Transient Analysis of Networks of Stochastic Timed Automata using Stochastic State Classes

Paolo Ballarini, Nathalie Bertrand, Andras Horvath, Marco Paolieri, Enrico Vicario

Transient Analysis of Networks of Stochastic Timed Automata Using Stochastic State Classes

Paolo Ballarini[1], Nathalie Bertrand[2], András Horváth[3], Marco Paolieri[4], and Enrico Vicario[4]

[1] École Centrale Paris, France
[2] Inria Rennes, France
[3] Università di Torino, Italy
[4] Università di Firenze, Italy

Abstract. Stochastic Timed Automata (STA) associate logical locations with continuous, generally distributed sojourn times. In this paper, we introduce Networks of Stochastic Timed Automata (NSTA), where the components interact with each other by message broadcasts. This results in an underlying stochastic process whose state is made of the vector of logical locations, the remaining sojourn times, and the value of clocks. We characterize this general state space Markov process through transient stochastic state classes that sample the state and the absolute age after each event. This provides an algorithmic approach to transient analysis of NSTA models, with fairly general termination conditions which we characterize with respect to structural properties of individual components that can be checked through straightforward algorithms.

1 Introduction

Timed Automata (TA) extend standard automata by adding real-time clocks to states and clock constraints to transitions [4]. While this produces a continuous, infinite state-space, various finite abstractions based on regions [4], zones [17], or clock difference diagrams [7] were developed to allow the solution of verification problems in a qualitative perspective, i.e., with reference to possible or necessary behaviors. Various probabilistic extensions were then proposed to enable quantitative evaluation of the probability of feasible behaviors, or to restrain qualitative verification to behaviors with non-null probability.

In Probabilistic Timed Automata (PTA) [24], non-deterministic continuous-time delays are mixed with discrete distributions over actions, and models are checked against PTCTL [22]. In real-time probabilistic processes [2,3], also event durations are randomized. The underlying stochastic process becomes continuous time and may fall in the class of Generalized Semi-Markov Processes (GSMPs). Model-checking can be performed with respect to PTCTL by relying on finite-state abstraction. Continuous Probabilistic Timed Automata (CPTA) [23] extend PTA with randomized clock updates and enable approximate checking against PTCTL.

Stochastic Timed Automata (STA) [5] were proposed with the aim of relaxing the idealized aspects of TA through a semantics that distinguishes null-probability behaviors. To this end, STA associate locations with sojourn time distributions and transition edges with weights for the probabilistic choice among multiple enabled transitions. In the most general formulation, both of these quantities may depend on clock valuations. For single-clock STA, almost-sure verification of LTL specifications was shown decidable [6] and an approximated technique for quantitative model-checking was proposed [8]. Decidability of almost sure verification was also shown for *reactive* timed automata [10], even with multiple-clocks but under restrictions on sojourn times.

Combination of qualitative real-time constraints with quantitative probabilistic information was also largely addressed on the ground of various classes of Stochastic Petri Nets (SPNs) with generally distributed transitions. In general, the underlying stochastic process of such models belongs to the class of GSMPs [18], for which simulation or statistical model checking are the only general viable approaches to quantitative evaluation. Analytic treatment becomes possible under the so-called enabling restriction that basically requires that no more than one generally distributed transition be enabled at the same time [15,9].

More recently, the method of stochastic state classes addressed models with multiple generally distributed transitions possibly supported over bounded domains, through the symbolic characterization of supports and distributions of remaining times to fire after each transition firing. This was first proposed for steady state analysis [12,26], and then extended to transient analysis [20] and probabilistic model checking [19]. A similar approach was developed for the analysis of Duration Probabilistic Automata (DPA) [25], which compose a set of acyclic semi-Markov processes under control of a non-deterministic scheduler. Symbolic derivation of probability density functions over equivalence classes was proposed also in [1] with a calculus similar to that of [12] but leveraging finite state-space abstractions based on regions rather than zones.

In this paper, we extend the STA formalism by introducing the so-called Networks of Stochastic Timed Automata (NSTA), where multiple STA may synchronize through message passing over a broadcast channel (Sect. 2). We then propose an analytic approach to transient analysis of NSTA models based on the method of stochastic state classes. To this end, we describe the construction of stochastic state classes that sample the state after each transition, we show how these classes provide transient probabilities, and we characterize conditions for termination of the analysis (Sect. 3). An example is then discussed to highlight modeling patterns of NSTA, and analysis results are provided through a preliminary tool-chain implemented on top of the ORIS tool [11,14] (Sect. 4).

2 Model Definition and Semantics

2.1 Timed Automata

Timed automata were introduced in the 90's as a model for real-time systems [4]. Given a finite set of clocks X, we write $\mathcal{G}(X)$ for the set of *guards*, i.e., conjunctions

of atomic constraints of the form $x \sim c$ where $x \in X$, $\sim \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{N}$. For a *clock valuation* $v \in \mathbb{R}^X_{\geq 0}$ and a guard $g \in \mathcal{G}(X)$, we write $v \models g$ when v satisfies g.

Definition 1 (Timed automaton). *A* timed automaton *is a tuple* $\langle L, \ell_0, \Sigma, X, E\rangle$ *where* L *is a finite set of locations,* $\ell_0 \in L$ *is the initial location,* Σ *is the action alphabet,* X *is a set of clocks and* $E \subseteq L \times \Sigma \times \mathcal{G}(X) \times 2^X \times L$ *is a set of edges.*

The semantics of a timed automaton is a transition system where the states are pairs (ℓ, v) of a location $\ell \in L$ and a valuation $v \in \mathbb{R}^X_{\geq 0}$ for the clocks. From any state (ℓ, v) and for any delay $\tau \in \mathbb{R}_{\geq 0}$, there is a delay transition leading to the state $(\ell, v + \tau)$, where $v + \tau$ is a notation for the valuation defined by $(v+\tau)(x) = v(x)+\tau$ for all $x \in X$. Also, for every edge $e = (\ell, a, g, r, \ell') \in E$ and from every state (ℓ, v) such that $v \models g$, there exists a discrete transition leading to $(\ell', v_{[r\leftarrow 0]})$ where $v_{[r\leftarrow 0]}$ denotes the valuation defined by $v_{[r\leftarrow 0]}(x) = 0$ if $x \in r$, and $v_{[r\leftarrow 0]}(x) = x$ otherwise. In this case, we say that edge e is enabled in (ℓ, v) and we write $(\ell, v) \xrightarrow{e} (\ell', v')$ or $e(\ell, v) = (\ell', v')$, with $v' = v_{[r\leftarrow 0]}$.

