# Verifying Probabilistic Timed Automata Against Timed-Automata Specifications

Anonymous Author(s)

## ABSTRACT

Probabilistic timed automata (PTAs) are timed automata extended with discrete probability distributions. They serve as a mathematical model for a wide range of applications that involve both stochastic and timed behaviours. In this paper, we study the model-checking problem of linear *dense-time* temporal properties over PTAs. In particular, we consider linear dense-time properties that can be encoded by timed automata with both finite and infinite acceptance criteria. We first show that the problem of model-checking PTAs against deterministic-timed-automata specifications with infinite acceptance criterion can be solved through a product construction and is EXPTIME-complete. Then we show that when relaxed to general (nondeterministic) timed automata, the model-checking problem becomes undecidable. Finally, we investigate the situation where the acceptance criterion is restricted to be finite. We show that for deterministic timed automata, the model-checking problem can be solved using known zone-based algorithms for reachability probabilities on PTAs. Furthermore, for nondeterministic timed automata, we show an approximation algorithm for solving the problem whose correctness is based on a translation to infinite-state Markov decision processes.

**ACM Reference Format:**
Anonymous Author(s). 1997. Verifying Probabilistic Timed Automata Against Timed-Automata Specifications. In *Proceedings of 21st ACM International Conference on Hybrid Systems: Computation and Control (HSCC 2018).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.475/123_4

## 1 INTRODUCTION

Stochastic timed systems are systems that exhibit both timed and stochastic behaviours. Such systems play a dominant role in many real-world applications [3], hence addressing fundamental issues such as safety and performance over these systems are important. *Probabilistic timed automata* (PTAs) [5, 20, 24] serve as a good mathematical model for these systems. They extend the well-known model of timed automata [1] (for nonprobabilistic timed systems) with discrete probability distributions, and Markov Decision Processes (MDPs) [25] (for untimed probabilistic systems) with timing constraints.

Formal verification of PTAs has received much attention in recent years [24]. For branching-time model-checking of PTAs, the problem is reduced to computation of reachability probabilities over MDPs through well-known finite abstraction for timed automata (namely *regions* and *zones*) [5, 13, 20]. Advanced techniques for branching-time model checking of PTAs such as inverse method and symbolic method have been further explored in [2, 14, 17, 21]. Extension with *cost* or *reward*, resulting in *priced* PTAs, has also been well investigated. Jurdzinski *et al.* [15] and Kwiatkowska *et al.* [19] proved that several notions of accumulated or discounted cost are computable over priced PTAs, while cost-bounded reachability probability over priced PTAs is shown to be undecidable by Berendsen *et al.* [6]. Most verification algorithms for PTAs have been implemented in the model checker PRISM [18]. Computational complexity of several verification problems for PTAs is studied in [15, 16, 22].

For linear-time model-checking, much less is known. As far as we know, the only relevant result is by ? [**?** ] who proved that the problem of model-checking PTAs against linear *discrete-time* properties encoded by deterministic omega-regular automata can be solved by a product construction. In their paper, ? [**?** ] first devised a production construction that produces a PTA out of the input PTA and the omega-regular automaton; then they proved that the problem can be reduced to omega-regular verification of MDPs through maximal end components.

In this paper, we study the problem of model-checking linear *dense-time* properties over PTAs. Compared with discrete-time properties, dense-time properties take into account timing constraints and therefore is more expressive. We focus on linear dense-time properties that can be encoded by timed automata [1]. Timed automata are normal automata extended with *clocks* and *timing constraints*. Due to the ability to model dense-time behaviours, they can be used to model real-time systems, while they can also act as language recognizers for timed omega-regular languages. Here we treat timed automata as language recognizers for timed paths from a PTA, and study the problem to compute the probability that

*HSCC 2018, April 2018, Porto, Portugal*
2016. ACM ISBN 123-4567-24-567/08/06...$15.00
https://doi.org/10.475/123_4

a timed path from the PTA is accepted by the timed automaton. The intuition is that a timed automaton can recognize the set of "good" (or "bad") timed paths emitting from a PTA, so the problem is to compute the probability that the PTA behaves in a good (or bad) manner.

*Our Contributions.* We distinguish between the subclass of *deterministic* timed automata (DTAs) and general *nondeterministic* timed automata. DTAs are the deterministic version of timed automata. Although DTA is weaker than general timed automata, it can recognize a wide class of formal timed languages, and express interesting linear dense-time properties which cannot be expressed in branching-time logics (cf. [11]). For infinite acceptance criterion, we show that the problem of model-checking PTAs against DTA specifications can be solved through a nontrivial product construction which tackles the integrated feature of timing constraints and randomness. From the product construction, we further show that the problem is EXPTIME-complete. We also prove that the problem becomes undecidable when one considers general nondeterministic timed automata. For finite acceptance criterion, we show that the problem with DTA specifications can be solved by using efficient zone-based algorithms [5, 13, 20] through the same product construction. Moreover, we devised an approximation algorithm for the problem with general timed automata through translation to infinite-state MDPs.

## 2 PRELIMINARIES

We denote by $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{Z}$, and $\mathbb{R}$ the sets of all positive integers, non-negative integers, integers and real numbers, respectively.

For any infinite word $w = b_0 b_1 \ldots$, we denote by $\mathit{inf}(w)$ the set of symbols (i.e., $b_i$'s) that occur infinitely many times in $w$. Given a finite word $w = b_0 \ldots b_n$ $(n \geq 0)$, the last symbol $b_n$ is denoted by $\mathit{last}\,(w)$.

A *clock* is a variable for a nonnegative real number. Below we fix a finite set $\mathcal{X}$ of clocks.

*Clock Valuations.* A *clock valuation* is a function $\nu : \mathcal{X} \to [0, \infty)$. The set of clock valuations is denoted by $\mathit{Val}\,(\mathcal{X})$. Given a clock valuation $\nu$, a subset $X \subseteq \mathcal{X}$ of clocks and a non-negative real number $t$, we let (i) $\nu[X := 0]$ be the clock valuation such that $\nu[X := 0](x) = 0$ for $x \in X$ and $\nu[X := 0](x) = \nu(x)$ otherwise, and (ii) $\nu{+}t$ be the clock valuation such that $(\nu{+}t)(x) = \nu(x) + t$ for all $x \in \mathcal{X}$. We denote by $\mathbf{0}$ the clock valuation such that $\mathbf{0}(x) = 0$ for $x \in \mathcal{X}$.

*Clock Constraints.* The set $CC(\mathcal{X})$ of *clock constraints* over $\mathcal{X}$ is generated by the following grammar:

$$\phi := \textbf{true} \mid x \leq d \mid c \leq x \mid x{+}c \leq y{+}d \mid \neg\phi \mid \phi \wedge \phi$$

where $x, y \in \mathcal{X}$ and $c, d \in \mathbb{N}_0$. We write **false** for a short hand of $\neg$**true**. The satisfaction relation $\models$ between valuations $\nu$ and clock constraints $\phi$ is defined through substituting every $x \in \mathcal{X}$ appearing in $\phi$ by $\nu(x)$ and standard semantics for logical connectives. For a given clock constraint $\phi$, we denote by $[\![\phi]\!]$ the set of all clock valuations that satisfy $\phi$.

*Clock Equivalence.* Consider a nonnegative integer $N$ such that values held by clocks are treated equivalent if they both exceed $N$. With such a threshold, the standard notion of clock equivalence [1] is an equivalence relation $\sim_N$ over $\mathit{Val}\,(\mathcal{X})$ as follows: for any two clock valuations $\nu, \nu'$, $\nu \sim_N \nu'$ iff the following conditions hold:

- for all $x \in \mathcal{X}$, $\nu(x) > N$ iff $\nu'(x) > N$;
- for all $x \in \mathcal{X}$, if $\nu(x) \leq N$ then (i) $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ and (ii) $\mathit{frac}(\nu(x)) > 0$ iff $\mathit{frac}(\nu'(x)) > 0$;
- for all $x, y \in \mathcal{X}$, if $\nu(x), \nu(y) \leq N$ then it holds that $\mathit{frac}(\nu(x)) \bowtie \mathit{frac}(\nu(y))$ iff $\mathit{frac}(\nu'(x)) \bowtie \mathit{frac}(\nu'(y))$ for $\bowtie \in \{<, =, >\}$.

Equivalence classes of $\sim_N$ are called *regions*. The equivalence class that contains a given clock valuation $\nu$ is denoted by $[\nu]_{\sim_N}$. We simply write $[\nu]_\sim$ if $N$ is clear from the context.

### 2.1 Probabilistic Timed Automata

A *discrete probability distribution* over a countable nonempty set $U$ is a function $q : U \to [0, 1]$ such that $\sum_{z \in U} q(z) = 1$. The *support* of $q$ is defined as $\mathit{supp}(q) := \{z \in U \mid q(z) > 0\}$. We denote the set of discrete probability distributions over $U$ by $\mathcal{D}(U)$.

