mathcal{X}$, we let $val[X := 0]$ be the clock valuation for which all clocks within X are set to 0; formally, we let $val[X := 0]$ be the clock valuation val' obtained from letting $val'_x = 0$ for all $x \in X$ and letting $val'_x = val_x$ for all other clocks

[1] Our definition of clock valuations allows both positive and negative reals to be assigned to clocks. In later sections, we will consider usually clock valuations which assign *only* non-negative or *only* non-positive values to clocks.

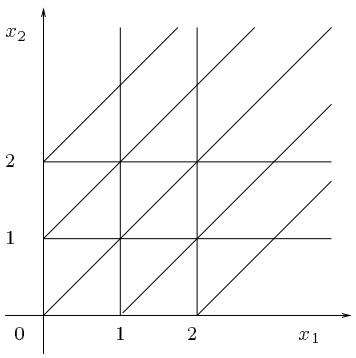

Fig. 1. The partition induced by clock equivalence with max = 2.

$x \in \mathcal{X} \setminus X$. Operations such as $val + \delta$ are called *clock increments* and those such as $val[X := 0]$ are called *clock assignments*.

For a real number $q \in \mathbb{R}$, let $\lfloor q \rfloor$ be the integer part of q. Let max $\in \mathbb{N}$ be a natural number. Given the clock set $\mathcal{X}$, we define *clock equivalence (with respect to* max*)* to be the relation $\equiv_{\max} \subseteq \mathbb{R}_{\geq 0}^{|\mathcal{X}|} \times \mathbb{R}_{\geq 0}^{|\mathcal{X}|}$ over valuations within non-negative real-valued space, where $val \equiv_{\max} val'$ if and only if:

- for each clock $x \in \mathcal{X}$, either both $val_x >$ max and $val'_x >$ max or:
  - $\lfloor val_x \rfloor = \lfloor val'_x \rfloor$ (clock values have the same integer part), and
  - $val_x - \lfloor val_x \rfloor = 0$ if and only if $val'_x - \lfloor val'_x \rfloor = 0$ (the fractional parts of the clocks' values are either all zero or all positive); and
- for each pair of clocks $x, y \in \mathcal{X}$, either $\lfloor val_x - val_y \rfloor = \lfloor val'_x - val'_y \rfloor$ (the integer part of the difference between clocks values is the same) or both $val_x - val_y >$ max and $val'_x - val'_y >$ max.

Note that the second point 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 max. For an example of two clock valuations val, val' which have the same ordering on the fractional parts on clocks in the set $\mathcal{X} = \{x_1, x_2, x_3\}$, consider $val_{x_1} = 0.1$, $val_{x_2} = 0.25$ and $val_{x_3} = 0.8$ and $val'_{x_1} = 0.5$, $val'_{x_2} = 0.9$ and $val'_{x_3} = 0.95$. Instead, the clock valuation $val''_{x_1} = 0.3$, $val''_{x_2} = 0.75$ and $val''_{x_3} = 0.6$ does *not* have the same ordering on the fractional parts as val and val'. 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 1, where the natural number max is equal to 2; each vertical, horizontal or diagonal line segment, open area, and point of intersection of lines is a distinct clock equivalence class.

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 $val \in \alpha$, there exists $\delta \in \mathbb{R}_{\geq 0}$ such that $val + \delta \in \beta$ and $val + \delta' \in \alpha \cup \beta$ for all $0 \leq \delta' \leq \delta$. A clock equivalence class α is *unbounded* if, for all $val \in \alpha$ and

$\delta \in \mathbb{R}_{\geq 0}$, we have $val + \delta \in \alpha$; in the case of an unbounded clock equivalence class α, we let the time successor of α be α itself. Finally, we define the effect of clock assignment on clock equivalence classes. For a clock equivalence class α and clock set $X \subseteq \mathcal{X}$, we use $\alpha[X := 0]$ to denote the clock equivalence class defined by the set $\{val[X := 0] \mid val \in \alpha\}$ (that this set defines a clock equivalence class can be verified easily). For a clock equivalence class α and a subset $X \subseteq \mathcal{X}$ of clocks, let $\alpha \lfloor_X$ be the restriction of the valuations contained within α to the clocks in X.

The set $\Psi_{\mathcal{X}}$ of *clock constraints* over $\mathcal{X}$ is defined by the following syntax:

$$\psi ::= x \sim c \mid x - y \sim c \mid \psi \wedge \psi \mid \neg\psi$$

where $x, y \in \mathcal{X}$ are clocks, $\sim \in \{<, \leq\}$ is a comparison operator, and $c \in \mathbb{N}$ is a natural. We abbreviate the clock constraint $\neg(x < c)$ by $x \geq c$, the clock constraint $\neg(x \leq c)$ by $x > c$, and the clock constraint $x \leq c \wedge x \geq c$ by $x = c$. Furthermore, we denote a trivially-satisfied clock constraint such as $\neg(x < c \wedge x > c)$ by `true`. The clock valuation *val satisfies* the clock constraint $\psi \in \Psi_{\mathcal{X}}$, written $val \models \psi$, if and only if ψ resolves to true after substituting each clock $x \in \mathcal{X}$ with the corresponding clock value val_x from val. Similarly, a clock equivalence class α satisfies ψ, written $\alpha \models \psi$, if and only if $val \models \psi$ for all $val \in \alpha$; in this case, we say that α is *ψ-satisfying*. Observe that every clock equivalence class is a set of valuations definable by a clock constraint.

3 Timed automata

3.1 Modelling with timed automata

We consider the task of modelling systems using the various formalisms featured in this paper by using the running example of a light switch. In this section, we model the action of a light which, after its switch is pressed, stays illuminated for between 2 and 3 minutes. A timed automaton modelling this simple system is shown in Figure 2. The model comprises of two nodes, called *locations*, which are connected by *edges*. The annotations on the nodes and edges denote clock constraints and clock assignments, both of which refer to the single clock x of the model. The behaviour of the timed automaton takes the following form. Time elapses while the timed automaton remains within a certain location, increasing the value of the clock x. It is possible to use a particular edge to switch between locations only if the condition on the values of clocks which annotate the edge is satisfied by the current values of the clocks. Edge traversals are instantaneous. On traversal of an edge, the clocks within the clock set which forms the second annotation to the edge (within { }) are reset to the value 0; the values of all other clocks remain the same. We also note that conditions on the values of clocks which label the locations of the timed automaton can prevent time elapsing beyond a certain bound. More precisely, the timed automaton can remain within a location, letting time pass, only as long as the current values of the clocks satisfy the condition which annotates that location (for example, the

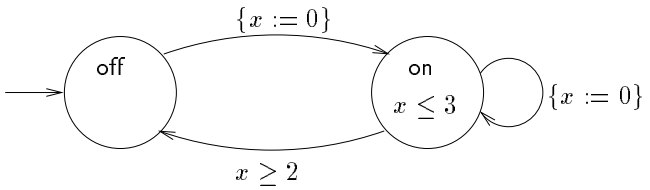

Fig. 2. A timed automaton modelling a switch.

clock constraint $x \leq 3$ which labels the location `on` in Figure 2). The absence of a clock constraint annotation on locations or edges refers to `true`, whereas the absence of a clock assignment annotation refers to the empty assignment $\{\emptyset := 0\}$.

The timed automaton in Figure 2 commences its behaviour in the location `off`. The edge to location `on` can be taken at any point, corresponding to the switch being pressed, and results in the value of x being reset to 0. In location `on`, the timed automaton can return to `off` after 2 minutes have elapsed, and not after 3 time units have elapsed, provided that the switch is not pressed again. The self-loop edge of `on` represents the switch being pressed while the light is on, in which case the timed automaton may leave (must leave) the location after 2 minutes (before more than 3 minutes, respectively).

3.2 Syntax of timed automata

Let AP be a set of atomic propositions, which we henceforth assume to be fixed. A *timed automaton* $\mathsf{TA} = (L, \bar{l}, \mathcal{X}, \mathit{inv}, E, \mathcal{L})$ comprises:

- a finite set L of *locations*, of which one $\bar{l} \in L$ is designated as being the *initial location*,
- a finite set $\mathcal{X}$ of *clocks*,
- a function $\mathit{inv} : L \to \Psi_{\mathcal{X}}$ associating an *invariant condition* with each location,
- a set $E \subseteq L \times \Psi_{\mathcal{X}} \times 2^{\mathcal{X}} \times L$ of *edges*,
- and a *labelling function* $\mathcal{L} : L \to 2^{AP}$ associating a set of atomic propositions with each location.

The invariant condition *inv* takes the form of an assignment of a clock constraint to each location, which gives the set of clock valuations that are admissible in the location. Each edge $(l, g, X, l') \in E$ is determined by (1) its source location l and target location l', (2) its *guard* g, and (3) the set X of clocks called its *clock reset*. The role of a guard is to define a condition on the values of the clocks which enable the edge for choice, while the role of the clock reset is to specify which clocks should be reset to 0 on traversal of the edge.

The behaviour of a timed automaton starts in the initial location with all clocks set to 0. Time may then elapse, making the clocks increase in value, while

the model remains in the initial location. Such passage of time is limited by the invariant condition of the location, in the sense that time can elapse only while the clock values satisfy the location's invariant condition. Control of the timed automaton can leave the location via one of its outgoing edges at any point when the current values of the clocks satisfy the guard of the relevant edge. When an edge is traversed, the clocks within the clock reset are set to 0; then the behaviour of the timed automaton continues anew in the manner described above in the new location. Observe that the choice of when an edge is taken, and of which edge is selected, is purely nondeterministic; there is no probabilistic choice within the model. The time spent in any given location may be 0, given that a guard is satisfied upon entry of the location.

Finally, note that the use of timed automata for modelling complex, real-life systems is aided by a naturally-defined parallel composition operator [5], which we omit from this paper for simplicity. In order to achieve parallel composition, the definition of timed automata is extended with a set of *actions*, where every edge is labelled by an action. Intuitively, two timed automata which are composed in parallel synchronize edge traversals with shared actions, and are able to traverse edges labelled with other, "non-shared" actions independently, following the precedent of process-algebraic theories of parallel composition.

3.3 Semantics of timed automata

A *transition system* $\mathsf{TS} = (S, \bar{s}, T, lab)$ comprises a set S of states, of which $\bar{s} \in S$ is declared as being the initial state, a transition relation $T \subseteq S \times S$, and a labelling function $lab : S \to 2^{AP}$ associating a set of atomic propositions with each state. We often denote a transition $(s, s') \in T$ as $s \to s'$, and choose occasionally to label the arrow with an extra symbol, for example $s \xrightarrow{symbol} s'$. A *path* of TS is an infinite sequence of transitions $s_0 \to s_1 \to \cdots$. We denote by $Path(s)$ the set of paths of TS starting in the state $s \in S$.

The semantics of a timed automaton $\mathsf{TA} = (L, \bar{l}, \mathcal{X}, inv, E, \mathcal{L})$ is expressed formally in terms of an infinite-state transition system $\mathsf{TS} = (S, \bar{s}, T, lab)$. The state set S of the transition system comprises pairs of the form (l, val), the first element of which is a location, and the second element of which is a clock valuation which satisfies the location's invariant condition. Formally, we let $S = \{(l, val) \in L \times \mathbb{R}_{\geq 0}^{|\mathcal{X}|} \mid val \models inv(l)\}$. The initial state $\bar{s}$ of the transition system is defined as $(\bar{l}, \mathbf{0})$. The labelling function lab of TS is derived from the labelling function of TA by letting $lab(l, val) = \mathcal{L}(l)$. Transitions between states are derived from the time-elapse and edge-traversal behaviour of the timed automaton. Formally, we define $T \subseteq S \times S$ to be the smallest relation such that $((l, val), (l', val')) \in T$ if both of the following conditions hold:

Time elapse: there exists $\delta \in \mathbb{R}_{\geq 0}$ such that $val + \delta' \models inv(l)$ for all $0 \leq \delta' \leq \delta$, and

Edge traversal: there exists an edge $(l, g, X, l') \in E$ such that $val + \delta \models g$, $val' \models inv(l')$ and $val' = val[X := 0]$.

We use $s \xrightarrow{\delta} s'$ to denote that $(s, s') \in T$ and that $\delta \in \mathbb{R}_{\geq 0}$ can be used in the time-elapse part of this transition. Intuitively, the duration of a transition $s \xrightarrow{\delta} s'$ is the non-negative real δ. Note that the fact that clocks are real-valued means that the transition system is generally infinite-state, and that the real-valued durations of time that may elapse from a state means that the transition system is generally infinitely branching.

A behaviour of the timed automaton corresponds to an infinite path comprising of transitions of its transition system. Not all behaviours of timed automata are of interest: certain behaviours correspond to an infinite number of edge traversals made within a bounded amount of time, which contradicts our intuition that the operation of a real-time system cannot be "infinitely fast". Therefore, we would like to disregard such unrealisable behaviours during model checking analyses. Formally, a path $s_0 \xrightarrow{\delta_0} s_1 \xrightarrow{\delta_1} \cdots$ of the transition system of a timed automaton is a *divergent* path if the infinite sum $\sum_{i \geq 0} \delta_i$ diverges. We use $Path_{div}(s)$ to denote the set of divergent paths starting in state $s \in S$. As will be seen in the next section, we consider model checking over divergent paths only. Finally, we say that a timed automaton is *non-Zeno* if there exists a divergent path commencing in each state.

3.4 Model checking timed automata

We now proceed to describe how model checking of timed automata can be made possible using clock equivalence, following [4,5]. First, we consider the property description languages that can be used to express requirements of timed automata.

Timed properties. The timed temporal logic TCTL adapts the classical branching-time temporal logic CTL [15] with the means to express quantitative timing constraints on the system's timed behaviour. For example, the logic TCTL can express requirements such as "a response always follows a request within 5 microseconds". TCTL has been introduced in two main forms: in [4], the syntax of the "until" operator of CTL is augmented by a time constraint to obtain a "time-bounded until" path operator $\phi_1 \mathcal{U}_{\sim c} \phi_2$, where ϕ_1, ϕ_2 are TCTL formulae, $\sim \in \{<, \leq, =, \geq, >\}$ is a comparison operator and $c \in \mathbb{N}$. The path operator $\phi_1 \mathcal{U}_{\sim c} \phi_2$ is evaluated, not on a state, but on a path, and, if satisfied by the path, means intuitively "ϕ_2 is true at some point along the path, where the total time elapsed at this point satisfies $\sim c$, and ϕ_1 is true until this point". As with CTL, such a path operator must be nested immediately within an existential or universal path quantifier to obtain a CTL-like formula which is true or false in a state. The syntax of TCTL is stated formally as follows:

$$\phi ::= a \mid \phi \vee \phi \mid \neg\phi \mid \phi \exists \mathcal{U}_{\sim c} \phi \mid \phi \forall \mathcal{U}_{\sim c} \phi$$

where $a \in AP$ is an atomic proposition, and $\sim$ and $c \in \mathbb{N}$ are defined as above. In the sequel, we use occasionally $\Diamond_{\sim c} \phi$ to abbreviate the path formula $\mathtt{true} \mathcal{U}_{\sim c} \phi$ (where `true` denotes a formula which is trivially true in all states, such as $a \vee \neg a$).

