Active Learning of Deterministic Timed Automata with Myhill-Nerode Style Characterization

Masaki Waga(✉)
Graduate School of Informatics, Kyoto University, Kyoto, Japan
mwaga@fos.kuis.kyoto-u.ac.jp

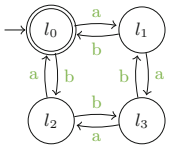
Abstract. We present an algorithm to learn a deterministic timed automaton (DTA) via membership and equivalence queries. Our algorithm is an extension of the L* algorithm with a Myhill-Nerode style characterization of recognizable timed languages, which is the class of timed languages recognizable by DTAs. We first characterize the recognizable timed languages with a Nerode-style congruence. Using it, we give an algorithm with a smart teacher answering *symbolic* membership queries in addition to membership and equivalence queries. With a symbolic membership query, one can ask the membership of a certain set of timed words at one time. We prove that for any recognizable timed language, our learning algorithm returns a DTA recognizing it. We show how to answer a symbolic membership query with finitely many membership queries. We also show that our learning algorithm requires a polynomial number of queries with a smart teacher and an exponential number of queries with a normal teacher. We applied our algorithm to various benchmarks and confirmed its effectiveness with a normal teacher.

Keywords: timed automata · active automata learning · recognizable timed languages · L* algorithm · observation table

1 Introduction

Active automata learning is a class of methods to infer an automaton recognizing an unknown target language $\mathcal{L}_{\mathrm{tgt}} \subseteq \Sigma^*$ through finitely many queries to a teacher. The L* algorithm [8], the best-known active DFA learning algorithm, infers the minimum DFA recognizing $\mathcal{L}_{\mathrm{tgt}}$ using *membership* and *equivalence* queries. In a membership query, the learner asks if a word $w \in \Sigma^*$ is in the target language $\mathcal{L}_{\mathrm{tgt}}$, which is used to obtain enough information to construct a hypothesis DFA $\mathcal{A}_{\mathrm{hyp}}$. Using an equivalence query, the learner checks if the hypothesis $\mathcal{A}_{\mathrm{hyp}}$ recognizes the target language $\mathcal{L}_{\mathrm{tgt}}$. If $\mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) \neq \mathcal{L}_{\mathrm{tgt}}$, the teacher returns a counterexample $cex \in \mathcal{L}_{\mathrm{tgt}} \triangle \mathcal{L}(\mathcal{A}_{\mathrm{hyp}})$ differentiating the target language and the current hypothesis. The learner uses cex to update $\mathcal{A}_{\mathrm{hyp}}$ to classify cex correctly. Such a learning algorithm has been combined with formal verification, e. g., for testing [22,24,26,28] and controller synthesis [31].

C. Enea and A. Lal (Eds.): CAV 2023, LNCS 13964, pp. 3–26, 2023.
https://doi.org/10.1007/978-3-031-37706-8_1



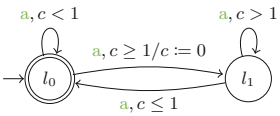
(a) A DFA $\mathcal{A}$

	ε	a
ε	⊤	⊥
a	⊥	⊥
⋮		
aa	⊥	⊥

add b to S

	ε	a	b
ε	⊤	⊥	⊥
a	⊥	⊥	⊤
⋮			
aa	⊥	⊥	⊥

(b) Intermediate observation tables for learning $\mathcal{A}$. a and aa are deemed equivalent with extensions $S = \{\varepsilon, \mathrm{a}\}$ but distinguished with $S = \{\varepsilon, \mathrm{a}, \mathrm{b}\}$.



(c) A DTA $\mathcal{A}'$ with one clock variable c

	$\{\tau_0' \mid \tau_0' = 0\}$	$\{\tau_0'\mathrm{a} \mid \tau_0' \in (0,1)\}$
$\{\tau_0 \mid \tau_0 = 0\}$	⊤	⊤
$\{\tau_0 \mid \tau_0 \in (0,1)\}$	⊤	$\tau_0 + \tau_0' \in (0,1)$
⋮		
$\{\tau_0 \mathrm{a} \tau_1 \mid \tau_0 \in (0,1), \tau_1 \in (0,1), \tau_0 + \tau_1 \in (0,1)\} (= p_1)$	⊤	$\tau_0 + \tau_1 + \tau_0' \in (0,1)$
⋮		
$\{\tau_0 \mathrm{a} \tau_1 \mathrm{a} \tau_2 \mid \tau_0 \in (1,2), \tau_1 \in (0,1), \tau_2 \in (0,1), \tau_1 + \tau_2 \in (0,1)\} (= p_2)$	⊤	$\tau_1 + \tau_2 + \tau_0' \in (0,1)$
⋮		