2.2 Stochastic Timed Automata

We consider stochastic timed automata, a continuous-time probabilistic model associating locations with sojourn time probability density functions (PDFs) and edges with probabilistic choices based on weights [6].

Definition 2 (Stochastic timed automaton). *A* stochastic timed automaton *is a tuple* $\mathcal{A} = \langle L, \ell_0, \Sigma, X, E, \mu, w\rangle$ *consisting of a timed automaton* $\langle L, \ell_0, \Sigma, X, E\rangle$ *equipped with sojourn time probability density functions* $\mu = (\mu_\ell)_{\ell \in L}$ *and natural weights* $w = (w_e)_{e \in E}$.

Transitions of a STA are determined as follows. When a location l is entered with clock valuation v, then (1) the sojourn time T is chosen according to the PDF μ_ℓ, (2) after the delay T has elapsed, denoting by $E(\ell, v + T)$ the set of edges enabled in state $(\ell, v+T)$, edge $e \in E(\ell, v+T)$ is selected with probability $w_e / \sum_{f \in E(\ell, v+T)} w_f$, (3) assuming e was selected, and if $(\ell', v') = e(\ell, v + T)$, then a transition to ℓ' with clock valuation v' occurs.

The underlying stochastic process of STA ranges from CTMCs to GSMPs, and it will be discussed in Sect. 2.4. In general, the underlying process of a STA is a general state space Markov chain whose state is composed of a discrete component (the location) and a continuous one (the clock valuation and the remaining sojourn times). In this Markov process, if the current state is (ℓ, v, T), the probability to fire an edge $e \in E(\ell, v+T)$ with $(\ell, v+T) \xrightarrow{e} (\ell', v')$ and sample a new sojourn time $T' \leq t$ is given by $(w_e / \sum_{f \in E(\ell, v+T)} w_f) \cdot \int_0^t \mu_{\ell'}(\tau)\, \mathrm{d}\tau$.

Remark 1 (Comparison with the model from [6]). Note that we consider a restricted class of stochastic timed automata where the sojourn time probability

density functions only depend on the current location, and not on the clock valuation when entering that location. This has consequences on the PDFs that are possible in a location, but also on the structure of the underlying timed automaton itself. However, natural large classes of stochastic timed automata such as reactive STA [10] are covered in our framework.

Also, differently from the traditional approach, we choose a slightly alternative view of STA by considering states (ℓ, v, T) where the sojourn time has already been sampled, rather than states (ℓ, v). This choice is motivated by the extension to networks that we propose in the following section.

2.3 Networks of Stochastic Timed Automata

We now introduce a model where several stochastic timed automata form a network and interact by broadcasting messages. Therefore, in each component, the set of actions is partitioned into sending and receiving actions.

Definition 3 (Network of STA). *A* network *of STA is a tuple* $\langle \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle$ *of* n *stochastic timed automata associated with global weights* $w(\mathcal{A}_1), \ldots, w(\mathcal{A}_n)$ *and sharing an alphabet* $\Sigma = \Sigma_b \cup \Sigma_r$ *partitioned into broadcasts* (Σ_b) *and receptions* (Σ_r). *Broadcasts are of the form* $!m$ *and receptions of the form* $?m$, *for some message* m *from a fixed alphabet* M, *i.e.,* $\Sigma_b = !\mathsf{M}$ *and* $\Sigma_r = ?\mathsf{M}$.

The intuitive semantics of a network is as follows. The network starts in a configuration where each STA is in its initial location and samples an initial sojourn time. When the minimum sampled sojourn time elapses, say for STA $\mathcal{A}_i$, the component $\mathcal{A}_i$ performs an action selected according to the weights of its enabled edges, and broadcasts the associated message (races among equal, deterministic times to fire are solved by the global weights $w(\mathcal{A}_i)$). When $\mathcal{A}_i$ performs a broadcast (sending action), then all other components for which the corresponding receiving action is enabled must synchronize and perform the corresponding reception. On occurrence of such inter-process communication, all components involved in the exchange (the sender and the receivers) update their locations and sample new sojourn times, and then the execution in the network proceeds.

Formally, states of the network $\langle \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle$ are n-tuples of triplets (ℓ_i, v_i, T_i), one for each component, consisting of current location (ℓ_i), clock valuation (v_i), and remaining sojourn time (T_i). Given the current state of the network $\mathbf{s} = \langle (\ell_1, v_1, T_1), \ldots, (\ell_n, v_n, T_n) \rangle$, the next transition is determined by selecting the component $\mathcal{A}_i$ with lowest remaining sojourn time T_i; deterministically, the delay T_i is elapsed from $\mathbf{s}$ and a broadcast edge e_i enabled in $(\ell_i, v_i + T_i)$ is selected according to weights (as in STA), resulting in the local transition $(\ell_i', v_i') = e_i(\ell_i, v_i + T_i)$. For every other component $\mathcal{A}_j$ with $j \neq i$, an enabled matching receiving edge e_j (if any) is selected according to the weights, and the state of components with a selected (synchronizing) action is updated as $(\ell_j', v_j') = e_j(\ell_j, v_j + T_i)$. New sojourn times are sampled for components that performed an action: T_i' according to $\mu_{\ell_i'}$, and possibly T_j' according to $\mu_{\ell_j'}$; the resulting state is then $\mathbf{s}' = \langle (\ell_1', v_1', T_1'), \cdots (\ell_n', v_n', T_n') \rangle$ where $(\ell_k', v_k', T_k') = (\ell_k, v_k + T_k, T_k' - T_i)$ for components $\mathcal{A}_k$ that did not take an action.

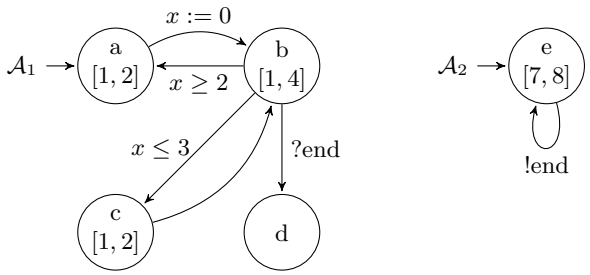


Fig. 1. Network of STA

Example. Consider the NSTA depicted in Fig. 1, where all sojourn times of $\mathcal{A}_1$ and $\mathcal{A}_2$ are uniformly distributed over the depicted supports. Automaton $\mathcal{A}_2$ broadcasts an END message at each termination of its computation: if $\mathcal{A}_1$ is in b at the moment of the broadcast, its current computation is interrupted by the receiving edge (b, d) and the network reaches the absorbing state $\vec{\ell} = (d, e)$. The clock x in $\mathcal{A}_1$ tracks the time elapsed since the last transition from a to b.