A second syntax of TCTL is given in [27], and differs with respect to the manner in which time-constrained properties are specified. Here, a formula can refer explicitly to clocks of the system using clock constraints, and can also refer to a new set of *formula clocks* which can be set to 0 by special *reset quantifiers* of the formula. Although this second syntax can express all of the operators of the first, we do not consider it here for simplicity.

We return to the previous syntax of TCTL, and introduce the notion of position along a path to identify the exact state at any point along a path, including at a time point which falls during the time-elapse part of a transition. A *position* of a path $\omega = s_0 \xrightarrow{\delta_0} s_1 \xrightarrow{\delta_1}$ is a pair $(i, \delta) \in \mathbb{N} \times \mathbb{R}_{\geq 0}$ such that δ is not greater than the duration δ_i of the $(i+1)$th transition $s_i \xrightarrow{\delta_i} s_{i+1}$ of ω. We define a total order $\prec$ on positions of a path by defining $(i, \delta) \prec (j, \delta')$ if and only if either $i < j$, or $i = j$ and $\delta < \delta'$. The *state at position* (i, δ) of ω, denoted by $\omega(i, \delta)$, is defined as $(l_i, val_i + \delta)$, where $(l_i, val_i) = s_i$ is the $(i+1)$th state along ω. We define $\mathsf{duration}(\omega, i, \delta)$ along the path ω at position (i, δ) to be the sum $\delta + \sum_{0 \leq k < i} \delta_k$. Given a state (l, val) of TA, and a TCTL formula ϕ, the satisfaction relation $\models^{\mathrm{T}}$ of TCTL is defined inductively as follows. Note that we express the satisfaction relation for the "until" formulae $\phi_1 \exists \mathcal{U}_{\sim c} \phi_2$ and $\phi_1 \forall \mathcal{U}_{\sim c} \phi_2$ using the formula $\phi_1 \mathcal{U}_{\sim c} \phi_2$, the satisfaction of which relates to a path, rather than to a state.

$$
\begin{array}{lll}
(l, val) \models^{\mathrm{T}} a & \text{iff} & a \in lab(l, val) \\
(l, val) \models^{\mathrm{T}} \phi_1 \vee \phi_2 & \text{iff} & (l, val) \models^{\mathrm{T}} \phi_1 \text{ or } (l, val) \models^{\mathrm{T}} \phi_2 \\
(l, val) \models^{\mathrm{T}} \neg\phi & \text{iff} & \text{not } (l, val) \models^{\mathrm{T}} \phi \\
(l, val) \models^{\mathrm{T}} \phi_1 \exists \mathcal{U}_{\sim c} \phi_2 & \text{iff} & \text{there exists a divergent path } \omega \in Path_{div}(l, val) \text{ such} \\
 & & \text{that } \omega \models^{\mathrm{T}} \phi_1 \mathcal{U}_{\sim c} \phi_2 \\
(l, val) \models^{\mathrm{T}} \phi_1 \forall \mathcal{U}_{\sim c} \phi_2 & \text{iff} & \text{for all divergent paths } \omega \in Path_{div}(l, val), \text{ we have} \\
 & & \omega \models^{\mathrm{T}} \phi_1 \mathcal{U}_{\sim c} \phi_2 \\
\omega \models^{\mathrm{T}} \phi_1 \mathcal{U}_{\sim c} \phi_2 & \text{iff} & \text{there exists a position } (i, \delta) \text{ of } \omega \text{ such that} \\
 & & \omega(i, \delta) \models^{\mathrm{T}} \phi_2, \ \mathsf{duration}(\omega, i, \delta) \sim c, \text{ and for all} \\
 & & \text{positions } (j, \delta') \text{ of } \omega \text{ such that } (j, \delta') \prec (i, \delta), \text{ either} \\
 & & \omega(j, \delta') \models^{\mathrm{T}} \phi_1 \text{ or both } \omega(j, \delta') \models^{\mathrm{T}} \phi_2 \text{ and} \\
 & & \mathsf{duration}(\omega, j, \delta') \sim c \ .
\end{array}
$$

We say that the timed automaton TA with the initial location $\bar{l}$ satisfies the TCTL formula ϕ if and only if $(\bar{l}, \mathbf{0}) \models^{\mathrm{T}} \phi$.

In addition to TCTL, timed automata can themselves be used as a property specification language, given that they are suitably augmented with acceptance conditions, to provide an timed extension of linear-time specification mechanisms such as finite-state automata [5].

The region graph. The primary tool for obtaining decidability and complexity results for timed automata verification is clock equivalence, as presented in Section 2, which is used to define the key notion of *regions*. In the following definition of regions, we assume the timed automaton TA and the TCTL formula ϕ to be fixed, and use $\mathsf{TS} = (S, \bar{s}, Act, T, lab)$ to denote the underlying

transition system of the timed automaton TA. Let max be the maximal natural constant against which any clock is compared in the invariant conditions and guards of TA, or which is included as a time bound of a subformula of ϕ. We define *region equivalence of* TA *and* ϕ to be the relation $\cong_{\max} \subseteq S \times S$, where $(l, val) \cong_{\max} (l', val')$ if and only if:

- $l = l'$ (the location components are equal);
- $val \equiv_{\max} val'$ (the valuations are clock equivalent with respect to max).

We define a *region* to be an equivalence class of $\cong_{\max}$. Region equivalent states satisfy the same TCTL formulae [4]. Note that region equivalence is a bisimulation of the underlying transition system of a timed automaton (not necessarily the coarsest), provided that the duration of time-elapse transitions is abstracted.

A fundamental property of region equivalence is that the number of regions of any timed automaton is *finite*. This allows us to construct a finite-state transition system which represents the behaviour of the timed automaton, where the states of the new system are regions, and transitions are obtained in a natural manner: if some state from region r_1 can reach a state in a region r_2 either by letting time elapse or by traversing an edge, then there is a sequence of transitions in the region graph from r_1 to r_2. Formally, given a timed automaton TA, its underlying transition system TS and its maximal constant max, we define the *region graph* to be the transition system $\mathsf{RG} = (R, \bar{r}, RT, Rlab)$, where R is the finite set of regions, and the initial region $\{(\bar{l}, \mathbf{0})\}$ is denoted by $\bar{r}$. The transition relation of the region graph $RT \subseteq R \times R$ is obtained in the following manner. We define RT to be the smallest relation such that $((l, \alpha), (l', \beta)) \in RT$ if either of the following conditions hold:

Time transition: $l = l'$ and β is the $inv(l)$-satisfying time successor of the clock equivalence class α;

Discrete transition: there exists an edge $(l, g, X, l') \in E$ such that $\alpha \models g$ and $\beta = \alpha[X := 0]$.

Finally, the labelling function $Rlab$ is derived from the labelling function of the timed automaton by letting $Rlab(l, \alpha) = \mathcal{L}(l)$ for all regions $(l, \alpha) \in R$. The significance of the region graph lies in the close correspondence between its paths and those of the underlying transition system of the timed automaton. More precisely, for any region r and any path ω of the region graph which commences in r, we can find a path of the transition system which commences in a state contained within the region r, and which passes through states contained in the regions of the path ω in the correct sequence. Conversely, for any state s of the transition system, and any divergent path ω' which commences in s, we can find a path of the region graph which commences in the region which contains s, and which passes through regions which contain the states of the path ω' in the correct sequence.

Model checking using regions. The region graph RG obtained from a non-Zeno timed automaton TA and a TCTL formula ϕ can be used to establish

whether $\mathsf{TA} \models^{\mathrm{T}} \phi$. The model-checking method is based on the classical labelling algorithms of [15, 44]: it proceeds up through levels of the parse tree of ϕ, from the smallest subformulae of ϕ to the largest (ϕ itself), labelling regions with the subformulae that they satisfy. The manner in which the labelling is done for the atomic propositions and the boolean combinations $\neg\phi$ and $\phi_1 \vee \phi_2$ follows straightforwardly from classical model checking. The case for formulae of the form $\phi_1 \exists\mathcal{U}_{\geq 0} \phi_2$ is also standard: we label a region with $\phi_1 \exists\mathcal{U}_{\geq 0} \phi_2$ if there exists a path from the region in which ϕ_2 is true at some point, and $\phi_1 \vee \phi_2$ is true at all preceding points. Such a labelling can be computed by iterating an "existential" predecessor operator on regions until a fixpoint is obtained. Formulae of the form $\phi_1 \forall\mathcal{U}_{\geq 0} \phi_2$ label a region if, for all paths from the region, we have ϕ_2 true at some point, and $\phi_1 \vee \phi_2$ is true at all preceding points. We can compute a labelling of such formulae again by iterating a "universal" predecessor operation on regions until a fixpoint is obtained.

Formulae of the form $\phi_1 \exists\mathcal{U}_{\sim c} \phi_2$ and $\phi_1 \forall\mathcal{U}_{\sim c} \phi_2$ are handled by introducing an extra clock $z \notin \mathcal{X}$ to the clock set of the timed automaton, and two extra atomic propositions a_b and $a_{\sim c}$ used to label the resulting region graph. More precisely, the atomic proposition a_b labels those "boundary" regions in which no time is allowed to pass (that is, when the value of at least one clock equals an integer), while the atomic proposition $a_{\sim c}$ labels those regions for which $val_z \sim c$, for all valuations val of that region. Then a region (l, α) is labelled with $\phi_1 \exists\mathcal{U}_{\sim c} \phi_2$ if there exists a path starting in the region $(l, \alpha[z := 0])$ in which $\phi_2 \wedge a_{\sim c} \wedge (a_b \vee \phi_1)$ is true at some point and ϕ_1 is true at all preceding points. The reason for the conjunct $(a_b \vee \phi_1)$ is that, if the region is not a boundary region, then some time may pass in the region, and, as we want ϕ_1 to hold at all points before ϕ_2 holds, then ϕ_1 must also hold in the region. A similar rule holds for $\phi_1 \forall\mathcal{U}_{\sim c} \phi_2$. These algorithms can also be implemented using fixpoint computations.

The model-checking algorithm will terminate, because the predecessor operations used to compute fixpoints return sets of regions, of which there are a finite number. The correctness of the algorithm follows from the fact that TA is non-Zeno, and because region-equivalent states satisfy the same TCTL formulae. Hence, we have a method for establishing whether TA satisfies ϕ. Combining this with a complexity-theoretic argument considering the size of the region graph yields the result that model checking of timed automata against TCTL specifications is PSPACE-complete [4].

Note that we can check whether a timed automaton is non-Zeno by checking that all states satisfy the formula $\exists\Diamond_{=1}\mathtt{true}$ (that is, all states allow time to progress by one time unit). A method for strengthening the invariant conditions of a timed automaton so that non-Zenoness is guaranteed is given in [27]. Furthermore, [47] gives syntactic (and thus easily checked) conditions for a timed automaton to be *strongly non-Zeno*, in which all loops of the model are required to let at least one time unit elapse. Strong non-Zenoness implies non-Zenoness.

Beyond regions. Although regions are useful to supply the basic decidability and complexity results of timed automata model checking, in practice more

sophisticated methods are used in the verification of timed automata, which we now summarize briefly. The main approach is to use clock constraints to define *sets* of regions, which are then manipulated during model checking algorithms. This approach was pioneered in the papers [27] for TCTL properties, in [50, 39] for reachability properties, and in [12] for mixed logic-automata properties. Both the TCTL and reachability algorithms have been implemented in the tool KRONOS [52], while algorithms for reachability and a subset of liveness properties have been implemented in the tool UPPAAL [6].

The problem of model checking a timed automaton using semantics based on discrete-time, rather than continuous-time, has been addressed by [26, 7, 13].

4 Real-time probabilistic processes

4.1 Modelling with real-time probabilistic processes

In order to introduce the first model for *probabilistic* timed systems featured in this paper, we revisit the example of a light switch. In this section, we assume that the light switches off deterministically 3 minutes after being switched on, and that the light is switched on at a point in time given by a uniform distribution between 1 minute and 30 minutes. The real-time probabilistic process modelling this switch is shown in Figure 3. The model consists of locations and edges, and has two clocks: the clock y models the amount of time remaining before a person wishing to switch on the light arrives, and the clock x models the amount of time remaining before the light switches off automatically. If a clock annotates a location, then it is *scheduled* in that location. Upon entry to a location l, those clocks scheduled in l that were not scheduled in the previous location l' are reset to new, negative values. A scheduled clock may be used to trigger edge traversals from the location when its value reaches 0. An edge which can be triggered by a clock is labelled by the clock within { }. More than one clock may reach 0 at the same time, and therefore edges can also be labelled with sets of clocks, with the interpretation that the edge is triggered if all clocks within the set labelling the edge reach 0. In Figure 3, the clock y is scheduled in the initial location off, and is set initially to a value given by the amount of time before the light switch is used next. More precisely, if this amount of time is δ, then y is set to $-\delta$. Time then elapses and the values of all scheduled clocks increase, until the value of at least one clock becomes 0. In our example, the clock y is the only clock scheduled in off, and will become 0 after δ time units. The clock(s) which are now 0 then trigger outgoing edges labelled with the clock(s). Thus, the edge from off to on is traversed in our example, indicating that a user has switched the light on. This alternation between letting time elapse until at least one clock reaches 0 continues in the subsequent location. In our example, when we newly enter the location on, it is possible that another user may arrive while the light is still on, so y takes a new value corresponding to the arrival time of the next user. Also, the clock x is deterministically set to the value -3 to denote that the light will turn off automatically in 3 minutes. In contrast to the example in Section 3, we assume that if another person arrives, he will not switch the

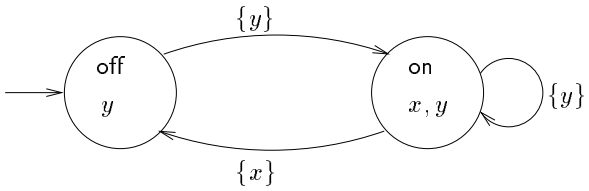

Fig. 3. A real-time probabilistic process modelling a switch.

light on again; this is denoted by the self-loop of the location on labelled by $\{y\}$. Naturally, if x reaches the value 0, the real-time probabilistic process will return to the location off.

The reader familiar with stochastic processes will note that real-time probabilistic processes have many similarities with generalized semi-Markov processes [49, 24].