*Definition 2.1 (Probabilistic Timed Automata [24]).* A *probabilistic timed automaton* (PTA) $\mathcal{C}$ is a tuple

$$\mathcal{C} = (L, \ell^*, \mathcal{X}, \mathit{Act}, \mathit{inv}, \mathit{enab}, \mathit{prob}, \mathit{AP}, \mathcal{L}) \qquad (1)$$

where:

- $L$ is a finite set of *locations*;
- $\ell^* \in L$ is the *initial* location;
- $\mathcal{X}$ is a finite set of *clocks*;
- $\mathit{Act}$ is a finite set of *actions*;
- $\mathit{inv} : L \to CC(\mathcal{X})$ is an *invariant condition*;
- $\mathit{enab} : L \times \mathit{Act} \to CC(\mathcal{X})$ is an *enabling condition*;
- $\mathit{prob} : L \times \mathit{Act} \to \mathcal{D}\big(2^{\mathcal{X}} \times L\big)$ is a *probabilistic transition function*;
- $\mathit{AP}$ is a finite set of *atomic propositions*;
- $\mathcal{L} : L \to 2^{\mathit{AP}}$ is a *labelling function*.

W.l.o.g, we consider that both $Act$ and $AP$ is disjoint from $[0,\infty)$. Below we fix a PTA $\mathcal{C}$ in the form (1). The semantics of PTAs is as follows.

*States and Transition Relation.* A *state* of $\mathcal{C}$ is a pair $(\ell,\nu)$ in $L \times Val(\mathcal{X})$ such that $\nu \models inv(\ell)$. The set of all states is denoted by $S_{\mathcal{C}}$. The *transition relation* $\rightarrow$ consists of all triples $((\ell,\nu),a,(\ell',\nu'))$ satisfying the following conditions:

- $(\ell,\nu),(\ell',\nu')$ are states and $a \in Act \cup [0,\infty)$;
- if $a \in [0,\infty)$ then $\nu+\tau \models inv(\ell)$ for all $\tau \in [0,a]$ and $(\ell',\nu') = (\ell,\nu+a)$;
- if $a \in Act$ then $\nu \models enab(\ell,a)$ and there exists a pair $(X,\ell'') \in supp(prob(\ell,a))$ such that $(\ell',\nu') = (\ell'',\nu[X:=0])$.

By convention, we write $s \xrightarrow{a} s'$ instead of $(s,a,s') \in \rightarrow$. We omit '$\mathcal{C}$' in '$S_{\mathcal{C}}$' if the underlying context is clear.

*Probability Transition Kernel.* The *probability transition kernel* $\mathbf{P}$ is the function $\mathbf{P} : S \times Act \times S \rightarrow [0,1]$ such that

$$\mathbf{P}((\ell,\nu),a,(\ell',\nu')) = \begin{cases} 1 & \text{if } (\ell,\nu) \xrightarrow{a} (\ell',\nu') \text{ and } a \in [0,\infty) \\ \sum_{Y \in B} prob(\ell,a)(Y,\ell') & \text{if } (\ell,\nu) \xrightarrow{a} (\ell',\nu') \text{ and } a \in Act \\ 0 & \text{otherwise} \end{cases}$$

where $B := \{X \subseteq \mathcal{X} \mid \nu' = \nu[X:=0]\}$.

*Well-formedness.* We say that $\mathcal{C}$ is *well-formed* if for every state $(\ell,\nu)$ and action $a \in Act$ such that $\nu \models enab(\ell,a)$ and every $(X,\ell') \in supp(prob(\ell,a))$, one has that $\nu[X:=0] \models inv(\ell')$. The well-formedness is to ensure that when an action is enabled, the next state after taking this action will always be legal. In the following, we always assume that the underlying PTA is well-formed. Non-well-formed PTAs can be repaired into well-formed PTAs [21].

*Paths.* A *finite path* $\rho$ (under $\mathcal{C}$) is a finite sequence

$$\langle s_0, a_0, s_1, \ldots, a_{n-1}, s_n \rangle \ \ (n \geq 0)$$

in $S \times ((Act \cup [0,\infty)) \times S)^*$ such that (i) $s_0 = (\ell^*, \mathbf{0})$, (ii) $a_{2k} \in [0,\infty)$ (resp. $a_{2k+1} \in Act$) for all integers $0 \leq k \leq \frac{n}{2}$ (resp. $0 \leq k \leq \frac{n-1}{2}$) and (iii) for all $0 \leq k \leq n-1$, $s_k \xrightarrow{a_k} s_{k+1}$. The length $|\rho|$ of $\rho$ is defined by $|\rho| := n$. An *infinite path* (under $\mathcal{C}$) is an infinite sequence

$$\langle s_0, a_0, s_1, a_1, \ldots \rangle$$

in $(S \times (Act \cup [0,\infty)))^\omega$ such that for all $n \in \mathbb{N}_0$, the prefix $\langle s_0, a_0, \ldots, a_{n-1}, s_n \rangle$ is a finite path. The set of finite (resp. infinite) paths under $\mathcal{C}$ is denoted by $Paths^*_{\mathcal{C}}$ (resp. $Paths^\omega_{\mathcal{C}}$).

*Schedulers.* A *scheduler* is a function $\sigma$ from the set of finite paths into $Act \cup [0,\infty)$ such that for all finite paths $\rho = s_0 a_0 \ldots s_n$, (i) $\sigma(\rho) \in Act$ (resp. $\sigma(\rho) \in [0,\infty)$) if $n$ is odd (resp. even) and (ii) there exists a state $s'$ such that $s_n \xrightarrow{\sigma(\rho)} s'$.

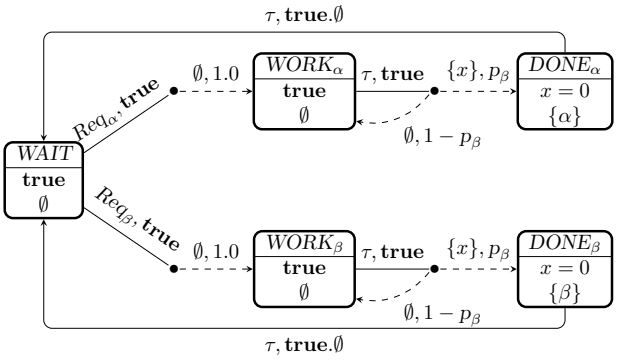


**Figure 1: A Task-Handling Example**

*Paths under Schedulers.* A finite path $s_0 a_0 \ldots s_n$ *follows* a scheduler $\sigma$ if for all $0 \leq m < n$, $a_m = \sigma(s_0 a_0 \ldots s_m)$. An infinite path $s_0 a_0 s_1 a_1 \ldots$ *follows* $\sigma$ if for all $n \in \mathbb{N}_0$, $a_n = \sigma(s_0 a_0 \ldots s_n)$. The set of finite (resp. infinite) paths following a scheduler $\sigma$ is denoted by $Paths^*_{\mathcal{C},\sigma}$ (resp. $Paths^\omega_{\mathcal{C},\sigma}$). We note that the set $Paths^*_{\mathcal{C},\sigma}$ is countably infinite from definition.

*Probability Spaces under Schedulers.* Let $\sigma$ be any scheduler. The probability space w.r.t $\sigma$ is defined as

$$(\Omega^{\mathcal{C},\sigma}, \mathcal{F}^{\mathcal{C},\sigma}, \mathbb{P}^{\mathcal{C},\sigma})$$

where (i) $\Omega^{\mathcal{C},\sigma} := Paths^\omega_{\mathcal{C},\sigma}$, (ii) $\mathcal{F}^{\mathcal{C},\sigma}$ is the smallest sigma-algebra generated by all cylinder sets induced by finite paths for which a finite path $\rho$ induces the cylinder set $Cyl(\rho)$ of all infinite paths in $Paths^\omega_{\mathcal{C},\sigma}$ with $\rho$ being their (common) prefix, and (iii) $\mathbb{P}^{\mathcal{C},\sigma}$ is the unique probability measure such that for all finite paths $\rho = s_0 a_0 \ldots a_{n-1} s_n$ in $Paths^*_{\mathcal{C},\sigma}$,

$$\mathbb{P}^{\mathcal{C},\sigma}(Cyl(\rho)) = \prod_{k=0}^{n-1} \mathbf{P}(s_k, \sigma(s_0 a_0 \ldots a_{k-1} s_k), s_{k+1}).$$

For details see [20].

*Zenoness and Time-Divergent Schedulers.* An infinite path $\pi = s_0 a_0 s_1 a_1 \ldots$ is *zeno* if $\sum_{n=0}^{\infty} d_n < \infty$, where $d_n := a_n$ if $a_n \in [0,\infty)$ and $d_n := 0$ otherwise. Then a scheduler $\sigma$ is *time divergent* if $\mathbb{P}^{\mathcal{C},\sigma}(\{\pi \mid \pi \text{ is zeno}\}) = 0$. In the following, we only consider time-divergent schedulers. The purpose is to eliminate non-realistic zeno behaviours (i.e., performing infinitely many actions within a finite amount of time).

In the following example, we illustrate a PTA which models a simple task-handling process.