2.4 Underlying Stochastic Process of STA and NSTA

The underlying stochastic process of a STA is a CTMC only under severe restrictions: (1) the (non-null) sojourn times of locations must be exponentially distributed and (2) the choice of the next location cannot depend on the value of clocks, so that clocks do not have an impact on the behavior (i.e., the control-flow) of the STA. Under these conditions, the STA is *memoryless* in every location and every time instant is a regeneration point, i.e., the current location alone determines the future of the process in a stochastic sense.

If clocks are reset at every transition, or those that are not reset cannot affect future choices of the next location (i.e., they do not appear in guards), then every transition constitutes a regeneration point and the underlying stochastic process is a semi-Markov process (SMP). When clocks can carry memory from a location to another and influence the behavior of the STA, then there can be transitions that do not result in a regeneration point. If regeneration points are guaranteed to appear infinitely often, the process is Markov regenerative (MRP); otherwise, we have a generalized semi-Markov process (GSMP). Regeneration points can be exploited in the analysis of the process [21]; if regeneration is not guaranteed, two general analysis techniques can be adopted: the so-called supplementary variable approach [16] and the method of stochastic state classes [20].

In the case of a NSTA, the underlying process is a CTMC if all STA are memoryless in every location, i.e., their underlying process is also a CTMC. If this condition does not hold, we can still have regeneration points when (1) all automata are memoryless in the current location, or (2) some automata are memoryless in the current location and the others "lost memory" at the same time due to one or more receiving transitions, i.e., they performed a transition that reset all clocks except those that cannot have an impact on the future

behavior. The way regeneration points occur determines if the underlying process of the NSTA is a SMP, a MRP or a GSMP. The classification cannot be based on the analysis of the automata in isolation because their interplay can be decisive. The transient analysis we propose in this paper can be easily modified to provide information on the underlying process and compute the kernels of the MRP required in calculations exploiting regeneration points (see [20]).

3 Transient Analysis

3.1 Stochastic State Classes for NSTA

Stochastic state classes characterize the underlying stochastic process by representing explicitly, for each state transition, the resulting logical state and joint probability density function of the continuous random variables that govern the evolution of the system (clocks and remaining sojourn times). To support transient analysis, we include an additional clock x_{age} to encode the absolute time of the last transition [20]. Let us formalize the concept of stochastic state class.

Definition 4 (Stochastic state class). *A* stochastic state class *for the STA network* $\langle \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle$ *is a tuple* $\langle \ell, D, f \rangle$ *where* $\ell = (\ell_1, \ldots, \ell_n)$ *specifies the current location of each automaton and* $f \colon D \to [0,1]$ *is the probability density function of the random remaining sojourn times* $\vec{\tau} = (\tau_1, \ldots, \tau_n)$ *and clocks* $\langle x_{age}, \vec{x} \rangle$*, with* $\vec{x} = (x_1, \ldots, x_m)$*, on the support* $D \subseteq \mathbb{R}^n_{\geq 0} \times \mathbb{R}^{m+1}_{\leq 0}$.

Note that the support of remaining sojourn times τ_i is $\mathbb{R}_{\geq 0}$, while that of clock random variables x_i is $\mathbb{R}_{\leq 0}$; this is required to allow an efficient representation of the joint support D. In fact, stochastic state classes were originally developed in the context of Stochastic Time Petri Nets (STPNs), where times to fire of transitions decrease with unitary rate and the support of joint PDFs can be represented as a Difference Bounds Matrix zone (DBM zone), i.e., the set of solutions of a system of linear inequalities $\tau_i - \tau_j \leq b_{ij}$ for all $i \neq j \in \{*, 1, \ldots, n\}$ with $b_{ij} \in \mathbb{R} \cup \{+\infty\}$ and $\tau_* = 0$. This form allows a compact representation of the state space, and it is preserved by all the operations required in the computation of successor state classes through the firing of a transition; these operations include: reducing all the variables of a stochastic class by one of them, marginalizing variables, and adding new variables in product form. In STA, sojourn time random variables can be managed with the same operations: they can be reduced by the minimum one, or marginalized and added in product form for the automata performing the transition. In contrast, clocks should be *increased* by the minimum sojourn time; in order to preserve the DBM form for the support of PDFs of random variables, we thus encode clocks as negative variables that are initially set to zero and *decreased* at each transition. In so doing, all the random variables of the stochastic state class (sojourn times and clocks) are simply decreased by one of them, as in STPNs.

We now define the stochastic state classes of a NSTA, starting from the initial class, and then describing the derivation of successor classes.

Initial class. In the initial class Σ_0, all the clocks are set to zero and each automaton $\mathcal{A}_i$ is in its initial location ℓ_i with a sojourn time independently distributed according to the probability density function μ_{ℓ_i}. Hence, we have $\Sigma_0 = \langle \vec{\ell}_0, D_0, f_0 \rangle$ with $\vec{\ell}_0 = (\ell_1, \ldots, \ell_n)$ and

$$D_0 = ([a_0, b_0] \times \cdots \times [a_n, b_n]) \times [0,0] \times [0,0]^m$$
$$f_0(\vec{\tau}, x_{age}, \vec{x}) = \prod_{i=0}^{n} \mu_{\ell_i}(\tau_i) \cdot \delta(x_{age}) \cdot \prod_{i=0}^{m} \delta(x_i)$$

where δ is the Dirac delta function, μ_{ℓ_i} is the PDF associated with location ℓ_i, and $[a_i, b_i]$ its support (we indicate the Cartesian product of supports by $\times$).