4.2 Syntax of real-time probabilistic processes

Probability distributions. A *continuous density function* on $\mathbb{R}$ is a function f such that $f(q) \geq 0$ for all $q \in \mathbb{R}$ and $\int_{-\infty}^{\infty} f(q)dq = 1$. Let $\mathsf{CDF}(\mathbb{R})$ be the set of continuous density functions on $\mathbb{R}$. The *support* of a continuous density function is the smallest closed subset of $\mathbb{R}$ which has probability 1. A continuous density function is *positive-bounded* if its support lies within an interval of $\mathbb{R}_{\geq 0}$ with natural numbered end-points. Let $PB \subseteq \mathsf{CDF}(\mathbb{R})$ be the set of positive-bounded continuous density functions.

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

Syntax. A *real-time probabilistic process* $\mathsf{RTPP} = (L, \bar{p}, \mathcal{X}, sched, prob, delay, \mathcal{L})$ [2, 3] comprises:

- a finite set L of *locations*,
- an *initial distribution* $\bar{p} \in \mathsf{Dist}(L)$,
- a finite set $\mathcal{X}$ of *clocks*,
- a *scheduling function* $sched : L \to 2^{\mathcal{X}}$,
- a *probabilistic transition function* $prob : L \times 2^{\mathcal{X}} \to \mathsf{Dist}(L)$,
- a *delay distributions function* $delay : \mathcal{X} \to \mathsf{CDF}(\mathbb{R})$, and
- a *labelling function* $\mathcal{L} : L \to 2^{AP}$.

The initial distribution $\bar{p}$ is a discrete probability distribution which gives the probability of the real-time probabilistic process starting its behaviour in any location. A set $sched(l)$ of clocks from the set $\mathcal{X}$ are *scheduled* when the process

is in a location l, with the intuition that only scheduled clocks can trigger a discrete transition from a location. The probabilistic transition function *prob* is a partial function which associates a discrete probability distribution with pairs comprising a location and a clock set. For any $l \in L$ and $X \subseteq \mathcal{X}$, we require that $prob(l, X)$ is defined only if X is a non-empty subset of $sched(l)$; then $prob(l, X)(l')$ is the probability of making a transition to location l' if the clock set X "expires" (reaches 0) in l. Finally, the delay distribution function *delay* associates a continuous density function with each clock, where we restrict the distributions to be of the following forms:

- distributions of the form $f_x(q) = 1$ for some $q \in \mathbb{N}$; we say that any clock $x \in \mathcal{X}$ is a *fixed-delay* clock if $delay(x)$ is of this form;
- positive bounded continuous density functions f_x with support $[a, b]$ where $a < b$ (therefore $\int_a^b f_x(q)\ dq = 1$); we say that any clock $x \in \mathcal{X}$ is a *variable-delay* clock if $delay(x)$ is of this form;
- exponential distributions f_x with the associated cumulative distribution function $F_x(t) = 1 - e^{-\lambda t}$ for some positive $\lambda \in \mathbb{R}_{>0}$. Therefore, the interpretation of the distribution is that $F_x(t)$ gives the probability that the value selected by the distribution is less than or equal to t. We say that any clock $x \in \mathcal{X}$ is a *exponential-delay* clock if $delay(x)$ is of this form.

Note that [17] defines ways to obtain distributions with more permissive conditions from syntactic combinations of the above distributions. Finally, we note that the previous definitions of real-time probabilistic processes [2, 3], in addition to the literature on generalized semi-Markov processes [49, 24], refers to *events* where we have clocks in our definition above. This changes our *interpretation* of the apparatus used to trigger changes in location, but, as a unique clock is associated with each event in these previous studies, the change does not have a effect on the technicalities with regard to model checking. We therefore choose to include clocks explicitly in our presentation of real-time probabilistic processes in order to be consistent with the definitions of the other models in this paper. The reader more familiar with generalized semi-Markov processes may prefer to regard a clock as indicating an event.

The behaviour of a real-time probabilistic process RTPP takes the following form. The process commences in some location $l \in L$ such that $\bar{p}(l) > 0$, and schedules the set of clocks $sched(l)$ by choosing independently a real value d_x for each clock $x \in sched(l)$ according to the clock's delay distribution $delay(x)$. The value of the clock x is set to the chosen value $-d_x$. Time then passes, and the values of the clocks increase at the same rate, until one or more of the clocks reach the value 0. At this point, a discrete transition is made according to the probability transition function *prob* in the following manner: recalling that the current location is l, and using $\mathsf{early} \subseteq sched(l)$ to denote the set of clocks which have reached 0, we use the discrete distribution $prob(l, \mathsf{early})$ to choose probabilistically the location to which we make the transition. The set of scheduled clocks in the new location l' is determined by partitioning the set $sched(l')$ into two sets:

1. let $old(l, \mathsf{early}, l') = sched(l') \cap (sched(l) \setminus \mathsf{early})$ be the set of clocks that were scheduled in the previous location and did not elapse, and
2. let $new(l, \mathsf{early}, l') = sched(l') \setminus old(l, \mathsf{early}, l')$ be the set of clocks that are newly scheduled.

These sets are used to define the manner in which the clocks take values in the new location l'. Each clock in the set $old(l, \mathsf{early}, l')$ retains its value from before the discrete transition. In contrast, for each clock x in the set $new(l, \mathsf{early}, l')$, a value d_x is randomly and independently chosen from the clock's continuous density function $delay(x)$, and the value of x is set to $-d_x$. The behaviour of the process then proceeds anew with time elapsing until at least one clock reaches the value 0, at which point another discrete transition occurs.

A significant difference between real-time probabilistic processes and timed automata is that the former exhibit *only* probabilistic choice (which manifests itself within the probabilistic setting of clock values and the probabilistic branching between locations on discrete transitions), whereas the latter exhibit *only* nondeterministic choice (which manifests itself within the nondeterministic choice as to the exact time to select edges for choice, and the nondeterministic choice between simultaneously enabled edges).

4.3 Semantics of real-time probabilistic processes

Dense Markov chains. A *dense Markov chain* $\mathsf{DMC} = (S, \Sigma, \bar{\mathbf{p}}, \mathbf{P}, lab)$ consists of a set S of states, a sigma-field Σ over S, an initial distribution taking the form of a probability measure $\bar{\mathbf{p}} : \Sigma \to [0, 1]$, a probabilistic transition function $\mathbf{P} : S \times \Sigma \to [0, 1]$, where $\mathbf{P}(s, \cdot)$ is a probability measure for all states $s \in S$, and a labelling function $lab : S \to 2^{AP}$ such that $\{s \in S \mid lab(s) = a\} \in \Sigma$ for each atomic proposition $a \in AP$.

Semantics. Real-time probabilistic processes are expressed typically in terms of continuous-time Markov processes with continuous state spaces [2, 3, 17]. For consistency with Section 6, we instead present the formal semantics of real-time probabilistic processes using dense Markov chains. More precisely, the semantics of a real-time probabilistic process RTPP is expressed formally in terms of a dense Markov chain $\mathsf{DMC} = (S, \Sigma, \bar{\mathbf{p}}, \mathbf{P}, lab)$. The state set S consists of location-valuation pairs (l, val), although with the following caveat: the valuations are defined only over the *scheduled* clocks of the location l. Hence, for a state $(l, val) \in S$ of the underlying dense Markov chain of a real-time probabilistic process, we have $val \in \mathbb{R}^{|sched(l)|}$. We note that this choice is made for notational convenience rather than from necessity.

The initial distribution of the dense Markov chain $\bar{\mathbf{p}} : \Sigma \to [0, 1]$ is derived from the initial distribution $\bar{p}$ of RTPP and the delay distribution function $delay$ in the following manner. First, for every location $l \in L$, we define the set $fix(l) \subseteq sched(l)$ to be the set of fixed-delay clocks scheduled in l, and $nonfix(l) = sched(l) \setminus fix(l)$ to be the set of variable- and exponential-delay

clocks scheduled in l. Next, we choose an interval or point for each scheduled clock $x \in sched(l)$ in the following way:

1. for each non-fixed, scheduled clock $x \in nonfix(l)$, let $I_x \subseteq (-\infty, 0)$ be an interval of the negative real-line;
2. for each fixed, scheduled clock $x \in fix(l)$, let $I_x = [-q, -q]$, where $-q$ is the unique value for which $f_x(q) = 1$.

Let $(l, I) \subseteq S$ be the set of states with the location l and valuation val for which $val_x \in I_x$ for all scheduled clocks $x \in sched(l)$. Noting that $(l, I) \in \Sigma$, we let:

$$\bar{\mathbf{p}}(l, I) = \bar{p}(l) \cdot \prod_{x \in nonfix(l)} \int_{I_x} f_x \ .$$

An *initial state* of RTPP is a state $(l, val) \in S$ such that $\bar{p}(l) > 0$ and, for all clocks $x \in sched(l)$ scheduled in l, the following conditions hold:

- if $x \in fix(l)$ is a fixed-delay clock, then $-val_x = q$ for the unique value q for which $f_x(q) = 1$;
- if $x \in nonfix(l)$ is a variable- or exponential-delay clock, then $-val_x \in Int_x$, where Int_x is the interior of the support set of f_x.

The transition probability function $\mathbf{P} : S \times \Sigma \to [0, 1]$ is defined in a similar manner to the initial distribution. Consider the source state $(l, val) \in S$ of a transition, let $\mathsf{early} \subseteq sched(l)$ be the set of clocks which have the greatest value in val, and let δ_{early} be this value multiplied by -1 (formally, let $\delta_{\mathsf{early}} = -\max_{x \in sched(l)} val_x$). Now, for each clock $x \in \mathcal{X}$, let I_x be an interval of the real-line obtained in the following manner:

1. for each non-fixed, *newly*-scheduled clock $x \in nonfix(l) \cap new(l, \mathsf{early}, l')$, let $I_x \subseteq (-\infty, 0)$ be an interval of the negative real-line;
2. for each fixed, *newly*-scheduled clock $x \in fix(l) \cap new(l, \mathsf{early}, l')$, let $I_x = [-q, -q]$, where $-q$ is the unique value for which $f_x(q) = 1$;
3. for each *old* clock $x \in old(l, \mathsf{early}, l')$ that continues to be scheduled in the new location, let $I_x = [val_x + \delta_{\mathsf{early}}, val_x + \delta_{\mathsf{early}}]$.

Again, let $(l', I) \subseteq S$ be the set of states with the location l' and valuation val for which $val_x \in I_x$ for all scheduled clocks $x \in sched(l)$. Noting that $(l, I) \in \Sigma$, we then define:

$$\mathbf{P}\big((l, val), (l', I)\big) = prob(l, \mathsf{early})(l') \cdot \prod_{x \in nonfix(l) \cap new(l, \mathsf{early}, l')} \int_{I_x} f_x \ .$$

We use the notation $(l, val) \xrightarrow{\delta_{\mathsf{early}}} (l', val')$ to signify a transition from (l, val) to (l', val'), where $prob(l, \mathsf{early})(l') > 0$ and the valuation val' satisfies the following properties; for each clock $x \in sched(l')$ scheduled in the new location l', we have:

- if $x \in fix(l) \cap new(l, \mathsf{early}, l')$ is a fixed, newly-scheduled clock, then $-val'_x = q$ for the unique value q for which $f_x(q) = 1$;

- if $x \in \mathit{nonfix}(l) \cap \mathit{new}(l, \mathsf{early}, l')$ is a non-fixed, newly-scheduled clock, then $-\mathit{val}'_x \in \mathit{Int}_x$, where Int_x is the interior of the support set of f_x;
- if $x \in \mathit{old}(l, \mathsf{early}, l')$ is an old clock, then $\mathit{val}'_x = \mathit{val}_x$.

Details of how the probability measure *Prob* over sets of infinite sequences of a dense Markov chain of a real-time probabilistic process is obtained can be found in [34].

A computation of a real-time probabilistic process corresponds to an infinite path of its dense Markov chain. As in the case of timed automata, some of these computations correspond to unrealisable behaviour in which an infinite number of transitions occur in a finite amount of time. First we identify the duration of a transition $(l, \mathit{val}) \xrightarrow{\delta_{\mathsf{early}}} (l', \mathit{val}')$ as δ_{early} for (l, val). Then a path $s_0 \xrightarrow{\delta_0} s_1 \xrightarrow{\delta_1} \cdots$ of the dense Markov chain of a real-time probabilistic process is divergent if the infinite sum $\sum_{i \geq 0} \delta_i$ diverges. We say that a real-time probabilistic process is non-Zeno if, for each state, the probability measure over its divergent paths is 1. Unfortunately, it is possible to construct pathological examples of real-time probabilistic processes which are not non-Zeno (for example, a model with a reachable bottom strongly connected component in which only fixed-delay transitions of duration 0 are taken). However, it possible to test whether a real-time probabilistic process is non-Zeno using a property expressed as a deterministic timed Muller automaton.

4.4 Model checking real-time probabilistic processes

Timed properties. The property description language considered in [2] takes the form of a variant of the temporal logic TCTL. Our presentation of the syntax of the logic differs from [2], to emphasize the following point: in contrast to the traditional presentation of TCTL in the context of timed automata, the path quantifiers $\exists$ and $\forall$ of [2] do not refer to the satisfaction of some or all paths, but instead to the satisfaction of paths *with probability greater than 0*, and *with probability 1*, respectively, rather than to some or all paths. We prefer to make this distinction between the interpretation of TCTL on timed automata and on real-time probabilistic processes explicit, and, borrowing notation from quantitative probabilistic logics such as PCTL [25], replace $\exists$ with $\mathbb{P}_{>0}$ and $\forall$ with $\mathbb{P}_{=1}$. Formally, the syntax of the logic, denoted by $\overline{\text{TCTL}}$, is given by:

$$\phi ::= a \mid \phi \vee \phi \mid \neg\phi \mid \mathbb{P}_{>0}(\phi \mathcal{U}_{\sim c} \phi) \mid \mathbb{P}_{=1}(\phi \mathcal{U}_{\sim c} \phi)$$

where $a \in AP$ is an atomic proposition, $\sim \in \{<, \leq, =, \geq, >\}$ is a comparison operator, and $c \in \mathbb{N}$ is a natural constant.

