From Stochastic Automata to Timed Automata: Abstracting probability in a compositional manner
(Extended Abstract)

Pedro R. D'Argenio

FaMAF, Universidad Nacional de Córdoba,
Ciudad Universitaria, 5000 Córdoba, Argentina
dargenio@famaf.unc.edu.ar

Abstract. We present a translation from stochastic automata [10, 8] into timed automata with deadlines [5]. The translation abstracts probabilities and preserves trace behaviour. Moreover, it is compositional in the sense that the translation of the parallel composition of two stochastic automata is equivalent to the parallel composition of the timed automata resulting from the translation of each component.

1 Introduction

The last few years have witnessed a growing interest in combining formal methods with performance and reliability techniques, the reason being that systems, like embedded systems or communication protocols, do not only require to be functionally correct but also efficient. Approaches that combine qualitative and quantitative analysis of system models have been addressed from many different perspectives including process algebra and model checking (e.g. [1, 15, 16, 22, 12, 4, 8, 17, 6]). Thus, these models provide a framework both for functional analysis and for obtaining performance and reliability measures of the system design under consideration.

One such approach is based on stochastic automata. The stochastic automata model [10, 8] is a variant of the automata model inspired by the so-called *generalised semi-Markov processes (GSMPs)* [13] and timed automata [2]. Stochastic automata are intended to describe timed systems in which the occurrence time of an event is a random variable. Moreover, they provide an adequate framework for composition and it serves as the underlying semantic model for the process algebra $\spadesuit$ [10, 8]. Stochastic automata properly contain stochastic models such as continuous time (semi-)Markov chains and (semi-)Markov decision chains, and a large class of GSMPs [8].

In its generality, stochastic automata are amenable to discrete event simulation techniques as a tool to gather performance and reliability information. Alternatively, restricted instances, such as continuous-time Markov chains, can be used to apply analytical and numerical techniques. In [11, 8] we already showed the potential use of stochastic automata both as a model for performance analysis and as a model for verification. In this article we are concerned on developing further in this last direction.

The timed automata model [2] has been adopted with large success for the verification of real-time systems. Efficient techniques and tools have been developed and successfully used in many industrial-based case studies. Stochastic automata can be seen as an extension of timed automata in which the occurrence time of events is stochastic rather than non-deterministic. As a consequence one might think that a timed automaton is an abstraction of stochastic automata, in which stochastic information has been "forgotten".

In this article, we formalise such an abstraction by translating stochastic automata into timed automata preserving *probable behaviour*. More specifically, the target of our translation is a variant of timed automata with deadlines [5]. The translation preserves *(timed) trace behaviour* in the sense that traces that are likely to occur in the original stochastic automaton are possible in the translation timed automaton and vice-versa. To show adequacy of the translation, we proceed in two steps. First, a probabilistic abstracted semantics is given to SA. Such semantic preserves the probable executions of the probabilistic semantics. The translation timed automata is then shown to define the same set of traces as the probabilistic abstracted semantics. Moreover, we also show that this translation commutes with the parallel composition. That is, the translation of the parallel composition of two stochastic automata is equivalent to the parallel composition of the timed automata resulting from the translation of each stochastic automaton.

The advantage of the translation is to profit in the stochastic automata world from the successful development in the timed automata world.

Outline of the Article. Section 2 introduces the basic concepts including probabilistic and timed transition systems. In Section 3 the stochastic automata model is defined together with its probabilistic behaviour as well as the probability abstracted behaviour. Section 4 recalls timed automata with deadlines. The translation and its adequacy criteria are given in Section 5. The compositionality of the translation is proved in Section 6. Section 7 discusses related work and the article concludes in Section 8.

2 Transition Systems

Probabilistic transition systems (PTS for short) are transition systems whose transitions impose a probabilistic jump. More precisely, in a PTS, a transition does not lead to a single state but to a probability space whose sample space is a set of states. The model defined below deals with any kind of probability space. It is a modification of the model introduced in [10, 8] and is basically a generalisation of models that only deal with discrete probabilities, e.g. [23, 22].

A *probability space* is a tuple $(\Omega, \mathcal{F}, P)$ where Ω is the *sample space*, $\mathcal{F}$ is a σ-*algebra* containing subsets of Ω, and P is a *probability measure* on the measurable space $(\Omega, \mathcal{F})$. If $\mathcal{P}$ is a probability space, we write $\Omega_{\mathcal{P}}$, $\mathcal{F}_{\mathcal{P}}$ and $P_{\mathcal{P}}$ for its sample space, σ-algebra, and probability measure, respectively[1]. Let $\mathit{Prob}(H)$ denote the set of probability spaces $\mathcal{P}$ such that $\Omega_{\mathcal{P}} \subseteq H$.

Definition 1. *A* probabilistic transition system *(PTS for short) is a structure* $\mathit{PTS} = (\Sigma, \mathcal{L}, \longrightarrow)$ *where (1)* Σ *is a set of* states*, (2)* $\mathcal{L}$ *is a set of* labels*, and (3)* $\longrightarrow \subseteq \Sigma \times \mathcal{L} \times \mathit{Prob}(\Sigma)$ *is the* (probabilistic) transition relation*. We write* $\sigma \xrightarrow{\ell} \mathcal{P}$ *whenever* $\langle \sigma, \ell, \mathcal{P} \rangle \in \longrightarrow$.

A probabilistic transition $\sigma \xrightarrow{\ell} \mathcal{P}$ is said to be *trivial* if its probability space $\mathcal{P}$ is trivial, i.e., if the sample space contains only one element. In this case we write $\sigma \xrightarrow{\ell} \sigma'$ provided $\Omega_{\mathcal{P}} = \{\sigma'\}$. A *(Labelled) Transition System* (*LTS* for short) is a *PTS* where all transitions are trivial. A *Timed Probabilistic Transition System* is a $\mathit{PTS} = (\Sigma, \mathcal{L}, \longrightarrow)$ where:

- The set of labels $\mathcal{L}$ is the disjoint union of a set $\mathcal{A}$ of *actions* and the set $\mathbb{R}_{>0}$ of positive real numbers intended to represent the passage of time.
- Transitions labelled with $d \in \mathbb{R}_{>0}$ are trivial and called *timed transitions*.
- Timed transitions satisfy *time additivity* ($\exists \sigma''.\ \sigma \xrightarrow{d} \sigma'' \xrightarrow{d'} \sigma'$ iff $\sigma \xrightarrow{d+d'} \sigma'$) and *time determinism* ($\sigma \xrightarrow{d} \sigma' \wedge \sigma \xrightarrow{d} \sigma'' \implies \sigma' = \sigma''$) [24].

A LTS that is also a timed PTS is called a *Timed Transition System*.

An *execution fragment* of a PTS is a path obtained by traversing the PTS. An *execution* is a maximal path on PTS. A *supported execution* is an execution likely to occur. More precisely, an *execution fragment* of *PTS* is a is a finite sequence $\xi \equiv \sigma_0 \ell_1 \sigma_1 \ldots \ell_n \sigma_n$ such that, for all $0 \leq i < n$, $\sigma_i \xrightarrow{\ell_{i+1}} \mathcal{P}_{i+1}$ for some $\mathcal{P}_{i+1}$ with $\sigma_{i+1} \in \Omega_{\mathcal{P}_{i+1}}$. An *execution* of *PTS* is either a maximal execution fragment (i.e. $\sigma_n \not\longrightarrow$) or an infinite sequence $\rho \equiv \sigma_0 \ell_1 \sigma_1 \ell_2 \sigma_2 \ldots$ such that, for all $i \geq 0$, $\sigma_i \xrightarrow{\ell_{i+1}} \mathcal{P}_{i+1}$ for some $\mathcal{P}_{i+1}$ with $\sigma_{i+1} \in \Omega_{\mathcal{P}_{i+1}}$. ρ is *supported* if in addition every σ_i is in the support set of $P_{\mathcal{P}_i}$[2]. For a fragment or an execution ρ, let $\mathit{first}(\rho)$ and $\mathit{last}(\rho)$ denote the *first* and *last* state in ρ, respectively; $\mathit{last}(\rho)$ is not defined if ρ is infinite. Provided $\mathit{last}(\xi) = \mathit{first}(\rho)$, $\xi\rho$ represents the *concatenation* of ξ and ρ; in this case we say that ξ is a *prefix* of $\xi\rho$.