Computation of successor classes. The computation of successor classes characterizes the set of states (locations, remaining sojourn times and clock valuations) that can be reached after a state transition in the STA network, their probability density function, and the probability of the state transition itself.
In general, a transition is identified by the automaton $\mathcal{A}_i$ with minimum remaining sojourn time, an edge e in $\mathcal{A}_i$, and a possible set E of receiving edges for the message broadcast by e. In addition, a transition also depends on the clock valuation, which may restrict the set of enabled edges because of the guards (in $\mathcal{A}_i$ or in the rest of the network). This partitions the space of clock valuations in *decision domains*, such that any two valuations in the same domain satisfy the same guards and thus result in a probabilistic choice within the same set of enabled edges. Since guards are expressed by conjunctions of simple inequalities, each guard is satisfied within a hyper-rectangular domain, but a decision domain may be the difference among different hyper-rectangles. When guards on the outgoing edges of a location involve multiple clocks, decision domains are not hyper-rectangular, but they can be anyway partitioned into a set of hyper-rectangular subdomains, which we call *decision zones*. We denote with $\mathcal{R}(\vec{\ell})$ the set of decision zones r associated with location $\vec{\ell}$ of the network.

Example. For the NSTA depicted in Fig. 1, $\mathcal{A}_1$ has outgoing edges in location b with guards on x that are both satisfied for $x \in [2,3]$, and exclusively satisfied for $x \notin [2,3]$. The decision zones in b are thus $r_{b,1} = [0,2)$, $r_{b,2} = [2,3]$, $r_{b,3} = [3,+\infty)$, while any other location is associated with a single decision zone $[0,+\infty)$. If $x \in r_{b,2}$, assuming equally weighted edges $w(b,a) = w(b,c)$, either enabled edge (b,a) or (b,c) is selected with probability $1/2$.

Definition 5 (Succession relation). *We say that $\Sigma' = \langle \vec{\ell}', D', f' \rangle$ is the successor of $\Sigma = \langle \vec{\ell}, D, f \rangle$ through the edge e of $\mathcal{A}_i$, for a decision zone $r \in \mathcal{R}(\vec{\ell})$ and a set of receiving edges E, with probability p (and we write $\Sigma \overset{\xi,p}{\Longrightarrow} \Sigma'$ with $\xi = (\mathcal{A}_i, r, e, E)$), if, given that the location of the NSTA is $\vec{\ell}$ and the sojourn times and clocks are random variables distributed over D according to f, then:*

(i) with non-null probability p, $\mathcal{A}_i$ is the automaton with minimum remaining sojourn time in Σ, the random clock valuation $\vec{x}$ belongs to the decision zone

$r \in \mathcal{R}(\vec{\ell})$, *the sender edge* e *and the receiving edges* E *are enabled by* r, *and they are selected as outgoing event;*

(ii) conditioned to (i), the state transition yields the location $\vec{\ell}'$ *with sojourn times and clock random variables distributed over* D' *according to* f'.

Given this definition, the successors of a stochastic state class can be derived through the following steps.

1. *Conditioning on the minimum sojourn time.* For each automaton of the network, we compute the probability that its remaining time to fire is the minimum, and we condition the sojourn time and clock random variables by this event. Up to a renaming of the components, we assume that the automaton $\mathcal{A}_1$ has minimum remaining time to fire with probability

$$p_{\tau_1} = \int_{\{\langle \vec{\tau}, x_{age}, \vec{x} \rangle \in D \,|\, \tau_1 \leq \tau_j \ \forall j\}} f(\vec{\tau}, x_{age}, \vec{x}) \, d\vec{\tau} \, dx_{age} \, d\vec{x} \, .$$

   By conditioning on this event, we obtain the random vector of sojourn times and clocks $\vec{v}_a = \langle \vec{\tau}, x_{age}, \vec{x} \mid \{\tau_1 \leq \tau_j \ \forall j\} \rangle$ distributed over $D_a = \{\langle \vec{\tau}, x_{age}, \vec{x} \rangle \in D \mid \tau_1 \leq \tau_j \ \forall j\}$ according to $f_a = f/p_{\tau_1}$. Note that races among equal deterministic times to fire are resolved by the global weights $w(\mathcal{A}_1), \ldots, w(\mathcal{A}_n)$ associated with the automata.

2. *Shifting all the variables by the minimum sojourn time and marginalizing the minimum sojourn time.* In order to account for the time elapsed in the previous state, all the variables are decreased by the minimum sojourn time τ_1, that is in turn marginalized. This leads to a random vector of sojourn times and clocks $\vec{v}_b = \langle \tau_2 - \tau_1, \ldots, \tau_n - \tau_1, x_{age} - \tau_1, \vec{x} - \tau_1 \rangle$ distributed over

$$D_b = \{(\tau_2, \ldots, \tau_n, x_{age}, \vec{x}) \in \mathbb{R}_{\geq 0}^{n-1} \times \mathbb{R}_{\leq 0}^{m+1} \mid \exists \tau_1 \in \mathbb{R}_{\geq 0} : (\tau_1, \tau_2 + \tau_1, \ldots, \tau_n + \tau_1, x_{age} + \tau_1, \vec{x} + \tau_1) \in D_a\}$$

   according to

$$f_b(\tau_2, \ldots, \tau_n, x_{age}, \vec{x}) = \int_{L_1}^{U_1} f_a(\tau_1, \tau_2 + \tau_1, \ldots, \tau_n + \tau_1, x_{age} + \tau_1, \vec{x} + \tau_1) \, d\tau_1$$

   where

$$L_1(\tau_2, \ldots, \tau_n, x_{age}, \vec{x}) = \min_{j \neq \tau_1} \{b_{\tau_1, j} + j\}$$
$$U_1(\tau_2, \ldots, \tau_n, x_{age}, \vec{x}) = \max_{i \neq \tau_1} \{-b_{i, \tau_1} + i\}$$

   are the piecewise linear functions of the minimum and maximum constraints on the variable τ_1 within the DBM zone D_a of coefficients b_{ij} and variables $i, j \in \{\tau_1, \ldots, \tau_n, x_{age}, x_1, \ldots, x_m, *\}$. Because of this piecewise dependency of integration bounds on different expressions of the form $b_{\tau_1, j} + j$ or $-b_{i, \tau_1} + i$, the result of the symbolic integration is in general a piecewise continuous function on a partitioning of D_b in DBM subzones [13]. In this case, all of the following steps have to be performed individually on each subzone.

