L^*-Based Learning of Probabilistic Timed Automata with One Clock

Rong Wu*
*School of Software Engineering, Tongji University, Shanghai, China
Email: wu_rong@tongji.edu.cn

***Abstract*—Probabilistic timed automata (PTAs) allow for analyzing systems which operate under the existence of both probability and time. Probabilistic model checking for PTAs requires formal models as a basis. In this paper, we consider a learning algorithm for PTAs with one clock (OPTAs) which extends Angluin's L^* algorithm. For the black-box systems, we estimate the probabilities by sampling system traces, and apply the framework of Probably Approximately Correct (PAC) learning to provide correctness guarantees respect to error and confidence parameters. Experiments with the implementation of our learning algorithm include the sender of CSMA/CD protocol and randomly generated examples. Evaluating results show the effectiveness of our learning algorithm.**

***Index Terms*—model inference, active automata learning, probabilistic timed automata**

I. Introduction

Model learning is a efficient technique where we are able to get formal models for black-box systems. It enables the formal techniques, such as model checking, for systems. Passive learning and active learning are different two strategies in model learning. Specifically, passive learning outputs the learned models through before-hand data, but the data active learning uses grow gradually by asking for the systems under learning (SUL) during learning. Increased active learning algorithms are derived from the Angluin's L^* algorithm [1] which learns deterministic finite automata (DFAs), e.g. learning other automata variants like Mealy machines [2], and symbolic automata [3] [4].

However, active learning in stochastic and black-box settings has received less attention, except for Markov Decision Processes (MDPs) [5] and Stochastic Reactive Systems (SMMs) [6], where the learned probabilities converge to the true values in the large sample limit. Instead of finite automata with finite alphabets, the learning algorithms for real-time systems are complicated since continuous-time semantics. In [7], the learner can generate equivalent models with the One-clock Timed Automata (OTAs) under the real-time systems, through transforming the learning problem from the timed language observed outside to the timed language observed inside. Since the goal of Probably Approximately Correct (PAC) learning is the hypothesis generated by the learner has low error with high probability, [8] applies it to the learning of OTAs, which approximates equivalence between the learned model and the SUL by random sampling.

The contributions of this paper are as follows:

1) We present a L^*-based learning algorithm for One-clock Probabilistic Timed Automata (OPTAs), where OPTAs allow for analyzing systems which operate under the existence of both probability and time.
2) In L^*, the learner queries the teacher for collecting information of the SUL. Instead the existence of the teacher owning perfect knowledge of the SUL, we approximate the queries offered by the teacher under the black-box system via sampling. Besides, we apply PAC learning to check the correctness of the hypothesis generated by the learner. Thus, we carefully design the learning mechanism, the queries, the behaviors of the learner and the teacher.
3) We implemented and evaluated in the sender of CSMA/CD protocol and random examples. Experiment results show the effectiveness of our algorithm.

Structure: In Sect. II, we introduce preliminaries like OPTAs and the L^* algorithm. We present L^*-based learning algorithm for OPTAs in detail in Sect. III. Sect. IV discusses the evaluation and we provide a concluding in Sect. V.

II. Preliminaries

Let $\mathbb{R}_{\geq 0}$ and $\mathbb{N}$ denote the sets of non-negative real numbers and natural numbers, respectively, and $\mathbb{B} = \{\mathsf{T}, \mathsf{F}\}$ the Boolean set where T represents true. For a finite set Z, a probabilistic $distribution$ is defined by function $\eta : Z \mapsto [0, 1]$ with $\sum_{z \in Z} \eta(z) = 1$, and we refer with $Dist(Z)$ to the set of distributions over Z. The concatenation of two sequences $a, a' \in Z^*$ is denoted by $a \cdot a'$, where a is a prefix of $a \cdot a'$, denoted $a \ll a \cdot a'$, and $prefixes(a) = \{a' | a' \in Z^* : a' \ll a\}$ is the set of all prefixes of sequence a. A set of sequences $A \subseteq Z^*$ is prefix-closed, iff $\forall a \in A : prefixes(a) \subseteq A$. Suffixes and suffix-closedness are defined analogously. We use $\mathcal{S}(s) \in \mathbb{N}$ to denote the multiplicity of s in multiset $\mathcal{S}$.

Probabilistic timed automata (PTAs) [9] use clocks to model real-time behaviour like classical timed automata [10]. We consider models with only one clock denoted by x in this paper. A clock valuation is a value $\nu \in \mathbb{R}_{\geq 0}$ interpreted as the current value of clock x. Let $[\nu]_{\mathsf{T}} = 0$ denote that clock x is reset to 0, and $[\nu]_{\mathsf{F}} = \nu$. The set of clock constraints over clock x, denoted by Φ, is defined by the form $\phi ::= \mathsf{T} | x \bowtie n \, | \phi \wedge \phi$, where $n \in \mathbb{N}$ and $\bowtie \in \{=, <, >, \leq, \geq\}$. we use $\nu \models \phi$ to denote a clock valuation ν satisfies a clock constraint ϕ.

DOI reference number: 10.18293/SEKE2023-156

Definition 1. *A one-clock probabilistic timed automata (OPTA) is a tuple* $\mathcal{P} = (\Sigma^I, \Sigma^O, Q, q_0, x, prob, \mathcal{L})$ *where*

- Σ^I *and* Σ^O *are finite sets of input and output symbols respectively,*
- Q *is a set of locations,* $q_0 \in Q$ *is the initial location,*
- x *is the unique clock,*
- $prob$ *is the probabilistic transition relation consisting of elements with the form* (q, g, i, η)*, where* $q \in Q$*, guard* $g \in \Phi$*,* $i \in \Sigma^I$*, and* $\eta \in Dist(\mathbb{B} \times Q)$*, and*
- $\mathcal{L} : Q \to \Sigma^O$ *is a labeling function.*

A state of OPTA $\mathcal{P}$ is a pair (q, ν) which transitions to either $(q, \nu + t)$ after elapsing a certain amount of time $t \in \mathbb{R}_{\geq 0}$, or $(q', [\nu]_b)$ with probability $\eta(b, q')$ after traversing an enabled probabilistic transition $(q, g, i, \eta) \in prob$. i.e. $\nu \models g$, where outcome $(b, q') \in \mathbb{B} \times Q$ consists of a reset indicator b signaling whether to reset the clock x and a successor location q'. We use $(q, \nu) \xrightarrow{(i,t)o} (q', [\nu + t]_b)$ to denote the successor relation after both transitions, where $\mathcal{L}(q') = o$.