*Example 2.2.* In the PTA depicted in Figure 1, $WAIT$, $WORK_s$ and $DONE_s$ ($s \in \{\alpha, \beta\}$) are locations and $x$ is the only clock. Below each location first comes (vertically) its invariant condition and then the set of labels assigned to the location. For example, $inv(DONE_\alpha) = (x = 2)$ and $\mathcal{L}(DONE_\alpha) = \{\alpha\}$. The four dot points together with corresponding arrows refer to four actions and their enabling conditions and probability transition functions. For example, the upper dot at the right of $WORK_\alpha$ refers to an action whose name is $\tau$, the enabling condition for $\tau$ (from $WORK_\alpha$) is **true** (cf. the solid line emitting from $WORK_\alpha$), and the probability distribution for this action is to reset $x$ and go to $DONE_\alpha$ with probability $p_\alpha$ and to reset $x$ and go back to $DONE_\alpha$ with probability $1 - p_\alpha$. The PTA models a machine whiche deals with two different kinds of jobs.

## 2.2 Timed Automata

*Definition 2.3 (Timed Automata [10–12]).* A *timed automaton* (TA) $\mathcal{A}$ is a tuple

$$\mathcal{A} = (Q, \Sigma, \mathcal{Y}, \Delta) \tag{2}$$

where

- $Q$ is a finite set of *modes*;
- $\Sigma$ is a finite *alphabet* of *symbols* disjoint from $[0, \infty)$;
- $\mathcal{X}$ is a finite set of *clocks*;
- $\Delta \subseteq Q \times \Sigma \times CC(\mathcal{Y}) \times 2^{\mathcal{Y}} \times Q$ is a finite set of *rules*.

$\mathcal{A}$ is a *deterministic* TA (DTA) if the following holds:

(1) (*determinism*) for $(q_i, b_i, \phi_i, X_i, q_i') \in \Delta$ ($i \in \{1, 2\}$), if $(q_1, b_1) = (q_2, b_2)$ and $[\![\phi_1]\!] \cap [\![\phi_2]\!] \neq \emptyset$ then $(\phi_1, X_1, q_1') = (\phi_2, X_2, q_2')$;

(2) (*totality*) for all $(q, b) \in Q \times \Sigma$ and $\nu \in Val(\mathcal{X})$, there exists $(q, b, \phi, X, q') \in \Delta$ such that $\nu \models \phi$.

Below we illustrate the semantics of TAs. We fix a TA $\mathcal{A}$ in the form (2).

*Configurations and One-Step Transition Relation.* A *configuration* is a pair $(q, \nu)$, where $q \in Q$ and $\nu \in Val(\mathcal{Y})$. The *one-step transition relation*

$$\Rightarrow \subseteq (Q \times Val(\mathcal{Y})) \times (\Sigma \cup [0, \infty)) \times (Q \times Val(\mathcal{Y}))$$

is defined by: $((q, \nu), a, (q', \nu')) \in \Rightarrow$ iff either (i) $a \in [0, \infty)$ and $(q', \nu') = (q, \nu + a)$ or (ii) $a \in \Sigma$ and there exists a rule $(q, a, \phi, X, q') \in \Delta$ such that $\nu \models \phi$ and $\nu' = \nu[X := 0]$. For the sake of convenience, we write $(q, \nu) \overset{a}{\Rightarrow} (q', \nu')$ instead of $((q, \nu), a, (q', \nu')) \in \Rightarrow$. Note that if $\mathcal{A}$ is deterministic, then there is a unique $(q', \nu')$ such that $(q, \nu) \overset{a}{\Rightarrow} (q', \nu')$ given any $(q, \nu), a$.

*Infinite Time Words and Runs.* An *infinite time word* is an infinite sequence $w = \{a_n\}_{n \in \mathbb{N}_0}$ such that $a_{2n} \in [0, \infty)$ and $a_{2n+1} \in \Sigma$ for all $n$; the infinite timed word $w$ is *time-divergent* if $\sum_{n \in \mathbb{N}_0} a_{2n} = \infty$. A *run* of $\mathcal{A}$ on an infinite timed word $w = \{a_n\}_{n \in \mathbb{N}_0}$ with *initial configuration* $(q, \nu)$, is an infinite sequence $\xi = \{(q_n, \nu_n, a_n)\}_{n \in \mathbb{N}_0}$ satisfying that $(q_0, \nu_0) = (q, \nu)$ and $(q_n, \nu_n) \overset{a_n}{\Rightarrow} (q_{n+1}, \nu_{n+1})$ for all $n \in \mathbb{N}_0$; the *trajectory* $traj(\xi)$ of the run $\xi$ is defined as an infinite word over $Q$ such that $traj(\xi) := q_0 q_1 \dots$ . Note that if $\mathcal{A}$ is deterministic, then there is a unique run on every infinite timed word.

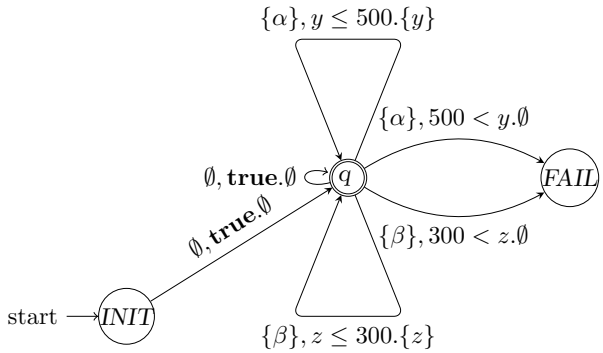


**Figure 2: A DTA Specification**

Below we illustrate the acceptance conditions for TAs. We first illustrate infinite acceptance condition for which we consider Rabin acceptance condition.

*Definition 2.4 (Rabin Acceptance Condition).* A TA with *Rabin acceptance condition* (TRA) is a tuple

$$\mathcal{A} = (Q, \Sigma, \mathcal{Y}, \Delta, \mathcal{F}) \tag{3}$$

where $(Q, \Sigma, \mathcal{Y}, \Delta)$ is a TA and $\mathcal{F}$ is a finite set of pairs $\mathcal{F} = \{(H_1, K_1), \dots, (H_n, K_n)\}$ representing a Rabin condition for which $H_i$ and $K_i$ are subsets of $Q$ for all $i < n$. $\mathcal{A}$ is a deterministic TRA (DTRA) if $(Q, \Sigma, \mathcal{Y}, \Delta)$ is a DTA. A set $Q' \subseteq Q$ is *Rabin-accepting* by $\mathcal{F}$, written as the predicate $\mathbf{ACC}(Q', \mathcal{F})$, if there is $1 \leq i \leq n$ such that $Q' \cap H_i = \emptyset$ and $Q' \cap K_i \neq \emptyset$. An infinite timed word $w$ is *Rabin-accepted* by $\mathcal{A}$ with *initial configuration* $(q, \nu)$ iff there exists a run $\xi$ of $(Q, \Sigma, \mathcal{Y}, \Delta)$ on $w$ with $(q, \nu)$ such that $inf(\xi)$ is Rabin-accepting by $\mathcal{F}$.

*Example 2.5.* Consider the DTA depicted in Figure 2 which works as a specification for the PTA in Example 2.2. $INIT$, $q$ and $FAIL$ are modes with $\mathcal{F} = \{\{FAIL\}, \{q\}\}$, $y, z$ are clocks and arrows between modes are rules. For example, there are five rules emitting from $q$ , one is $(q, \{\beta\}, 300 < z, \emptyset, FAIL)$ and another is $(q, \emptyset, \mathbf{true}, \emptyset, q)$. $INIT$ is the initial mode to read the label of the initial location of a PTA in the product construction, and $FAIL$ is a trap mode. Note that this DTA does not satisfy the totality condition. However, this can be remedied by

adding rules leading to a deadlock mode without changing the acceptance behaviour of the DTA. This DTA specified the property that every $\alpha$ job should be done within 500 units of time after last $\alpha$ job done and every $\beta$ job should be done within 300 units of time after last $\beta$ job done.

*Definition 2.6 (Finite Acceptance Condition).* A TA with *finite acceptance condition* (TFA) is a tuple

$$\mathcal{A} = (Q, \Sigma, \mathcal{Y}, \Delta, F) \tag{4}$$

where $(Q, \Sigma, \mathcal{Y}, \Delta)$ is a TA and $F$ is a subset of $Q$ representing the set of *final* modes. $\mathcal{A}$ is a deterministic TFA (DTFA) if $(Q, \Sigma, \mathcal{Y}, \Delta)$ is a DTA. An infinite timed word $w$ is *finitely-accepted* by $\mathcal{A}$ with *initial configuration* $(q, \nu)$ if there exists a run $\xi$ of $(Q, \Sigma, \mathcal{Y}, \Delta)$ on $w$ with $(q, \nu)$ such that some final mode in $F$ appear in the infinite word $inf(\xi)$.