3. *Conditioning on a decision zone.* In order to analyze fixed sets of enabled edges, each decision zone $r \in \mathcal{R}(\vec{\ell})$ is taken into account separately by imposing that the clock variables $\vec{x}$ belong to $r \subseteq \mathbb{R}^m_{\leq 0}$. This event has probability

$$p_r = \int_{\{\langle \vec{\tau}, x_{age}, \vec{x} \rangle \in D_b \mid \vec{x} \in r\}} f_b(\vec{\tau}, x_{age}, \vec{x})\, d\vec{\tau}\, dx_{age}\, d\vec{x}$$

and, by conditioning on it, we obtain the vector of sojourn times and clocks $\vec{v}_c = \langle \vec{v}_b \mid \{\vec{x} \in r\} \rangle$ distributed over $D_c = \{\langle \vec{\tau}, x_{age}, \vec{x} \rangle \in D_b \mid \vec{x} \in r\}$ according to $f_c = f_b / p_r$.

4. *Selection of a sender edge.* Since the set of enabled edges W is fixed within the decision zone r, the edge $e \in W$ is selected by automaton $\mathcal{A}_1$ in r with probability $p_e = w(e) / \sum_{e' \in W} w(e')$.

5. *Selection of receiving edges.* For each automaton with more than one receiving edge for the symbol broadcast by the edge e and enabled in r, the choice is resolved with weights, so that the probability p_E of each distinct set E of receiving edges is determined.

6. *Locations update.* The locations are updated according to the transitions performed by the automaton with minimum sojourn time and by those with receiving edges in E. A new locations vector $\vec{\ell}'$ is computed.

7. *Variables removal.* For automata that updated their locations, remaining sojourn time variables and reset clocks are marginalized. As an example, marginalization of a variable τ_2 is performed as

$$f_d(\tau_3, \ldots, \tau_n, x_{age}, \vec{x}) = \int_{E_2}^{L_2} f_c(\tau_2, \ldots, \tau_n, x_{age}, \vec{x})\, d\tau_2.$$

Similarly to *shift and project* operations, subzones can be introduced by the piecewise integration bounds L_2 and E_2.

8. *Variables addition.* Similarly to the definition of the initial stochastic state class, new Dirac deltas are added in product form for all reset clocks, and sojourn time PDFs are added in product form to f_d for automata that updated their locations. This results in the final domain D' and PDF f'.

Given a stochastic class $\Sigma = \langle \vec{\ell}, D, f \rangle$, the probability associated with the successor resulting from the transition given by the sender edge e of $\mathcal{A}_1$ and receiving edges E within the decision zone $r \in \mathcal{R}(\vec{\ell})$ is thus $p = p_{\tau_1} p_r p_e p_E$. Note that, in general, the automaton with minimum time to fire can select different edges, and same edge e can result in several stochastic successors with distinct sets of receiving edges or distinct decision zones.

Moreover, if no edge is enabled in the current location of $\mathcal{A}_1$ when the clocks belong to the decision zone $r \in \mathcal{R}(\vec{\ell})$, the remaining sojourn time variable τ_1 is marginalized (step 1) but not reintroduced in product form (step 8). According to the semantics of NSTA, $\mathcal{A}_1$ reaches (with probability $p = p_{\tau_1} p_r$) a state from

which it can perform a transition only by receiving a broadcast symbol; if all of the automata are in such receive-only condition, a deadlock has occurred.

3.2 Transient Tree Enumeration and Transient Measures

The transient evolution of the logical state $\vec{\ell}$ in a NSTA can be analyzed through the probability and time distribution of discrete events representing the end of sojourn times sampled according to the PDFs μ_ℓ. We consider as discrete event abstraction the tuple $(\mathcal{A}_i, r, e, E)$: given the current locations $\vec{\ell}$, the event $(\mathcal{A}_i, r, e, E)$ is the next event of the network if

- the sojourn time of $\mathcal{A}_i$ is the minimum;
- the clock valuation $\vec{v}$ belongs to the decision zone $r \in \mathcal{R}(\vec{\ell})$;
- the sender and receiving edges e and E (respectively) are randomly selected ($e = \text{NIL}$ and $E = \emptyset$ if no edge is enabled for $\mathcal{A}_i$ in ℓ_i when $\vec{v} \in r$).

Since decision zones represent a partition of the space of clock valuations, the events $(\mathcal{A}_i, r, e, E)$ for each automaton, decision zone, and distinct enabled sender and receiving edges, are mutually exclusive and collectively exhaustive. Moreover, given a PDF and support for clocks and remaining sojourn times in the current locations, the computation of successor classes presented in Sect. 3.1 allows to compute the probability of an event $(\mathcal{A}_i, r, e, E)$ and the PDF and support conditioned to it. Successive events can then be evaluated independently after conditioning, since a stochastic state class is a full characterization of the future evolution of the probabilistic model.

This construction leads to the enumeration of a tree in which each node is labeled with a stochastic state class, and where each edge carries an event $\xi = (\mathcal{A}_i, r, e, E)$ and a probability p.

Definition 6 (Transient tree). *The* transient tree *from an initial stochastic state class Σ_0 is a tuple* $\textsc{Transient-Tree}(\Sigma_0) = \langle N, A, n_0, \Sigma, p, \xi \rangle$ *where*

- *N is a set of nodes and $n_0 \in N$ is the root of the tree;*
- *the labeling function Σ associates each node $n \in N$ with a stochastic state class $\Sigma(n)$, with $\Sigma(n_0) = \Sigma_0$;*
- *A is the smallest set of edges (n, n') with $n, n' \in N$ such that $\Sigma(n')$ is a successor of $\Sigma(n)$, i.e., $\Sigma(n) \overset{\xi,p}{\Longrightarrow} \Sigma(n')$; in such a case, the edge is labeled with the probability $p(n, n')$ and the event $\xi(n, n')$ that it bears.*

The transient tree from an initial stochastic state class Σ_0 can be enumerated by repeatedly computing the successor classes of a leaf node until some stopping criterion is satisfied. A node n_k in the transient stochastic tree can thus be associated with a sequence of events $\xi_1, \xi_2, \ldots, \xi_k$ such that

$$\Sigma(n_0) \overset{\xi_1,p_1}{\Longrightarrow} \Sigma(n_1) \overset{\xi_2,p_2}{\Longrightarrow} \cdots \overset{\xi_k,p_k}{\Longrightarrow} \Sigma(n_k)\,.$$

The probability that the sequence of events $\xi_1, \xi_2, \ldots, \xi_k$ happens, leading from the initial node n_0 to the node n_k, is given by the *reaching probability*