Given a path $\rho = (q_0, \nu_0) \xrightarrow{t_1, i_1, \eta_1} (q_1, \nu_1) \cdots \xrightarrow{t_n, i_n, \eta_n} (q_n, \nu_n)$, we have a delay-timed trace $\mathcal{L}(\rho) = o_0(i_1, t_1)o_1 \cdots (i_n, t_n)o_n$, also called as delay-timed trace, where $\mathcal{L}(q_k) = o_k$. Obviously, reset-delay-timed traces integrate reset information, denoted $\omega_r = o_0(i_1, t_1)(b_1, o_1) \cdots (i_n, t_n)(b_n, o_n)$ where $b_k \in \mathbb{B}$ taken from $\eta_k(b_k, q_k)$. We extend the representation of successor relation to delay-timed traces ω by $(q, \nu) \xrightarrow{\mathcal{L}(q)} (q, \nu)$, and $(q, \nu) \xrightarrow{\omega \cdot (i,t)o} (q', \nu')$ if $\exists (q'', \nu'') : (q, \nu) \xrightarrow{\omega} (q'', \nu'') \wedge (q'', \nu'') \xrightarrow{(i,t)o} (q', \nu')$.

Delay-timed traces are observed outside according to the global clock. We introduce logical-timed traces and reset-logical-timed traces to denote the observations inside, i.e. from the view of the local clock. We use Γ to denote the mapping of observations outside and inside, e.g. $\Gamma(\omega_r) = \gamma_r$ represents reset logical-timed trace γ_r is the same sequence with reset delay-timed trace ω, except for the time recorded is the clock valuation μ_k after delaying time t_k, i.e. $\gamma_r = \Gamma(\omega_r) = o_0(i_1, \mu_1)(b_1, o_1) \cdots (i_n, \mu_n)(b_n, o_n)$ with $\mu_k = t_k$ if $k = 1 \vee b_{k-1} = \mathsf{T}$ and $\mu_k = \mu_{k-1} + t_k$. We use $\Pi(\gamma_r) = \gamma$ to denote the projection of reset-logical-timed trace γ_r without reset information, and let $last(\gamma_r) = o_n$, $\nu(\gamma_r) = [\mu_n]_{b_n}$.

Let $\mathbf{\Sigma^I} = \Sigma^I \times \mathbb{R}_{\geq 0}$ and $\mathbf{\Sigma^O} = \mathbb{B} \times \Sigma^O$. Then $\mathcal{TR}_d = \Sigma^O \times (\mathbf{\Sigma^I} \times \Sigma^O)^*$ denotes the set of delay-timed traces, and $\mathcal{TR}_{lr} = \Sigma^O \times (\mathbf{\Sigma^I} \times \mathbf{\Sigma^O})^*$ denotes the set of reset-logical-timed traces. We say OPTA $\mathcal{P}$ is $deterministic$ if each delay-timed trace in $\mathcal{TR}_d$ corresponds to at most one path. And, OPTA $\mathcal{P}$ is $complete$ if for all $q \in Q$ and $i \in \Sigma^I$, the guards form a partition of $\mathbb{R}_{\geq 0}$. We assume that the OPTAs we work are complete and deterministic in the following sections.

The L^* algorithm [1] learns an unknown regular language L, which has two different roles for learning: a learner, and a teacher which is able to answer membership and equivalence queries. First the learner checks which strings are in L using membership queries. While has collected sufficient information, the learner builds a hypothesis (DFA) $\mathcal{H}$ using the membership query results. It then checks the equivalence between language L and the language accepted by $\mathcal{H}$ via equivalence query. If not, the teacher returns a counterexample to the learner for refining $\mathcal{H}$. After processing a counterexample, the learner starts a new round until the equivalence query returns yes, i.e. $\mathcal{H}$ accepts language L.

III. Learning OPTAs via Sampling

Here we present a learning algorithm for OPTAs referring to the L^*-based algorithms of OTAs [7] and MDPs [5]. We first describe how we capture the behaviour of an OPTA.

For a timed language $\mathbb{L} \subseteq (\Sigma \times \mathbb{R}_{\geq 0})^*$ accepted by a timed automata over alphabet Σ, we associate to a characteristic function $f : (\Sigma \times \mathbb{R}_{\geq 0})^* \mapsto \{\mathsf{T}, \mathsf{F}\}$. Similarly, we explain OPTA $\mathcal{P}$ as a function $P : (\Sigma^O \times \mathbf{\Sigma^I})^* \mapsto Dist(\Sigma^O) \cup \{\perp\}$.

Definition 2. *For an OPTA* $\mathcal{P} = (\Sigma^I, \Sigma^O, Q, q_0, x, prob, \mathcal{L})$*, its characteristic function is* P*, defined for delay-timed input* $(i, t) \in \mathbf{\Sigma^I}$*, and delay-timed trace* $\omega \in \mathcal{TR}_d$ *as follows:*

- $P(\epsilon)(\mathcal{L}(q_0)) = 1$*;*
- $P(\omega \cdot (i, t)) = \perp$ *if* $\nexists (q, \nu) : (q_0, \nu_0) \xrightarrow{\omega} (q, \nu)$*;*
- $P(\omega \cdot (i, t))(o) = p$ *otherwise if* $\eta(b, q') = p > 0 \wedge \mathcal{L}(q') = o$*, where* $(q, g, i, \eta) \in prob$ *with* $\nu + t \models g$*.*

We say OPTAs $\mathcal{P}$ *and* $\mathcal{P}'$ *with characteristic function* P *and* P' *are equivalent, iff* $P = P'$*.*

We define reset-logical-timed characteristic function P_{lr}, which is the characteristic function observed inside, by $P_{lr}(\gamma_r \cdot (i, \mu))(b, o) = P(\omega \cdot (i, t))(o)$ if γ_r and ω refer to the same path, where $b \in \mathbb{B}$ is taken from $\eta(b, q')$ and $\mu = \nu(\gamma_r) + t$. We transform the learning problem to that of learning a reset-logical-timed characteristic function as noted in [7], that is, we can build a hypothesis that is equivalent with the target OPTA over their reset-logical-timed characteristic functions.