(d) Timed observation table for learning $\mathcal{A}'$. Each cell is indexed by a pair $(p, s) \in P \times S$ of elementary languages. The cell indexed by (p, s) shows a constraint Λ such that $w \in p \cdot s$ satisfies $w \in \mathcal{L}_{\mathrm{tgt}}$ if and only if Λ holds. Elementary languages p_1 and p_2 are deemed equivalent with the equation $\tau_0^1 + \tau_1^1 = \tau_1^2 + \tau_2^2$, where τ_i^j represents τ_i in p_j.

Fig. 1. Illustration of observation tables in the L* algorithm for DFA learning (Fig. 1b) and our algorithm for DTA learning (Fig. 1d)

Most of the DFA learning algorithms rely on the characterization of regular languages by *Nerode's congruence*. For a language $\mathcal{L}$, words p and p' are equivalent if for any extension s, $p \cdot s \in \mathcal{L}$ if and only if $p' \cdot s \in \mathcal{L}$. It is well known that if $\mathcal{L}$ is regular, such an equivalence relation has finite classes, corresponding to the locations of the minimum DFA recognizing $\mathcal{L}$ (known as *Myhill-Nerode theorem*; see, e. g., [18]). Moreover, for any regular language $\mathcal{L}$, there are finite extensions S such that p and p' are equivalent if and only if for any $s \in S$, $p \cdot s \in \mathcal{L}$ if and only if $p' \cdot s \in \mathcal{L}$. Therefore, one can learn the minimum DFA by learning such finite extensions S and the finite classes induced by Nerode's congruence.

The L* algorithm learns the minimum DFA recognizing the target language $\mathcal{L}_{\mathrm{tgt}}$ using a 2-dimensional array called an *observation table*. Figure 1b illustrates observation tables. The rows and columns of an observation table are indexed with finite sets of words P and S, respectively. Each cell indexed by $(p, s) \in P \times S$ shows if $p \cdot s \in \mathcal{L}_{\mathrm{tgt}}$. The column indices S are the current extensions approximating Nerode's congruence. The L* algorithm increases P and S until: 1) the equivalence relation defined by S converges to Nerode's congruence and 2) P covers all the classes induced by the congruence. The equivalence between $p, p' \in P$ under S can be checked by comparing the rows in the observation table indexed with p and p'. For example, Fig. 1b shows that a and aa are deemed equivalent with extensions $S = \{\varepsilon, \mathrm{a}\}$ but distinguished by adding b to S. The refinement of P and S is driven by certain conditions to validate the DFA construction and by addressing the counterexample obtained by an equivalence query.

Timed words are extensions of conventional words with real-valued dwell time between events. *Timed* languages, sets of timed words, are widely used to formalize real-time systems and their properties, e. g., for formal verification. Among various formalisms representing timed languages, *timed automata (TAs)* [4] is one of the widely used formalisms. A TA is an extension of an NFA with finitely many clock variables to represent timing constraints. Figure 1c shows an example.

Despite its practical relevance, learning algorithms for TAs are only available for limited subclasses of TAs, e. g., real-time automata [6,7], event-recording automata [15,16], event-recording automata with unobservable reset [17], and one-clock deterministic TAs [5,30]. Timing constraints representable by these classes are limited, e. g., by restricting the number of clock variables or by restricting the edges where a clock variable can be reset. Such restriction simplifies the inference of timing constraints in learning algorithms.

Contributions. In this paper, we propose an active learning algorithm for *deterministic* TAs (DTAs). The languages recognizable by DTAs are called *recognizable timed languages* [21]. Our strategy is as follows: first, we develop a Myhill-Nerode style characterization of recognizable timed languages; then, we extend the L* algorithm for recognizable timed languages using the similarity of the Myhill-Nerode style characterization.

Due to the continuity of dwell time in timed words, it is hard to characterize recognizable timed languages by a Nerode-style congruence between timed words. For example, for the DTA in Fig. 1c, for any $\tau, \tau' \in [0,1)$ satisfying $\tau < \tau'$, $(1-\tau')\mathrm{a}$ distinguishes τ and τ' because $\tau(1-\tau')\mathrm{a}$ leads to l_0 while $\tau(1-\tau)\mathrm{a}$ leads to l_1. Therefore, such a congruence can make *infinitely* many classes.