Given a real-time probabilistic process $\mathsf{RTPP} = (L, \bar{p}, \mathcal{X}, \mathit{sched}, \mathit{prob}, \mathit{delay}, \mathcal{L})$ and a $\overline{\text{TCTL}}$ formula ϕ, we define the satisfaction relation $\models^{\overline{\text{T}}}$ of $\overline{\text{TCTL}}$ as follows. As with TCTL, we express the satisfaction of the "until" formulae $\mathbb{P}_{>0}(\phi_1 \mathcal{U}_{\sim c} \phi_2)$ and $\mathbb{P}_{=1}(\phi_1 \mathcal{U}_{\sim c} \phi_2)$ using the path formula $\phi_1 \mathcal{U}_{\sim c} \phi_2$. We omit the semantics of the formulae a, $\phi_1 \vee \phi_2$, $\neg\phi$, and $\phi_1 \mathcal{U}_{\sim c} \phi_2$, as it is the same as that

presented in Section 3 with $\models^{\overline{\mathrm{T}}}$ substituted for $\models^{\mathrm{T}}$. For the remaining operators, we define:

$$(l, val) \models^{\overline{\mathrm{T}}} \mathbb{P}_{>0}(\phi_1 \mathcal{U}_{\sim c} \phi_2) \text{ iff } Prob\{\omega \in Path(l, val) \mid \omega \models^{\overline{\mathrm{T}}} \phi_1 \mathcal{U}_{\sim c} \phi_2\} > 0$$
$$(l, val) \models^{\overline{\mathrm{T}}} \mathbb{P}_{=1}(\phi_1 \mathcal{U}_{\sim c} \phi_2) \text{ iff } Prob\{\omega \in Path(l, val) \mid \omega \models^{\overline{\mathrm{T}}} \phi_1 \mathcal{U}_{\sim c} \phi_2\} = 1 \ .$$

Note that we can use duality to write $\mathbb{P}_{=1}(\phi_1 \mathcal{U}_{\sim c} \phi_2)$ as

$$\neg[\mathbb{P}_{>0}(\Box_{\sim c} \neg\phi_2) \vee \mathbb{P}_{>0}((\neg\phi_2) \mathcal{U}_{\sim c} (\neg\phi_1 \wedge \neg\phi_2))] \ ,$$

where the semantics of $\mathbb{P}_{>0}(\Box_{\sim c}\phi)$ is expressed as:

$$\begin{aligned} (l, val) \models^{\overline{\mathrm{T}}} \mathbb{P}_{>0}(\Box_{\sim c}\phi) \text{ iff } & Prob\{\omega \in Path(l, val) \mid \omega \models^{\overline{\mathrm{T}}} \Box_{\sim c}\phi\} > 0 \\ \omega \models^{\overline{\mathrm{T}}} \Box_{\sim c}\phi \quad \text{iff } & \text{for all positions } (i, \delta) \text{ of } \omega \text{ such that} \\ & \mathsf{duration}(\omega, i, \delta) \sim c, \text{ we have } \omega(i, \delta) \models^{\overline{\mathrm{T}}} \phi \ . \end{aligned}$$

That is, $\Box_{\sim c}$ is a time-bounded version of the standard "globally" operator of temporal logic. Thus we can write any $\overline{\text{TCTL}}$ formula in terms of probabilistic quantifiers of the form $\mathbb{P}_{>0}$ only. Then we say that the real-time probabilistic process RTPP satisfies the $\overline{\text{TCTL}}$ formula ϕ if and only if there exists an initial state s of the underlying Markov process such that $s \models^{\overline{\mathrm{T}}} \phi$.

Finally, we note that deterministic timed Muller automata can be used as property specifications of real-time probabilistic processes, rather than temporal logic properties, as described in [3]. This result is extended in [42] to consider nondeterministic timed Büchi automata (which have the same expressive power as nondeterministic timed Muller automata).

The region graph. In order the define a decidable model checking algorithm for real-time probabilistic processes against $\overline{\text{TCTL}}$ formulae, we resort to a version of clock equivalence, similar to that used for the definition of the region graph of timed automata. In this subsection, we show how a new definition of clock equivalence can be used to obtain a finite-state transition system which corresponds to the real-time probabilistic process. We consider a basic construction of the region graph, which suffices for $\overline{\text{TCTL}}$ formulae of the form $\mathbb{P}_{>0}(\phi_1 \mathcal{U}_{\geq 0} \phi_2)$ and $\mathbb{P}_{=1}(\phi_1 \mathcal{U}_{\geq 0} \phi_2)$ with trivial time bounds, and extend it to non-trivial time bounds in the next subsection.

Assume the real-time probabilistic process RTPP to be fixed. We use $\mathsf{DMC} = (S, \Sigma, \bar{\mathbf{p}}, \mathbf{P}, lab)$ to denote the underlying dense Markov chain of RTPP. Let max be the maximum constant used in the definition of any fixed- or variable-delay clock distribution, and let $\min = -\max$. Given the clock set $X \subseteq \mathcal{X}$, we define *RTPP clock equivalence (with respect to X and* min*)* to be the relation $\equiv_X^{\min} \subseteq [\min, 0]^{|X|} \times [\min, 0]^{|X|}$, where $val \equiv_X^{\min} val'$ if and only if:

- for each clock $x \in X$,
  - $\lfloor val_x \rfloor = \lfloor val'_x \rfloor$ (clock values have the same integer part) and
  - $val_x - \lfloor val_x \rfloor = 0$ if and only if $val'_x - \lfloor val'_x \rfloor = 0$ (the fractional parts of the clocks' values are either all zero or all positive); and

– for each pair of clocks $x, y \in X$, we have $\lfloor val_x - val_y \rfloor = \lfloor val'_x - val'_y \rfloor$ (the integer part of the difference between clocks values is the same).

Observe that RTPP clock equivalence has several minor differences with clock equivalence presented in Section 2. Most obviously, RTPP clock equivalence partitions a section of *negative* clock valuation space, which is bounded from below by min and from above by 0. Instead, the clock equivalence of Section 2 partitions *positive* clock valuation space, and is unbounded from above. Note that we do not consider non-scheduled clocks in clock valuations in this section, and therefore it is not necessary for us to take into account the value of non-scheduled clocks in the clock equivalence.

Now we define *bounded-delay* clocks to be scheduled, non-exponential-delay clocks (therefore, they are either fixed- or variable-delay), and denote the set of bounded-delay clocks scheduled in a location $l \in L$ by $bounded(l)$. By definition, $bounded(l) \subseteq sched(l)$. We define *region equivalence for* RTPP *and* ϕ to be the relation $\asymp \subseteq S \times S$, where $(l, val) \asymp (l', val')$ if and only if:

– $l = l'$ (location components are equal);
– $val \equiv^{\min}_{bounded(l)} val'$ (the valuations over bounded clocks scheduled in l are RTPP clock equivalent with respect to a minimum of min and the clock set $bounded(l)$).

We define a region to be an equivalence class of $\asymp$, and let R be the set of all regions. [2] Note that the equivalence relation $\asymp$ has a finite number of equivalence classes. We identify the regions which are reached just before an edge transition as a *discrete trigger*. Formally, a region $(l, \alpha) \in R$ is a discrete trigger if, for some clock $x \in bounded(l)$, the time successor of α includes a valuation val such that $val_x = 0$.

We are now able to define a finite-state transition system called the region graph $\mathsf{RG} = (R, \bar{R}, RT, Rlab)$ in the following manner. Note that we consider a variant of transition systems which have an initial *set* of states; hence, the initial set of regions is denoted by $\bar{R}$, where $(l, \alpha) \in \bar{R}$ if and only if there exists an initial state (l, val) such that $val \in \alpha$. With respect to the transition relation, let $RT \subseteq R \times R$ be the smallest set such that $(r, r') \in RT$ if either of the following conditions:

Time transitions: we have $r = (l, \alpha)$ where r is not a discrete trigger, and $r' = (l, \beta)$ where β is the time successor of α;

[2] Some comments are in order, as our presentation is somewhat different to that of [2, 3, 17]. The minimum constant min is not used in these previous papers, as the idea there is to define an equivalence relation with an infinite number of equivalence classes, but for which the region graph can be constructed so that it navigates only a finite number of these classes. This can be seen by the fact that, although the partitioning of negative real-valued space in [2, 3, 17] extends towards $-\infty$, clocks will not be set to values lower than min, and therefore regions corresponding to clock values below min will never be considered as targets of the transition relation in the region graph.

Bounded-delay discrete transitions: we have $r = (l, \alpha)$, where r is a discrete trigger, and $r' = (l', \beta)$ is such that:

1. $prob(l, \mathsf{early})(l') > 0$;
2. for each newly-scheduled, fixed-delay clock $x \in \mathit{fix}(l') \cap \mathit{new}(l, \mathsf{early}, l')$, we have $\beta \lfloor_x = -r$ where $f_x(r) = 1$;
3. for each newly-scheduled, variable-delay clock $x \in \mathit{nonfix}(l') \cap \mathit{new}(l, \mathsf{early}, l')$, we have $\beta \lfloor_x \subseteq I_x$ such that $\beta \lfloor_x$ is not an integer, and there does not exist another clock $y \in \mathit{sched}(l')$ such that $\mathit{val}_x - \mathit{val}_y$ is an integer, for some $\mathit{val} \in \beta$;
4. for each old clock $x \in \mathit{old}(l, \mathsf{early}, l')$, we have $\beta_x = \gamma_x$, where γ is the time successor of α.

Exponential-delay discrete transitions: we have $r = (l, \alpha)$, and $r' = (l', \alpha)$ is such that there exists some $x \in \mathit{sched}(l) \setminus \mathit{bounded}(l)$ for which $prob(l, \{x\})(l') > 0$.

Finally, the labelling function Rlab is obtained in the usual manner by letting $\mathit{Rlab}(l, \alpha) = \mathcal{L}(l)$ for all regions $(l, \alpha) \in R$.

As with the timed automaton case, we can identify a correspondence between the paths of the underlying dense Markov chain of a real-time probabilistic process and the associated region graph. That is, for any path of the region graph, we can find a path of the dense Markov chain, the states of which are contained within the regions of the region graph in an appropriate sequence. Similarly, for any path of the dense Markov chain, we can find a path of the region graph with the same kind of correspondence between states and regions.

Model checking using regions. Given the construction of the region graph RG of a real-time probabilistic process RTPP and a $\overline{\text{TCTL}}$ formula ϕ, we are now in a position to consider whether RTPP satisfies ϕ. Without loss of generality, we assume that the $\overline{\text{TCTL}}$ formula ϕ is written using probabilistic quantifiers of the form $\mathbb{P}_{>0}$ only (potentially introducing the path formula $\Box_{\sim c}$). As in Section 3, our strategy is to use the classical labelling algorithm to successively label the regions of RG with the subformulae of ϕ.

Despite the correspondence between paths of a real-time probabilistic process and the region graph, it is not immediate that RG can be used for model checking of RTPP against the $\overline{\text{TCTL}}$ formula ϕ. While the idea of the model checking techniques presented below is to perform graph-theoretical analysis on the region graph to infer probabilistic properties that are sufficient for establishing $\overline{\text{TCTL}}$ formulae, the probabilistic quantifiers $\mathbb{P}_{>0}$ of such formulae do not consider paths with probability 0, and therefore we have to rule out the possibility that a region graph path may correspond to a set of real-time probabilistic process paths which has probability 0. The case in which we have subformulae of the form $\mathbb{P}_{>0}(\phi_1 \mathcal{U}_{\sim c} \phi_2)$ is not problematic: if there exists a finite path in the region graph which satisfies $\phi_1 \mathcal{U}_{\sim c} \phi_2$ (given suitable adjustments to the region graph to cater for the time bound $\sim c$), then there will exist at least one finite path of the real-time probabilistic process which satisfies $\phi_1 \mathcal{U}_{\sim c} \phi_2$, and, as this is a finite path, its probability must be greater than 0. However, in the case of subformulae

of the form $\mathbb{P}_{>0}(\Box_{\sim c}\phi')$ we encounter the following problems. Recall that, in the case of model checking finite-state Markov chains [18], a subformula $\mathbb{P}_{>0}(\Box_{\sim c}\phi')$ is established by checking for a reachable *bottom strongly connected component* (a strongly connected component of the graph of the Markov chain which cannot be exited), all of the states of which are labelled with ϕ'. All of the transition probabilities of states within a bottom strongly connected component of a finite-state Markov chain are fixed, and therefore are bounded from below. Hence, once the bottom strongly connected component is entered, each state within the component is visited infinitely often with probability 1. Instead, it is not immediate that the probabilities of transition between regions of RG obtained from RTPP are bounded from below: for example, if the region r is such that the fractional parts of the clocks x, y, z have the order $0 < x < y < z < 1$, and there occurs a transition to a region in which the fractional parts have the order $0 < y < x < z < 1$, then the probability of the transition is proportional to the difference $z - y$. Given that this transition of the region graph is visited infinitely often, if the value of $z - y$ converges to 0, then the probability of the transition also converges to 0. Therefore, it is established in [2, 3] that this case can never occur: there is a positive probability that the clock values are separated by at least some fixed amount, and hence the graph-theoretical approach, applied to the region graph, can also be used to verify $\mathbb{P}_{>0}(\Box_{\sim c}\phi')$. These results are also necessary in the case of properties expressed as deterministic timed Muller automata [3].

Now we turn to the labelling algorithms used to label RG. The cases of subformulae of the form $\mathbb{P}_{>0}(\phi_1 \mathcal{U}_{\geq 0}\phi_2)$ and $\mathbb{P}_{>0}(\Box_{\geq 0}\phi')$ follow from the model checking algorithms of [18]. Such classical labelling algorithms require adjustment when we consider subformulae of the general form $\mathbb{P}_{>0}(\phi_1 \mathcal{U}_{\sim c}\phi_2)$ and $\mathbb{P}_{>0}(\Box_{\sim c}\phi')$. We proceed by adding a single clock z to the clock set of the real-time probabilistic process, which has the role to measure global time, and to compare its value to c. The clock z is a fixed-delay clock, and is deterministically set to $-c$. We also introduce a special atomic proposition $a_{\sim c}$, which labels regions for which the value of z is ~ 0, and the atomic propositions a_b, which labels boundary regions in which no time can pass. Formally, we define the new real-time probabilistic process $\mathsf{RTPP}^{\sim c} = (L_1 \cup L_2, \bar{p}', \mathcal{X} \cup \{z\}, sched', prob', delay', \mathcal{L}')$. For each location $l \in L$, there are two copies $l_1 \in L_1$ and $l_2 \in L_2$ in the location set of $\mathsf{RTPP}^{\sim c}$; we then let $\bar{p}'(l_1) = \bar{p}'(l_2) = \frac{\bar{p}(l)}{2}$, $sched'(l_1) = sched(l) \cup \{z\}$ and $sched'(l_2) = sched(l)$. Hence, the new clock z is scheduled in L_1 but not in L_2. For each source location $l \in L$, non-empty clock set $X \subseteq \mathcal{X}$ and target location $l' \in L$ such that there exists a probabilistic transition from l to l' via the expiration of clocks in X (formally, $prob(l, X)(l') > 0$), then there exists probabilistic transitions from l_1 to l'_1 and probabilistic transitions from l_2 to l'_2, both of which are triggered by the expiration of clocks in X, where $prob'(l_1, X)(l'_1) = prob'(l_1, X)(l'_2) = prob'(l_2, X)(l'_1) = prob'(l_2, X)(l'_2) = \frac{prob(l,X)(l')}{2}$. Furthermore, for all $l \in L$, we also let $prob'(l_1, \{z\})(l_2) = 1$ (that is, if z expires, then the new real-time probabilistic process makes a transition from the copy of l in which z is scheduled to the other copy).