Let $\mathit{frags}(\mathit{PTS})$, $\mathit{execs}(\mathit{PTS})$, and $\mathit{supp\_execs}(\mathit{PTS})$ denote the set of all execution fragments, all executions, and all supported executions of *PTS*, respectively. Let $\mathit{frags}(\mathit{PTS}, \sigma)$, $\mathit{execs}(\mathit{PTS}, \sigma)$, and $\mathit{supp\_execs}(\mathit{PTS}, \sigma)$ be their respective subsets restricted to the sequences that start from σ. Notice that in the context of LTSs, the sets of executions and supported executions are the same, i.e., $\mathit{execs}(\mathit{LTS}) = \mathit{supp\_execs}(\mathit{LTS})$.

The probability measure of a set of executions depends on how non-determinism is resolved which is done by a *scheduler* or *adversary* [23].

[1] We assume the reader is familiar with the basics of probability and measure theory.
[2] Recall that the support set of a probability measure is the smallest closed subset of the sample space whose measure is 1.

Definition 2. *A* scheduler *on a PTS is a function* $\mathsf{S} : frags(PTS) \to (\longrightarrow)$ *such that* $\mathsf{S}(\xi) = last(\xi) \xrightarrow{\ell} \mathcal{P}$ *for some* $\ell \in \mathcal{L}$ *and* $\mathcal{P} \in Prob(\Sigma)$ *whenever they exist (i.e.* ξ *is not maximal).*

The set $execs(PTS, \mathsf{S}) \subseteq execs(PTS)$ of *executions induced by a scheduler* S *in PTS* contains all the executions $\rho \equiv \sigma_0\ell_1\sigma_1\ell_2\sigma_2\ldots$ and for all $0 \leq k < |\rho|$, $\mathsf{S}(\sigma_0\ell_1\sigma_1\ldots\sigma_k) = \sigma_k \xrightarrow{\ell_{k+1}} \mathcal{P}$ and $\sigma_{k+1} \in \Omega_{\mathcal{P}}$ for some $\mathcal{P}$. Let $execs(PTS, \mathsf{S}, \sigma) \stackrel{\text{def}}{=} execs(PTS, \mathsf{S}) \cap execs(PTS, \sigma)$. We write $execs(\mathsf{S})$ and $execs(\mathsf{S}, \sigma)$ instead of $execs(PTS, \mathsf{S})$ and $execs(PTS, \mathsf{S}, \sigma)$ if it is clear from the context.

Every scheduler S define a probability space on the executions of *PTS* as follows. Given a state $\sigma \in \Sigma$, $bcones(\mathsf{S}, \sigma)$ is the smallest set containing all subsets $\Xi \subseteq execs(PTS)$ that can be defined inductively as follows:

1. $\Xi = execs(\mathsf{S}, \sigma)$, or
2. For $\mathsf{S}(\sigma) = \sigma \xrightarrow{\ell} \mathcal{P}$ there is a measurable set $A \in \mathcal{F}_{\mathcal{P}}$ and a function $\hat{\Xi}$, with $\hat{\Xi}(\sigma') \in bcones(\mathsf{S}/(\sigma\ell\sigma'), \sigma')$ for all $\sigma' \in A$, such that

$$\Xi = \{(\sigma\ell\sigma')\rho \in execs(\mathsf{S}) \mid \sigma' \in A \wedge \rho \in \hat{\Xi}(\sigma')\}$$

where $(\mathsf{S}/\xi)(\rho) \stackrel{\text{def}}{=} \textbf{if } last(\xi) = first(\rho) \textbf{ then } \mathsf{S}(\xi\rho) \textbf{ else } \mathsf{S}(\rho)$.

Every $\Xi \in bcones(\mathsf{S}, \sigma)$ is called a *basic cone*. Let $\mathcal{F}(bcones(\mathsf{S}, \sigma))$ be the σ-algebra generated by all sets in $bcones(\mathsf{S}, \sigma)$. For every a scheduler S there is probability measure P^{S}_{σ} on the measurable space $(execs(\mathsf{S}, \sigma), \mathcal{F}(bcones(\mathsf{S}, \sigma)))$. P^{S}_{σ} is defined as the unique probability measure such that for every $\Xi \in bcones(\mathsf{S}, \sigma)$

$$P^{\mathsf{S}}_{\sigma}(\Xi) = \begin{cases} 0 & \text{if } \Xi \cap execs(\mathsf{S}, \sigma) = \varnothing \\ 1 & \text{if } \Xi = execs(\mathsf{S}, \sigma) \\ \displaystyle\int_{\mathcal{F}_{\mathcal{P}}} f \, \mathbf{d}P_{\mathcal{P}} & \text{if } \varnothing \neq \Xi \cap execs(\mathsf{S}, \sigma) \neq execs(\mathsf{S}, \sigma) \text{ with} \\ & \mathsf{S}(\sigma) = \sigma \xrightarrow{\ell} \mathcal{P} \text{ and } f(\sigma') \stackrel{\text{def}}{=} P^{\mathsf{S}/(\sigma\ell\sigma')}_{\sigma'}(\Xi/(\sigma\ell\sigma')) \end{cases}$$

where $\Xi/\xi \stackrel{\text{def}}{=} \{\rho \in execs(\mathsf{S}/\xi, last(\xi)) \mid \xi\rho \in \Xi\}$. (Clearly $\Xi/\xi \in bcones(\mathsf{S}/\xi, last(\xi))$ if $\Xi \in bcones(\mathsf{S}, first(\xi))$.)

Executions will be used as fundamentals in the main step to abstract from the measure of a stochastic system. Once the probability measure is abstracted, we will resort to non-probabilistic semantic relations that relate states of a transition system according to their observable behaviour (see e.g. [19, 18]).

Let $(\Sigma, \mathcal{L}, \longrightarrow)$ be a *LTS*. A relation $R \subseteq \Sigma \times \Sigma$ is a *simulation* if, for all $\langle\sigma_1, \sigma_2\rangle \in R$, whenever $\sigma_1 \xrightarrow{\ell} \sigma_1'$, there exists σ_2' such that $\sigma_2 \xrightarrow{\ell} \sigma_2'$ and $\langle\sigma_1', \sigma_2'\rangle \in R$. σ_1 *is simulated by* σ_2, notation $\sigma_1 \lesssim \sigma_2$, if there is a simulation R with $\langle\sigma_1, \sigma_2\rangle \in R$. If R is a symmetric simulation, R is called *bisimulation*, and in this case we denote $\sigma_1 \sim \sigma_2$ whenever $\langle\sigma_1, \sigma_2\rangle \in R$.

A sequence $\ell_1, \ell_2, \ldots, \ell_n \in \mathcal{L}^*$, $n \geq 0$, is a *trace* of a state σ in a *LTS* if there is an execution $\sigma\ell_1\sigma_1\ell_2\sigma_2\ldots\sigma_{n-1}\ell_n\sigma_n \in execs(LTS, \sigma)$. The *set of all traces* of a state σ is denoted by $tr(\sigma)$. σ_1 and σ_2 are *trace-equivalent*, notation $\sigma_1 =_{tr} \sigma_2$, if they have the same set of traces, i.e., $tr(\sigma_1) = tr(\sigma_2)$.

If σ_1 and σ_2 are states of LTS_1 and LTS_2 respectively, then $\sigma_1 \bowtie \sigma_2$ (where $\bowtie$ is either $\lesssim$, $\sim$, $=_{tr}$) whenever $\sigma_1 \bowtie \sigma_2$ in the (disjoint) union of LTS_1 and LTS_2.

It is known that two states that are bisimilar can simulate each other, and that if a state σ_1 can be simulated by σ_2, i.e. $\sigma_1 \lesssim \sigma_2$, then $tr(\sigma_1) \subseteq tr(\sigma_2)$.

3 Stochastic Automata

A stochastic automaton [10, 8] is a LTS extended with *clock variables* that can be set set to 0 (in which case it becomes *active*) and check whether it reaches a random value, in which moment it is *terminated*. Each clock x has associated a random variable which takes a random value according to the probability distribution function F_x. This random value is the *termination value* of clock x. Thus, clock x may *enable* different transitions when it reaches the termination value.