$\eta(n_k) = \prod_{i=1}^{k} p_i$. The stochastic state class $\Sigma(n_k)$ defines the PDF of remaining sojourn times and clocks after the sequence of events $\xi_1, \xi_2, \ldots, \xi_k$. This information allows to impose additional constraints on the execution time and compute more specific measures. Notably, if $\Sigma(n_k) = \langle \vec{\ell}, D, f \rangle$, the probability that the system has performed, at time t, all and only the events $\xi_1, \xi_2, \ldots, \xi_k$ is given by

$$\pi(n_k, t) = \eta(n_k) \cdot \int_{D(t)} f(\vec{\tau}, x_{age}, \vec{x}) \, d\vec{\tau} \, dx_{age} \, d\vec{x}$$

with $D(t) = \{\langle \vec{\tau}, x_{age}, \vec{x} \rangle \in D \mid -x_{age} \leq t \text{ and } -x_{age} + \tau_i > t \text{ for } i = 1, \ldots, n\}$. The reaching probability $\eta(n)$ accounts for the probability of performing the sequence of events, while the restricted domain $D(t)$ imposes that the last event happened at a time $-x_{age} \leq t$ and the next one will happen at a time $\min_i\{-x_{age}+\tau_i\}$ greater than t. By definition of the PDF f, integrating over this restricted set of remaining sojourn time and age values results in the required measure.

Transient probabilities for a specific vector of locations $\vec{\ell}^*$ can then be defined as the sum of measures $\pi(n, t)$ for all nodes associated with a stochastic state class $\Sigma(n) = \langle \vec{\ell}, D, f \rangle$ with $\vec{\ell} = \vec{\ell}^*$:

$$\pi(\vec{\ell}^*, t) = \sum_{n \in N : \Sigma(n) = \langle \vec{\ell}^*, D, f \rangle} \pi(n, t) \, .$$

Events associated with the measures $\pi(n, t)$ are in fact mutually exclusive for distinct n: at any given time instant t, a node n uniquely identifies a sequence of transitions performed by the system and $\pi(n, t)$ is the probability that all and only the transitions in the sequence have been performed.

Note that, in the enumeration of the transient tree, each decision zone has to be taken into account separately and, moreover, the piecewise partition of PDFs in DBM subzones requires that the derivation of density functions be repeated on each subdomain. In the worst case, the number of successors and subdomains within each successor grows exponentially with the depth of the transient tree, comprising the dominating factor of complexity in practical implementations.

3.3 Termination

The *exact* evaluation of transient probabilities up to a given time bound T requires the enumeration of all and only the stochastic state classes that can be reached within T, i.e., all classes where the support of x_{age} includes values greater or equal to $-T$. If an *approximation* bound $\epsilon \geq 0$ is allowed, construction of the tree can be halted as soon as the total probability of reaching any leaf node before T is lower than ϵ; such probability, indicated as $\eta_T(n)$, can be computed for any node n associated with $\Sigma(n) = \langle \vec{\ell}, D, f \rangle$ as

$$\eta_T(n) = \eta(n) \cdot \int_{\{\langle \vec{\tau}, x_{age}, \vec{x} \rangle \in D \mid -x_{age} \leq T\}} f(\vec{\tau}, x_{age}, \vec{x}) \, d\vec{\tau} \, dx_{age} \, d\vec{x} \, .$$

For both exact and approximate evaluation, termination involves two orthogonal aspects pertaining to: (1) the cyclic behaviors arising in the composition of multiple automata; (2) the guarantee of time progression in each cyclic execution, in certainty (i.e., surely) or in probability (i.e., almost surely). We characterize termination through sufficient conditions that can be checked on the structure of individual STA, relaxing on aspect (1) so as to obtain conditions that can be easily checked by the designer in the construction of a model without resorting to state space analysis. Necessary and sufficient conditions require instead a non-deterministic analysis of the state class graph of the NSTA, with concepts analogous to those applied in [19] and without affecting the conditions given in the following for the advancement of time along cyclic behaviors.

Termination in exact analysis depends on *necessarily tangible* and *possibly vanishing* locations, i.e., locations where the *minimum* sojourn time is strictly greater than zero or equal to zero, respectively.

Theorem 1 (Exact time-bounded termination). *Given an initial stochastic state class Σ_0, the enumeration of the transient tree* TRANSIENT-TREE(Σ_0) *within any time bound T terminates in a finite number of steps provided that every cycle of each automaton visits at least one necessarily tangible location that cannot be preempted by messages broadcast on exit from possibly vanishing locations.*

Proof. Since the NSTA has a finite number of locations and the branching factor of TRANSIENT-TREE(Σ_0) is bounded, an infinite path would traverse some cycle within some automaton infinitely often. In so doing, at least one necessarily tangible location is visited infinitely often; at each visit, the minimum time to complete the path would increase by at least the minimum sojourn time of the location or the minimum sojourn time of the location in some other automaton that broadcasts a preempting message, thus exceeding T in a finite number of steps. □

When a cycle can be traversed in a time arbitrarily close to zero, termination can still be guaranteed if we exclude cycles that are *bound* to complete in zero time. In this case, termination depends on *possibly tangible* and *necessarily vanishing* locations, i.e., locations where the *maximum* sojourn time is strictly greater than zero or equal to zero, respectively.

Theorem 2 (Time-bounded termination with ϵ-error). *Given an initial stochastic state class Σ_0, a time bound T and an allowed error ϵ, if the transient tree is enumerated in breadth first order, there exists a number $N(\Sigma_0, T, \epsilon)$ such that, after $N(\Sigma_0, T, \epsilon)$ classes have been traversed, the probability of reaching any leaf node before T is lower than ϵ, provided that every cycle within each automaton visits at least one possibly tangible location that cannot be preempted by a message broadcast on exit from a necessarily vanishing location.*

Proof. Since each cycle of the NSTA visits at least some possibly tangible location of a STA, there exists a finite number $m \in \mathbb{N}$ such that the sojourn time of a possibly tangible location elapses every m state transitions of the NSTA. For

every number n, after $n \cdot m$ state transitions, at least n tangible sojourn times have been completed and the total elapsed time $T_{n \cdot m}$ is at least $\sum_{i=1}^{n} \tau_i$, where τ_i denotes the i-th sojourn time completed in a possibly tangible location. Since the random variables τ_i are independent and distributed according to a finite number of PDFs with supports $[a_i, b_i]$ with $b_i > 0$, for all $\epsilon > 0$ there exists $k(\epsilon, T) \in \mathbb{N}$ such that $Prob\{T_{k(\epsilon,T)} < T\} < \epsilon$. According to the construction of the transient tree, $Prob\{T_{k(\epsilon,T)} < T\}$ is the total probability obtained by summing up the quantity $\eta_T(\cdot)$ over all nodes at depth $k(\epsilon, T)$. Since the tree is constructed in breadth first order, for any allowed approximation ϵ, there exists $k(\epsilon, T)$ such that, when the tree reaches depth $k(\epsilon, T)$, the sum of probabilities $\eta_T(\cdot)$ over all the nodes of the frontier of the tree is lower than ϵ. □