Our next task is to construct the region graph of $\mathsf{RTPP}^{\sim c}$, which we denote by $\mathsf{RG}^{\sim c} = (R', \bar{R}', RT', Rlab')$. Although the definition of $\mathsf{RG}^{\sim c}$ follows similarly to that presented in the previous subsection, the definition of the labelling function is somewhat different. For each region $r \in R'$, we let $Rlab'(r)$ be the set of labels as described in the previous definition of $Rlab$, but also including the atomic proposition a_b for all boundary regions and the atomic proposition $a_{\sim c}$ if:

- the location component of r is in the set L_1 and $(c + 1 - val_z) \sim c$ for some valuation val contained in the clock equivalence class component of r;
- the location component of r is in the set L_2 and $c + 1 \sim c$.

Then we can apply traditional model checking algorithms to establish which regions satisfy the time-bounded subformulae. For example, for $\mathbb{P}_{>0}(\phi_1 \mathcal{U}_{\sim c} \phi_2)$, we check for a path along which $\phi_2 \wedge a_{\sim c} \wedge (a_b \vee \phi_1)$ holds at some point and ϕ_1 holds at all preceding points (the justification of the conjunct $a_b \vee \phi_1$ is the same as in Section 3).

To conclude, we can establish whether RTPP satisfies ϕ by verifying whether the labelling algorithm labels an initial region in $\bar{R}$ with ϕ.

4.5 Related work

An approach to the qualitative verification of reachability properties of stochastic automata is presented in [19]. Stochastic automata are similar to real-time probabilistic processes but have some nondeterministic choice over edges when clocks expire, and can permit parallel composition of stochastic automaton subcomponents. The approach of D'Argenio is to transform each sub-component into a timed automaton, following which reachability analysis may be performed using timed automata model checking tools such as UPPAAL or KRONOS. The verification is conservative in the sense that if a set of states is reached in the timed automaton model then it is also reached with non-zero probability in the stochastic automaton model, but not necessarily vice versa.

We also report on methods for the verification of models similar to real-time probabilistic processes against *quantitative* probabilistic properties, in which probabilistic quantifiers are more general than those used in this section (that is, the probabilistic quantifiers are of the form $\mathbb{P}_{\bowtie \lambda}$ for $\bowtie \in \{<, \leq, \geq, >\}$ and $\lambda \in [0, 1]$, as introduced in the probabilistic temporal logic PCTL [25]). Model checking quantitative probabilistic properties of semi-Markov chains is considered in [41]. Algorithms for the verification of a restricted class of stochastic automata (for which the model's clocks must be reset on entry to a location, similarly to semi-Markov processes) against quantitative probabilistic properties, are presented in [14]. These algorithms assume that nondeterminism is resolved before they are executed. In [51], a verification method based on *sampling* paths of a generalized-semi Markov process is presented. The method can both be used to obtain an estimate of the probability that a path formula is satisfied and to obtain error bounds for the estimate.

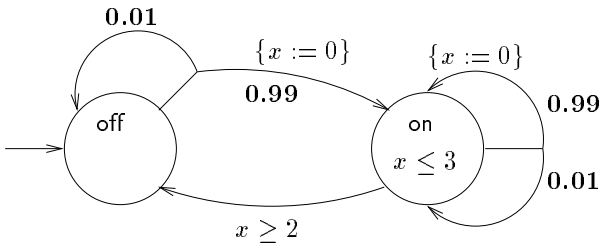

Fig. 4. A probabilistic timed automaton modelling a faulty switch.

5 Probabilistic timed automata

5.1 Modelling with probabilistic timed automata

A probabilistic timed automaton can be used to model the light switch of Section 3 in the presence of faults. Suppose that the mechanism of the switch is faulty, so that there is a 1% chance that, when the switch is pressed, the light will not switch on. The resulting probabilistic timed automaton is shown in Figure 4. Like timed automata, the model consists of locations annotated by clock constraints. However, the edges have changed form: from off and on there exist bifurcating edges that represent probabilistic choice. For example, in off, the light may be switched on at any time, corresponding to the bifurcating edge leaving this location. Once the decision to leave the location has been made, then a probabilistic choice of one branch (for example, the switch fails and we return to off with probability 0.01) or of the other branch (the switch works and we move to on while assigning 0 to the clock x with probability 0.99) is performed.

5.2 Syntax of probabilistic timed automata

A *probabilistic timed automaton* $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, inv, prob, \mathcal{L})$ [30,35] is a tuple, the first four and the final elements of which are borrowed directly from the definition of timed automata: that is, L is a finite set of locations with the initial location $\bar{l}$, we have a finite set $\mathcal{X}$ of clocks, and inv and $\mathcal{L}$ are functions associating an invariant condition and a set of atomic propositions, respectively, with each location. In contrast to the definition of timed automata, the finite set $prob \subseteq L \times \Psi_{\mathcal{X}} \times \mathsf{Dist}(2^{\mathcal{X}} \times L)$ denotes a set of *probabilistic edges*. Each probabilistic edge $(l, g, p) \in prob$ is determined by (1) its source l location, (2) its guard g, and (3) its probability distribution p which assigns probability to pairs of the form (X, l') for some clock reset X and target location l'. Observe that p is a *discrete* probability distribution, because the sets $\mathcal{X}$ and L are finite.

The behaviour of a probabilistic timed automata takes a similar form to that of timed automata: in any location time can advance to an upper limit

given by the restrictions of the invariant condition on clock values, and a probabilistic edge can be taken if its guard is satisfied. However, probabilistic timed automata generalize timed automata in the sense that, once a probabilistic edge is nondeterministically selected, then the choice of which clock reset to use and which target location to make the transition to is *probabilistic*. For example, in Figure 4 there is a probabilistic choice between the switch failing, resulting in a self-loop to the same location with no clock assignments, and the switch working, resulting in a transition to a location with an assignment of 0 to the clock x. Probabilistic timed automata extend timed automata with probabilistic branching, but are incomparable to the real-time probabilistic processes of Section 4, because they feature nondeterministic choice and are unable to set clocks according to continuous density functions.

Note that parallel composition for probabilistic timed automata can be defined by combining the parallel composition theories of timed automata [5] and probabilistic transition systems [46], given the extension of probabilistic timed automata with actions, as shown in [38].

5.3 Semantics of probabilistic timed automata

The semantics of a probabilistic timed automaton is expressed in terms of Markov chains and Markov decision processes.

Markov chains. A *Markov chain* $\mathsf{MC} = (S, \bar{p}, P, lab)$ consists of a countable set S of states, an initial distribution $\bar{p} \in \mathsf{Dist}(S)$ over states, a probabilistic transition function $P : S \times S \to [0,1]$ such that $\sum_{s' \in S} P(s,s') = 1$ for all states $s \in S$, and a labelling function $lab : S \to 2^{AP}$. A probability measure $Prob$ over infinite sequences of states of the Markov chain can be defined in the classical manner [31].

Markov decision processes. A *Markov decision process* $\mathsf{MDP} = (S, \bar{s}, D, lab)$ comprises a set of states S with an initial state $\bar{s}$, a labelling function $lab : S \to 2^{AP}$, and a probabilistic-nondeterministic transition relation $D \subseteq S \times \mathsf{Dist}(S)$. The transitions from state to state of a Markov decision process are performed in two steps: the first step concerns a nondeterministic selection of a discrete probability distribution associated with the current state (that is, if the current state is s, then a distribution is chosen nondeterministically from the set $\{\mu \mid (s,\mu) \in D\}$); the second step comprises a probabilistic choice according to the chosen distribution (that is, given the choice of μ in the first step, we then make a transition to a state $s' \in S$ with probability $\mu(s')$). We often denote such a transition by $s \xrightarrow{\mu} s'$. Note that probabilistic timed automata extend Markov decision processes with the ability of resetting and testing the value of clocks, in the same way as timed automata extend finite automata with the ability of resetting and testing the value of clocks.

An infinite or finite *path* of MDP is defined as an infinite or finite sequence of transitions, respectively. We use $Path_{fin}$ to denote the set of finite paths of

MDP, and $Path_{ful}$ the set of infinite paths of MDP. If ω is finite, we denote by $last(\omega)$ the last state of ω. We abuse notation by letting a state $s \in S$ to denote a path starting in s but consisting of no transitions, and by using $\omega \xrightarrow{\mu} s$ to refer to a path comprising the sequence of transitions of ω followed by the transition $last(\omega) \xrightarrow{\mu} s$. Finally, let $Path^{A}_{fin}(s)$ and $Path^{A}_{ful}(s)$ refer to the set of finite and infinite paths, respectively, commencing in state $s \in S$.

In contrast to a path, which corresponds to a resolution of nondeterministic and probabilistic choice, an *adversary* represents a resolution of nondeterminism *only*. Formally, an adversary of a Markov decision process MDP is a function A mapping every finite path $\omega \in Path_{fin}$ to a distribution $\mu \in \mathsf{Dist}(S)$ such that $(last(\omega), \mu) \in D$ [48]. Let Adv be the set of adversaries of MDP.

We now see how adversaries can be used to express probability measures of certain behaviours of a Markov decision process. For any adversary $A \in Adv$, let $Path^{A}_{fin}$ and $Path^{A}_{ful}$ denote the set of finite and infinite paths associated with A (more precisely, the paths resulting from the choices of distributions of A). Then, for a state $s \in S$, we define the probability measure $Prob^{A}_{s}$ over $Path^{A}_{ful}(s)$ in the following, standard way [48]. We define the Markov chain $\mathsf{MC}^{A}_{s} = (Path^{A}_{fin}(s), \{s \mapsto 1\}, \mathsf{A}, \_)$, where the set of states of the Markov chain is the set of finite paths generated by A starting in s, the initial distribution and the labelling function are arbitrary, and the probabilistic transition function function $\mathsf{A} : Path^{A}_{fin}(s) \times Path^{A}_{fin}(s) \to [0, 1]$ is defined such that:

$$\mathsf{A}(\omega, \omega') = \begin{cases} A(\omega)(s') & \text{if } \omega' = \omega \xrightarrow{A(\omega)} s' \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, $\mathsf{A}(\omega, \omega')$ refers to the probability of obtaining the finite path ω' when extending the finite path ω with one transition under the control of the adversary A. Now we are can define the probability measure $Prob^{A}_{s}$ over infinite paths which share the same finite prefixes, using the Markov chain MC^{A}_{s} obtained from the adversary A and classical methods [31].

Semantics. The semantics of the probabilistic timed automaton $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, inv, prob, \mathcal{L})$ is the infinite-state Markov decision process $\mathsf{MDP} = (S, \bar{s}, D, lab)$ derived in the following manner. The state set S comprises location-valuation pairs, where a valuation must satisfy the invariant of a location with which it is paired. As with timed automata, we let the initial state $\bar{s}$ be $(\bar{l}, \mathbf{0})$, and let the labelling function lab be such that $lab(l, val) = \mathcal{L}(l)$ for all states $(l, val) \in S$. The probabilistic transitions of MDP are defined in the following manner. We define D to be the smallest set such that $((l, val), \mu) \in D$ if both of the following conditions hold:

Time elapse: there exists $\delta \in \mathbb{R}_{\geq 0}$ such that $val + \delta' \models inv(l)$ for all $0 \leq \delta' \leq \delta$, and

Edge traversal: there exists a probabilistic edge $(l, g, p) \in E$ such that $val + \delta \models g$ and, for any $(l', val') \in S$, we have:

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

We refer to the real δ used in the time-elapse part of a transition as the *duration* of the transition. For each $s' \in S$ such that $\mu(s') > 0$, we use $s \xrightarrow{\mu[\delta,p]} s'$ to denote the discrete transition from s to s' with the duration δ using the distribution p. The summation in the definition of discrete transitions is required for the cases in which multiple clock resets result in the same target state (l', val'). Note that we assume a well-formedness condition which stipulates that the invariant condition of a target location of a transition is always satisfied immediately after a transition is taken (for more details, see [38]).

As with our treatment of the semantics of timed automata and real-time probabilistic processes, a behaviour of a probabilistic timed automaton takes the form of an infinite sequence of transitions. We can also extend the idea of divergence and non-Zenoness to probabilistic timed automata: using the precedents of [45, 22], we associate a notion of divergence with *adversaries*. First we say that a path $s_0 \xrightarrow{\mu[\delta_0,p_0]} s_1 \xrightarrow{\mu[\delta_1,p_1]} \cdots$ of the underlying Markov decision process of a probabilistic timed automaton is *divergent* if the infinite sum $\sum_{>0} \delta_i$ diverges. Then, for a state $s \in S$, the adversary $A \in Adv$ is a *divergent adversary from* s if $Prob_s^A\{\omega \in Path_{ful}^A(s) \mid \omega \text{ is divergent}\} = 1$. A probabilistic timed automaton is *non-Zeno* if there exists a divergent adversary from all states.

5.4 Model checking probabilistic timed automata

Probabilistic timed properties. We now proceed to define a *probabilistic*, *timed* temporal logic which can be used to specify properties of probabilistic timed automata. In particular, we note that the logic that we present in this section has the power to refer to *exact* probability values, in contrast to $\overline{\text{TCTL}}$ presented in Section 4, which only had the ability to refer to probability strictly greater than 0 or equal to 1. Our approach is to define a logic which draws ideas from TCTL, and the probabilistic temporal logic PCTL [25], in which a probabilistic quantifier of the form $\mathbb{P}_{\bowtie\lambda}$ is used in place of a path quantifier of CTL, where $\bowtie \in \{<, \leq, \geq, >\}$ is a comparison operator and $\lambda \in [0, 1]$ is a probability.

Formally, the syntax of PTCTL (Probabilistic Timed Computation Tree Logic) [35] is defined as follows:

$$\phi ::= a \mid \phi \vee \phi \mid \neg\phi \mid \mathbb{P}_{\bowtie\lambda}(\phi \mathcal{U}_{\sim c} \phi) \mid \mathbb{P}_{\bowtie\lambda}(\phi \mathcal{U}_{\sim c} \phi)$$

where $a \in AP$ is an atomic proposition, and $\bowtie$ and λ are the comparison operator and the probability defined above. Given a probabilistic timed automaton $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, inv, prob, \mathcal{L})$ and a PTCTL formula ϕ, we define the satisfaction relation

$\models^{\mathrm{PT}}$ of PTCTL as follows. The semantics of the formulae a, $\phi_1 \vee \phi_2$, $\neg\phi$, and $\phi_1 \mathcal{U}_{\sim c} \phi_2$ is omitted, as it is the same as that presented for TCTL in Section 3 with $\models^{\mathrm{PT}}$ substituted for $\models^{\mathrm{T}}$. We then define:

$(l, val) \models^{\mathrm{PT}} \mathbb{P}_{\bowtie \lambda}(\phi_1 \mathcal{U}_{\sim c} \phi_2)$ iff
for all adversaries $A \in Adv$ which are divergent from (l, val),
we have $Prob^A_{(l,val)}\{\omega \in Path^A_{ful}(l, val) \mid \omega \models^{\mathrm{PT}} \phi_1 \mathcal{U}_{\sim c} \phi_2\} \bowtie \lambda$.