Remark 1. *Finite acceptance condition is a special case of Rabin acceptance condition. Given a TFA in the form (4), one can transform it into an equivalent TRA by first removing all rules emitting from final modes, then making all final modes "absorbing" through adding self-loop rules $(q, b, \mathbf{true}, \emptyset, q)$ for all final modes $q$ and symbols $b$, and finally setting the Rabin condition $\mathcal{F}$ to be the singleton set $\{(\emptyset, F)\}$.*

## 3 PROBLEM STATEMENT

In this part, we define the PTA-TA problem of model-checking PTAs against TA-specifications. The problem takes a PTA and a TRA/TFA as input, and computes the probability that infinite paths under the PTA are accepted by the TRA/TFA. Informally, the TRA/TFA encodes the linear dense-time property by judging whether an infinite path is accepted or not through its external behaviour, then the problem is to compute the probability that an infinite path is accepted by the TRA/TFA. In practice, the TRA/TFA can be used to capture all good (or bad) behaviours, so the problem can be treated as a task to evaluate to what extent the PTA behaves in a good (or bad) way.

Below we fix a well-formed PTA $\mathcal{C}$ taking the form (1) and a TRA (or TFA) $\mathcal{A}$ taking the form (3) (or (4))) . W.l.o.g., we assume that $\mathcal{X} \cap \mathcal{Y} = \emptyset$ and $\Sigma = 2^{AP}$. We first show how an infinite path in $Paths^{\omega}_{\mathcal{C}}$ can be interpreted as an infinite timed word.

*Definition 3.1 (Infinite Paths as Infinite Timed Words).* Given an infinite path

$$\pi = (\ell_0, \nu_0)a_0(\ell_1, \nu_1)a_1(\ell_2, \nu_2)a_2 \dots$$

under $\mathcal{C}$, the infinite timed word $\mathcal{L}(\pi)$ is defined as

$$\mathcal{L}(\pi) := a_0\mathcal{L}(\ell_2)a_2\mathcal{L}(\ell_4)\dots a_{2n}\mathcal{L}(\ell_{2n+2})\dots \quad .$$

Recall that $\nu_0 = \mathbf{0}$, $a_{2n} \in [0, \infty)$ and $a_{2n+1} \in Act$ for $n \in \mathbb{N}_0$.

Remark 2. *Informally, the interpretation in Definition 3.1 works by (i) dropping (a) the initial location $\ell_0$, (b) all clock valuations $\nu_n$'s, (c) all locations $\ell_{2n+1}$'s following a time-elapse, (d) all internal actions $a_{2n+1}$'s of $\mathcal{C}$ and (ii) replacing every $\ell_{2n}$ $(n \geq 1)$ by $\mathcal{L}(\ell_{2n})$. The interpretation captures only external behaviours including time-elapses and labels of locations upon state-change, and discards internal behaviours such as the concrete locations, clock valuations and actions. Although the interpretation ignores the initial location, we deal with it in our acceptance condition where the initial location is preprocessed by the TRA/TFA.*

*Definition 3.2 (Path Acceptance).* An infinite path $\pi$ under $\mathcal{C}$ is *accepted* by $\mathcal{A}$ w.r.t initial configuration $(q, \nu)$, written as the single predicate $\mathbf{ACC}(\mathcal{A}, (q, \nu), \pi)$, if there exists a mode $q \in Q$ such that $(q, \nu) \xrightarrow{\mathcal{L}(\ell^*)} (q', \nu')$ and the infinite word $\mathcal{L}(\pi)$ is Rabin- (or finitely-) accepted by $\mathcal{A}$ w.r.t $((q', \nu'), \mathbf{0})$.

The initial location omitted in Definition 3.1 is preprocessed by specifying explicitly that the first label $\mathcal{L}(\ell^*)$ is read by the initial configuration $(q, \nu)$. Below we define acceptance probabilities over infinite paths under $\mathcal{C}$.

*Definition 3.3 (Acceptance Probabilities).* The probability that $\mathcal{C}$ *observes* $\mathcal{A}$ under scheduler $\sigma$ and initial mode $q \in Q$, denoted by $\mathfrak{p}^{\sigma}_{q}$, is defined by:

$$\mathfrak{p}^{\sigma}_{q} := \mathbb{P}^{\mathcal{C},\sigma}\left(AccPaths^{\mathcal{A},q}_{\mathcal{C},\sigma}\right)$$

where $AccPaths^{\mathcal{A},q}_{\mathcal{C},\sigma}$ is the set of infinite paths under $\mathcal{C}$ that are accepted by the TRA (or TFA) $\mathcal{A}$

$$AccPaths^{\mathcal{A},q}_{\mathcal{C},\sigma} = \left\{\pi \in Paths^{\omega}_{\mathcal{C},\sigma} \mid \mathbf{ACC}(\mathcal{A}, (q, \mathbf{0}), \pi)\right\}.$$

Since the set $Paths^{*}_{\mathcal{C},\sigma}$ is countably-infinite, $AccPaths^{\mathcal{A},q}_{\mathcal{C},\sigma}$ is measurable since it can be represented as a countable intersection of certain countable unions of some cylinder sets (cf. [3, Chap. ?] for details).

Now we introduce the PTA-TA problem.

- **Input:** a well-formed PTA $\mathcal{C}$, a TRA (or TFA) $\mathcal{A}$ and an initial mode $q$ in $\mathcal{A}$;
- **Output:** $\inf_{\sigma} \mathfrak{p}^{\sigma}_{q}$ and $\sup_{\sigma} \mathfrak{p}^{\sigma}_{q}$, where $\sigma$ ranges over all time-divergent schedulers.

We refer to the problem as PTA-DTA if $\mathcal{A}$ is deterministic.

## 4 THE PTA-DTA PROBLEM

In this section, we solve the PTA-DTA problem through a product construction. Based on the product construction, we also settle the complexity of the problem. Below

we fix a well-formed PTA $\mathcal{C}$ in the form (1) and a DTRA (or resp. DTFA) $\mathcal{A}$ in the form (3) (or resp. (4)). W.l.o.g, we consider that $\mathcal{X} \cap \mathcal{Y} = \emptyset$ and $\Sigma = 2^{AP}$.

**The Main Idea.** The core part of the product construction is a PTA which preserves the probability of the set of infinite paths accepted by $\mathcal{A}$. The intuition is to let $\mathcal{A}$ reads external actions of $\mathcal{C}$ while $\mathcal{C}$ evolves along the time axis. The major difficulty is that when $\mathcal{C}$ performs actions in $Act$, there is a probabilistic choice between the target locations. Then $\mathcal{A}$ needs to know the labelling of the target location and the rule (in $\Delta$) used for the transition. A naive solution is to integrate each single rule in $\Delta$ into the enabling condition *enab* in $\mathcal{C}$. However, this simple solution does not work since a single rule fixes the labelling of a location in $\mathcal{C}$, while the probability distribution (given by *prob*) can jump to locations with different labels. We solve this difficulty by integrating into the enabling condition enough information on clock valuations under $\mathcal{A}$ so that the rule used for the transition is clear.

**The Product Construction.** For each $q \in Q$, we let

$$\mathcal{T}_q := \{h : \Sigma \to CC(\mathcal{Y}) \mid \forall b \in \Sigma.\, (q, b, h(b), X, q') \in \Delta \text{ for some } X, q')\}\ .$$

The totality of $\Delta$ ensures that $\mathcal{T}_q$ is non-empty. Intuitively, every element of $\mathcal{T}_q$ is a tuple of clock constraints $\{\phi_b\}_{b\in\Sigma}$, where each clock constraint $\phi_b$ is chosen from the rules emitting from $q$ and $b$. The *product PTA* $\mathcal{C}\otimes\mathcal{A}_q$ (between $\mathcal{C}$ and $\mathcal{A}$ with initial mode $q$) is defined as

$$\left(L_\otimes, \ell^*_\otimes, \mathcal{X}_\otimes, Act_\otimes, inv_\otimes, enab_\otimes, prob_\otimes, \mathcal{L}_\otimes\right)$$

where:

- $L_\otimes := L \times Q$;
- $\ell^*_\otimes := (\ell^*, q^\star)$ where $q^\star$ is the unique mode such that $(q, \mathbf{0}) \xrightarrow{\mathcal{L}(\ell^*)} (q^\star, \mathbf{0})$;
- $\mathcal{X}_\otimes := \mathcal{X} \cup \mathcal{Y}$;
- $Act_\otimes := Act \times \bigcup_q \mathcal{T}_q$;
- $inv_\otimes(\ell, q) := inv(\ell)$ for all $(\ell, q) \in L_\otimes$;
- $enab_\otimes((\ell, q), (a, h)) := enab(\ell, a) \wedge \bigwedge_{b\in\Sigma} h(b)$ for all $(\ell, q) \in L_\otimes$ and $h \in \mathcal{T}_q$, and $enab_\otimes((\ell, q), (a, h)) :=$ **false** otherwise;
- $\mathcal{L}_\otimes(\ell, q) := \{q\}$ for all $(\ell, q) \in L_\otimes$;
- $prob_\otimes$ is given by

$$prob_\otimes((\ell, q), (a, h))(Y, (\ell', q')) := \begin{cases} prob(\ell, a)(Y \cap \mathcal{X}, \ell') & \text{if } (q, \mathcal{L}(\ell'), h(\mathcal{L}(\ell')), Y \cap \mathcal{Y}, q') \\ & \quad \text{is a (unique) rule in } \Delta \\ 0 & \text{otherwise} \end{cases}.$$

Besides standard constructions (e.g., the Cartesian product between $L$ and $Q$), the product construction also has Cartesian product between $Act$ and $\bigcup_q \mathcal{T}_q$. For each extended action $(a, h)$, the enabling condition for this action is the conjunction between $enab(\ell, a)$ and all clock constraints from $h$. This is to ensure that when the action $(a, h)$ is taken, the clock valuation under $\mathcal{A}$ satisfies every clock constraint in $h$. Then in the definition for $prob_\otimes$, upon the action $(a, h)$ and the target location $\ell'$, the product PTA first chooses the unique rule $(q, \mathcal{L}(\ell'), h(\mathcal{L}(\ell')), Y \cap \mathcal{Y}, q')$ from the emitting mode $q$ and the label $\mathcal{L}(\ell')$ for which the uniqueness comes from the determinism of $\Delta$, then perform probabilistic jump from $\mathcal{C}$ and the discrete transition from $\mathcal{A}$. Finally, we label each $(\ell, q)$ by $q$ to meet the Rabin or finite acceptance condition.