4 Example

Fig. 2 shows the NSTA model of a flexible manufacturing system where one consumer $\mathcal{C}$ alternates between two producers $\mathcal{A}$ and $\mathcal{B}$, scheduling optional activities when time laxity occurs. In so doing, the consumer uses time observations to reduce the probability of performing an optional task while the next producer is waiting for the consumption of the previously produced item: two clocks (a and b) keep memory of the time elapsed since the last task accepted from the two producers. An optional activity is scheduled when the time elapsed since the last synchronization with the next producer is lower than a given threshold. For simplicity of the model, sojourn times are assumed to be uniformly distributed on the supports (but any expolynomial probability density function could be managed as well).

The synchronization between the STA occurs through symmetric handshakes: after the end of a production, producer $\mathcal{A}$ broadcasts a message A and waits for the echo AA; on reception of the echo, it immediately broadcasts a new A message. With this protocol, the first A message (indicating the end of a production phase of $\mathcal{A}$) can be lost if $\mathcal{C}$ is still consuming the item produced by producer $\mathcal{B}$, forcing producer $\mathcal{A}$ to wait for the end of consumption signaled by $\mathcal{C}$ with message AA; the availability of a produced unit is then communicated again with the second A message (the protocol for producer $\mathcal{B}$ is analogous).

The example is intended to point out some basic modeling patterns of the NSTA formalism: (1) clock values do not affect the distribution of location sojourn times, but they can be used to restrict the choice of the next location; (2) reception of a broadcast preempts the sojourn time in the location; (3) broadcast is non-blocking for the sender, and it is relevant only for receivers in an accepting location; (4) on completion of the sojourn time in any location, if no outgoing edge is enabled, an automaton remains blocked until some accepted action is received or clocks reach a value that satisfy some guard on an outgoing edge.

Structural complexities with relevant impact on the analysis are also illustrated by this example: (1) the overall state is composed of the vector of (discrete) locations in the three automata, and of the (continuous) values of the two consumer clocks and three remaining sojourn times; (2) the two consumer clocks

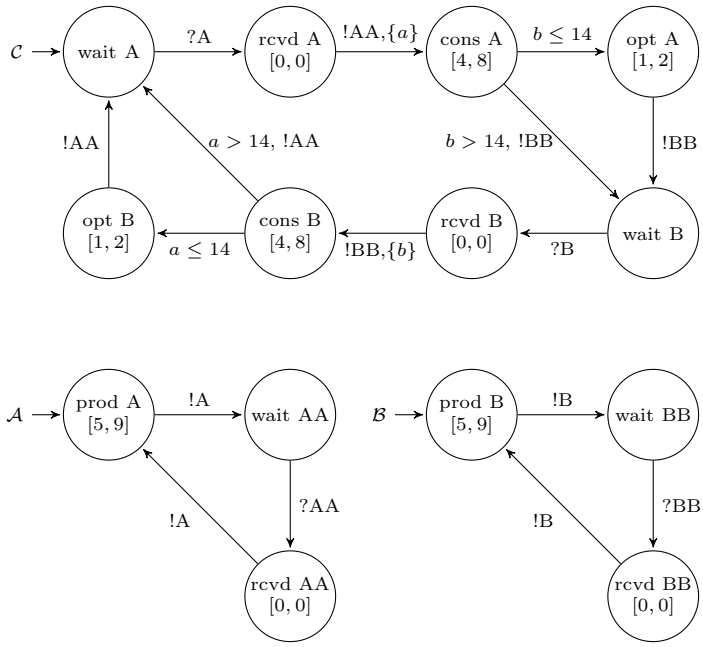
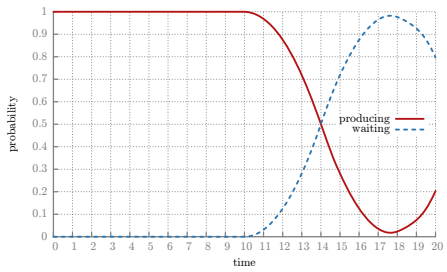
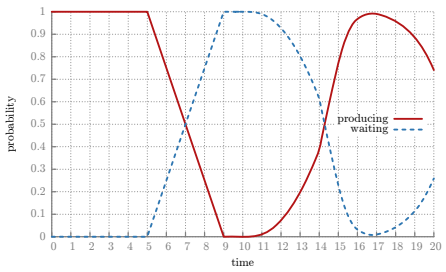


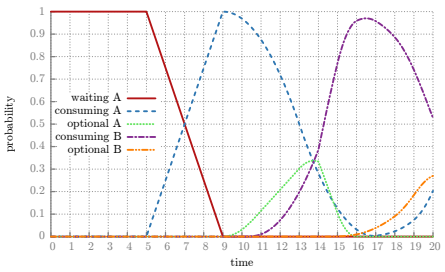
Fig. 2. An alternating producer/consumer scheme



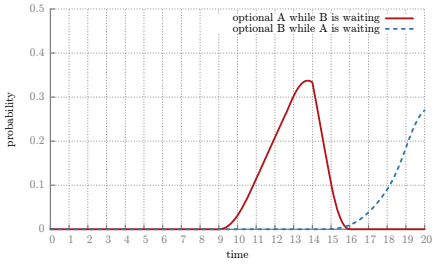
(a) Producer $\mathcal{A}$



(b) Producer $\mathcal{B}$



(c) Consumer $\mathcal{C}$



(d) Consumer causing a wait with optional tasks

Fig. 3. Transient analysis of the producer/consumer example

are independent and overlap their activity cycles between reset points and guard usage; (3) automata include both tangible and vanishing states, and broadcast actions can also be issued on exit from vanishing locations.

For any given time bound T, termination of transient analysis is guaranteed by Theorem 1: every cycle in every automaton visits at least one location with minimum sojourn time greater than zero that cannot be preempted by any broadcast issued on exit from some vanishing state. If sojourn times of all the locations included a minimum sojourn equal to zero, then termination would be guaranteed by Theorem 2 for any allowed error $\epsilon > 0$.