The probabilistic timed automaton PTA with the initial location $\bar{l}$ satisfies the PTCTL formula ϕ if and only if $(\bar{l}, \mathbf{0}) \models^{\mathrm{PT}} \phi$.

Probabilistic timed automata can also be verified against properties which refer to the maximal probability with which a Büchi condition is satisfied [10].

The region graph. The fundamental notion of clock equivalence, defined in Section 2, can be used to obtain a finite-state Markov decision process from a probabilistic timed automaton which contains sufficient information to establish or refute PTCTL properties. For the remainder of this section, assume that the probabilistic timed automaton $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, inv, prob, \mathcal{L})$ and the PTCTL formula ϕ are fixed, and that we use $\mathsf{MDP} = (S, \bar{s}, D, lab)$ to denote the underlying Markov decision process of PTA. Let max be the maximal constant against which any clock is compared in the invariant conditions and guards of PTA. Then we define *region equivalence for* PTA as the relation $\simeq \subseteq S \times S$, where $(l, val) \simeq (l', val')$ if and only if:

- $l = l'$ (the location components are equal);
- $val \equiv_{\max} val'$ (the valuations are clock equivalent with respect to max).

Observe that this is the same definition of region equivalence defined for timed automata in Section 3, now applied to the states of the underlying Markov decision process of a probabilistic timed automaton. We define a *region* to be an equivalence class of $\simeq$, and use R to denote the set of regions. Region equivalent states of a probabilistic timed automaton satisfy the same PTCTL formulae [35], and region equivalence is a probabilistic bisimulation (in the sense of [40, 46], but not necessarily the coarsest) of a probabilistic timed automaton, provided that the duration of transitions is abstracted.

The property that the set of clock equivalence classes is finite, and therefore the number of regions is finite, permits us to define a finite-state region graph. Many of the principles involved in the construction of the region graph are borrowed from the corresponding constructions that we have seen for timed automata and real-time probabilistic processes. However, in contrast with the definitions in Section 3 and Section 4, the region graph now incorporates probability values. More precisely, the *region graph of* PTA *and* ϕ is a Markov decision process $\mathcal{RG} = (R, \bar{r}, \mathcal{RT}, Rlab)$, where the initial region $\bar{r} = \{(\bar{l}, \mathbf{0})\}$ and region labelling function $Rlab(l, \alpha) = \mathcal{L}(l)$, for all regions $(l, \alpha) \in R$, are defined in a similar manner as in the case of the region graph of a timed automaton. The probabilistic transition relation $\mathcal{RT} \subseteq R \times \mathsf{Dist}(R)$ is the smallest relation such that $((l, \alpha), \rho) \in \mathcal{RT}$ if either of the following conditions hold:

Time elapse: $\rho = \{(l, \beta) \mapsto 1\}$, where β is an $inv(l)$-satisfying time successor of α;

Probabilistic edge traversal: there exists a probabilistic edge $(l, g, p) \in prob$ such that $\alpha \models g$ 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 fact that the region graph defined above includes probabilistic information, derived from the probabilistic edges of the associated probabilistic timed automaton, allows us to strengthen the notion of correspondence between behaviours of our probabilistic timed system and its associated region graph. More precisely, in addition to showing that every path of the region graph corresponds to at least one path of the probabilistic timed automaton, and that every path of the probabilistic timed automaton corresponds to a path of the region graph, we can also make a stronger statement: every adversary of the probabilistic timed automaton has a corresponding adversary of the region graph, and that every adversary of the region graph corresponds to at least one adversary of the probabilistic timed automaton.

Model checking using regions. Given the construction of the region graph $\mathcal{RG}$ of a probabilistic timed automaton PTA and a PTCTL formula ϕ, we can now proceed to model check $\mathcal{RG}$ to establish whether PTA satisfies ϕ. In analogy with the corresponding case of timed automata, we borrow ideas from finite-state model checking, in this case probabilistic model checking of finite-state Markov decision processes [11, 9]. The technique of extending PTA with an additional clock to count time relative to the time-bound subscripts of the PTCTL formula ϕ, and adding extra atomic propositions a_b and $a_{\sim c}$ to the region graph, is adopted directly from TCTL model checking. Then it remains to use probabilistic model checking techniques to calculate the probability of satisfying subformulae of ϕ in each region, progressing up the parse tree of ϕ until we reach ϕ itself. Then if the initial region $\bar{r}$ satisfies ϕ, we can conclude that PTA satisfies ϕ.

As in the case of model checking timed automata, the assumption of non-Zenoness of the probabilistic timed automaton guarantees that all regions have some successor. We can adapt strong non-Zenoness [47] to probabilistic timed automata to give syntactical conditions which guarantee the non-Zenoness property.

Finally, we note that a region-graph-based construction can also be used to obtain the maximum probability with which a probabilistic timed automaton satisfies a Büchi property [10].

Beyond regions. As with timed automata, the region graph construction for probabilistic timed automata is expensive, and therefore efficient model checking techniques which do not rely explicitly on the region graph are required. Adaptations of methods based on the manipulation of clock constraints are presented in

[35, 36], which consider properties referring to the maximal probability of reaching a certain state set (that is, properties of the form $\mathbb{P}_{\leq c}(\Diamond a)$ or $\mathbb{P}_{<c}(\Diamond a)$). Furthermore, a discrete-time semantics for probabilistic timed automata is defined in [38], and permits direct use of the probabilistic model checker PRISM [32]. Probabilistic timed properties of the IEEE1394 FireWire root contention protocol have been verified using algorithms based on manipulation of clock constraints [38, 21] and the discrete-time semantics [38]. The discrete-time semantics has also been used to facilitate probabilistic model checking of properties of a IEEE802.11 wireless local area network [37] and the IPv4 link-local address configuration protocol [33]. The latter case study also illustrates how the discrete-time semantics may be used to refer to expected costs incurred before the probabilistic timed automaton reaches a certain set of states.

6 Continuous probabilistic timed automata

6.1 Modelling with continuous probabilistic timed automata

We revisit the example of the light switch for a final time in order to introduce the model of continuous probabilistic timed automata. As in Section 4, we consider randomly distributed arrivals of people wanting to use the switch; more precisely, we consider arrivals uniformly distributed between 1 and 30 minutes. Furthermore, as in Section 5, we assume that the switch is faulty. The resulting continuous probabilistic timed automaton is shown in Figure 5. The model combines location annotations indicating clocks (such as y in $\mathsf{off1}$) and clock constraints (such as $x \leq 3 \wedge y \leq 30$ in on). The first type of location annotation gives the clocks which must be reset on entry to the location, whereas the second indicates the location's invariant condition. The edges leaving a location can split into multiple branches, indicating probabilistic edges, such as those used for probabilistic timed automata in Section 5.

In Figure 5, we always set the clock x to 0 on entry to a location with which it is annotated, whereas y is set uniformly to a real value in the interval $[0, 29]$. Then, if y reaches the value 30, we know that between 1 and 30 time units have elapsed since it was set. We use two locations $\mathsf{off1}$ and $\mathsf{off2}$ to model the light being off, as we need to distinguish between the case in which the clock y needs to be set to a new value ($\mathsf{off1}$) and the case in which y should retain its old value ($\mathsf{off2}$).

6.2 Syntax of continuous probabilistic timed automata

A *continuous probabilistic timed automaton* $\mathsf{CPTA} = (L, \bar{l}, \mathcal{X}, \mathit{inv}, \underline{\mathit{prob}}, \mathit{reset}, \mathcal{L})$ [34] is a tuple, the first four and final elements of which are borrowed directly from timed automata. That is, L is a finite set of locations with initial location $\bar{l}$, the finite set $\mathcal{X}$ comprises clocks, and inv and $\mathcal{L}$ are functions associating an invariant condition and a set of atomic propositions, respectively, with each location. The finite set $\underline{\mathit{prob}} \subseteq L \times \Psi_{\mathcal{X}} \times \mathsf{Dist}(L)$ is the set of probabilistic

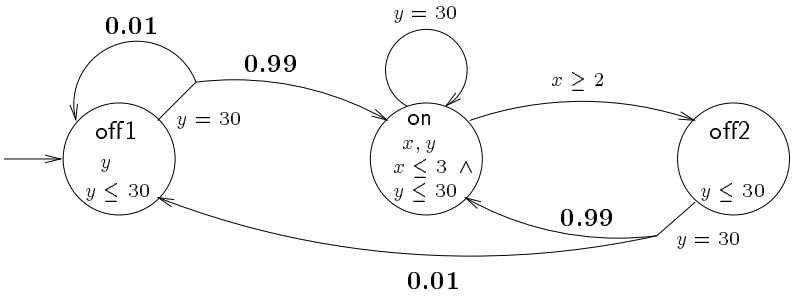

Fig. 5. A continuous probabilistic timed automaton modelling a faulty switch.

edges of the continuous probabilistic timed automaton. Observe that, for each probabilistic edge $(l, g, p) \in \underline{prob}$, the distribution component p is a distribution over the set of locations L *only*, rather than over the set $L \times 2^{\mathcal{X}}$ of pairs of locations and clock resets, as we saw in probabilistic timed automata.

The partial function $reset : L \times \mathcal{X} \rightarrow PB$, referred to as the *clock reset function*, associates a positive bounded density function with certain location-clock pairs. The interpretation of the clock reset function is as follows: if $reset(l, x)$ is defined for a location $l \in L$ and a clock $x \in \mathcal{X}$, then $reset(l, x)$ is the positive-bounded density function which captures the manner in which the clock x is reset upon entry into l. That is, in contrast to the case of probabilistic timed automata, we now do not only consider resetting the values of clocks to the value 0, but we also consider resetting to clocks to *exact, natural numbered values* (if *reset* returns a distribution which assigns probability 1 to a natural number) or according to *random assignments* (if *reset* returns a more general positive-bounded distribution function) in the manner of real-time probabilistic processes. Instead, if $reset(l, x)$ is undefined, then the clock x retains its old value upon entry to the location l. Thus, the behaviour of a continuous probabilistic timed automata derives aspects from all of the previous three models featured previously in this paper: it adopts the ability to select the exact time at which transitions are taken according to nondeterministic choice from timed automata, the discrete probability distributions which can be used to express relative likelihoods of transitions between locations from probabilistic timed automata, and the resetting of clocks to values taken from continuous density functions from real-time probabilistic processes. The invariant conditions and guards of probabilistic edges retain their usual interpretation from timed automata. That is, time can elapse in a location only for as long as the invariant condition is satisfied by the current values of the clocks, and a probabilistic edge can be selected (nondeterministically) for choice at all times when its guard is satisfied by the current values of the clocks. Furthermore, probabilistic edges retain their interpretation from probabilistic timed automata, in the sense that, once nondeterministically selected for choice, they involve a probabilistic choice made according to their

distribution component. The manner in which clocks are reset bears more of a resemblance to the analogous mechanism of real-time probabilistic processes.

Note that we require a well-formedness condition which requires a clock to be set explicitly to a value before it is referred to in the invariant condition of a location or in the guards of one of the outgoing probabilistic edges. Once we have imposed this condition, we can adjust the density function *reset* so that, in the initial location $\bar{l}$, the clocks that were not originally assigned a value are assigned an arbitrary value, for example 0. This final adjustment is made for purely technical reasons, and ensures that our clock valuations are always points in $\mathbb{R}_{\geq 0}^{|\mathcal{X}|}$ space, rather than in positive real-valued space of dimension smaller than $|\mathcal{X}|$, as we saw in the case of real-time probabilistic processes.

Observe that both real-time probabilistic processes and continuous probabilistic timed automata associate a set of clocks to each location (using the functions *sched* and *reset*, respectively), and that these sets determine the manner in which clocks are reset. However, the interpretation of these clock sets is somewhat different for the two models. In real-time probabilistic processes, the set $sched(l)$ contains the clocks which are *scheduled* in a location l; therefore, if such a clock has been scheduled in the previous location, it keeps its old value, otherwise it is reset to a new value. On the other hand, in continuous probabilistic timed automata, *sched* defines the clocks that must be reset to a new value on entry to the location, regardless of the previous location, and all clocks *not* within this set retain their old values. (This type of mechanism is also witnessed in stochastic automata [20].)

It is important to note that continuous probabilistic timed automata are *not* a superclass of real-time probabilistic processes, as they do not support exponentially distributed functions. However, continuous probabilistic timed automata are a superclass of real-time probabilistic processes without such distributions (for example, the real-time probabilistic processes of [2]). A continuous probabilistic timed automaton which emulates a real-time probabilistic process can be constructed in the following way. First, we note that the clocks of continuous probabilistic timed automata take non-negative values only, whereas the clocks of real-time probabilistic processes take values in the interval $[\min, 0]$ only, where min is defined as in Section 4. Therefore, we "shift the axes" by adding the value $-\min$ to each of the constant used in the description of the real-time probabilistic process. Note that this involves constructing invariants and guards using the constraints $x \leq -\min$ and $x = -\min$, respectively, for all scheduled clocks and all locations. Our second task is to ensure that the continuous probabilistic timed automaton which represents the real-time probabilistic process resets the clocks in the correct manner, which can be achieved by adding extra copies of locations.

We can also observe the probabilistic timed automata are a subclass of continuous probabilistic timed automata. In probabilistic timed automata, all clock assignments can be regarded as fixed-delay distributions which set clocks to 0. Then, as in the previous example, we require some bookkeeping to require that the correct set of clocks are reset when the continuous probabilistic timed

automaton makes a discrete transition, which is achieved by adding copies of locations.

6.3 Semantics of continuous probabilistic timed automata

Dense Markov decision processes. A *dense Markov decision process* $\mathsf{DMDP} = (S, \Sigma, \bar{\mathbf{d}}, \mathbf{D}, lab)$ is a quintuple, the first three and the final elements of which are adopted directly from dense Markov chains. That is, DMDP comprises a set S of states, a sigma-field Σ over S, an initial distribution taking the form of a probability measure $\bar{\mathbf{d}} : \Sigma \to [0,1]$, and a labelling function $lab : S \to 2^{AP}$ such that $\{s \in S \mid lab(s) = a\} \in \Sigma$ for each atomic proposition $a \in AP$. The difference between dense Markov chains and dense Markov processes resides in the probabilistic transition relation $\mathbf{D} \subseteq S \times (\Sigma \to [0,1])$, which is defined such that $\mathbf{d}$ is a probability measure for each pair $(s, \mathbf{d}) \in \mathbf{D}$.