Theorem 1. *Given OPTAs* $\mathcal{P}$ *and* $\mathcal{P}'$ *with reset-logical-timed characteristic functions* P_{lr} *and* P'_{lr}*, if* $P_{lr} = P'_{lr} \Rightarrow \mathcal{P} \equiv \mathcal{P}'$*.*

A. PAC learning of OPTAs

We assume $\mathcal{P} = (\Sigma^I, \Sigma^O, Q, q_0, x, prob, \mathcal{L})$ is the OPTA underlying the SUL, and P is its characteristic function. For a hypothesis $\mathcal{H}$ with characteristic function H, let $P \oplus H = \{\omega \cdot (i, t) | \omega \in \mathcal{TR}_d, (i, t) \in \mathbf{\Sigma^I} : P(\omega \cdot (i, t)) \neq H(\omega \cdot (i, t))\}$ be the symmetric difference between P and H. We use $\mathcal{D}$ to denote a probabilistic distribution over $(\Sigma^O \times \mathbf{\Sigma^I})^*$. The quality of hypothesis $\mathcal{H}$ for $\mathcal{P}$ is defined by $\mathcal{D}(P \oplus H)$.

Let ϵ and δ be the error and confidence parameters respectively. We say a learning algorithm for OPTA $\mathcal{P}$ is PAC(ϵ, δ)-correct if its output learned model $\mathcal{H}$ satisfies:

$$Pr(\mathcal{D}(P \oplus H) \leq \epsilon) \geq 1 - \delta \tag{1}$$

where Pr is the probability of the event $\mathcal{D}(P \oplus H) \leq \epsilon$.

Thus, in PAC learning framework, equivalence queries require to sample delay-timed test sequences, i.e. elements in $(\Sigma^O \times \mathbf{\Sigma^I})^*$, then test whether they belong to the symmetric difference. Theorem 2 defines the minimum number of samples required to ensure Equation 1. We discuss equivalence queries in more detail in Sect. III-D4.

Theorem 2. *A learning algorithm PAC-learns an OPTA if the k-th equivalence query tests*

$$r_k = \frac{1}{\epsilon}(\ln\frac{1}{\delta} + \ln 2(k+1)). \quad (2)$$

random delay-timed test sequences from a fixed distribution over $(\Sigma^O \times \mathbf{\Sigma^I})^*$ without finding a counterexample.

B. Queries

Let $\lambda = \gamma \cdot e \in (\Sigma^O \times \mathbf{\Sigma^I})^*$ be a logical-timed test sequence, where γ is a logical-timed trace, and e is called as logical-timed continuation sequence. Obviously, reset-logical-timed test sequences is the logical-timed test sequences extending with reset information.

The teacher gains information about the reset-logical-timed characteristic function by sampling system traces. Let $\mathcal{S}$ be the multiset of reset-logical-timed traces collected from the SUL during learning. Thus, the teacher offers four queries:

$Frequency$ queries: take a logical-timed test sequence λ as input, and return the corresponding reset-logical-timed test sequence $\boldsymbol{\pi}(\lambda)$ and output frequencies $\mathbf{freq}(\lambda) : \mathbf{\Sigma^O} \mapsto \mathbb{N}$ collected in multiset $\mathcal{S}$.

$Complete$ queries: given a logical-timed test sequence λ, complete query $\mathbf{cq}(\lambda)$ returns whether $\mathbf{freq}(\lambda)$ has sufficient information to approximate the true distribution.

$Refine$ queries: given a set $rare$ of logical-timed test sequences which require to refine the knowledge about the SUL, refine query $\mathbf{rq}(rare)$ returns a multiset of reset-logical-timed traces sampled from the SUL along $rare$.

$Equivalence$ queries: given a hypothesis $\mathcal{H}$, and the parameters: error ϵ, confidence δ and the amount of previous equivalence queries k, equivalence query $\mathbf{eq}(\mathcal{H}, \epsilon, \delta, k)$ returns a pair (r, ctx) where r is the Boolean value signaling whether $\mathcal{H}$ passes all tests, and ctx is a counterexample.

C. Learner

1) Timed observation table: Timed Observation table is used to store information growing during learning in our work.

Definition 3. *A timed observation table is a tuple $\mathcal{T} = (\mathbf{\Sigma^I}, S, R, E, f, \psi)$, where $\mathbf{\Sigma^I} = \Sigma^I \times \mathbb{R}_{\geq 0}$ is a infinite set of logical-timed inputs, $S, R \subset \mathcal{TR}_{lr}$ are finite sets of reset-logical-timed traces, $E \subset (\mathbf{\Sigma^I} \times \Sigma^O)^* \times \mathbf{\Sigma^I}$ is a suffix-closed set of logical-timed continuation sequences, $\psi : S \cup R \mapsto 2^{\mathbf{\Sigma^I}}$, and $\forall \gamma_r \in S \cup R, e \in E \cup \psi(\gamma) : f(\gamma_r \cdot e) = \mathbf{freq}(\Pi(\gamma_r) \cdot e)$. Specifically,*

- *S is called the set of prefixes, R is called the boundary, and E is called the set of suffixes.*
- *S and R are disjoint, and $S \cup R$ is prefix-closed.*
- *$\forall \gamma_r \in S \cup R, \boldsymbol{i} \in \psi(\gamma_r), \boldsymbol{o} \in \mathbf{\Sigma^O} : \gamma_r \cdot \boldsymbol{io} \in R$ if $\mathbf{freq}(\Pi(\gamma_r) \cdot \boldsymbol{i})(\boldsymbol{o}) > 0$.*

We represent the content of a row by $row(\gamma_r)(e) = f(\gamma_r \cdot e)$ for $\gamma_r \in S \cup R$, $e \in E \cup \psi(\gamma)$. We initially set $S = \{\mathcal{L}(q_0)\}$, where $\mathcal{L}(q_0)$ is the initial output of the SUL, $E = \{(i, 0) | \Sigma^I\}$, and $\psi(\mathcal{L}(q_0)) = \{(i, 0) | \Sigma^I\}$.