Definition 3. *Let $\mathcal{C}$ be a set of clocks and $\mathcal{A}$ a set of action names. A* stochastic automaton *(SA for short) is a tuple (LTS, h_s) where $LTS = (\mathcal{S}, \mathcal{A}, \longrightarrow)$ is a labelled transition system and $h_s \subseteq \longrightarrow \times (\wp(\mathcal{C}) \times \wp(\mathcal{C}))$ is a relation called* stochastic annotation. *$s \xrightarrow{a,C_t,C_r} s'$ denotes $\langle s \xrightarrow{a} s', C_t, C_r\rangle \in h_s$ and is called* edge *with C_t being the* trigger set *and C_r the* resetting set*. In this context, states are called* locations *and are denoted with $s, s', s_1, \ldots$. (The same nomenclature will be used for timed automata.)*

The edge $s \xrightarrow{a,C_t,C_r} s'$ becomes *enabled* when every clock in C_t terminates, i.e., every clock has reached or passed its termination value. When the system is in location s and $s \xrightarrow{a,C_t,C_r} s'$ becomes enabled, it moves to location s' performing action a and resetting every clock in C_r. When a clock x is reset, its value is set to zero and its termination value is sampled according to F_x. Once location s' is reached, the system should idle there until an outgoing edge becomes enabled.

Example 1. Consider a switch that controls a light in a stairway. People arrive and turn on the light once 30 minute average according to a Poisson process. They press the switch even if the light is still on. It switches automatically off exactly 2 minutes after the last time the switch was pressed.

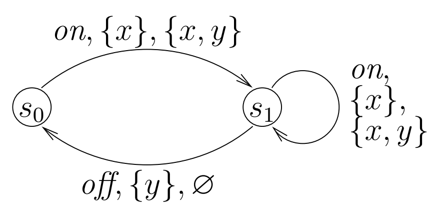


Fig. 1. The switch

Fig. 1 represents this switch. In the picture, circles represent locations and edges are represented by arrows. Besides, $F_x(t) = 1 - e^{-\frac{t}{30}}$ and $F_y(t) = \textbf{if } t < 2 \textbf{ then } 0 \textbf{ else } 1$.

Probabilistic Semantics. A valuation is a function that assigns a real number in $\mathbb{R}$ to each clock in $\mathcal{C}$. There are two kinds of valuations. The first kind, ranging over v, v', v_1,..., record the passage of time, while the other, ranging over e, e', e_1,..., are used to save the termination value of the clocks. The latter ones are called *termination conditions* and the first ones, just *valuations*.

Let SA be a stochastic automaton with clocks in $\mathcal{C}$. Let n be the cardinality of $\mathcal{C}$. A state is a triple consisting of a location, a valuation, and a termination condition. Thus, $\Sigma_{\mathcal{S}} \stackrel{\text{def}}{=} \mathcal{S} \times \mathbb{R}^n_{\geq 0} \times \mathbb{R}^n$ is the set of states. Notice that for any location s, valuation v, and termination condition e there is a unique tuple $\langle s, v(x_1), \ldots, v(x_n), e(x_1), \ldots, e(x_n)\rangle \in \Sigma_{\mathcal{S}}$. It is denoted by $\langle s, v, e\rangle$.

Definition 4. *The* semantics *of SA is given by* $PTS(SA) = (\Sigma_{\mathcal{S}}, \mathcal{A} \cup \mathbb{R}_{>0}, \longrightarrow)$ *where* $\longrightarrow$ *is defined in the following.*

Discrete (untimed) case: *Let* $v[C_r\!: =\!0](x) \stackrel{\text{def}}{=}$ **if** $x \in C_r$ **then** 0 **else** $v(x)$. *Then:*

$$\frac{s \xrightarrow{a, C_t, C_r} s' \quad \forall x \in C_t.\ v(x) \geq e(x) \quad v' = v[C_r\!: =\!0]}{\langle s, v, e\rangle \stackrel{a}{\longrightarrow} (\Sigma_{\mathcal{S}}, \mathcal{B}(\Sigma_{\mathcal{S}}), P^{s'}_{v',e})}$$

where $\mathcal{B}(\Sigma_{\mathcal{S}})$ *is the Borel algebra with elements in* $\Sigma_{\mathcal{S}}$ *and* $P^{s'}_{v',e}$ *is the unique probability measure on* $\mathcal{B}(\Sigma_{\mathcal{S}})$ *induced by the distribution functions*

$$F_0 = \mathbf{I}_{s'} \quad F_i = \mathbf{I}_{v'(x_i)} \quad F_{n+i} = \begin{cases} F_{x_i} & \textbf{if } x_i \in C_r \\ \mathbf{I}_{e(x_i)} & \textbf{otherwise} \end{cases}$$

with $0 < i \leq n$ *and* $\mathbf{I}_d(d') \stackrel{\text{def}}{=}$ **if** $d = d'$ **then** 1 **else** 0.

Timed case: *Let* $(v + d)(x) \stackrel{\text{def}}{=} v(x) + d$. *Then:*

$$\frac{\forall\, d' \!\leq\! d.\ \forall\, s \xrightarrow{a, C_t, C_r}.\ \exists x \in C_t.\ (v + d')(x) \leq e(x)}{\langle s, v, e\rangle \stackrel{d}{\longrightarrow} \langle s, v + d, e\rangle}$$

The **discrete** case captures the intuition described above: $s \xrightarrow{a, C_t, C_r} s'$ can be taken whenever it becomes enabled, that is, every $x \in C_t$ is terminated in the current valuation v and the current termination condition e, namely $v(x) \geq e(x)$. Then clocks in C_r are set to 0 and their termination values are sampled according to the respective distribution function. Indicator functions take care that the location and valuation are the newly defined, and that the termination values of clocks not in C_r remain unchanged.

In the **timed** case, the stochastic automaton is allowed to stay at location s for d units of time as long as no edge becomes enabled during this time.

Probability Abstracted Semantics. To abstract from probabilities in a SA, its semantics should be given in terms of timed LTS rather than PTS. To do so, one probabilistic step is instead interpreted as several trivial (non-deterministic) transitions whose target states are probable states. What "probable state" means in the context of continuous probabilities is not so clear. Timed transition should be as before.

A first (not so naive) view may be to consider probable those states that fall in the support set of the probability measure of the probabilistic transition. In this sense the discrete transition relation of the LTS should be defined by:

$$\frac{s \xrightarrow{a,C_t,C_r} s' \quad \forall x \in C_t.\ v(x) \geq e(x) \quad v' = v[C_r:=0] \qquad \textbf{if } x \in C_r \textbf{ then } e'(x) \in \mathit{supp}(F_x) \textbf{ else } e'(x) = e(x)}{\langle s, v, e\rangle \xrightarrow{a} \langle s', v', e'\rangle}$$

Notice that the enabling condition $\forall x \in C_t.\ v(x) \geq e(x)$ is the same as before but, now, it induces many non-deterministic transitions, one for each possible termination condition.

Consider the SA given in Fig. 2. Let F_x and F_y be uniform distributions in $[0, 1]$ and $[1, 2]$, respectively. In $\mathit{PTS}(SA_{ex1})$, the probability that c occurs before b is 0 for any scheduler. However, in the LTS obtained from SA_{ex1} and according to the previous rule, the following execution is possible:

$$\langle s_0, v_0, e_0\rangle\, a\, \langle s_1, v_0, e_1\rangle\, 1\, \langle s_1, v_1, e_1\rangle\, c\, \langle s_3, v_1, e_1\rangle\, b\, \langle s_4, v_1, e_1\rangle \qquad (1)$$

where $v_d(z) = e_d(z) = d$ for $z \in \{x, y\}$ and $d \in \mathbb{R}_{\geq 0}$. Execution (1) is in fact a supported execution in $\mathit{PTS}(SA_{ex1})$ (i.e., it is in $\mathit{supp\_execs}(\mathit{PTS}(SA_{ex1}), \langle s_0, v_0, e_0\rangle)$). Nevertheless, its presence in the probability abstracted semantics is undesirable due to the fact that execution of c before b is improbable.

Therefore, we follow a different approach. We still look at the support set, but instead eliminate the improbable borders. We call this set the *useful domain* of a distribution.

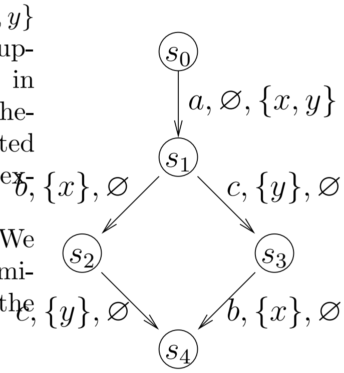


Fig. 2. SA_{ex1}

Definition 5. *Let F be a distribution function with support set $\mathit{supp}(F)$. $\mathit{supp}(F)$ can be written as $\bigcup_i I_i$ where each I_i is an interval on the real line such that if $I_i \cup I_j$ is also an interval then $i = j$ (i.e., each interval I_i is "maximal"). The* useful domain *of F is the set $\mathit{udom}(F) = \bigcup_i I'_i$ where each interval I'_i satisfies:*

1. *$\mathit{lub}(I_i) = \mathit{lub}(I'_i)$ and $\mathit{glb}(I_i) = \mathit{glb}(I'_i)$, and*
2. *for $d \in \{\mathit{lub}(I'_i), \mathit{glb}(I'_i)\}$, $d \in I'_i \Leftrightarrow P_F(\{d\}) > 0$;*

where lub and glb are respectively the lowest upper bound and the greatest lower bound of a given interval, and P_F is the unique probability measure defined by F. (Note that $\mathit{glb}(I_i)$ may be $-\infty$ and $\mathit{lub}(I_i)$ may be ∞).