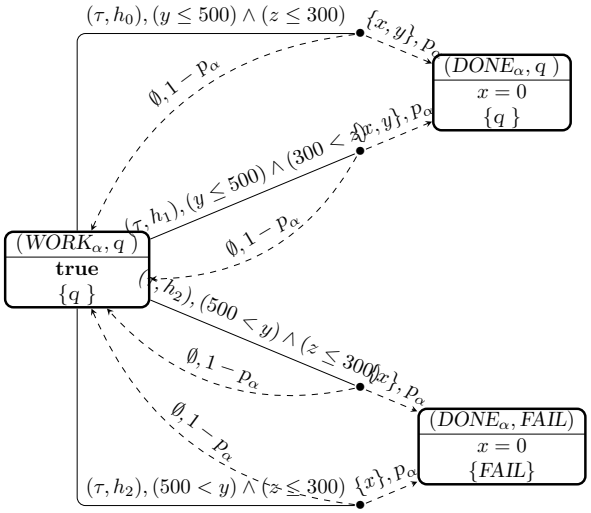


**Figure 3: A Part of Product PTA**

Remark 3 (Technical Novelty). *The novelty for our product construction is that by adopting extended actions and integrating them into enabling condition and probabilistic transition function, the DTA $\mathcal{A}$ can know which rule to use upon any symbol to be read. This solves the problem that probabilistic jumps can lead to different locations.*

Remark 4. *It is easy to see that the PTA $\mathcal{C}\otimes\mathcal{A}_q$ is well-formed as $\mathcal{C}$ is well-formed and $\mathcal{A}$ does not introduce extra invariant conditions.*

*Example 4.1.* Here we represent an running example to show how our product construction works. Here for

the accepting mod $q$ in DTA, we have

$\mathcal{T}_q = \{h_0 = \{\emptyset \mapsto \mathbf{true}, \{\alpha\} \mapsto (x \le 500), \{\beta\} \mapsto (y \le 300), \{\alpha,\beta\} \mapsto \mathbf{true}\},$
$h_1 = \{\emptyset \mapsto \mathbf{true}, \{\alpha\} \mapsto (x \le 500), \{\beta\} \mapsto (300 < y), \{\alpha,\beta\} \mapsto \mathbf{true}\},$
$h_2 = \{\emptyset \mapsto \mathbf{true}, \{\alpha\} \mapsto (500 < x), \{\beta\} \mapsto (y \le 300), \{\alpha,\beta\} \mapsto \mathbf{true}\},$
$h_3 = \{\emptyset \mapsto \mathbf{true}, \{\alpha\} \mapsto (500 < x), \{\beta\} \mapsto (300 < y), \{\alpha,\beta\} \mapsto \mathbf{true}\}\}.$

And a part of the product of Example 2.2 and Example 2.5 is depicted in Figure 2.

Below we clarify the correspondence between $\mathcal{C}, \mathcal{A}$ and $\mathcal{C}\otimes\mathcal{A}_q$. We first show the relationship between paths under $\mathcal{C}$ and those under $\mathcal{C}\otimes\mathcal{A}_q$. Informally, paths under $\mathcal{C}\otimes\mathcal{A}_q$ are just paths under $\mathcal{C}$ extended with runs of $\mathcal{A}$.

**Transformation $\mathcal{T}$ for Paths from $\mathcal{C}$ into $\mathcal{C}\otimes\mathcal{A}_q$.** The transformation is defined as the function $\mathcal{T} : Paths^*_{\mathcal{C}} \cup Paths^{\omega}_{\mathcal{C}} \to Paths^*_{\mathcal{C}\otimes\mathcal{A}_q} \cup Paths^{\omega}_{\mathcal{C}\otimes\mathcal{A}_q}$ which transform a finite or infinite path under $\mathcal{C}$ into one under $\mathcal{C}\otimes\mathcal{A}_q$. For a finite path $\rho = (\ell_0,\nu_0)a_0\dots a_{n-1}(\ell_n,\nu_n)$ under $\mathcal{C}$ (note that $(\ell_0,\nu_0) = (\ell^*,\mathbf{0})$ by definition), we define $\mathcal{T}(\rho)$ to be the unique finite path

$$\mathcal{T}(\rho) := ((\ell_0,q_0),\nu_0\cup\mu_0)a'_0\dots a'_{n-1}((\ell_n,q_n),\nu_n\cup\mu_n) \tag{5}$$

under $\mathcal{C}\otimes\mathcal{A}_q$ such that the following conditions (†) hold:

- $(q,\mathbf{0}) \xrightarrow{\mathcal{L}(\ell^*)} (q_0,\mu_0)$ (note that $\mu_0 = \mathbf{0}$);
- for all $0 \le k < n$, if $a_k \in [0,\infty)$ then $a'_k = a_k$ and $(q_k,\mu_k) \xRightarrow{a_k} (q_{k+1},\mu_{k+1})$;
- for all $0 \le k < n$, if $a_k \in Act$ then $a'_k = (a_k,\xi_k)$ and $(q_k,\mu_k) \xRightarrow{\mathcal{L}(\ell_{k+1})} (q_{k+1},\mu_{k+1})$ where $\xi_k$ is the unique function such that for each symbol $b \in \Sigma$, $\xi_k(b)$ is the unique clock constraint appearing in a rule emitting from $q_k$ and with symbol $b$ such that $\mu_k \models \xi_k(b)$.

Likewise, for an infinite path $\pi = (\ell_0,\nu_0)a_0(\ell_1,\nu_1)a_1\dots$ under $\mathcal{C}$, we define $\mathcal{T}(\pi)$ to be the unique infinite path

$$\mathcal{T}(\pi) := ((\ell_0,q_0),\nu_0\cup\mu_0)a'_0((\ell_1,q_1),\nu_1\cup\mu_1)a'_1\dots \tag{6}$$

under $\mathcal{C}\otimes\mathcal{A}_q$ such that the three conditions in (†) hold for all $k \in \mathbb{N}_0$ instead of all $0 \le k < n$.

LEMMA 4.2. *The function $\mathcal{T}$ is a bijection. Moreover, for any infinite path $\pi$, $\pi$ is non-zeno iff $\mathcal{T}(\pi)$ is non-zeno.*

PROOF. The first claim follows directly from the determinism and totality of DTAs. The second claim follows from the fact that $\mathcal{T}$ preserves time elapses in the transformation. □

Below we also show the correspondence on schedulers before and after the product construction.

**Transformation $\theta$ for Schedulers from $\mathcal{C}$ into $\mathcal{C}\otimes\mathcal{A}_q$.** We define the function $\theta$ from the set of schedulers under $\mathcal{C}$ into the set of schedulers under $\mathcal{C}\otimes\mathcal{A}_q$ as follows: for any scheduler $\sigma$ for $\mathcal{C}$, $\theta(\sigma)$ (for $\mathcal{C}\otimes\mathcal{A}_q$) is defined such that for any finite path $\rho$ under $\mathcal{C}$ where $\rho = (\ell_0,\nu_0)a_0\dots a_{n-1}(\ell_n,\nu_n)$ and $\mathcal{T}(\rho)$ is given as in (5),

$$\theta(\sigma)(\mathcal{T}(\rho)) := \begin{cases} \sigma(\rho) & \text{if } n \text{ is even} \\ (\sigma(\rho),\lambda(\rho)) & \text{if } n \text{ is odd} \end{cases}$$

where $\lambda(\rho)$ is the unique function such that for each symbol $b \in \Sigma$, $\lambda(\rho)(b)$ is the clock constraint in the unique rule emitting from $q_k$ and with symbol $b$ such that $\mu_n \models \lambda(\rho)(b)$. Note that the well-definedness of $\theta$ follows from Lemma 4.2.

From Lemma 4.2, the product construction, the determinism and totality of $\Delta$, one can prove directly the following lemma.

LEMMA 4.3. *The function $\theta$ is a bijection.*

Now we prove the relationship between infinite paths accepted by a DTRA/DTFA before product construction and infinite paths satisfying certain Rabin/reachability condition. We first consider that $\mathcal{A}$ is a DTRA in the form (3).