Semantics. We now describe formally the semantics of a continuous probabilistic timed automaton $\mathsf{CPTA} = (L, \bar{l}, \mathcal{X}, inv, \underline{prob}, reset, \mathcal{L})$ in terms of a dense Markov decision process $\mathsf{DMDP} = (S, \Sigma, \bar{\mathbf{d}}, \mathbf{D}, \overline{lab})$. As usual, the set S of states consists of pairs of the form (l, val) where $val \models inv(l)$, and the labelling function is defined as $lab(l, val) = \mathcal{L}(l)$ for each state $(l, val) \in S$.

Before proceeding to the definitions of the initial distribution $\bar{\mathbf{d}}$ and the probabilistic transition relation $\mathbf{D}$, we introduce some notation. For each location $l \in L$, let $entry(l) \subseteq \mathcal{X}$ be the set of clocks such that, for each clock $x \in entry(l)$, the function $reset(l, x)$ is defined (that is, $entry(l)$ comprises the clocks reset on entry to l). As in the case of real-time probabilistic processes, let $\mathit{fix}(l) \subseteq entry(l)$ be the set of fixed-delay clocks which are reset in l (formally, we have $x \in \mathit{fix}(l)$ if and only if $reset(l, x)$ is defined and equals $\{q \mapsto 1\}$ for some $q \in \mathbb{N}$). Then let $\mathit{nonfix}(l) = entry(l) \setminus \mathit{fix}(l)$ be the set of variable-delay clocks which are reset on entry to l.

The initial distribution $\bar{\mathbf{d}} : \Sigma \to [0,1]$ is derived from the initial location $\bar{l}$ of CPTA and probability distributions $reset(\bar{l}, x)$, for each clock $x \in \mathcal{X}$ (recall that we assumed $entry(\bar{l}) = \mathcal{X}$). As we saw in the case of real-time probabilistic processes, we choose an interval or point for each clock $x \in \mathcal{X}$ in the following manner:

1. for each non-fixed clock $x \in \mathit{nonfix}(l)$, let $I_x \subseteq (0, \infty)$ be an interval of the positive real-line;
2. for each fixed clock $x \in \mathit{fix}(l)$, let $I_x = [q, q]$, where q is the unique value for which $f_x(q) = 1$.

Let $(\bar{l}, I) \subseteq S$ be the set of states with the initial location $\bar{l}$ and valuation val for which $val_x \in I_x$ for all scheduled clocks $x \in \mathcal{X}$. Noting that $(l, I) \in \Sigma$, we then let:

$$\bar{\mathbf{d}}(\bar{l}, I) = \prod_{x \in nonfix(\bar{l})} \int_{I_x} reset(\bar{l}, x) \ .$$

The probabilistic transition relation $\mathbf{D} \subseteq S \times (\Sigma \to [0,1])$ is defined in a similar manner. As with all of the other models studied in this paper, transitions consist of a time-elapse phase and an edge-traversal phase. We let $\mathbf{D}$ be the smallest relation such that $((l, val), \mathbf{d}) \in \mathbf{D}$ if both of the following conditions hold:

Time elapse: there exists $\delta \in \mathbb{R}_{\geq 0}$ such that $val + \delta' \models inv(l)$ for all $0 \leq \delta' \leq \delta$, and

Probabilistic edge traversal: there exists a probabilistic edge $(l, g, p) \in \underline{prob}$, such that $val + \delta \models g$ and, for each set (l', I) of states comprising some location $l' \in L$ and a set I of valuations satisfying the following conditions:

1. for each $x \in nonfix(l')$, the interval $I_x \subseteq (0, \infty)$ is an interval of the real-line;
2. for each $x \in fix(l')$, the interval I_x is such that $I_x = [q, q]$, where $q \in \mathbb{N}$ is the unique value for which $reset(l', x) = \{q \mapsto 1\}$;
3. for each $x \in \mathcal{X} \setminus entry(l')$, the interval I_x is such that $I_x = [val_x + \delta, val_x + \delta]$;

then we have:

$$\mathbf{d}(l', I) = p(l') \cdot \prod_{x \in nonfix(l')} \int_{I_x} reset(l', x) \ .$$

We denote a transition of the dense Markov process as $(l, val) \xrightarrow{\mathbf{d}[\delta, p]} (l', val')$ if $\delta \in \mathbb{R}_{\geq 0}$ is the duration and p the distribution used to define the transition above, and the target state (l', val') is such that the following conditions hold:

- for each defined clock $x \in nonfix(l')$ for which $reset(l', x)$ has support $[q_1, q_2]$, we have $val'_x \in (q_1, q_2)$;
- for each defined clock $x \in fix(l')$ for which $reset(l', x)$ has support $[q, q]$, we have $val'_x = q$;
- for each non-defined clock $x \in \mathcal{X} \setminus entry(l')$, we have $val'_x = val_x$.

As with probabilistic timed automata, we assume a well-formedness condition which guarantees that a clock never has a value which does not satisfy an invariant condition on entry to a location.

We can define notions of path and adversaries of dense Markov decision processes. An infinite or finite *path* of DMDP is defined as an infinite or finite sequence of transitions, respectively. We use $Path_{fin}$ to denote the set of finite paths of DMDP, and $Path_{ful}$ the set of infinite paths of DMDP. If ω is finite, we denote by $last(\omega)$ the last state of ω, and let a state $s \in S$ denote a path consisting of no transitions. We use $\omega \xrightarrow{\mathbf{d}} s$ to refer to a path comprising the sequence of transitions of ω followed by the transition $last(\omega) \xrightarrow{\mathbf{d}} s$.

An adversary of a dense Markov decision process DMDP is a function A mapping every finite path $\omega \in Path_{fin}$ to a probability measure $\mathbf{d} : \Sigma \to [0,1]$ such that $(last(\omega), \mathbf{d}) \in \mathbf{D}$. Let Adv be the set of adversaries of DMDP. For any adversary $A \in Adv$, let $Path^A_{fin}$ and $Path^A_{ful}$ denote the set of finite and infinite paths associated with A (more precisely, the paths resulting from the choices of dense distributions of A).

Given an adversary $A \in Adv$ of DMDP and a finite path $\bar{\omega} \in Path^A_{fin}$ of A, we define the probability measure $Prob^A_{\bar{\omega}}$ over $Path^A_{ful}$ in the following way. First we define the dense Markov chain $\mathsf{DMC}^A_{\bar{\omega}} = (Path^A_{fin}, \Sigma^A, \{\bar{\omega} \mapsto 1\}, \mathbf{A}, \_)$ obtained from A and $\bar{\omega}$. The set of states of $\mathsf{DMC}^A_{\bar{\omega}}$ are the set of finite paths of A. The sigma-field Σ^A is generated by all of the sets of the following form, where $\omega \in Path^A_{fin}$ is a finite path, $l \in L$ is a location and I is a set of clock valuations obtained from intervals $I_x \subseteq [0, \infty)$ for each defined clock $x \in entry(l)$:

$$\Xi[A, \omega, l, I] = \{\omega \xrightarrow{A(\omega)} (l, val) \in Path^A_{fin} \mid val_x \in I_x \text{ for all } x \in entry(l, x)\} .$$

The initial probability distribution $\{\bar{\omega} \mapsto 1\}$ assigns probability 1 to $\bar{\omega}$. The transition relation $\mathbf{A} : Path^A_{fin} \times \Sigma^A \to [0, 1]$ is defined such that:

$$\mathbf{A}(\omega, \Xi[A, \omega', l', I]) = \begin{cases} A(\omega)(l', I) & \text{if } \omega' = \omega \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, $\mathbf{A}(\omega, \Xi[A, \omega', l', I])$ refers to the probability of obtaining the set of finite paths $\Xi[A, \omega', l', I]$ when extending the finite path ω with one transition under the control of the adversary A. Now that we have defined the dense Markov chain $\mathsf{DMC}^A_{\bar{\omega}}$ obtained from A and $\bar{\omega}$, we are in a position to define the probability measure $Prob^A_{\bar{\omega}}$ over sets of its infinite paths: refer to [34] for details.

6.4 Model checking continuous probabilistic timed automata

In this section, we consider how the notion of clock equivalence can be used to facilitate a model checking algorithm for continuous probabilistic timed automata. The property description language that we use is PTCTL, the syntax and semantics of which have been presented in Section 5. For simplicity, assume that $reset(\bar{l}, x)$ is defined as $\{[0, 0] \mapsto 1\}$ for all clocks $x \in \mathcal{X}$; this is not a serious restriction, as we can ensure that no time passes in $\bar{l}$ before a transition to the "proper" initial location which includes more liberal clock resetting functions is made. A continuous probabilistic timed automaton then satisfies a PTCTL formula ϕ if and only if $(\bar{l}, \mathbf{0}) \models^{\mathrm{PT}} \phi$. We now proceed directly to the definition of the region graph of a continuous probabilistic timed automaton.

The region graph. Assume that the continuous probabilistic timed automaton $\mathsf{CPTA} = (L, \bar{l}, \mathcal{X}, inv, \underline{prob}, reset, \mathcal{L})$ and the PTCTL formula ϕ are fixed. We use $\mathsf{DMDP} = (S, \Sigma, \bar{\mathbf{d}}, \mathbf{D}, lab)$ to denote the underlying dense Markov decision process of CPTA. Let max be the maximal constant against which any clock is compared in the invariant conditions and guards of CPTA. We define *region equivalence for* CPTA as the relation $\approx \subseteq S \times S$, where $(l, val) \approx (l', val')$ if and only if:

- $l = l'$ (the location components are equal);
- $val \equiv_{\max} val'$ (the valuations are clock equivalent with respect to max).

This is the same definition of region equivalence defined for timed automata in Section 3, and for probabilistic timed automata in Section 5, applied in the context of continuous probabilistic timed automata. As usual, we define a *region* to be an equivalence class of $\approx$, and use R to denote the set of regions. As in the previous sections, the fact that region equivalence is finite permits us to define a finite-state region graph which represents the behaviour of the continuous probabilistic timed automaton.

The definition of the region graph is somewhat different to the definition presented in Section 5. The difference resides in the fact that regions do not encode enough information to calculate transition probabilities when clocks are reset according to continuous density functions. (Indeed, this problem also arises in the context of real-time probabilistic processes, meaning that it is in general not possible to obtain a Markovian continuous-time stochastic process from region equivalence.) Consider the following example, originally due to Alur: assume that a clock x is set to some value within the interval $(0, 1)$ on entry to the location l. Time then elapses in location l, and, before the value of x reaches the value 1, a transition is made to location l'. In this new location l', the clock x retains its existing value, whereas another clock y is set to some value in the interval $(0, 1)$. We have three possible relationships between the clocks x and y in location l': either (1) the value of x is less than that of y, (2) the value of x is greater than that of y, or (3) the values of x and y are equal. While we can encode the probability of the value of the clocks being equal is 0, it is impossible to use region equivalence to obtain the exact probabilities of outcomes (1) and (2). The problem is that region equivalence abstracts from the exact amount of time that elapses in location l, upon which the probabilities of outcomes (1) and (2) depends.

However, observe that we can identify the relationship between *only* the clocks that are reset on entry to a location (namely, those in the set $\mathit{entry}(l)$), simply by combining continuous density functions and performing integrations. The values of all other clocks remain unchanged by the change of location. Our problem then reduces to that of describing how the newly-reset clocks and the unchanged clocks relate to one another. Unfortunately, it is impossible to obtain information giving the relative orders the fractional parts of reset and unchanged clocks. Therefore, we let the choice as to which relative order is chosen in the region graph be *nondeterministic*. The presence of such a nondeterministic choice in the place of a probabilistic choice introduces the possibility of an error in the probabilistic model checking results.

To define formally these concepts, we introduce the following notation. For a subset $X \subseteq \mathcal{X}$ of clocks, let $\equiv^X_{\max} \subseteq \mathbb{R}^{|X|}_{>0} \times \mathbb{R}^{|X|}_{\geq 0}$ be a variant of clock equivalence of Section 2 restricted to clocks in X. A *union region* is a triple of the form (l, α_u, α_r), where $l \in L$ is a location, α_u is a clock equivalence class of unchanged clocks in $\mathcal{X} \setminus \mathit{entry}(l)$, and α_r is a clock equivalence class of reset clocks in $\mathit{entry}(l)$. We let U denote the set of union regions, which is finite. The set of *combinations* of two clock equivalence classes, one α_X defined over a set $X \subseteq \mathcal{X}$ of clocks, the other $\alpha_{(\mathcal{X} \setminus X)}$ defined over the remainder $\mathcal{X} \setminus X$, is defined as the set of clock

equivalence classes over $\mathcal{X}$ obtained by combining the values of the clocks from the two distinct sets X and $\mathcal{X} \setminus X$ in any way (that is, with any relative order on the fractional parts between the clocks in X and the clocks in $\mathcal{X} \setminus X$). Formally, let:

$$\mathsf{Comb}(\alpha_X, \alpha_{(\mathcal{X} \setminus X)}) = \{\alpha \mid \alpha \lfloor_X = \alpha_X \text{ and } \alpha \lfloor_{\mathcal{X} \setminus X} = \alpha_{(\mathcal{X} \setminus X)}\} \ .$$

Finally, for a location $l \in L$, we define μ_l to be a discrete probability distribution over the clock equivalence classes of $\mathit{entry}(l)$, such that $\mu_l(\alpha)$ is the probability that, when the location l is entered (and thus when the clocks in $\mathit{entry}(l)$ are reset), the clock equivalence class over $\mathit{entry}(l)$ is α. The probabilities can be obtained using standard integration.

We now proceed to define the *region graph of* CPTA *and* ϕ as the Markov decision process $\mathcal{RG} = (R \cup U, \bar{r}, \mathcal{RT}, \mathit{lab})$. The initial region is defined as $\bar{r} = \{(\bar{l}, \mathbf{0})\}$, while the labelling function is defined by $\mathit{lab}(l, \alpha) = \mathcal{L}(l)$, for all regions $(l, \alpha) \in R$, and by $\mathit{lab}(l, \alpha_u, \alpha_r) = \mathcal{L}(l)$, for all union regions $(l, \alpha_u, \alpha_r) \in U$. Then the probabilistic transition relation $\mathcal{RT} \subseteq R \times \mathsf{Dist}(R)$ is defined in the following way. Let $\mathcal{RT}$ be the smallest set such that $(r, \rho) \in \mathcal{RT}$ if any of the following conditions hold:

Time elapse: $r = (l, \alpha) \in R$ is a region, and $\rho = \{(l, \beta) \mapsto 1\}$, where β is an $\mathit{inv}(l)$-satisfying time successor of α;

Probabilistic edge traversal: $r = (l, \alpha) \in R$ is a region, there exists a probabilistic edge $(l, g, p) \in \underline{\mathit{prob}}$ such that $\alpha \models g$ and, for each union region $(l', \beta_u, \beta_r) \in U$, we have $\overline{\rho(l', \beta_u, \beta_r)} = p(l') \cdot \mu_{l'}(\beta_r)$, where $\beta_u = \alpha \lfloor_{\mathcal{X} \setminus \mathit{entry}(l)}$;

Combination: $r = (l, \alpha_u, \alpha_r) \in U$ is a union region, and $\rho = \{(l, \beta) \mapsto 1\}$, where $\beta \in \mathsf{Comb}(\alpha_u, \alpha_r)$ is a combination of α_u, α_r.