For example, if F_u is a uniform distribution in the interval $[1, 2]$, then $\mathit{udom}(F_u) = (1, 2)$; if $F_{mix}(d) = \textbf{if } d{=}2 \textbf{ then } \frac{1}{2} \textbf{ else } \frac{F_u(d)}{2}$ then $\mathit{udom}(F_{mix}) = (1, 2]$; if F_g is a geometric distribution, $\mathit{udom}(F_g) = \mathbb{N}$; if F_e is a negative exponential distribution, $\mathit{udom}(F_e) = (0, \infty)$. Notice that their support sets are $[1, 2]$, $[1, 2]$, $\mathbb{N}$, and $[0, \infty)$, respectively.

Definition 6. *The* probability abstracted semantics *of SA is given by* $LTS(SA) = (\Sigma_{\mathcal{S}}, \mathcal{A} \cup \mathbb{R}_{>0}, \longrightarrow)$ *where* $\longrightarrow$ *is defined by*

$$\frac{s \xrightarrow{a, C_t, C_r} s' \quad \forall x \in C_t.\ v(x) \geq e(x) \quad v' = v[C_r\!:=\!0] \qquad \textbf{if } x \in C_r \textbf{ then } e'(x) \in udom(F_x) \textbf{ else } e'(x) = e(x)}{\langle s, v, e \rangle \xrightarrow{a} \langle s', v', e' \rangle}$$

for the ***discrete*** *case, and the* ***timed*** *case is as in Definition 4.*

Notice that execution (1) is not present in $LTS(SA)$. This is a consequence of considering the useful domain rather than the support set of a distribution function in the definition of the **discrete** case.

Finding a criterion to assert adequacy is not straightforward due to the fact that, in the probability abstracted semantics, there is no notion of measure. Besides, there might be an execution in the probabilistic semantics that, point-wise, has probability 0, but any open cone[3] containing it has probability greater than 0 (execution (1) is precisely an example of it). This is a consequence of continuous probabilities.

A first approach to adequacy may be to verify that every execution ρ of $LTS(SA)$ is an execution of $PTS(SA)$ such that every open cone containing ρ has probability greater than 0 (for some scheduler), and vice-versa. Technically speaking, this adequacy criteria states that, for all state $\sigma \equiv \langle s, v, e \rangle$, $execs(LTS(SA), \sigma) = supp\_execs(PTS(SA), \sigma)$.

However, execution (1) *is* a supported execution of $PTS(SA_{ex1})$ and it is not present in $LTS(SA_{ex1})$. Therefore, a weaker notion of adequacy is necessary. Although $execs(LTS(SA), \sigma) \subseteq supp\_execs(PTS(SA), \sigma)$ will be required, the converse is relaxed. We still want that $execs(LTS(SA), \sigma)$ contains enough probable executions so that, for any scheduler of $PTS(SA)$, there is a set of executions with probability 1 (in $PTS(SA)$) that is contained in $execs(LTS(SA), \sigma)$. This adequacy criterion is stated in Theorems 1 and 2.

Theorem 1. *For every* $\sigma \in \Sigma_{\mathcal{S}}$, $execs(LTS(SA), \sigma) \subseteq supp\_execs(PTS(SA), \sigma)$.

The proof of this theorem should be clear from the fact that useful domains of distribution functions are included in their support set.

Theorem 2. *For every state* $\sigma \equiv \langle s, v, e \rangle$ *and for every scheduler* S *for* σ *there is a set* $\Xi \subseteq execs(PTS(SA), \sigma)$ *such that* $P^{\mathsf{S}}_{\sigma}(\Xi_{\sigma}) = 1$ *and* $\Xi \subseteq execs(LTS(SA), \sigma)$.

Proof. For every scheduler S, $\sigma = \langle s, v, e \rangle$ and $n \geq 0$, define the sets $\Xi^{\mathsf{S}}_{\sigma}(n)$ inductively as follows:
(1) $\Xi^{\mathsf{S}}_{\sigma}(0) = execs(\mathsf{S}, \sigma)$.

[3] An *open cone* is a cone in which every probabilistic transition defines an open rectangle rather than an arbitrary measurable set (see item 2 in the definition of cone).

(2) If $\mathsf{S}(\langle s,v,e\rangle) = \langle s,v,e\rangle \xrightarrow{a} \mathcal{P}^{s'}_{v',e}$, comes from edge $s \xrightarrow{a,C_t,C_r} s'$ as defined in rule **discrete**, Definition 4, then

$$\begin{aligned}\Xi^{\mathsf{S}}_{\sigma}(n+1) \;=\; \{(\langle s,v,e\rangle a\langle s',v',e'\rangle)\rho \mid & \\ & \textbf{if } x\in C_r \textbf{ then } e'(x)\in \mathit{udom}(F_x) \textbf{ else } e'(x)=e(x) \\ & \wedge\, \rho \in \Xi^{\mathsf{S}/(\langle s,v,e\rangle a\langle s',v',e'\rangle)}_{\langle s',v',e'\rangle}(n)\}.\end{aligned}$$

(3) If $\mathsf{S}(\langle s,v,e\rangle) = \langle s,v,e\rangle \xrightarrow{d} \langle s,v+d,e\rangle$ is a timed transition, then

$$\Xi^{\mathsf{S}}_{\sigma}(n+1) \;=\; \{(\langle s,v,e\rangle d\langle s,v+d,e\rangle)\rho \mid \rho \in \Xi^{\mathsf{S}/(\langle s,v,e\rangle d\langle s,v+d,e\rangle)}_{\langle s,v+d,e\rangle}(n)\}.$$

Define $\Xi^{\mathsf{S}}_{\sigma} = \bigcap_{n\geq 0}\Xi^{\mathsf{S}}_{\sigma}(n)$. Clearly $\Xi^{\mathsf{S}}_{\sigma}(n) \in \mathit{bcones}(\mathsf{S},\sigma)$; as a consequence $\Xi^{\mathsf{S}}_{\sigma} \in \mathit{bcones}(\mathsf{S},\sigma)$. By induction, it can be proven that $P^{\mathsf{S}}_{\sigma}(\Xi^{\mathsf{S}}_{\sigma}(n)) = 1$ for all $n \geq 0$. Then $P^{\mathsf{S}}_{\sigma}(\Xi^{\mathsf{S}}_{\sigma}) = P^{\mathsf{S}}_{\sigma}(\bigcap_{n\geq 0}\Xi^{\mathsf{S}}_{\sigma}(n)) = \lim_{n\to\infty} P^{\mathsf{S}}_{\sigma}(\Xi^{\mathsf{S}}_{\sigma}(n)) = 1$. □

4 Timed Automata with Deadlines

Timed automata with deadlines [5] are a variant of traditional timed automata though both models share the same expressive power.

For a set of clocks $\mathcal{C}$, define the set Φ of *constraints* on $\mathcal{C}$ to be the set of propositional logic formulas with atomic propositions $x \leq d$ and $x < d$ where $x \in \mathcal{C}$ and $d \in \mathbb{R}_{\geq 0}$. Valuations can be lifted to clock constraints in the usual way. Denote $v \models \phi$ if constraint ϕ holds in valuation v.

Definition 7. *Let $\mathcal{C}$ be a set of clocks and $\mathcal{A}$ a set of action names. A* timed automaton (with deadline) *(TA for short) is a tuple (LTS, h_t) where $LTS = (\mathcal{S},\mathcal{A},\longrightarrow)$ is a labelled transition system and $h_t \subseteq \longrightarrow \times (\Phi\times\Phi\times\wp(\mathcal{C}))$ is a relation called* time annotation. *Let $s \xrightarrow{a,\phi_g,\phi_d,C} s'$ denotes $\langle s \xrightarrow{a} s', \phi_g, \phi_d, C\rangle \in h_t$ and is called* edge. *ϕ_g is called* guard *and ϕ_d,* deadline *and it must hold that $\phi_d \Rightarrow \phi_g$.*

In $s \xrightarrow{a,\phi_g,\phi_d,C} s'$, constraint ϕ_g states the moment the transition *may* be taken, constraint ϕ_d indicates when it *must* be taken, and the set C is the set of clocks that should be reset at the moment the transition occurs. The system is allowed to idle in a location s as long as all the deadlines of its outgoing edges are invalid. This behaviour is formalised as follows.