Since can not directly determine equivalence of rows, we apply Hoeffding bounds [11] to determine whether two logical-timed test sequences produce statistically different distributions. Put differently, for logical-timed test sequences λ_1, λ_2, we approximate $P_{lr}(\boldsymbol{\pi}(\lambda_1)) \neq P_{lr}(\boldsymbol{\pi}(\lambda_2))$ by checking whether $\mathbf{freq}(\lambda_1)$ and $\mathbf{freq}(\lambda_2)$ have been sampled from different distributions. Thus, we say λ_1, λ_2 produce statistically $different$ distributions, denoted $diff(\lambda_1, \lambda_2)$, iff one of the following conditions holds:

1) $\mathbf{cq}(\lambda_1) \wedge \mathbf{cq}(\lambda_1) \wedge (n_1 > 0 \oplus n_2 > 0)$, or
2) $n_1 > 0 \wedge n_2 > 0 \wedge \exists \boldsymbol{o} \in \mathbf{\Sigma^O}$:

$$\left| \frac{fq_1(\boldsymbol{o})}{n_1} - \frac{fq_2(\boldsymbol{o})}{n_2} \right| > (\frac{1}{\sqrt{n_1}} + \frac{1}{\sqrt{n_2}})\sqrt{\frac{1}{2}\ln\frac{2}{\alpha}} \quad (3)$$

where $fq_k = \mathbf{freq}(\lambda_k)$, $n_k = \sum_{\boldsymbol{o} \in \mathbf{\Sigma^O}} fq_k(\boldsymbol{o})$ for $k = 1, 2$, and α specifies the confidence level $(1 - \alpha)^2$.

We say $f(\gamma_r \cdot e)$ and $f(\gamma'_r \cdot e')$ are compatible, denoted $f(\gamma_r \cdot e) \approx f(\gamma'_r \cdot e')$ if $\neg diff(\Pi(\gamma_r) \cdot e, \Pi(\gamma'_r) \cdot e')$ for $\gamma_r \in S \cup R$, $e \in E \cup \psi(\gamma)$. We extend this notion to rows, i.e. we say two rows labeled γ_r and γ'_r are compatible, denoted $row(\gamma_r) \approx row(\gamma'_r)$, if all of the following conditions hold: (1) $last(\gamma_r) = last(\gamma'_r)$, (2) $\forall e \in E : row(\gamma_r)(e) \approx row(\gamma'_r)(e)$, and (3) $\forall \boldsymbol{i} \in \psi(\gamma_r) \cap \psi(\gamma'_r) : row(\gamma_r)(\boldsymbol{i}) \approx row(\gamma'_r)(\boldsymbol{i})$.

The hypothesis generated from a timed observation table $\mathcal{T}$ is well-formed if $\mathcal{T}$ is $prepared$, which satisfies all of the following properties:

- $reduced$: $\forall s, s' \in S \Rightarrow row(s) \not\approx row(s')$
- $closed$: $\forall r \in R, \exists s \in S : row(s) \approx row(r)$.
- $consistent$: for all compatible pairs of rows labeled $\gamma_r, \gamma'_r \in S \cup R$, the logical-timed input $\boldsymbol{i} \in \psi(\gamma_r) \cap \psi(\gamma'_r)$, and output $o \in \Sigma^O$, we have $row(\gamma_r \cdot \boldsymbol{i} \cdot (\_, o)) \approx row(\gamma'_r \cdot \boldsymbol{i} \cdot (\_, o))$ if $\gamma_r \cdot \boldsymbol{i} \cdot (\_, o), \gamma'_r \cdot \boldsymbol{i} \cdot (\_, o) \in S \cup R$.
- $evidence\text{-}closed$: $\forall s \cdot e \in S \cdot E : \gamma_r \in S \cup R, \boldsymbol{i} \in \psi(\gamma_r)$, where $\gamma_r \cdot \boldsymbol{i} \ll \boldsymbol{\pi}(\Pi(s) \cdot e)$ is the longest prefix such that $\gamma_r \in \mathcal{S}$.

We apply repeatedly following operations to get a prepared table $\mathcal{T}$. We move $r \in R$ to S and update $\psi(r) = \psi(r) \cup \{(i, 0) | i \in \Sigma^I\}$ as noted in [7], if r breaks the closedness. Notably, we add column $\boldsymbol{i} \in \psi(r)$ to E if $row(r)(\boldsymbol{i})$ makes $row(r)$ different from all of the rows in S. As for consistent, if there exist $\gamma_r, \gamma_r \in S \cup R$ such that $row(\gamma_r) \approx row(\gamma'_r)$ but $row(\gamma_r \cdot \boldsymbol{i} \cdot (\_, o)) \not\approx row(\gamma'_r \cdot \boldsymbol{i} \cdot (\_, o))$, we add suffix $\boldsymbol{i} \cdot o \cdot e$ to E, where $e \in E$ such that $f(\gamma_r \cdot \boldsymbol{i} \cdot (\_, o) \cdot e) \not\approx f(\gamma'_r \cdot \boldsymbol{i} \cdot (\_, o) \cdot e)$. To make table $\mathcal{T}$ evidence-closed, we add all the prefixes $\gamma_r \cdot \boldsymbol{i} \in prefixes(\boldsymbol{\pi}(\Pi(s) \cdot e))$ such that $\gamma_r \in \mathcal{S}$ to the table $\mathcal{T}$, i.e. add γ_r to R and update $\psi(\gamma_r) = \psi(\gamma_r) \cup \{\boldsymbol{i}\}$.

However, compatibility is not transitive in general, that is, a row in R may be compatible with multiple rows in S. Thus we create compatible classes for partitioning R as noted in [5], where every trace $\gamma_r \in S \cup R$ in compatible class $cg(s)$ are compatible with its representative $s \in S$. We use $rep(\gamma_r) = s$ to denote s is the representative of trace γ_r, which has the largest rank among S, where rank is before-defined.

2) Hypothesis generation: We build a hypothesis $\mathcal{H}$ from a prepared timed observation table $\mathcal{T}$, where table $\mathcal{T}$ is used to construct an $intermediary\ automaton$ $\mathcal{M}$, then $\mathcal{M}$ is transformed to hypothesis $\mathcal{H}$ via partition function [7].