We introduce more notations. First, we lift the function $\mathcal{T}$ to all subsets of paths in the standard fashion: for all subsets $A \subseteq Paths^*_{\mathcal{C}} \cup Paths^{\omega}_{\mathcal{C}}$, $\mathcal{T}(A) := \{\mathcal{T}(\omega) \mid \omega \in A\}$. Then for an infinite path $\pi = (\ell_0,\nu_0)a_0(\ell_1,\nu_1)a_1\dots$ with $\mathcal{T}(\pi)$ taking the form (6), we define the *trace* of $\mathcal{T}(\pi)$ as an infinite word over $Q$ defined by $trace(\mathcal{T}(\pi)) := q_0q_1\dots$ . Finally, for any scheduler $\sigma$ for $\mathcal{C}\otimes\mathcal{A}_q$, we define the set $RPaths_\sigma$ by

$$RPaths_\sigma := \left\{\pi \in Paths^{\omega}_{\mathcal{C}\otimes\mathcal{A}_q,\sigma} \mid \mathbf{ACC}\left(inf(trace(\pi)),\mathcal{F}\right)\right\}.$$

Intuitively, $RPaths_\sigma$ is the set of infinite paths that meet the Rabin condition $\mathcal{F}$ from $\mathcal{A}$.

PROPOSITION 4.4. *For any scheduler $\sigma$ for $\mathcal{C}$ and any initial mode $q$ for $\mathcal{A}$, we have $\mathcal{T}\left(AccPaths^{\mathcal{A},q}_{\mathcal{C},\sigma}\right) = RPaths_{\theta(\sigma)}$.*

PROOF. By definition, the set $AccPaths^{\mathcal{A},q}_{\mathcal{C},\sigma}$ equals

$$\left\{\pi \in Paths^{\omega}_{\mathcal{C},\sigma} \mid \mathbf{ACC}\left(inf(traj(\xi_\pi)),\mathcal{F}\right)\right\}$$

where $\xi_\pi$ is the unique run of $\mathcal{A}$ on $\mathcal{L}(\pi)$ with initial configuration $(q^*,\mathbf{0})$ for which $q^*$ is the unique location such that $(q,\mathbf{0}) \xrightarrow{\mathcal{L}(\ell^*)} (q^*,\mathbf{0})$. Let $\pi = (\ell_0,\nu_0)a_0(\ell_1,\nu_1)a_1\dots$ be any infinite path. By the definition of $\mathcal{T}$ we have

$$\mathcal{T}(\pi) = ((\ell_0,q_0),\nu_0\cup\mu_0)a'_0((\ell_1,q_1),\nu_1\cup\mu_1)a'_1\dots$$

in the form (6) with $\xi_\pi = \{(q_n,\mu_n,a_n)\}_{n\in\mathbb{N}_0}$ being the unique run on $\mathcal{L}(\pi) = a_0a_1\dots$ . Then it is obvious that

$$trace(\mathcal{T}(\pi)) = q_0q_1\cdots = traj(\mathcal{A}_{(q^*,\mathbf{0})}(\mathcal{L}(\pi))).$$

It follows that $\mathit{inf}(\mathit{trace}(\mathcal{T}(\pi)))$ is Rabin accepting by $\mathcal{F}$ iff $\mathit{inf}(\mathit{traj}(\mathcal{A}_{(q^*,\mathbf{0})}(\mathcal{L}(\pi))))$ is Rabin accepting by $\mathcal{F}$. □

Finally, we demonstrate the relationship between acceptance probabilities before product construction and reachability probabilities after product construction. We also clarify the probability of zenoness before and after the product construction.

LEMMA 4.5. *For any scheduler $\sigma$ and mode $q$, the followings hold:*

- $\mathfrak{p}_q^\sigma = \mathbb{P}^{\mathcal{C},\sigma}\left(\mathit{AccPaths}_{\mathcal{C},\sigma}^{\mathcal{A},q}\right) = \mathbb{P}^{\mathcal{C}\otimes\mathcal{A}_q,\theta(\sigma)}\left(\mathit{RPaths}_{\theta(\sigma)}\right)$;
- $\mathbb{P}^{\mathcal{C},\sigma}(\{\pi \mid \pi \textit{ is zeno}\}) = \mathbb{P}^{\mathcal{C}\otimes\mathcal{A}_q,\theta(\sigma)}(\{\pi' \mid \pi' \textit{ is zeno}\})$.

PROOF. Define the probability measure $\mathbb{P}'$ by: $\mathbb{P}'(A) = \mathbb{P}^{\mathcal{C}\otimes\mathcal{A}_q,\theta(\sigma)}(\mathcal{T}(A))$ for $A \in \mathcal{F}^{\mathcal{C},\sigma}$. We show that $\mathbb{P}' = \mathbb{P}^{\mathcal{C},\sigma}$. By [7, Theorem 3.3], it suffices to consider cylinder sets as they form a pi-system (cf. [7, Page 43]). Let $\rho = (\ell_0,\nu_0)a_0\ldots a_{n-1}(\ell_n,\nu_n)$ be any finite path under $\mathcal{C}$. By definition, we have that

$$\begin{aligned}\mathbb{P}^{\mathcal{C},\sigma}(\mathit{Cyl}(\rho)) &= \mathbb{P}^{\mathcal{C}\otimes\mathcal{A}_q,\theta(\sigma)}(\mathit{Cyl}(\mathcal{T}(\rho)))\\ &= \mathbb{P}^{\mathcal{C}\otimes\mathcal{A}_q,\theta(\sigma)}(\mathcal{T}(\mathit{Cyl}(\rho)))\\ &= \mathbb{P}'(\mathit{Cyl}(\rho))\ .\end{aligned}$$

The first equality comes from the fact that both versions of product construction preserves transition probabilities. The second equality is due to $\mathit{Cyl}(\mathcal{T}(\rho)) = \mathcal{T}(\mathit{Cyl}(\rho))$. The final equality follows from the definition. Hence $\mathbb{P}^{\mathcal{C},\sigma} = \mathbb{P}'$. Then the first claim follows from Proposition 4.4 and the second claim follows from Lemma 4.2. □

A side result from Theorem 4.5 says that $\theta$ preserves time-divergence for schedulers before and after product construction. From Theorem 4.5 and Lemma 4.3, one immediately obtains the following result which transforms the PTA-DTA problem into Rabin(-accepting) probabilities under the product PTA.

COROLLARY 4.6. *For any initial mode $q$,*

$$\mathit{opt}_\sigma \mathfrak{p}_q^\sigma = \mathit{opt}_{\sigma'} \mathbb{P}^{\mathcal{C}\otimes\mathcal{A}_q,\sigma'}(\mathit{RPaths}_{\sigma'})$$

*where opt refers to either* inf *(infimum) or* sup *(supremum), $\sigma$ (resp. $\sigma'$) range over all time-divergent schedulers for $\mathcal{C}$ (resp. $\mathcal{C}\otimes\mathcal{A}_q$).*

**Solving Rabin Probabilities.** We follow the approach in [27] to solve Rabin probabilities over PTAs. Below we briefly describe the approach. The approach can be divided into two steps. The first step is to ensure time-divergence. This is achieved by (i) making a copy for every location in the PTA, (ii) enforcing a transition from every location to its copy to happen after 1 time-unit elapses, (iii) enforcing a transition from every copy location back to the original one immediately with no time-delay, and (iv) putting a special label *tick* in every copy. Then time-divergence is guaranteed by adding the label *tick* to the Rabin condition. The second step is to transform the problem into limit Rabin properties over MDPs [3, Theorem 10.127]. This step constructs an MDP $\mathrm{Reg}[\mathcal{C}\otimes\mathcal{A}_q]$ from the PTA $\mathcal{C}\otimes\mathcal{A}_q$ through a *region-graph* construction so that the problem is reduced to solving limit Rabin properties over $\mathrm{Reg}[\mathcal{C}\otimes\mathcal{A}_q]$. Then standard methods based on *maximal end components* (MECs) are applied to $\mathrm{Reg}[\mathcal{C}\otimes\mathcal{A}_q]$. In detail, the algorithm computes the reachability probability to MECs that satisfy the Rabin acceptance condition; In order to guarantee time-divergence, the algorithm only picks up MECs with at least one location that has a *tick* label. Based on this approach, our result leads to an algorithm for solving the problem PTA-DTA.

Note that in $\mathcal{C}\otimes\mathcal{A}_q$, although the size of $\mathit{Act}_\otimes$ may be exponential due to possible exponential blow-up from $\mathcal{T}_q$, one easily sees that $|L_\otimes|$ is $|L|\cdot|Q|$ and $|\mathcal{X}_\otimes| = |\mathcal{X}| + |\mathcal{Y}|$. Hence, the size of $\mathrm{Reg}[\mathcal{C}\otimes\mathcal{A}_q]$ is still exponential in the sizes of $\mathcal{C}$ and $\mathcal{A}$. It follows that $\mathit{opt}_\sigma\mathfrak{p}_q^\sigma$ can be calculated in exponential time from the MEC-based algorithm illustrated in [3, Theorem 10.127], as is demonstrated by the following proposition.