Instead, we define a Nerode-style congruence between sets of timed words called *elementary languages* [21]. An elementary language is a timed language defined by a word with a conjunction of inequalities constraining the time difference between events. We also use an equality constraint, which we call, a *renaming equation* to define the congruence. Intuitively, a renaming equation bridges the time differences in an elementary language and the clock variables in a TA. We note that there can be multiple renaming equations showing the equivalence of two elementary languages.

Example 1. Let p_1 and p_2 be elementary languages $p_1 = \{\tau_0^1 \mathrm{a} \tau_1^1 \mid \tau_0^1 \in (0,1), \tau_1^1 \in (0,1), \tau_0^1 + \tau_1^1 \in (0,1)\}$ and $p_2 = \{\tau_0^2 \mathrm{a} \tau_1^2 \mathrm{a} \tau_2^2 \mid \tau_0^2 \in (1,2), \tau_1^2 \in (0,1), \tau_2^2 \in (0,1), \tau_1^2 + \tau_2^2 \in (0,1)\}$. For the DTA in Fig. 1c, p_1 and p_2 are equivalent with the renaming equation $\tau_0^1 + \tau_1^1 = \tau_1^2 + \tau_2^2$ because for any $w_1 = \tau_0^1 \mathrm{a} \tau_1^1 \in p_1$ and $w_2 = \tau_0^2 \mathrm{a} \tau_1^2 \mathrm{a} \tau_2^2 \in p_2$: 1) we reach l_0 after reading either of w_1 and w_2 and 2) the values of c after reading w_1 and w_2 are $\tau_0^1 + \tau_1^1$ and $\tau_1^2 + \tau_2^2$, respectively.

We characterize recognizable timed languages by the finiteness of the equivalence classes defined by the above congruence. We also show that for any recognizable timed language, there is a finite set S of elementary languages such that the equivalence of any prefixes can be checked by the extensions S.

By using the above congruence, we extend the L* algorithm for DTAs. The high-level idea is the same as the original L* algorithm: 1) the learner makes membership queries to obtain enough information to construct a hypothesis DTA $\mathcal{A}_{\mathrm{hyp}}$ and 2) the learner makes an equivalence query to check if $\mathcal{A}_{\mathrm{hyp}}$ recognizes the target language. The largest difference is in the cells of an observation table. Since the concatenation $p \cdot s$ of an index pair $(p, s) \in P \times S$ is not a timed word but a set of timed words, its membership is not defined as a Boolean value. Instead, we introduce the notion of *symbolic* membership and use it as the value of each cell of the *timed* observation table. Intuitively, the symbolic membership is the constraint representing the subset of $p \cdot s$ included by $\mathcal{L}_{\mathrm{tgt}}$. Such a constraint can be constructed by finitely many (non-symbolic) membership queries.

Example 2. Figure 1d illustrates a *timed* observation table. The equivalence between $p_1, p_2 \in P$ under S can be checked by comparing the cells in the rows indexed with p_1 and p_2 with renaming equations. For the cells in rows indexed by p_1 and p_2, their constraints are the same by replacing $\tau_0 + \tau_1$ with $\tau_1 + \tau_2$ and vice versa. Thus, p_1 and p_2 are equivalent with the current extensions S.

Once the learner obtains enough information, it constructs a DTA via the monoid-based representation of recognizable timed languages [21]. We show that for any recognizable timed language, our algorithm terminates and returns a DTA recognizing it. We also show that the number of the necessary queries is polynomial to the size of the equivalence class defined by the Nerode-style congruence if symbolic membership queries are allowed and, otherwise, exponential to it. Moreover, if symbolic membership queries are not allowed, the number of the necessary queries is at most doubly exponential to the number of the clock variable of a DTA recognizing the target language and singly exponential to the number of locations of a DTA recognizing the target language. This worst-case complexity is the same as the one-clock DTA learning algorithm in [30].

We implemented our DTA learning algorithm in a prototype library LearnTA. Our experiment results show that it is efficient enough for some benchmarks taken from practical applications, e. g., the FDDI protocol. This suggests the practical relevance of our algorithm.

The following summarizes our contribution.

- We characterize recognizable timed languages by a Nerode-style congruence.
- Using the above characterization, we give an active DTA learning algorithm.
- Our experiment results suggest its practical relevance.