We define an intermediary automata is a tuple $\mathcal{M} = (Q_\mathcal{M}, \Sigma^I_\mathcal{M}, \Sigma^O_\mathcal{M}, q^0_\mathcal{M}, prob_\mathcal{M}, \mathcal{L}_\mathcal{M})$, where $Q_\mathcal{M}$ is a finite set of states, $\Sigma^I_\mathcal{M} \subset \mathbf{\Sigma^I}$ is a finite set of logical-timed inputs, $\Sigma^O_\mathcal{M}$ is a finite set of outputs, $q^0_\mathcal{M} \in Q_\mathcal{M}$ is the initial state, $prob_\mathcal{M} \subseteq Q_\mathcal{M} \times \Sigma^I_\mathcal{M} \times Dist(\mathbb{B} \times Q_\mathcal{M})$ is the transition relations, and $\mathcal{L}_\mathcal{M} : Q_\mathcal{M} \mapsto \Sigma^O_\mathcal{M}$ is a labelling function.

We build an intermediary automata $\mathcal{M} = (Q_\mathcal{M}, \Sigma^I_\mathcal{M}, \Sigma^O \cup \{undef\}, q^0_\mathcal{M}, prob_\mathcal{M}, \mathcal{L}_\mathcal{M})$ from a prepared timed observation table $\mathcal{T} = (\mathbf{\Sigma^I}, S, R, E, f, \psi)$, as follows:

- $Q_\mathcal{M} = \{\langle last(s), row(s)\rangle | s \in S\} \cup \{q_{ud}\}$,
  - for $q = \langle o, row(s)\rangle \in Q_\mathcal{M} \setminus \{q_{ud}\} : \mathcal{L}_\mathcal{M}(q) = o$
  - for $q_{ud} : \mathcal{L}_\mathcal{M}(q_{ud}) = undef$
- $q^0_\mathcal{M} = \langle \mathcal{L}(q_0), row(\mathcal{L}(q_0))\rangle$
- $\Sigma^I_\mathcal{M} = \{\boldsymbol{i} \in \psi(\gamma_r) | \gamma_r \in S \cup R\}$
- for $\gamma_r \in S \cup R, (i,\mu) \in \psi(\gamma_r)$: let $s = rep(\gamma_r)$, $q = \langle last(s), row(s)\rangle$, $n = \sum_{\boldsymbol{o} \in \mathbf{\Sigma^O}} \mathbf{freq}(\Pi(\gamma_r) \cdot (i,\mu))(\boldsymbol{o})$, and $repf(q,(i,\mu)) = \lambda$ where $\lambda \in \{\Pi(\gamma'_r) \cdot (i,\mu') | \gamma'_r \in cg(s), (i,\mu') \in \psi(\gamma'_r) : \neg diff(\Pi(\gamma_r) \cdot (i,\mu), \Pi(\gamma'_r) \cdot (i,\mu'))\}$ with the largest rank.
  1) if $\neg\mathbf{cq}(\Pi(\gamma_r) \cdot (i,\mu))$ and $\nexists \gamma'_r \in cg(s) : \mathbf{cq}(\Pi(\gamma'_r) \cdot (i,\mu))$, $prob_\mathcal{M} = prob_\mathcal{M} \cup \{(q,(i,\mu),\eta_{ud})\}$, where $\eta_{ud}(\mathsf{T}, q_{ud}) = 1$
  2) otherwise if $\mathbf{cq}(\Pi(\gamma_r) \cdot (i,\mu))$ and $n > 0$, $prob_\mathcal{M} = prob_\mathcal{M} \cup \{(q,(i,\mu),\eta)\}$, where for all $(b,o) \in \mathbf{\Sigma^O}$: $q' = \langle o, row(rep(\gamma_r \cdot (i,\mu)(b,o)))\rangle$ and $\eta(b,q') = \frac{\mathbf{freq}(repf(q,(i,\mu)))(b,o)}{\sum_{\boldsymbol{o} \in \mathbf{\Sigma^O}} \mathbf{freq}(repf(q,(i,\mu)))(\boldsymbol{o})}$
- $prob_\mathcal{M} = prob_\mathcal{M} \cup \{(q_{ud},(i,0),\eta_{ud}) | i \in \Sigma^I\}$ if q_{ud} is reachable

We create a state q for every $s \in S$, and transitions from state q lead to the representatives of the extensions $\gamma_r \cdot (i,\mu)(b,o)$ for each $\gamma_r \in cg(s)$, where transition probabilities are estimated by the representative frequencies $repf(q,(i,\mu))$. The representative frequencies are used to refer the compatible frequencies to the same distributions. If we have not sufficient information for a logical-timed input, we create a transition to a sink state q_{ud}.

We receive a OPTA $\mathcal{H} = (\Sigma^I, \Sigma^O, Q_\mathcal{H}, q^0_\mathcal{H}, x, prob_\mathcal{H}, \mathcal{L}_\mathcal{H})$ from an intermediary automata $\mathcal{M} = (Q_\mathcal{M}, \Sigma^I_\mathcal{M}, \Sigma^O \cup \{undef\}, q^0_\mathcal{M}, prob_\mathcal{M}, \mathcal{L}_\mathcal{M})$ as noted in [7], where $Q_\mathcal{H}$, $q^0_\mathcal{H}$ and $\mathcal{L}_\mathcal{H}$ is the same with $\mathcal{M}$, and $prob_\mathcal{H}$ are transformed from $prob_\mathcal{M}$ one by one: for a group of probabilistic transitions with the same source $q \in Q_\mathcal{H}$ and input $i \in \Sigma^I$, their guards are a partition of $\mathbb{R}_{\geq 0}$ such that each guard g includes the clock valuation μ recorded in the corresponding transition of $\mathcal{M}$, and we denote as $I_{q,i}(\mu') = \mu$ with $\mu' \models g$.