PROPOSITION 4.7. *The problem* PTA-DTA *is in EXPTIME in the size of the input PTA and DTA.*

Below we prove the hardness of the problem. In [23], ?? proved that the reachablity-probability problem for arbitrary PTAs is *EXPTIME*-complete. We show a polynomial-time reduction from the PTA reachibility problem to the PTA-DTA problem as follows. For an arbitrary PTA $\mathcal{C}$ in the form (1) and a set $F \subseteq L$ of final locations, we let $\mathcal{C}' = \left(L, \ell^*, \mathcal{X}, \mathit{Act}, \mathit{inv}, \mathit{enab}, \mathit{prob}, AP', \mathcal{L}'\right)$ where $AP' := AP \cup \{\mathit{acc}\}$ and $\mathcal{L}'$ is defined by

$$\mathcal{L}'(l) := \begin{cases} \mathcal{L}(l) & \text{if } \ell \notin F \\ \mathcal{L}(l) \cup \{\mathit{acc}\} & \ell \in F \end{cases}$$

for which *acc* is a fresh atomic proposition. We also construct the DTRA $\mathcal{A}'$ by

$$\mathcal{A}' := (\{q_0, q_1\}, \Sigma, \emptyset, \Delta, \{(\emptyset, \{q_1\})\})$$

where $\Sigma := \{\mathcal{L}'(\ell) \mid \ell \in L\}$ and $\Delta$ contains exactly the following rules:

- $(q_0, U, \mathit{true}, \emptyset, q_1) \in \Delta$ for all $U \in \Sigma$ such that $\mathrm{acc} \in U$;
- $(q_i, U, \mathit{true}, \emptyset, q_i) \in \Delta$ for all $U \in \Sigma$ and $i \in \{0,1\}$.

It is then straightforward from definition that an infinite path under $\mathcal{C}$ visits some location in $F$ iff the infinite path (under $\mathcal{C}'$) is accepted by $\mathcal{A}'$. Hence, under any scheduler (for both $\mathcal{C}$ and $\mathcal{C}'$), the probability to reach

$F$ in $\mathcal{C}$ equals the probability that $\mathcal{C}'$ observes $\mathcal{A}'$ under initial mode $q_0$. It follows that the maximum/minimum probability to reach $F$ can be solved by the algorithm demonstrated in Proposition **??** for Rabin acceptance condition. By the reduction, we obtain the main result of this section which settles the computational complexity of the problem PTA-DTA.

THEOREM 4.8. *The PTA-DTA problem is EXPTIME-complete.*

## 5 THE PTA-NTA PROBLEM

In this section, we study the PTA-NTA problem where the input timed automaton needs not to be deterministic. We show that for Rabin acceptance condition, the problem is undecidable, while the problem can be solved approximately through an infinite-state MDP construction when we restrict ourselves to finite acceptance condition.

We first consider Rabin acceptance condition. The main idea for the undecidability result is to reduce the universality problem of timed automata to the PTA-NTA problem. The universality problem over timed automata is well-known to be undecidable, as follows.

LEMMA 5.1. *([28, Theorem 5.2]) Given a timed automaton over an alphabet $\Sigma$ and an initial mode, the problem of deciding whether it accepts all time-divergent timed words w.r.t Büchi acceptance condition over $\Sigma$ is undecidable.*

Although Lemma 5.1 is on Büchi acceptance condition, it holds also for Rabin acceptance condition since Rabin acceptance condition extends Büchi acceptance condition. Actually the two acceptance conditions are equivalent over timed automata (cf. [28, Theorem 3.20]).

Now we prove the undecidability result as follows. The proof idea is that we construct a PTA that can generate every time-divergent timed words with probability 1 by some time-divergent scheduler. Then the TRA accepts all time-divergent timed words iff the minimal probability that the PTA observes the TRA equals 1.

THEOREM 5.2. *Given a PTA $\mathcal{C}$ and a TRA $\mathcal{A}$, the problem to decide whether the minimal probability that that $\mathcal{C}$* observes *$\mathcal{A}$ (under a given initial mode) is equal to 1 is undecidable.*

PROOF. Let $\mathcal{A} = (Q, \Sigma, \mathcal{Y}, \Delta, \mathcal{F})$ be any TRA where the alphabet $\Sigma = \{b_1, b_2, \cdots, b_k\}$ and the initial mode is $q_{start}$. W.l.o.g, we consider that $\Sigma \subseteq 2^{AP}$ for some finite set $AP$. This assumption is not restrictive since what $b_i$'s concretely are is irrelevant, while the only thing that matters is that $\Sigma$ has $k$ different symbols. We first construct the TRA $\mathcal{A}' = (Q', \Sigma', \mathcal{Y}, \Delta', \mathcal{F})$ where:

- $Q' = Q \cup \{q_{init}\}$ for which $q_{init}$ is a fresh mode;
- $\Sigma' = \Sigma \cup \{b_0\}$ for which $b_0$ is a fresh symbol;
- $\Delta' = \Delta \cup \{\langle q_{init}, b_0, \mathbf{true}, \mathcal{Y}, q_{start}\rangle\}$.

Then we construct the PTA

$$\mathcal{C}' = (L, \ell^*, \mathcal{X}, Act, inv, enab, prob, \mathcal{L})$$

where:

- $L := \Sigma'$;
- $\ell^* := b_0$;
- $\mathcal{X} := \emptyset$;
- $Act := \Sigma$;
- $inv(b_i) := \mathbf{true}$ for $b_i \in L$;
- $enab(b_i, b_j) := \mathbf{true}$ for $b_i \in L$ and $b_j \in Act$;
- $prob(b_i, b_j)$ is the Dirac distribution at $(\emptyset, b_j)$ (i.e., $\mu(\emptyset, b_j) = 1$ and $\mu(X, b) = 0$ whenever $(X, b) \neq (\emptyset, b_j)$), for $b_i \in L$ and $b_j \in Act$;
- $\mathcal{L}(b_i) := b_i$ for $b_i \in L$.

Note that we allow no clocks in the construction since clocks are irrelevant for our result. Since we omit clocks, we also treat states (of $\mathcal{C}'$) as single locations. Below we prove that $\mathcal{A}$ accepts all time-divergent timed words over $\Sigma$ with initial mode $q_{start}$ iff the minimal probability that $\mathcal{C}'$ observes $\mathcal{A}'$ with initial mode $q_{init}$ equals 1.

Consider any time-divergent infinite timed word $w = t_0 b'_0 t_1 b'_1 \cdots$ over $\Sigma$ (where $t_i \in \mathbb{R}$ and $b'_i \in \Sigma$). We define an infinite sequence $\{\rho_n\}_{n\in\mathbb{N}_0}$ of finite paths (of $\mathcal{C}'$) inductively as follows:

- $\rho_0 := b_0 (= \ell^*)$; (Note that we treat states as locations since clocks are irrelevant.)
- for $m \geq 0$, $\rho_{2m+1} := \langle s_0, a_0, s_1, \ldots, a_{k-1}, s_k, t_m, s_k\rangle$ if $\rho_{2m} = \langle s_0, a_0, s_1, \ldots, a_{k-1}, s_k\rangle$;
- for $m \geq 0$, $\rho_{2m+2} := \langle s_0, a_0, s_1, \ldots, a_{k-1}, s_k, b'_m, b'_m\rangle$ if $\rho_{2m+1} = \langle s_0, a_0, s_1, \ldots, a_{k-1}, s_k\rangle$.

Intuitively, the sequence $\{\rho_n\}_{n\in\mathbb{N}_0}$ is constructed by letting the PTA $\mathcal{C}'$ read the timed word $w$ in a stepwise fashion, while adjusting the next location upon reading a symbol (as an action) from $\Sigma$. Then one can define a scheduler $\sigma_w$ by:

- $\sigma_w(\rho_{2m}) := t_m$ for $m \geq 0$;
- $\sigma_w(\rho_{2m+1}) := b'_m$ for $m \geq 0$;
- $\sigma_w(\rho)$ is arbitrarily defined if $\rho$ is not from the sequence $\{\rho_n\}_{n\in\mathbb{N}_0}$.

Intuitively, $\sigma_w$ always chooses time-delays and actions from $w$. Note that $\sigma_w$ is time divergent since $w$ is time divergent. Moreover, from definition we have that $\mathbb{P}^{\mathcal{C},\sigma_w}(\{\pi \mid \mathcal{L}(\pi) = w\}) = 1$. Hence

$$\mathfrak{p}^{\sigma_w}_{q_{init}} = \begin{cases} 1 & \text{if } \mathcal{A} \text{ accepts } w \text{ w.r.t. } (q_{start}, \mathbf{0}) \ , \\ 0 & \text{if } \mathcal{A} \text{ rejects } w \text{ w.r.t. } (q_{start}, \mathbf{0}) \ . \end{cases}$$

Then we have that $\inf_\sigma \mathbb{P}^{\mathcal{C},\sigma}\left(AccPaths^{\mathcal{A}',q_{init}}_{\mathcal{C},\sigma}\right) = 1$ iff $\mathcal{A}$ accepts all time-divergent timed words w.r.t. $(q_{start}, \mathbf{0})$. □

# 6 CONCLUSION AND FUTURE WORK

In this paper, we studied the problem of model-checking PTAs against timed-automata specifications. We considered both Rabin and finite acceptance conditions. For Rabin acceptance condition, we first solved the problem with DTA specifications and Rabin acceptance condition through a product construction and prove that its computational complexity is EXPTIME-complete; then we proved that the problem with general timed-automata specifications is undecidable through a reduction from the universality problem of timed automata. For finite acceptance condition, we demonstrated that the problem with DTA specifications can be solved through efficient zone-based algorithms on verifying reachability probability of PTAs [20, 24], while the problem with general timed-automata specifications can be solved by an approximation algorithm based on value iteration.