Definition 8. *The* semantics *of TA is given by $LTS(TA) = (\mathcal{S}\times\mathbb{R}^n_{\geq 0}, \mathcal{A}\cup\mathbb{R}_{>0},\longrightarrow)$ where n is the cardinality of $\mathcal{C}$ and $\longrightarrow$ is defined by:*

discrete

$$\frac{s \xrightarrow{a,\phi_g,\phi_d,C} s' \quad v\models\phi_g \quad v' = v[C{:=}0]}{\langle s,v\rangle \xrightarrow{a} \langle s',v'\rangle}$$

timed

$$\frac{\forall\, d'{\leq} d.\ (v+d')\models \mathit{tpc}_s}{\langle s,v\rangle \xrightarrow{d} \langle s,v+d\rangle}$$

where $\mathit{tpc}_s \stackrel{\text{def}}{=} \neg\bigvee\{\phi_d \mid s \xrightarrow{a,\phi_g,\phi_d,C} s'\}$ is the time progress condition.

Rule **discrete** states that whenever the system is in location s, the edge $s \xrightarrow{a,\phi_g,\phi_d,C} s'$ can be taken if ϕ_g holds in the current valuation v. When taking the transition, clocks in C are reset to 0. Rule **timed** states that the system is allowed to idle in s for d units of times if within this period no deadline forced the execution of a transition, i.e., predicate tpc_s holds within this interval.

5 From Stochastic Automata to Timed Automata

In the following, the translation from SA to TA is presented. First an informal derivation of the definition is given which (hopefully) explain the rationale behind a not so intuitive translation rule in Definition 9. Afterwards, adequacy theorems for this translation are given showing that it preserve timed traces.

Consider $\mathbf{E} \equiv s_1 \xrightarrow{a,\{x\},\{y\}} s_2$ where $udom(F_x) = (1, 2)$. It is probable that $\mathbf{E}$ is performed at any time in which $x > 1$, but certainly, if $x \geq 2$, it *must* be performed. Therefore, a possible translation for $\mathbf{E}$ is the TA edge $s_1 \xrightarrow{a,x>1,x\geq 2,\{y\}} s_2$.

This translation is naive in the sense that $\mathbf{E}$ is taken out of context. Location s_1 may be reached after clock x has terminated. For example consider that $\mathbf{E}$ is preceded by $\mathbf{E}' \equiv s_0 \xrightarrow{a,\{x\},\varnothing} s_1$ and suppose the system is at state $\langle s_0, v, e\rangle$ with $v(x) = e(x) = 1.5$. After $\mathbf{E}'$ was taken, state $\langle s_1, v, e\rangle$ is reached and clock x has terminated. Then edge $\mathbf{E}$ *must* be performed. Instead the translation edge $s_1 \xrightarrow{a,x>1,x\geq 2,\{y\}} s_2$ will allow for a 0.5 units delay at state $\langle s_1, v\rangle$[4]. In this context, the correct interpretation of $\mathbf{E}$ is $s_1 \xrightarrow{a,\mathbf{tt},\mathbf{tt},\{y\}} s_2$ (clock x has terminated and hence it cannot delay executions).

To distinguish whether x has terminated or not in location s_1, we define a function $\mathcal{I}$ that ranges on clocks and helps to record the current context. So, define $\mathcal{I}(x) = \bot$ if and only if x is *not active*. (Remember that a clock is active if it was set but did not terminate.) Thus, $\mathbf{E}$ is translated in two different TA edges: $(s_1, \mathcal{I}_1) \xrightarrow{a,x>1,x\geq 2,\{y\}} (s_2, \mathcal{I}_2)$ and $(s_1, \mathcal{I}'_1) \xrightarrow{a,\mathbf{tt},\mathbf{tt},\{y\}} (s_2, \mathcal{I}'_2)$ where $\mathcal{I}_1(x) \neq \bot$, $\mathcal{I}'_1(x) = \bot$, and $\mathcal{I}_2(x) = \mathcal{I}'_2(x) = \bot$. Notice that y is set on $\mathbf{E}$, hence it becomes active at this point. As a consequence $\mathcal{I}_2(y) \neq \bot$ and $\mathcal{I}'_2(y) \neq \bot$.

Suppose now that F_x is such that $udom(F_x) = (1, 2) \cup (3, 4)$. Encoding the guard of $\mathbf{E}$ by $x > 1$ and its deadline by $x \geq 4$ is not a good idea since $\mathbf{E}$ may be taken after delaying 2.5 units of time which is an improbable value. So, it is better to split the pair guard-deadline in two possibilities: one pair is $x > 1$ and $x \geq 2$, and the other $x > 3$ and $x \geq 4$. To distinguish these two possible encodings let $\mathcal{I}(x)$ take the value of the interval that should be encoded. Therefore, $\mathbf{E}$ can be translated in three different TA edges: $(s_1, \mathcal{I}_1) \xrightarrow{a,x>1,x\geq 2,\{y\}} (s_2, \mathcal{I}_2)$ with $\mathcal{I}_1(x) = (1, 2)$, $(s_1, \mathcal{I}'_1) \xrightarrow{a,x>3,x\geq 4,\{y\}} (s_2, \mathcal{I}'_2)$ with $\mathcal{I}'_1(x) = (3, 4)$, and $(s_1, \mathcal{I}''_1) \xrightarrow{a,\mathbf{tt},\mathbf{tt},\{y\}} (s_2, \mathcal{I}''_2)$ with $\mathcal{I}''_1(x) = \bot$. Similarly, $udom(F_y)$ may also be split in several noncontiguous intervals. As a consequence each of these three

[4] e does not play a role in TA!

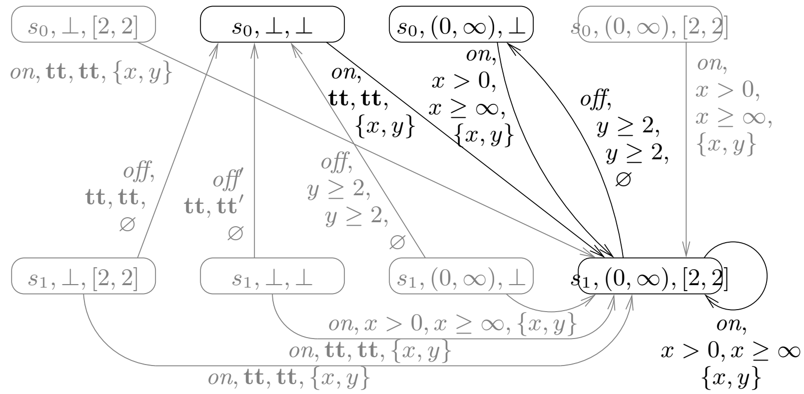


Fig. 3. Translation of the switch

edges may explode in several more since $\mathcal{I}_2(y)$, $\mathcal{I}'_2(y)$, and $\mathcal{I}''_2(y)$ may take as many values as noncontiguous intervals in $udom(F_y)$.

Translation function $\mathsf{sa2ta}$ is defined in the following. For a clock x such that $udom(F_x) = \bigcup_{i \in J} I_i$, where I_i and I_j are not contiguous nor intersecting for $i \neq j$, let $\mathcal{I}_x = \{I_i \mid i \in J\}$. Let $\Im$ be the set of all functions $\mathcal{I}$ that assigns a value $\mathcal{I}(x) \in \mathcal{I}_x \cup \{\bot\}$ to each clock $x \in \mathcal{C}$.