3) Learning algorithm: Algorithm 1 implements the learning algorithm for OPTAs via sampling. First, it initializes a timed observation table $\mathcal{T} = (S, R, E, f, \psi)$ with $S = \{\mathcal{L}(q_0)\}$ where $\mathcal{L}(q_0)$ is the initial output of SUL, $E = \{(i,0) | i \in \Sigma^I\}$, and $\psi(\mathcal{L}(q_0)) = \{(i,0) | i \in \Sigma^I\}$. Then, it performs the main loop until $equivalent = yes$ which represents the final equivalence query terminates without finding a counterexample. The loop starts with a refine query $\mathbf{rq}$ with the set of incomplete logical-timed test sequences of table $\mathcal{T}$, i.e. $get\_incomplete\_seq(\mathcal{T}) = \{\Pi(\gamma_r) \cdot e | \gamma_r \in S \cup R, e \in E \cup \psi(\gamma_r) : \neg\mathbf{cq}(\Pi(\gamma_r) \cdot e)\}$, then updates the extensions, where $Lt(S \cup R) = \{\gamma_r \cdot \boldsymbol{io} | \gamma_r \in S \cup R, \boldsymbol{i} \in \psi(\gamma_r), \boldsymbol{o} \in \mathbf{\Sigma^O} : \mathbf{freq}(\Pi(\gamma_r) \cdot \boldsymbol{i})(\boldsymbol{o}) > 0\}$, and fills table based on multiset $\mathcal{S}$. We adjust table $\mathcal{T}$ to be prepared before hypothesis generation. After building a hypothesis $\mathcal{H}$, we determine whether to perform equivalence queries depending on the uncertainty of table $\mathcal{T}$, which is discussed in detail later. We have $equivalent = no$ if not. Otherwise if finding a counterexample cex, we process cex with normalizing [7] and adding all the prefixes of cex to table $\mathcal{T}$, i.e. $\forall \gamma_r \cdot \boldsymbol{i} \in prefixes(cex) : R = R \cup \{\gamma_r\} \wedge \psi(\gamma_r) = \psi(\gamma_r) \cup \{\boldsymbol{i}\}$. Once the loop stops, we output the hypothesis.

Uncertainties in time observation tables mainly arise from the compatibility checks. Put differently, for $r \in R$, the state reached by r may be ambiguous if $row(r)$ is compatible with multiple rows $s \in S$. We quantify the uncertainty as the ratio r_{unamb} of unambiguous rows and all of the rows, where unambiguous rows are compatible with a row in S. Besides, there exist unknown distributions in hypothesis $\mathcal{H}$ if location q_{ud} is reachable. Thus, to decrease the amount of unnecessary equivalence queries, we perform equivalence queries once location q_{ud} is unreachable and $r_{unamb} \geq m_{unamb}$.

Algorithm 1 Learning of OPTAs by sampling

Input: timed observation table $\mathcal{T} = (\mathbf{\Sigma^I}, S, R, E, f, \psi)$, teacher capable of answering $\mathbf{freq}$, $\boldsymbol{\pi}$, $\mathbf{cq}$, $\mathbf{rq}$ and $\mathbf{eq}$
Output: final hypothesis $\mathcal{H}$

```
T ← initialized(T)
k ← 0
repeat
    k ← k + 1
    rare ← get_incomplete_seq(T)
    S ← S ⊎ rq(rare)
    R ← R ∪ Lt(S ∪ R)
    for all γ_r ∈ S ∪ R, e ∈ E do
        f(γ_r · e) ← freq(Π(γ_r) · e)
    while T is not prepared do
        T ← make_prepared(T)
    H ← build_hypothesis(T)
    if enable_eq(T, H) then
        equivalent, cex ← eq(H, k, ϵ, δ)
        if cex ≠ none then
            T ← process_cex(cex, T)
    else
        equivalent ← no
until equivalent = yes
return H
```

D. Teacher

We describe how to answer the four queries provided by the teacher in the following. We first introduce the SUL operations providing for the teacher, which include: (1) $\mathbf{reset}$: reset the SUL and return $\mathcal{L}(q_0)$; (2) $\mathbf{step}$: perform a delay-timed input, change the current state according to the distribution of the

enabled transition, and return $(b, \mathcal{L}(q'))$, where (b, q') is the selected outcome. Here we assume that the SUL is smart to answer the reset information in transitions.

1) Frequency queries: For a logical-timed test sequence λ, let $\lambda = \gamma \cdot e$ where γ is the longest logical-timed trace with $\exists \gamma_r \in \mathcal{S} : \Pi(\gamma_r) = \gamma$. The frequency query for λ returns pair $(\boldsymbol{\pi}(\lambda), \mathbf{freq}(\lambda))$, where $\boldsymbol{\pi}(\lambda) = \gamma_r \cdot e_r$ in which e_r sets all of the reset indicators of e to T, and $\mathbf{freq}(\lambda)(\boldsymbol{o}) = \mathcal{S}(\gamma_r \cdot e_r \cdot \boldsymbol{o})$ for all $\boldsymbol{o} \in \boldsymbol{\Sigma^O}$.

2) Complete queries: We have $\mathcal{S}(\gamma_r) \leq \mathcal{S}(\gamma_r')$ for all $\gamma_r' \in prefixes(\gamma_r)$, since we add all of the prefixes of samples to $\mathcal{S}$. For a logical-timed test sequence λ, we assume that $\mathbf{cq}(\lambda) = \mathsf{T}$ if $\sum_{\boldsymbol{o} \in \boldsymbol{\Sigma^O}} \mathcal{S}(\boldsymbol{\pi}(\lambda) \cdot \boldsymbol{o}) \geq n_c$, and we have already seen all of the outputs after λ. Thus the extensions of $\lambda \cdot o$ are complete if $\mathcal{S}(\boldsymbol{\pi}(\lambda) \cdot (\_, o)) = 0$. Besides, for $\gamma_r \in \mathcal{S}$, we have the extensions of $\Pi(\gamma_r) \cdot (i, \mu)$ are also complete if μ less than the clock valuation of the state reached by γ_r.

3) Refine queries: The procedure of refine queries we use is similar with [5], in which a prefix tree is the compact representations of the set $rare$ consisting of sequences requiring to refine the knowledge, and new sampled SUL traces generated by directed random walks on the prefix tree. But the prefix tree we use is a tree with edges labeled logical-timed inputs, and nodes labeled outputs. Notably, to represent as a prefix tree, the sequences in $rare$ are extended to logical-timed traces by add a special output $leaf \notin \Sigma^O$ at the end of every sequence. To get the delay-time inputs used to operation $\mathbf{step}$, we record the current clock valuation during sampling. We receive n_{resam} reset-logical-timed traces after performing a refine query.