Related Work. Among various characterization of timed languages [4,10–13,21], the characterization by *recognizability* [21] is closest to our Myhill-Nerode-style characterization. Both of them use finite sets of elementary languages for characterization. Their main difference is that [21] proposes a formalism to define a timed language by relating prefixes by a morphism, whereas we propose a technical gadget to define an equivalence relation over timed words with respect to suffixes using symbolic membership. This difference makes our definition suitable for an L*-style algorithm, where the original L* algorithm is based on Nerode's

congruence, which defines an equivalence relation over words with respect to suffixes using conventional membership.

As we have discussed so far, active TA learning [5,15–17,30] has been studied mostly for limited subclasses of TAs, where the number of the clock variables or the clock variables reset at each edge is fixed. In contrast, our algorithm infers both of the above information. Another difference is in the technical strategy. Most of the existing algorithms are related to the active learning of *symbolic automata* [9,14], enhancing the languages with clock valuations. In contrast, we take a more semantic approach via the Nerode-style congruence.

Another recent direction is to use a *genetic algorithm* to infer TAs in passive [27] or active [3] learning. This differs from our learning algorithm based on a formal characterization of timed languages. Moreover, these algorithms may not converge to the correct automaton due to a genetic algorithm.

2 Preliminaries

For a set X, its powerset is denoted by $\mathcal{P}(X)$. We denote the empty sequence by ε. For sets X, Y, we denote their symmetric difference by $X \triangle Y = \{x \mid x \in X \wedge x \notin Y\} \cup \{y \mid y \in Y \wedge y \notin X\}$.

2.1 Timed Words and Timed Automata

Definition 3 (timed word). *For a finite alphabet Σ, a* timed word w *is an alternating sequence $\tau_0 a_1 \tau_1 a_2 \dots a_n \tau_n$ of Σ and $\mathbb{R}_{\geq 0}$. The set of timed words over Σ is denoted by $\mathcal{T}(\Sigma)$. A* timed language $\mathcal{L} \subseteq \mathcal{T}(\Sigma)$ *is a set of timed words.*

For timed words $w = \tau_0 a_1 \tau_1 a_2 \dots a_n \tau_n$ and $w' = \tau'_0 a'_1 \tau'_1 a'_2 \dots a'_{n'} \tau'_{n'}$, their concatenation $w \cdot w'$ is $w \cdot w' = \tau_0 a_1 \tau_1 a_2 \dots a_n (\tau_n + \tau'_0) a'_1 \tau'_1 a'_2 \dots a'_{n'} \tau'_{n'}$. The concatenation is naturally extended to timed languages: for a timed word w and timed languages $\mathcal{L}, \mathcal{L}'$, we let $w \cdot \mathcal{L} = \{w \cdot w_{\mathcal{L}} \mid w_{\mathcal{L}} \in \mathcal{L}\}$, $\mathcal{L} \cdot w = \{w_{\mathcal{L}} \cdot w \mid w_{\mathcal{L}} \in \mathcal{L}\}$, and $\mathcal{L} \cdot \mathcal{L}' = \{w_{\mathcal{L}} \cdot w_{\mathcal{L}'} \mid w_{\mathcal{L}} \in \mathcal{L}, w_{\mathcal{L}'} \in \mathcal{L}'\}$. For timed words w and w', w is a *prefix* of w' if there is a timed word w'' satisfying $w \cdot w'' = w'$. A timed language $\mathcal{L}$ is *prefix-closed* if for any $w \in \mathcal{L}$, $\mathcal{L}$ contains all the prefixes of w.

For a finite set C of clock variables, a *clock valuation* is a function $\nu \in (\mathbb{R}_{\geq 0})^C$. We let $\mathbf{0}_C$ be the clock valuation satisfying $\mathbf{0}_C(c) = 0$ for any $c \in C$. For $\nu \in (\mathbb{R}_{\geq 0})^C$ and $\tau \in \mathbb{R}_{\geq 0}$, we let $\nu + \tau$ be the clock valuation satisfying $(\nu + \tau)(c) = \nu(c) + \tau$ for any $c \in C$. For $\nu \in (\mathbb{R}_{\geq 0})^C$ and $\rho \subseteq C$, we let $\nu[\rho := 0]$ be the clock valuation satisfying $(\nu[\rho := 0])(x) = 0$ for $c \in \rho$ and $(\nu[\rho := 0])(c) = \nu(c)$ for $c \notin \rho$. We let $\mathcal{G}_C$ be the set of constraints defined by a finite conjunction of inequalities $c \bowtie d$, where $c \in C$, $d \in \mathbb{N}$, and $\bowtie \in \{>, \geq, \leq, <\}$. We let $\mathcal{C}_C$ be the set of constraints defined by a finite conjunction of inequalities $c \bowtie d$ or $c - c' \bowtie d$, where $c, c' \in C$, $d \in \mathbb{N}$, and $\bowtie \in \{>, \geq, \leq, <\}$. We denote $\bigwedge \emptyset$ by $\top$. For $\nu \in (\mathbb{R}_{\geq 0})^C$ and $\varphi \in \mathcal{C}_C \cup \mathcal{G}_C$, we denote $\nu \models \varphi$ if ν satisfies φ.