Definition 9. *Let* $SA = (LTS, h_s)$ *with* $LTS = (\mathcal{S}, \mathcal{A}, \longrightarrow)$. *Define* $\mathsf{sa2ta}(SA) \stackrel{\text{def}}{=} (LTS', h_t)$ *with* $LTS' = (\mathcal{S}', \mathcal{A}, \longrightarrow')$ *where*

1. $\mathcal{S}' \stackrel{\text{def}}{=} \mathcal{S} \times \Im$,
2. $(s, \mathcal{I}) \xrightarrow{a}' (s', \mathcal{I}') \stackrel{\text{def}}{\iff} s \xrightarrow{a} s' \wedge \mathcal{I}, \mathcal{I}' \in \Im$, *and*
3. $(a, \phi_g, \phi_d, C_r) \in h_t((s, \mathcal{I}) \xrightarrow{a} (s', \mathcal{I}'))$ *(i.e.,* $(s, \mathcal{I}) \xrightarrow{a, \phi_g, \phi_d, C_r} (s', \mathcal{I}')$ *) if*
   (a) $s \xrightarrow{a, C_t, C_r} s'$
   (b) *if* $x \in C_r$, $\mathcal{I}'(x) \in \mathcal{I}_x$; *if* $x \in (C_t - C_r)$, $\mathcal{I}'(x) = \bot$; *otherwise* $\mathcal{I}'(x) = \mathcal{I}(x)$
   (c) $\phi_g = \bigwedge_{x \in C_t \wedge \mathcal{I}(x) \neq \bot} x \,\Box_{\mathcal{I}(x)}\, glb(\mathcal{I}(x))$
   (d) $\phi_d = \bigwedge_{x \in C_t \wedge \mathcal{I}(x) \neq \bot} x \boxtimes_{\mathcal{I}(x)} lub(\mathcal{I}(x))$

where $\Box_{\mathcal{I}(x)}$ *is* $\geq$ *if* $glb(\mathcal{I}(x)) \in \mathcal{I}(x)$ *and* $>$ *otherwise, and* $\boxtimes_{\mathcal{I}(x)}$ *is* $>$ *if* $lub(\mathcal{I}(x)) \in \mathcal{I}(x)$ *and* $\geq$ *otherwise.*

Fig. 3 depicts the translation of the switch (see Example 1). Labels on the location give the original location name together with function $\mathcal{I}$. For example $(s_0, (0, \infty), \bot)$ represents the location $(s_0, \mathcal{I})$ where $\mathcal{I}(x) = (0, \infty)$ and $\mathcal{I}(y) = \bot$. $(s_0, \bot, \bot)$ may be considered the initial state, since the light is originally off and none has been schedule to arrive. Locations and edges in gray are hence unreachable.

In the following, adequacy theorems are given. They state that the translation is correct in the sense that every trace of the original SA is also a trace

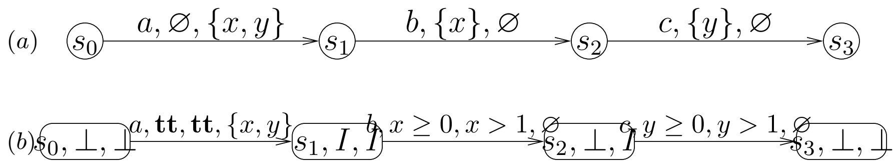


Fig. 4. SA_{ex2} and $\mathsf{sa2ta}(SA_{ex2})$ $(I = [0, 1])$

of its translation and vice-versa. The next theorem states the first inclusion: $LTS(\mathsf{sa2ta}(SA))$ simulates the abstract semantics $LTS(SA)$, and hence its traces are included in $LTS(SA)$.

Theorem 3. *If $\langle s, v, e\rangle$ and $\langle (s, \mathcal{I}), u\rangle$ are states of $LTS(SA)$ and $LTS(\mathsf{sa2ta}(SA))$ respectively, then $\langle s, v, e\rangle \lesssim \langle (s, \mathcal{I}), u\rangle$ if for every clock x, (1) $\mathcal{I}(x) \neq \bot$ implies $e(x) \in \mathcal{I}(x)$ and $v(x) = u(x)$, and (2) $\mathcal{I}(x) = \bot$ implies $v(x) \geq e(x)$.*

Proof. Define R to be the least relation containing all pairs $(\langle s, v, e\rangle, \langle (s, \mathcal{I}), u\rangle)$ that satisfy conditions (1) and (2) above. It is routine to prove that R is a simulation relation. □

The converse does not preserve simulation. Consider the automaton SA_{ex2} and its translation $\mathsf{sa2ta}(SA_{ex2})$ given in Fig. 4(a) and (b), respectively. Suppose $udom(F_x) = udom(F_y) = [0, 1]$. To be illustrative, assume time advances in discrete units of $\frac{1}{2}$ size. Their semantics can then be depicted as in Fig. 5(a) and (b), respectively[5].

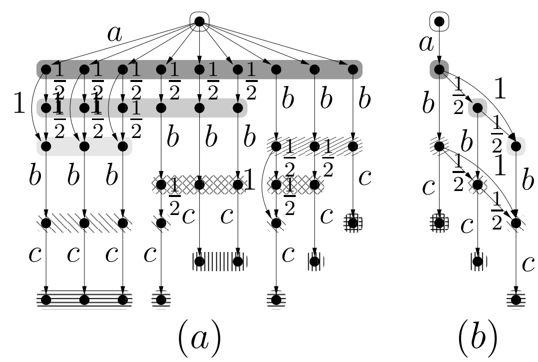


Fig. 5. The LTSs of the automata in Fig. 4

$LTS(SA_{ex2})$ does not simulate $LTS(\mathsf{sa2ta}(SA_{ex2}))$ as it is shown by the following scenario. Let $LTS(\mathsf{sa2ta}(SA_{ex2}))$ perform action a. $LTS(SA_{ex2})$ must then choose one of its many a branches; suppose it chooses the leftmost transition. $LTS(\mathsf{sa2ta}(SA_{ex2}))$ may choose now to perform the b-transition without waiting: $LTS(SA_{ex2})$ cannot simulate this step. This situation is due to the fact that the termination time of the clocks in $LTS(SA)$ is chosen at the moment they are started, while in $\mathsf{sa2ta}(SA)$ the termination time is only decided according to the guard and deadline of an edge, after the clock was started. Nevertheless, the translation does preserve trace inclusion, that is, traces of $LTS(\mathsf{sa2ta}(SA))$ are included in $LTS(SA)$. Forward-backward

[5] We insist: Fig. 5(a) and (b) are intended to be illustrative and *by no means* the actual LTS which should contain an uncountably large number of states and transitions.

simulation [18] is proven to coincide with trace inclusion. States equally shaded in Fig. 5(a) are, in fact, related by a forward-backward simulation. Details of the definition of forward-backward simulation are omitted as well as the proof of the next theorem which can be found in [9].

Theorem 4. *If $\langle(s,\mathcal{I}),v\rangle$ is a state of $LTS(\mathsf{sa2ta}(SA))$ and $\langle s,v,e\rangle$ is a state of $LTS(SA)$ such that, for every clock x, $\mathcal{I}(x) = \bot$ and $e(x) \leq v(x)$, then $tr(\langle(s,\mathcal{I}),v\rangle) \subseteq tr(\langle s,v,e\rangle)$.*

The extra requirement that for every clock x, $\mathcal{I}(x) = \bot$ and $e(x) \leq v(x)$, impose a small but insignificant restriction. It says that traces are preserved only from initial states: an initial state in SA has no active clocks, i.e. $e(x) \leq v(x)$ for all x. In consequence, the translated state must not have active clocks as well, i.e. $\mathcal{I}(x) = \bot$ for all x.

Theorems 1, 2, 3 and 4 state adequacy of the translation of a SA in a TA. Theorems 2 and 3 together say that likely traces in SA are present in $\mathsf{sa2ta}(SA)$. Therefore safety properties of SA are preserved by the translation. In particular, a location that is reachable in $\mathsf{sa2ta}(SA)$ is not likely to be reachable in SA. Theorems 1 and 4 state that $\mathsf{sa2ta}(SA)$ only produce likely traces in SA. Hence, a location that is reachable en $\mathsf{sa2ta}(SA)$ is likely to be reachable in SA. It should be observed that the translation preserve linear properties with *significant measures*. For instance, a property that requires that a particular state (i.e. location *and* valuations) is reachable in SA may have measure 0 in the context of continuous distributions, and yet be reachable in $\mathsf{sa2ta}(SA)$.

Finally, a note about the explosion introduced by the translation is in order. If $\mathcal{S}$ is the set of locations of SA, $\mathcal{S} \times \Im$ is the set of locations of $\mathsf{sa2ta}(SA)$. Therefore, the number of states in $\mathsf{sa2ta}(SA)$ is bounded by $|\mathcal{S}| \cdot \prod_{x\in\mathcal{C}}(|\mathcal{I}_x|+1)$. As a consequence, the translation induces a blow up in the number of locations that is exponential in the number of clocks. Moreover, notice that not every finite SA can be translated into a finite TA. If SA contains a clock whose useful domain is defined as an infinite set of intervals (as e.g. in a geometric distribution), then $\mathsf{sa2ta}(SA)$ is infinite.