4) Equivalence queries: Recall that the framework of PAC learning for OPTAs. We face the difficulty that a delay-timed test sequences $\omega \cdot \boldsymbol{i} \in (\Sigma^O \times \boldsymbol{\Sigma^I})^*$ sampled randomly can not directly determine $P(\omega \cdot \boldsymbol{i}) \neq H(\omega \cdot \boldsymbol{i})$. Thus, we apply two strategies to find counterexamples as in Algorithm 2. We first compute the number of samples r_k. Then we sample delay-timed traces from a distribution $\mathcal{D}$, and obtain the corresponding reset-delay-timed traces on the SUL by testing. The sampling mechanism is discussed in detail later. We return unreachable samples on $\mathcal{H}$ as counterexamples, or check for consistency between multiset $\mathcal{S}$ and hypothesis $\mathcal{H}$ respect to these samples according to Theorem 3. To not produce spurious results for $diff$, we perform refine queries for gaining more information about samples, until all of the distributions of samples are unambiguous, where $unamb(tests) = \frac{|\{\lambda \in tests | \mathbf{cq}(\lambda)\}|}{|tests|}$. Under the unambiguous setting, we have $\mathcal{H}$ is $PAC(\epsilon, \delta)$-*correct* if without finding a counterexample.

Theorem 3. *Let $C \subset \{\omega \cdot (i,t) \in (\Sigma^O \times \boldsymbol{\Sigma^I})^* | P(\omega \cdot (i,t) \neq \perp\}$, and multiset $\mathcal{S}$ has n samples of C. Given α_n such that $\sum_n \alpha_n n < \infty$, then with probability one, $\forall \omega \cdot (i,t) \in C : P(\omega \cdot (i,t)) \neq H(\omega \cdot (i,t)) \Leftrightarrow diff(\gamma \cdot (i,\mu), repf(q, (i, I_{q,i}(\mu))))$, where $\gamma = \Gamma(\omega)$, $q \in Q_{\mathcal{H}} : (q^0_{\mathcal{H}}, \nu_0) \xrightarrow{\omega} (q, \nu)$, and $\mu = \nu + t$, except for finitely many n.*

The purpose of a sample mechanism is to sample the SUL traces different from the hypothesis $\mathcal{H}$. Without Hoeffding checking, unreachable sampled traces on $\mathcal{H}$ diverge the SUL and the hypothesis. The sampling distribution $\mathcal{D}$ is that we randomly select delay-timed input $(i,t) \in \boldsymbol{\Sigma^I}$, test iteratively the SUL to observe output, and stop with probability p_{stop}. Besides, we choose some traces to extend randomly. More techniques used to improving sampling efficiency, can be found in [8].

Algorithm 2 Equivalence queries

Input: hypothesis $\mathcal{H}$, the count k of the current equivalence query, error ϵ and confidence δ
Output: Boolean $equivalent$ identifying whether $\mathcal{H}$ passes all tests, and counterexample cex

$tests \leftarrow \{\}$
$r_k \leftarrow \frac{1}{\epsilon}(\ln \frac{1}{\delta} + \ln 2(k+1))$
for $counter \leftarrow 1$ to r_k **do**
  $\omega \leftarrow sample(\mathcal{D})$
  $\omega_r \leftarrow test\_SUL(\omega)$
  $\gamma_r \cdot \boldsymbol{io} \leftarrow \Gamma(\omega_r)$
  if $\nexists q_{\mathcal{H}} \in Q_{\mathcal{H}} : (q^0_{\mathcal{H}}, \nu_0) \xrightarrow{\omega} (q_{\mathcal{H}}, \nu)$ **then**
    $equivalent \leftarrow \mathsf{F}, cex \leftarrow \gamma_r \cdot \boldsymbol{i}$
    return $equivalent, cex$
  $tests \leftarrow tests \cup \{\Pi(\gamma_r) \cdot \boldsymbol{i}\}$
repeat
  $\mathcal{S} \leftarrow \mathcal{S} \uplus \mathbf{rq}(tests)$
  for $\gamma \cdot (i, \mu)$ in $tests$ **do**
    $\omega \leftarrow \omega'$ with $\gamma = \Gamma(\omega')$
    $q \leftarrow q_{\mathcal{H}}$ with $(q^0_{\mathcal{H}}, \nu_0) \xrightarrow{\omega} (q_{\mathcal{H}}, \nu)$
    if $diff(\gamma \cdot (i, \mu), repf(q, (i, I_{q,i}(\mu))))$ **then**
      $equivalent \leftarrow \mathsf{F}, cex \leftarrow \boldsymbol{\pi}(\gamma) \cdot (i, \mu)$
      return $equivalent, ctx$
until $unamb(tests) = 1$
return $yes, none$

Theorem 4. *Algorithm 1 terminates and outputs a model that has an average error less than or equal ϵ with the the probability of at least $1 - \delta$, except for finitely many n.*

IV. Implementation and Experimental Results

In our work, we want to learn a model close to the true model, since equivalence can hardly be achieved. Our experiments aim to measure the distance between the learned models and the true models. More concretely, 2000 test cases generated from true models are calculated first whether executable on the learned models. If executable, we calculate the Kullback-Leibler Divergence (KLD) of the output distributions produced by both models, and record executable test cases are passing. And we compare the probability of the temporal properties for the true model and the learned model, if the true model can perform probabilistic model check by Prism [12]. We implemented our algorithm in JAVA, and performed the experiments on a PC with 16 GB RAM, an Intel Core i7-9750 CPU with 2.6 GHz and running Windows 10.