Model checking using regions. Given that we have constructed a region graph of a continuous probabilistic timed automaton CPTA, with respect to a particular PTCTL property ϕ, we are now in a position to define an approximate model-checking algorithm to determine whether CPTA satisfies ϕ. The algorithm proceeds according to the same principles established for verifying PTCTL properties of probabilistic timed automata: we convert the PTCTL property into an untimed probabilistic property, and extend the region graph with an extra clock and atomic propositions. However, in the context of continuous probabilistic timed automata, we must also consider the *error* that may arise in the computation of the probabilities.

Note that the minimum probability of satisfying a property established using the region graph will be equal to or less than the minimum probability of satisfying a property in the continuous probabilistic timed automaton; similarly, the maximal probability of satisfying a property in the region graph will be equal to or greater than the corresponding maximum probability in the continuous probabilistic timed automaton. This follows from the fact that, in the definition of the region graph, we replace some probabilistic choice (the manner in which the values of certain clocks are ordered) with nondeterministic choice.

Observe that we can envisage an adversary which, instead of selecting among nondeterministic alternatives *deterministically* (as in our previous definitions of adversaries) selects alternatives *probabilistically* according to some distribution. Then, considering that we have replaced a probabilistic choice in the continuous probabilistic timed automaton with a nondeterministic choice in the region graph, we must have one such randomized adversary that resolves the nondeterministic choice in the same manner as the original probabilistic choice (that is, using the same distribution). Now, recalling that for properties such as those of PTCTL the minimal and maximal probabilities of satisfying a property are given by non-randomized adversaries [11], it must be the case that the minimal and maximal probabilities of satisfying a property on the region graph using such non-randomized adversaries bound the minimal and maximal probabilities of satisfying the property using randomized adversaries, and then, as one of these randomized adversaries corresponds to the actual probabilistic choice made, they therefore bound the actual minimal and maximal probabilities of satisfying the property in the original continuous probabilistic timed automaton. Hence, the minimal and maximal probability bounds obtained using the region graph are *conservative*.

Our second point concerns the computation of the error. Note that we can identify regions which can potentially cause an error to arise as those in which there is doubt as to the relative ordering of at least a pair of clocks; that is, if we have at least one clock which keeps its old value after a transition, and at least one clock which is newly reset. Therefore, for any adversary, we can identify a bound on the error of satisfying an "until" property by that adversary by computing the probability of the adversary reaching such "doubtful" regions.

The third point concerns the manner in which it may be possible to obtain a more accurate estimation of the required probabilities. This can be achieved by *re-scaling* the time unit so that clock equivalence partitions the valuation space more finely. Consider the example in which the clock x keeps its old value in the interval $(d, d+1)$, and the other clock y is newly set in the interval $(d, d+1)$, and that two outgoing probabilistic edges of the current location have guards $x = d+1$ and $y = d+1$, respectively. Say that we re-scaled the time unit by a factor of two: then we would now consider the intervals $(d, d+\frac{1}{2})$ and $(d+\frac{1}{2}, d+1)$. Hence, say the old clock x was in the interval $(d, d+\frac{1}{2})$ and the new clock y was probabilistically reset in the interval $(d+\frac{1}{2}, d+1)$; then we would know that the value of y is greater than that of x, and that the probabilistic edge with the guard $y = d+1$ will be enabled before the probabilistic edge with the guard $x = d+1$. Note that there is still the possibility of error, as, say, clock y could nevertheless be probabilistically reset in the interval $(d+\frac{1}{2}, d+1)$, in which case we would still have no knowledge about the relative order of x and y, and therefore no knowledge about which probabilistic edge would be enabled first. However, refining clock equivalence in such a manner will not increase the error, and may potentially lead to improvements in the computation of the probabilities (see [34] for an example).

Note that this third point relates to the complexity of the procedure, as the size of the region graph is exponential in the size of the maximal constant used multiplied by the scaling factor. Therefore, the aims of future research in the area of model checking continuous probabilistic timed automata must either concern the adaptation of algorithms based on clock constraints or the identification of useful subclasses of continuous probabilistic timed automata for which model checking is less expensive.

7 Conclusions

This overview of probabilistic extensions of timed automata has been concerned primarily with theoretical issues centred around the use of clock equivalence to define model-checking algorithms. Future work in this area should bridge the gap between such theory and practical algorithms and tools. Note that timed automata tools such as UPPAAL [6] and KRONOS [52] have been available for several years, and are developed to an advanced stage of maturity. Probabilistic model checkers such as PRISM [32], $\mathsf{E} \vdash \mathsf{MC}^2$ [28] and RAPTURE [29] have also been developed, although more recently. Note that PRISM was used to probabilistically model check discrete-time probabilistic timed automata in [38, 37,33]; this approach shows promise, although the size of the state space of the obtained discrete-time representation of a probabilistic timed automaton is sensitive to the maximal constant used in the description model, which can prove to be a problem particularly if the description of the system uses different time scales. Hence, a complementary approach based on manipulation of clock constraints is required [35,36].

References

1. R. Alur. Timed automata. In N. Halbwachs and D. Peled, editors, *Proceedings of the 11th International Conference on Computer-Aided Verification (CAV'99)*, volume 1633 of *LNCS*, pages 8–22. Springer, 1999.
2. R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking for probabilistic real-time systems. In J. L. Albert, B. Monien, and M. A. Rodríguez, editors, *Proceedings of the 18th International Conference on Automata, Languages and Programming (ICALP'91)*, volume 510 of *LNCS*, pages 115–136. Springer, 1991.
3. R. Alur, C. Courcoubetis, and D. L. Dill. Verifying automata specifications of probabilistic real-time systems. In J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, editors, *Proceedings of Real-Time: Theory in Practice, REX Workshop*, volume 600 of *LNCS*, pages 28–44. Springer, 1991.
4. R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.
5. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
6. T. Amnell, G. Behrmann, J. Bengtsson, P. R. D'Argenio, A. David, A. Fehnker, T. Hune, B. Jeannet, K. G. Larsen, M. O. Möller, P. Pettersson, C. Weise, , and W. Yi. UPPAAL - Now, next, and future. In F. Cassez, C. Jard, B. Rozoy, and

M. Ryan, editors, *Proceedings of the Summer School on Modelling and Verification of Parallel Processes (MOVEP'2k)*, volume 2067 of *LNCS*, pages 100–125. Springer, 2001.

7. E. Asarin, O. Maler, and A. Pnueli. On discretization of delays in timed automata and digital circuits. In R. de Simone and D. Sangiorgi, editors, *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR'98)*, volume 1466 of *LNCS*, pages 470–484. Springer, 1998.
8. C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.
9. C. Baier and M. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11(3):125–155, 1998.
10. D. Beauquier. On probabilistic timed automata. *Theoretical Computer Science*, 292(1):65–84, 2003.
11. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In P. S. Thiagarajan, editor, *Proceedings of the 15th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'95)*, volume 1026 of *LNCS*, pages 499–513. Springer, 1995.
12. A. Bouajjani, S. Tripakis, and S. Yovine. On-the-fly symbolic model checking for real-time systems. In K.-J. Lin and S. H. Son, editors, *Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS'97)*. IEEE Computer Society Press, 1997.
13. M. Bozga, O. Maler, and S. Tripakis. Efficient verification of timed automata using dense and discrete time semantics. In L. Pierre and T. Kropf, editors, *Proceedings of the 10th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME'99)*, volume 1703 of *LNCS*, pages 125–141. Springer, 1999.
14. J. Bryans, H. Bowman, and J. Derrick. Model checking stochastic automata. To appear in *ACM Transactions on Computational Logic*, 2003.
15. E. M. Clarke and E. A. Emerson. The design and synthesis of synchronization skeletons using temporal logic. In D. Kozen, editor, *Proceedings of the Workshop on Logics of Programs*, volume 131 of *LNCS*, pages 52–71. Springer, 1981.
16. E. M. Clarke, O. Grümberg, and D. Peled. *Model checking*. MIT Press, 1999.
17. C. Courcoubetis and S. Tripakis. Probabilistic model checking: formalisms and algorithms for discrete and real-time systems. In M. K. Inan and R. P. Kurshan, editors, *Verification of Digital and Hybrid Systems*, pages 183–219. Springer, 2000.
18. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.
19. P. R. D'Argenio. A compositional translation of stochastic automata into timed automata. Technical Report CTIT 00-08, University of Twente, 2000.
20. P. R. D'Argenio, J.-P. Katoen, and E. Brinksma. An algebraic approach to the specification of stochastic systems (extended abstract). In D. Gries and W.-P. de Roever, editors, *Proceedings of the IFIP Working conference on Programming Concepts and Methods (PROCOMET'98)*, IFIP Series, pages 126–147. Chapman & Hall, 1998.
21. C. Daws, M. Kwiatkowska, and G. Norman. Automatic verification of the IEEE 1394 root contention protocol with KRONOS and PRISM. In R. Cleaveland and H. Garavel, editors, *Proceedings of the 7th International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2002)*, volume 66(2) of *Electronic Notes in Theoretical Computer Science*, 2002.

22. L. de Alfaro. *Formal verification of probabilistic systems.* PhD thesis, Stanford University, Department of Computer Science, 1997.
23. C. Derman. *Finite-State Markovian Decision Processes.* Academic Press, 1970.
24. P. W. Glynn. A GSMP formalism for discrete-event systems. *Proceedings of the IEEE*, 77:14–23, 1989.
25. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
26. T. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In W. Kuich, editor, *Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP'92)*, volume 623 of *LNCS*, pages 545–558. Springer, 1992.
27. T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.
28. H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. A Markov chain model checker. *Software Tools for Technology Transfer*, 4(2):153–172, 2003.
29. B. Jeannet, P. R. D'Argenio, and K. G. Larsen. RAPTURE: A tool for verifying Markov Decision Processes. In I. Cerna, editor, *Tools Day'02,* Brno, Czech Republic, Technical Report. Faculty of Informatics, Masaryk University Brno, 2002.
30. H. E. Jensen. Model checking probabilistic real time systems. In B. Bjerner, M. Larsson, and B. Nordström, editors, *Proceedings of the 7th Nordic Workshop on Programming Theory*, volume 86, pages 247–261. Chalmers Institute of Technology, 1996.
31. J. G. Kemeny, J. L. Snell, and A. W. Knapp. *Denumerable Markov Chains.* Graduate Texts in Mathematics. Springer, 2nd edition, 1976.
32. M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In T. Field, P. Harrison, J. Bradley, and U. Harder, editors, *Proceedings of the 12th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS 2002)*, volume 2324 of *LNCS*, pages 200–204. Springer, 2002.
33. M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. In K. G. Larsen and P. Niebert, editors, *Proceedings of the 1st International Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS 2003)*, *LNCS*. Springer, 2003.
34. M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Verifying quantitative properties of continuous probabilistic timed automata. In C. Palamidessi, editor, *Proceedings of the 11th International Conference On Concurrency Theory (CONCUR 2000)*, volume 1877 of *LNCS*, pages 123–137. Springer, 2000.
35. M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 286:101–150, 2002.
36. M. Kwiatkowska, G. Norman, and J. Sproston. Symbolic computation of maximal probabilistic reachability. In K. Larsen and M. Nielsen, editors, *Proceedings of the 13th International Conference on Concurrency Theory (CONCUR 2001)*, volume 2154 of *LNCS*, pages 169–183. Springer, 2001.
37. M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In H. Hermanns and R. Segala, editors, *Proceedings of the 2nd Joint International Workshop on Process Algebra and Performance Modelling and Probabilistic Methods in Verification (PAPM-PROBMIV 2002)*, volume 2399 of *LNCS*, pages 169–187. Springer, 2002.

38. M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol. *Formal Aspects of Computing*, 14(3):295–318, 2003.
39. K. G. Larsen, P. Pettersson, and W. Yi. Compositional and symbolic model-checking of real-time systems. In A. Burns, Y.-H. Lee, and K. Ramamritham, editors, *Proceedings of the 16th IEEE Real-Time Systems Symposium (RTSS'95)*, pages 76–87. IEEE Computer Society Press, 1995.
40. K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.
41. G. I. Lòpez, H. Hermanns, and J.-P. Katoen. Beyond memoryless distributions: Model checking semi-Markov chains. In L. de Alfaro and S. Gilmore, editors, *Proceedings of Process Algebra and Probabilistic Methods. Performance Modeling and Verification. Joint International Workshop (PAPM-PROBMIV 2001)*, volume 2165 of *LNCS*, pages 57–70. Springer, 2001.
42. A. V. Moura and G. A. Pinto. A note on the verification of automata specifications of probabilistic real-time systems. *Information Processing Letters*, 82(5):223–228, 2002.
43. M. L. Puterman. *Markov Decision Processes*. J. Wiley & Sons, 1994.
44. J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In M. Dezani-Ciancaglini and U. Montanari, editors, *Proceedings of the International Symposium on Programming*, volume 137 of *LNCS*, pages 337–351. Springer, 1982.
45. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.
46. R. Segala and N. A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.
47. S. Tripakis. Verifying progress in timed systems. In J.-P. Katoen, editor, *Proceedings of the 5th International AMAST Workshop on Real-Time and Probabilistic Systems (ARTS'99)*, volume 1601 of *LNCS*, pages 299–314. Springer, 1999.
48. M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science (FOCS'85)*, pages 327–338. IEEE Computer Society Press, 1985.
49. W. Whitt. Continuity of generalized semi-Markov processes. *Mathematics of Operations Research*, 5:494–501, 1980.
50. W. Yi, P. Pettersson, and M. Daniels. Automatic verification of real-time communicating systems by constraint-solving. In D. Hogrefe and S. Leue, editors, *Proceedings of the 7th International Conference on Formal Description Techniques*, pages 223–238. North–Holland, 1994.
51. H. L. S. Younes and R. G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In E. Brinskma and K. G. Larsen, editors, *Proceedings of the 14th International Conference on Computer Aided Verification (CAV2002)*, volume 2404 of *LNCS*, pages 223–235. Springer, 2002.
52. S. Yovine. Kronos: A verification tool for real-time systems. *International Journal of Software Tools for Technology Transfer*, 1(1/2):123–133, 1997.
53. S. Yovine. Model checking timed automata. In G. Rozenberg and F. Vaandrager, editors, *Embedded Systems*, volume 1494 of *LNCS*, pages 114–152. Springer, 1998.