6 Compositionality of the Translation

sa2ta commutes with respect to parallel composition. That is, the translation of the parallel composition of two stochastic automata is equivalent to the parallel composition of the timed automata resulting from the translation of each stochastic automaton. This is stated in the rest of this section.

Timed automata with deadlines allow for different definitions of parallel composition [5]. Some definitions are better suited for the modelling of *hard real time* systems, in which components cannot wait for synchronisation, and others oriented to *soft real time*, in which synchronisation can be delayed until all synchronising components are ready. This last type of parallel compositions are the same type of composition used in SA.

Table 1. Rules for Parallel Composition (symmetric rules where omitted)

(a) composition in SA

$$\frac{s_1 \xrightarrow{a,C_t,C_r} s_1' \quad a \notin A}{s_1 \,||_A\, s_2 \xrightarrow{a,C_t,C_r} s_1' \,||_A\, s_2} \qquad \frac{s_1 \xrightarrow{a,C_t^1,C_r^1} s_1' \quad s_2 \xrightarrow{a,C_t^2,C_r^2} s_2' \quad a \in A}{s_1 \,||_A\, s_2 \xrightarrow{a,C_t^1\cup C_t^2,C_r^1\cup C_r^2} s_1' \,||_A\, s_2'}$$

(b) composition in TA

$$\frac{s_1 \xrightarrow{a,\phi_g,\phi_d,C} s_1' \quad a \notin A}{s_1 \,||_A\, s_2 \xrightarrow{a,\phi_g,\phi_d,C} s_1' \,||_A\, s_2} \qquad \frac{s_1 \xrightarrow{a,\phi_g^1,\phi_d^1,C_1} s_1' \quad s_2 \xrightarrow{a,\phi_g^2,\phi_d^2,C_2} s_2' \quad a \in A}{s_1 \,||_A\, s_2 \xrightarrow{a,\phi_g^1\wedge\phi_g^2,\phi_d^1\wedge\phi_d^2,C_1\cup C_2} s_1' \,||_A\, s_2'}$$

Definition 10. *Let* SA_1 *and* SA_2 *be two stochastic automata with the same set of actions* $\mathcal{A}$*. Let* $A \subseteq \mathcal{A}$*. The* parallel composition of SA_1 and SA_2 *is defined by the stochastic automaton* $SA_1 \,||_A\, SA_2$ *with set of locations* $\mathcal{S} = \{s_1 \,||_A\, s_2 \mid s_1 \in \mathcal{S}_1 \wedge s_2 \in \mathcal{S}_2\}$ *and edges defined by rules in Table 1(a).*

For two timed automata TA_1 *and* TA_2 *with set of actions* $\mathcal{A}$*, the* parallel composition of TA_1 and TA_2 *is the timed automaton* $TA_1 \,||_A\, TA_2$ *with set of locations* $\mathcal{S} = \{s_1 \,||_A\, s_2 \mid s_1 \in \mathcal{S}_1 \wedge s_2 \in \mathcal{S}_2\}$ *and edges defined by rules in Table 1(b).*

The criteria to show that $\mathsf{sa2ta}(SA_1 \,||_A\, SA_2)$ and $\mathsf{sa2ta}(SA_1) \,||_A\, \mathsf{sa2ta}(SA_2)$ are equivalent is *structural bisimulation*. Structural bisimulation [8] is a bisimulation relation defined directly on timed automata and it is strictly finer than bisimulation on the underlying semantics of the timed automata.

Definition 11. $R \subseteq \mathcal{S}\times\mathcal{S}$ *is a* structural bisimulation *if it is symmetric and for all* $a \in \mathcal{A}$*,* $\phi_g, \phi_d \in \Phi$*, and* $C \in \mathcal{C}$*, whenever* $\langle s_1, s_2\rangle \in R$*, there exist* $s_2' \in \mathcal{S}$ *such that* $s_1 \xrightarrow{a,\phi_g,\phi_d,C} s_1'$ *implies* $s_2 \xrightarrow{a,\phi_g,\phi_d,C} s_2'$ *and* $\langle s_1', s_2'\rangle \in R$*.* s_1 *and* s_2 *are* structurally bisimilar*, denoted by* $s_1 \sim_s s_2$*, if there is a structural bisimulation* R *with* $\langle s_1, s_2\rangle \in R$*.*

$\sim_s$ is itself a structural bisimulation relation, and if $s_1 \sim_s s_2$ in TA, then $\langle s_1, v\rangle \sim \langle s_2, v\rangle$ in $LTS(TA)$ for every valuation v [8].

The next theorem states the compositional characteristic of $\mathsf{sa2ta}$.

Theorem 5. *Let* SA_1 *and* SA_2 *be stochastic automata with disjoint sets of clocks* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *respectively. Then* $(s_1 \,||_A\, s_2, \mathcal{I}_1 \oplus \mathcal{I}_2) \sim_s (s_1, \mathcal{I}_1) \,||_A\, (s_2, \mathcal{I}_2)$ *where* $(s_1 \,||_A\, s_2, \mathcal{I}_1 \oplus \mathcal{I}_2)$ *is a state of* $\mathsf{sa2ta}(SA_1 \,||_A\, SA_2)$ *with* $(\mathcal{I}_1 \oplus \mathcal{I}_2)(x) \stackrel{\text{def}}{=}$ **if** $x \in \mathcal{C}_1$ **then** $\mathcal{I}_1(x)$ **else** $\mathcal{I}_2(x)$*, and* $(s_1, \mathcal{I}_1) ||_A (s_2, \mathcal{I}_2)$ *is a state of* $\mathsf{sa2ta}(SA_1) ||_A \mathsf{sa2ta}(SA_2)$*.*

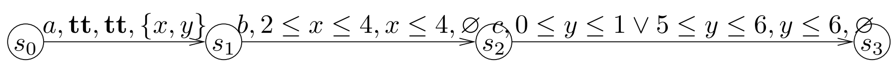
Proof. Let R be the smallest symmetric relation containing all tuples of the form $\langle (s_1 \,||_A\, s_2, \mathcal{I}_1 \oplus \mathcal{I}_2), (s_1, \mathcal{I}_1) \,||_A\, (s_2, \mathcal{I}_2)\rangle$ that satisfies the conditions of the theorem. It is routine to prove that R is a structural bisimulation. □

7 Related Work

To the author's knowledge, there are two works on a similar relation. Bryan & Derrick [7] also presented a translation from stochastic automata to timed automata with deadlines where they claim to preserve timed traces. Unfortunately this is not the case. Bryan & Derrick proposed to preserve the structure of the automata (in particular they do not consider whether clocks active) changing only the edge as follows[6]: $(a, \phi_g, \phi_d, C_r) \in h_t(s \xrightarrow{a} s')$ (i.e., $s \xrightarrow{a,\phi_g,\phi_d,C_r} s'$) if the following holds:

1. $s \xrightarrow{a,C_t,C_r} s'$
2. $\phi_g = \quad \left(\left(\bigwedge_{x\in C_t} x \geq glb(supp(F_x))\right) \wedge \left(\bigvee_{x\in C_t} x \in supp(F_x)\right)\right)$
   $\quad \vee \left(\bigwedge_{x\in C_t} x \geq lub(supp(F_x))\right)$
3. $\phi_d = \bigwedge_{x\in C_t} x \geq lub(supp(F_x))$

Consider SA_{ex2} given in Fig. 4 but take F_x and F_y such that $supp(F_x) = [2, 4]$ and $supp(F_y) = [0, 1] \cup [5, 6]$. The translation proposed by Bryan & Derrick yields the following TA:



It can be verified that $a\,3\,b\,c$ is a trace likely to occur in SA while it is not a trace of the TA. Therefore, this translation is not safe. Besides, Bryan & Derrick do not study compositionality of the translation.

The second work is a translation of IGSMPs in an ad-hoc variation of timed automata called ITA [6]. Bravetti proves that it commutes with parallel composition. However, no adequacy criteria has been provided in this case, which is unfortunate because it would have revealed it does not preserve safely the traces. Since, in this case, the relation to our models are not close, the example that shows it is provided apart in Appendix A.

In any case, both works consider support sets to obtain the guards. Hence, a pathological case similar to the one in Fig. 2 would yield to a translation where undesired executions like (1) are present.