TABLE I: Results for learning the CSMA/CD protocol

	true model	learned model
# traces	-	25075434
time [s]	-	1104.578
r_{pass}	-	1.0
KLD	-	0.000006
$P_{min}(F^{\leq 1800}(done))$	0.5833	0.5829
$P_{min}(F^{\leq 2200}(done))$	0.9663	0.9663

TABLE II: Results on random examples

Case ID	ϵ, δ	r_{pass}	KLD	# trace	t_{mean}
4_4_3_10	0.01	0.9997	0.2836	2196039	102.047
4_4_3_20	0.01	0.9996	0.3131	2305877	134.190
4_4_3_30	0.01	0.9988	0.1421	2401363	184.100
6_2_3_20	0.01	0.9959	0.1462	1978498	85.787
6_2_4_20	0.01	0.9951	0.1303	1940146	89.791
6_2_5_20	0.01	0.9974	0.1625	1848045	90.774
6_3_4_20	0.01	0.9978	0.0734	2197287	120.542
6_4_4_20	0.01	0.9999	0.0588	4095301	493.074
10_2_7_40	0.1	0.9000	0.0718	1330212	46.045
10_2_7_40	0.01	0.9956	0.0458	3363402	176.929
10_2_7_40	0.001	0.9991	0.0433	1587538	9277.404

A. CSMA/CD Protocol

We consider to learn the sender in IEEE 802.3 CSMA/CD protocol, which can be specified and verified formally within the framework of probabilistic timed automata [13]. In our setting, the corresponding OPTA $\mathcal{P}$ to be learned is configured to have $|Q| = 8$ locations [1] with two WAIT states moved with probability 0.5, $|\Sigma^I| = 5$, and the propagation delay is $26\mu s$, the time to send a data packet is $808\mu s$. For the evaluation of the CSMA/CD protocol, we perform probabilistic model checking for the minimum probability of reaching the goal $done$ within a varying number of time. We set the sampling parameters as $n_{resam} = 100$, $p_{stop} = 0.5$. Let $m_{unamb} = 0.999$ for the enable condition of equivalence queries. As the parameters of PAC, we set the error parameter $\epsilon = 0.001$ and the confident parameter $delta = 0.001$. We set $\alpha = 0.001$ for the compatibility check, and the complete threshold $n_c = 50$.

Table I shows the measurement results for learning the CSMA/CD protocol. The passing ratio achieves 100%, and the absolute difference to the true probabilities is at most 0.0004 in probabilistic model checking.

B. Random experiments

We randomly generated 90 OPTAs in nine groups, with each group labelled $n\_m\_c\_k$, having different numbers of locations, size of inputs, size of outputs, and maximum constant appearing in clock constraints. We set the sampling parameters as $n_{resam} = 200$, $p_{stop} = 0.5$. $m_{unamb} = 0.999$ for the enable condition of equivalence queries, $\alpha = 0.01$ for the compatibility check, and the complete threshold $n_c = 50$. As shown in Table II, the passing ratios at least 99%, and the sum of KLD for the passing test cases are at most 0.5 in all cases. While ϵ and δ are smaller, the passing rate increases.

[1]To ensure the completeness of $\mathcal{P}$, we complete it as noted in [7], i.e. any unspecified behavior in the original model is regarded as transitioning to a sink location

V. Conclusion

In this paper, we present a learning algorithm for one-clock probabilistic timed automata based on the framework of L^* via sampling. The traces sampled from the SUL are used to infer the structure, estimate transition probabilities, and check the compatibility between the target model and the hypothesis generated by learner. We apply the PAC learning framework to our learning algorithm for quantifying the learned model under the black-box setting. We evaluate our learning algorithm on randomly generated examples and the sender of CSMA/CD protocol. Future works include combines our learning algorithm with learning-based verification techniques for case studies on stochastic systems.

References

[1] D. Angluin, "Learning regular sets from queries and counterexamples," *Information and computation*, vol. 75, no. 2, pp. 87–106, 1987.

[2] M. Shahbaz and R. Groz, "Inferring mealy machines," in *FM 2009: Formal Methods: Second World Congress, Eindhoven, The Netherlands, November 2-6, 2009. Proceedings 2*. Springer, 2009, pp. 207–222.

[3] G. Argyros and L. D'Antoni, "The learnability of symbolic automata," in *Computer Aided Verification: 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I 30*. Springer, 2018, pp. 427–445.

[4] S. Drews and L. D'Antoni, "Learning symbolic automata," in *Tools and Algorithms for the Construction and Analysis of Systems: 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I*. Springer, 2017, pp. 173–189.

[5] M. Tappler, B. K. Aichernig, G. Bacci, M. Eichlseder, and K. G. Larsen, "L*-based learning of markov decision processes," in *Formal Methods–The Next 30 Years: Third World Congress, FM 2019, Porto, Portugal, October 7–11, 2019, Proceedings*. Springer, 2019, pp. 651–669.

[6] M. Tappler, E. Muškardin, B. K. Aichernig, and I. Pill, "Active model learning of stochastic reactive systems," in *Software Engineering and Formal Methods: 19th International Conference, SEFM 2021, Virtual Event, December 6–10, 2021, Proceedings*. Springer, 2021, pp. 481–500.

[7] J. An, M. Chen, B. Zhan, N. Zhan, and M. Zhang, "Learning one-clock timed automata," in *Tools and Algorithms for the Construction and Analysis of Systems: 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25–30, 2020, Proceedings, Part I*. Springer, 2020, pp. 444–462.

[8] W. Shen, J. An, B. Zhan, M. Zhang, B. Xue, and N. Zhan, "Pac learning of deterministic one-clock timed automata," in *Formal Methods and Software Engineering: 22nd International Conference on Formal Engineering Methods, ICFEM 2020, Singapore, Singapore, March 1–3, 2021, Proceedings*. Springer, 2020, pp. 129–146.

[9] H. Jensen, "Model checking probabilistic real time systems," in *Proc. 7th Nordic Workshop on Programming Theory*. Citeseer, 1996, pp. 247–261.

[10] R. Alur and D. Dill, "The theory of timed automata," in *Real-Time: Theory in Practice: REX Workshop Mook, The Netherlands, June 3–7, 1991 Proceedings*. Springer, 1992, pp. 45–73.

[11] R. C. Carrasco and J. Oncina, "Learning deterministic regular grammars from stochastic samples in polynomial time," *RAIRO-Theoretical Informatics and Applications*, vol. 33, no. 1, pp. 1–19, 1999.

[12] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," in *Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings 23*. Springer, 2011, pp. 585–591.

[13] M. Kwiatkowska, G. Norman, J. Sproston, and F. Wang, "Symbolic model checking for probabilistic timed automata," *Information and Computation*, vol. 205, no. 7, pp. 1027–1077, 2007.