Fig. 3 reports, for each automaton, the transient state probabilities derived through a preliminary implementation of the analysis algorithm of Sect. 3 based on the symbolic calculus library of the ORIS tool [11,14]. As an example of relevant measure, we consider also the probability (for each time instant) that the consumer is performing an optional task while the next producer is waiting.

5 Conclusions

In this paper, we introduced the model of networks of stochastic timed automata, an extension of timed automata with stochastic semantics and message broadcasts between its components. We provided a characterization of the underlying stochastic process of STA and NSTA, identifying the conditions corresponding to each family of stochastic processes (i.e., CTMCs, SMPs, MRPs and GSMPs).

A technique for the transient analysis of NSTA based on stochastic state classes was proposed. We defined an iterative procedure for the construction of the transient tree of stochastic state classes, and identified sufficient conditions for its termination. Modeling examples were presented in order to illustrate common design patterns of the formalism, providing transient state probabilities computed analytically with the ORIS tool (and validated through simulation). Notably, this result constitutes the basis for the application of time-bounded probabilistic model checking techniques based on stochastic state classes to networks of STA.

References

1. Alur, R., Bernadsky, M.: Bounded model checking for GSMP models of stochastic real-time systems. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 19–33. Springer, Heidelberg (2006)
2. Alur, R., Courcoubetis, C., Dill, D.: Model-checking for Probabilistic Real-time Systems. In: Leach Albert, J., Monien, B., Rodríguez-Artalejo, M. (eds.) ICALP 1991. LNCS, vol. 510, pp. 115–126. Springer, Heidelberg (1991)
3. Alur, R., Courcoubetis, C., Dill, D.L.: Verifying Automata Specifications of Probabilistic Real-time Systems. In: Huizing, C., de Bakker, J.W., Rozenberg, G., de Roever, W.-P. (eds.) REX 1991. LNCS, vol. 600, pp. 28–44. Springer, Heidelberg (1992)
4. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. 126(2), 183–235 (1994)

5. Baier, C., Bertrand, N., Bouyer, P., Brihaye, T., Größer, M.: Probabilistic and topological semantics for timed automata. In: Arvind, V., Prasad, S. (eds.) FSTTCS 2007. LNCS, vol. 4855, pp. 179–191. Springer, Heidelberg (2007)
6. Baier, C., Bertrand, N., Bouyer, P., Brihaye, T., Größer, M.: Almost-sure model checking of infinite paths in one-clock timed automata. In: LICS 2008, pp. 217–226. IEEE CS (2008)
7. Behrmann, G., Larsen, K.G., Pearson, J., Weise, C., Yi, W.: Efficient timed reachability analysis using clock difference diagrams. In: Halbwachs, N., Peled, D.A. (eds.) CAV 1999. LNCS, vol. 1633, pp. 341–353. Springer, Heidelberg (1999)
8. Bertrand, N., Bouyer, P., Brihaye, T., Markey, N.: Quantitative model-checking of one-clock timed automata under probabilistic semantics. In: QEST 2008, pp. 55–64. IEEE CS (2008)
9. Bobbio, A., Telek, M.: Markov regenerative SPN with non-overlapping activity cycles. In: IPDS 1995, pp. 124–133. IEEE CS (1995)
10. Bouyer, P., Brihaye, T., Jurdzinski, M., Menet, Q.: Almost-sure model-checking of reactive timed automata. In: QEST 2012, pp. 138–147. IEEE CS (2012)
11. Bucci, G., Carnevali, L., Ridi, L., Vicario, E.: Oris: a tool for modeling, verification and evaluation of real-time systems. Int. J. on Softw. Tools for Techn. Transfer 12(5), 391–403 (2010)
12. Bucci, G., Piovosi, R., Sassoli, L., Vicario, E.: Introducing probability within state class analysis of dense time dependent systems. In: QEST 2005, pp. 13–22. IEEE CS (2005)
13. Carnevali, L., Grassi, L., Vicario, E.: State-density functions over DBM domains in the analysis of non-Markovian models. IEEE Trans. Softw. Eng. 35(2), 178–194 (2009)
14. Carnevali, L., Ridi, L., Vicario, E.: A framework for simulation and symbolic state space analysis of non-Markovian models. In: Flammini, F., Bologna, S., Vittorini, V. (eds.) SAFECOMP 2011. LNCS, vol. 6894, pp. 409–422. Springer, Heidelberg (2011)
15. Ciardo, G., German, R., Lindemann, C.: A characterization of the stochastic process underlying a stochastic Petri net. IEEE Trans. Softw. Eng. 20(7), 506–515 (1994)
16. Cox, D.R.: The analysis of non-Markovian stochastic processes by the inclusion of supplementary variables. Math. Proc. Cambridge 51, 433–441 (1955)
17. Dill, D.L.: Timing assumptions and verification of finite-state concurrent systems. In: Sifakis, J. (ed.) CAV 1989. LNCS, vol. 407, pp. 197–212. Springer, Heidelberg (1990)
18. Haas, P.J.: Stochastic Petri Nets: Modelling, Stability, Simulation. Springer (2002)
19. Horváth, A., Paolieri, M., Ridi, L., Vicario, E.: Probabilistic model checking of non-Markovian models with concurrent generally distributed timers. In: QEST 2011, pp. 131–140. IEEE CS (2011)
20. Horváth, A., Paolieri, M., Ridi, L., Vicario, E.: Transient analysis of non-Markovian models using stochastic state classes. Perform. Eval. 69(7-8), 315–335 (2012)
21. Kulkarni, V.: Modeling and analysis of stochastic systems. Chapman & Hall (1995)
22. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
23. Kwiatkowska, M., Norman, G., Segala, R., Sproston, J.: Verifying quantitative properties of continuous probabilistic timed automata. In: Palamidessi, C. (ed.) CONCUR 2000. LNCS, vol. 1877, pp. 123–137. Springer, Heidelberg (2000)

24. Kwiatkowska, M., Norman, G., Segala, R., Sproston, J.: Automatic verification of real-time systems with discrete probability distributions. Theor. Comput. Sci. 282(1), 101–150 (2002)
25. Maler, O., Larsen, K.G., Krogh, B.H.: On zone-based analysis of duration probabilistic automata. In: INFINITY, pp. 33–46 (2010)
26. Vicario, E., Sassoli, L., Carnevali, L.: Using stochastic state classes in quantitative evaluation of dense-time reactive systems. IEEE Trans. Softw. Eng. 35(5), 703–719 (2009)