An interesting future direction is zone-based algorithms for Rabin acceptance condition. Another theoretical direction is to investigate timed-automata specifications with cost or reward. A more practical direction is to apply our approaches to industrial-level examples.

# 7 RELATED WORKS

Model-checking probabilistic timed models against linear dense-time properties are mostly considered for continuous-time Markov processes (CTMPs). First, Donatelli *et al.* [11] proved an expressibility result that the class of linear dense-time properties encoded by DTAs is not subsumed by branching-time properties. They also demonstrated an efficient algorithm for verifying continuous-time Markov chains [**?** ] against one-clock DTAs. Then various results on verifying CTMPs are obtained for specifications through DTAs and general timed automata (cf. [4, 8–12]). The fundamental difference between CTMPs and PTAs is that the former assign probability distributions to time elapses, while the latter treat time-elapses as pure nondeterminism. Because of this difference, the techniques for CTMPs cannot be applied to PTAs.

For PTAs, the only relevant result is by ?[**?** ] who developed an approach for verifying PTAs against deterministic discrete-time omega-regular automata through a similar product construction. Our results extend theirs in two ways. First, our product construction extends theirs with extra ability to tackle timing constraints from both the PTA and the DTA. The extension is non-trivial since it needs to resolve the integration between randomness and timing constraints, while ensuring the EXPTIME-completeness of the problem, matching the computational complexity in the discrete-time case [**?** ]. Second, our results also cover an undecidability result and an approximation algorithm in the case of general nondeterministic timed automata, extending [**?** ] with nondeterminism.

# REFERENCES


[1] Rajeev Alur and David L. Dill. 1994. A Theory of Timed Automata. *Theor. Comput. Sci.* 126, 2 (1994), 183–235.

[2] Étienne André, Laurent Fribourg, and Jeremy Sproston. 2013. An extension of the inverse method to probabilistic timed automata. *Formal Methods in System Design* 42, 2 (2013), 119–145.

[3] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. MIT Press.

[4] Benoît Barbot, Taolue Chen, Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. 2011. Efficient CTMC Model Checking of Linear Real-Time Objectives. In *TACAS (LNCS)*, Parosh Aziz Abdulla and K. Rustan M. Leino (Eds.), Vol. 6605. Springer, 128–142.

[5] Danièle Beauquier. 2003. On probabilistic timed automata. *Theor. Comput. Sci.* 292, 1 (2003), 65–84.

[6] Jasper Berendsen, Taolue Chen, and David N. Jansen. 2009. Undecidability of Cost-Bounded Reachability in Priced Probabilistic Timed Automata. In *TAMC (LNCS)*, Jianer Chen and S. Barry Cooper (Eds.), Vol. 5532. Springer, 128–137.

[7] Patrick Billingsley. 2012. *Probability and Measure* (anniversary edition ed.). Wiley.

[8] Luca Bortolussi and Roberta Lanciani. 2015. Fluid Model Checking of Timed Properties, See [26], 172–188.

[9] Tomás Brázdil, Jan Krcál, Jan Kretínský, Antonín Kucera, and Vojtech Rehák. 2011. Measuring performance of continuous-time stochastic processes using timed automata. In *HSCC*, Marco Caccamo, Emilio Frazzoli, and Radu Grosu (Eds.). ACM, 33–42.

[10] Taolue Chen, Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. 2011. Model Checking of Continuous-Time Markov Chains Against Timed Automata Specifications. *Logical Methods in Computer Science* 7, 1 (2011).

[11] Susanna Donatelli, Serge Haddad, and Jeremy Sproston. 2009. Model Checking Timed and Stochastic Properties with C-SLˆ{TA}. *IEEE Trans. Software Eng.* 35, 2 (2009), 224–240.

[12] Hongfei Fu. 2013. Approximating acceptance probabilities of CTMC-paths on multi-clock deterministic timed automata. In *HSCC*, Calin Belta and Franjo Ivancic (Eds.). ACM, 323–332.

[13] Henrik Ejersbo Jensen. 1996. Model Checking Probabilistic Real Time Systems. In *7th Nordic Workshop on Programming Theory*. Chalmers University of Technology, 247–261. Report 86.

[14] Aleksandra Jovanovic, Marta Z. Kwiatkowska, and Gethin Norman. 2015. Symbolic Minimum Expected Time Controller Synthesis for Probabilistic Timed Automata, See [26], 140–155.

[15] Marcin Jurdzinski, Marta Z. Kwiatkowska, Gethin Norman, and Ashutosh Trivedi. 2009. Concavely-Priced Probabilistic

Timed Automata. In *CONCUR (LNCS)*, Mario Bravetti and Gianluigi Zavattaro (Eds.), Vol. 5710. Springer, 415–430.

[16] Marcin Jurdzinski, Jeremy Sproston, and François Laroussinie. 2008. Model Checking Probabilistic Timed Automata with One or Two Clocks. *Logical Methods in Computer Science* 4, 3 (2008).

[17] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2009. Stochastic Games for Verification of Probabilistic Timed Automata. In *FORMATS (LNCS)*, Joël Ouaknine and Frits W. Vaandrager (Eds.), Vol. 5813. Springer, 212–227.

[18] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *CAV (LNCS)*, Ganesh Gopalakrishnan and Shaz Qadeer (Eds.), Vol. 6806. Springer, 585–591.

[19] Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Jeremy Sproston. 2006. Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods in System Design* 29, 1 (2006), 33–78.

[20] Marta Z. Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. 2002. Automatic verification of real-time systems with discrete probability distributions. *Theor. Comput. Sci.* 282, 1 (2002), 101–150.

[21] Marta Z. Kwiatkowska, Gethin Norman, Jeremy Sproston, and Fuzhi Wang. 2007. Symbolic model checking for probabilistic timed automata. *Inf. Comput.* 205, 7 (2007), 1027–1077.

[22] François Laroussinie and Jeremy Sproston. 2007. State explosion in almost-sure probabilistic reachability. *Inf. Process. Lett.* 102, 6 (2007), 236–241.

[23] François Laroussinie and Jeremy Sproston. 2007. State explosion in almost-sure probabilistic reachability. *Inf. Process. Lett.* 102, 6 (2007), 236–241.

[24] Gethin Norman, David Parker, and Jeremy Sproston. 2013. Model checking for probabilistic timed automata. *Formal Methods in System Design* 43, 2 (2013), 164–190.

[25] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

[26] Sriram Sankaranarayanan and Enrico Vicario (Eds.). 2015. *FORMATS*. LNCS, Vol. 9268. Springer.

[27] Jeremy Sproston. 2011. Discrete-Time Verification and Control for Probabilistic Rectangular Hybrid Automata. In *QEST*. IEEE Computer Society, 79–88.

[28] Frits W. Vaandrager. 1997. A Theory of Testing for Timed Automata (Abstract). In *TAPSOFT (LNCS)*, Michel Bidoit and Max Dauchet (Eds.), Vol. 1214. Springer, 39.

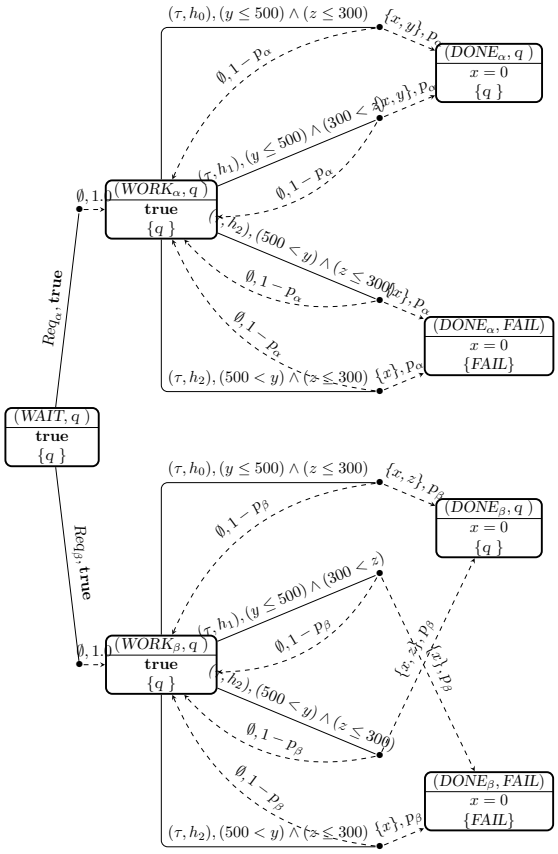


**Figure 4: A Big Part of Product PTA**

# A APPENDIX