8 Concluding Remarks

We defined a compositional translation from stochastic automata to timed automata. The translation abstracts probabilities and preserves trace behaviour. The simulation of SA by $\mathsf{sa2ta}(SA)$ guarantees that any state that cannot be reached in the translation timed automaton $\mathsf{sa2ta}(SA)$ can neither be reached in the original stochastic automaton SA. In fact, it guarantees the preservation of safety properties. The converse is slightly weaker due to the fact that there is no mean to measure how probable is that a state is reachable in the translation $\mathsf{sa2ta}(SA)$. In this sense it can only be guaranteed that whenever a trace leads

[6] The notation w.r.t. [7] was slightly changed but the definition is exactly the same.

to a reachable state in $\mathsf{sa2ta}(SA)$ then there is a supported execution defining the same trace that leads to a similar state in SA. This *does not* guarantee that the state is reachable with some probability in $PTS(SA)$. However, one can aim for reachability of sets of states and check that its measure is significant (greater than 0). For instance, the reachability of a particular location would usually have significant measure whenever it is reachable. Under this condition it can be concluded that if a location s is reachable in $\mathsf{sa2ta}(SA)$, then it is also reachable in SA.

The translation does introduce an exponential explosion on the size of the automata that, hopefully, would be seldom harmful (many of the newly generated location would turn to be unreachable).

The translation to timed automata may profit of well known and developed techniques for model checking [25, 20, 21, etc.]. Althoug techniques for model checking directly on a stochastic model exists [1, 17] they do have their restrictions. First, both of them require that the support set of all distributions is bounded excluding, thus, important cases like exponential distribution. Beside, they also use region construction, which makes the result not useful in practice. In particular, [17] give an approximation to the measure of the preoperty which make the technique more costly according the error of the approximation decreases.

An further study is to relate stochastic automata to probabilistic timed automata, i.e., timed automata with probabilistic jumps, or a combination of both models.

References

1. R. Alur, C. Courcoubetis, and D. Dill. Model-checking for probabilistic real-time systems. In J. Leach Albert, B. Monien, and M. Rodríguez, editors, *Proceedings of the* 18^{th} *International Colloquium Automata, Languages and Programming (ICALP'91),* Madrid, volume 510 of *Lecture Notes in Computer Science*, pages 113–126. Springer-Verlag, 1991.
2. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. J.C.M. Baeten and S. Mauw, editors. *Proceedings CONCUR 99,* Eindhoven, The Netherlands, volume 1664 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.
4. C. Baier, J.-P. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In Baeten and Mauw [3], pages 146–161.
5. S. Bornot, J. Sifakis, and S. Tripakis. Modeling urgency in timed systems. In Roever W.-P. de, H. Langmaack, and A. Pnueli, editors, *Compositionality: The Significant Difference*, volume 1536 of *Lecture Notes in Computer Science*, pages 103–129. Springer-Verlag, 1998.
6. M. Bravetti. *Specification and Analysis of Stochastic Real-Time Systems.* PhD thesis, Dottorato di Ricerca in Informatica. Università di Bologna, Padova, Venezia, February 2002.
7. J. Bryans and J. Derrick. Stochastic specification and verification. In *Irish Workshop on Formal Methods*, 1999.

8. P.R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, Department of Computer Science, University of Twente, 1999.
9. P.R. D'Argenio. A compositional translation of stochastic automata into timed automata. Technical Report CTIT 00-08, Department of Computer Science, University of Twente, 2000.
10. 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,* Shelter Island, New York, USA, IFIP Series, pages 126–147. Chapman & Hall, 1998.
11. P.R. D'Argenio, J.-P. Katoen, and E. Brinksma. Specification and analysis of soft real-time systems: quantity and quality. In *Proceedings of the 20th IEEE Real-Time Systems Symposium, RTSS'99,* Phoenix, Arizona, USA, pages 104–114. IEEE Computer Society Press, 1999.
12. L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, December 1997.
13. P.W. Glynn. A GSMP formalism for discrete event simulation. *Proceedings of the IEEE*, 77(1):14–23, 1989.
14. T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:193–244, 1994.
15. H. Hermanns. *Interactive Markov Chains – The Quest for Quantified Quality*, volume 2428 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.
16. J. Hillston. *A Compositional Approach to Performance Modelling*. Distinguished Dissertation in Computer Science. Cambridge University Press, 1996.
17. M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Verifying quantitative properties of continuous probabilistic timed automata. In C. Palamidessi, editor, *Proceedings CONCUR 2000,* State College, Pennsylvania, USA, volume 1877 of *Lecture Notes in Computer Science*, pages 123–137. Springer-Verlag, 2000.
18. N.A. Lynch and F.W. Vaandrager. Forward and backward simulations. part I: Untimed systems. *Information and Computation*, 121(2):214–233, September 1995.
19. R. Milner. *Communication and Concurrency*. Prentice-Hall International, 1989.
20. A. Olivero. *Modélisation et Analyse de Systèmes Temporisé et Hybrides*. PhD thesis, Institut National Politechnique de Grenoble, France, September 1994.
21. P. Pettersson. *Modelling and Verification of Real-Time Systems Using Timed Automata: Theory and Practice*. PhD thesis, Department of Computer Systems, Uppsala University, February 1999.
22. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1995.
23. M.Y. Vardi. Automatic verification of probabilistic concurrent finite state programs. In 26^{th} *Annual Symposium on Foundations of Computer Science, Portland, Oregon*, pages 327–338. IEEE Computer Society Press, 1985.
24. W. Yi. Real-time behaviour of asynchronous agents. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, volume 458 of *Lecture Notes in Computer Science*, pages 502–520. Springer-Verlag, 1990.
25. S. Yovine. *Méthodes et outils pour la vérification symbolique de systèmes temporisés*. PhD thesis, Institut National Politechnique de Grenoble, France, May 1993.

A The example on the translation of IGSMP into ITA

In this section, the reader needs to be familiarized with Bravetti's doctoral dissertation. To learn about IGSMP, ITA, and the translation the reader is referred respectively to Chapters 6, 5, and 8 in [6].

Consider the IGSMP in Fig. $6(a)$ and suppose the distributions of clocks C_1 and C_2 have support set in the intervals $[1, 2]$ and $[3, 4]$ respectively. Its translation is depicted in Fig. $6(b)$. In the *reset states* s_0, s_1, and s_2 of the

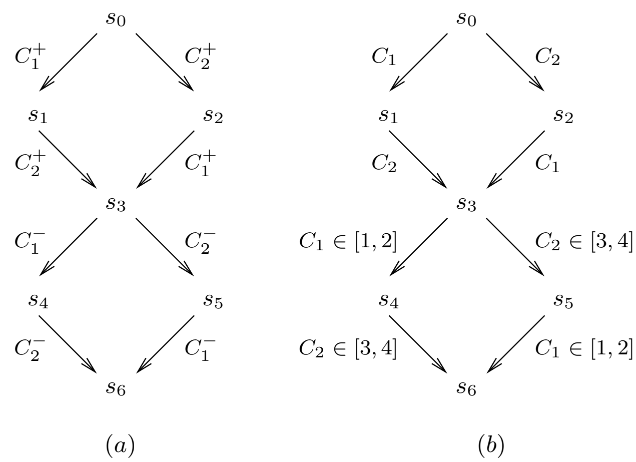


Fig. 6. The example IGSMP and its translation into an ITA

ITA, timed is not allowed to progress. So from s_0 to s_3, clocks C_1 and C_2 are reset without letting time pass. In a *timed state*, like s_3, s_4, or s_5, time progress is defined according to the guards in the outgoing edges. More precisely $(s, v) \xrightarrow{d} (s, v + d)$ if there exists $d' \geq d$ such that $(v + d') \models \bigwedge\{\phi \mid s \xrightarrow{\phi}\}$. At state s_3, the value of C_1 and C_2 can only be 0. Let v be this valuation. Then $C_1 \in [1, 2] \wedge C_2 \in [3, 4]$ does not hold in $(v + d)$ for every d[7]. As a consequence, a simple delaying trace is allowed in the IGSMP but not in the translation ITA.

[7] A possible solution to this problem would be to define $(s, v) \xrightarrow{d} (s, v + d)$ if for all $d' \leq d$, $(v + d') \models \bigwedge\{\overleftarrow{\phi} \mid s \xrightarrow{\phi}\}$, where $\overleftarrow{\cdot}$ is the usual past closure operation (see, e.g., [14]).

[Note added on 29-3-2006] The example above corresponds to ITA semantics reported on an early version of Mario Bravetti's Dissertation (published on February 2002). On a revised version appeared on April 2002, the semantics of ITA was changed in a similar manner to the one suggested in footnote 7. The new rule can be seen in Table 5.1, at page 112 of the last revision of Bravetti's dissertation.