Definition 4 (timed automaton). *A* timed automaton *(TA) is a 7-tuple* $(\Sigma, L, l_0, C, I, \Delta, F)$*, where:* Σ *is the finite alphabet,* L *is the finite set of locations,* $l_0 \in L$ *is the initial location,* C *is the finite set of clock variables,* $I \colon L \to \mathcal{C}_C$ *is the invariant of each location,* $\Delta \subseteq L \times \mathcal{G}_C \times (\Sigma \cup \{\varepsilon\}) \times \mathcal{P}(C) \times L$ *is the set of edges, and* $F \subseteq L$ *is the accepting locations.*

A TA is *deterministic* if 1) for any $a \in \Sigma$ and $(l, g, a, \rho, l'), (l, g', a, \rho', l'') \in \Delta$, $g \wedge g'$ is unsatisfiable, or 2) for any $(l, g, \varepsilon, \rho, l') \in \Delta$, $g \wedge I(l)$ is at most a singleton. Figure 1c shows a deterministic TA (DTA).

The semantics of a TA is defined by a *timed transition system (TTS).*

Definition 5 (semantics of TAs). *For a TA* $\mathcal{A} = (\Sigma, L, l_0, C, I, \Delta, F)$*, the* timed transition system (TTS) *is a 4-tuple* $\mathcal{S} = (Q, q_0, Q_F, \to)$*, where:* $Q = L \times (\mathbb{R}_{\geq 0})^C$ *is the set of* (concrete) states*,* $q_0 = (l_0, \mathbf{0}_C)$ *is the* initial state*,* $Q_F = \{(l, \nu) \in Q \mid l \in F\}$ *is the set of* accepting states*, and* $\to \subseteq Q \times Q$ *is the* transition relation *consisting of the following*[1]*.*

- *For each* $(l, \nu) \in Q$ *and* $\tau \in \mathbb{R}_{>0}$*, we have* $(l, \nu) \xrightarrow{\tau} (l, \nu + \tau)$ *if* $\nu + \tau' \models I(l)$ *holds for each* $\tau' \in [0, \tau)$*.*
- *For each* $(l, \nu), (l', \nu') \in Q$*,* $a \in \Sigma$*, and* $(l, g, a, \rho, l') \in \Delta$*, we have* $(l, \nu) \xrightarrow{a} (l', \nu')$ *if we have* $\nu \models g$ *and* $\nu' = \nu[\rho := 0]$*.*
- *For each* $(l, \nu), (l', \nu') \in Q$*,* $\tau \in \mathbb{R}_{>0}$*, and* $(l, g, \varepsilon, \rho, l') \in \Delta$*, we have* $(l, \nu) \xrightarrow{\varepsilon, \tau} (l', \nu' + \tau)$ *if we have* $\nu \models g$*,* $\nu' = \nu[\rho := 0]$*, and* $\forall \tau' \in [0, \tau).\, \nu' + \tau' \models I(l')$*.*

A *run* of a TA $\mathcal{A}$ is an alternating sequence $q_0, \to_1, q_1, \ldots, \to_n, q_n$ of $q_i \in Q$ and $\to_i \in \to$ satisfying $q_{i-1} \to_i q_i$ for any $i \in \{1, 2, \ldots, n\}$. A run $q_0, \to_1, q_1, \ldots, \to_n, q_n$ is accepting if $q_n \in Q_F$. Given such a run, the associated timed word is the concatenation of the labels of the transitions. The timed *language* $\mathcal{L}(\mathcal{A})$ of a TA $\mathcal{A}$ is the set of timed words associated with some accepting run of $\mathcal{A}$.

2.2 Recognizable Timed Languages

Here, we review the *recognizability* [21] of timed languages.

Definition 6 (timed condition). *For a set* $\mathbb{T} = \{\tau_0, \tau_1, \ldots, \tau_n\}$ *of ordered variables, a* timed condition Λ *is a finite conjunction of inequalities* $\mathbb{T}_{i,j} \bowtie d$*, where* $\mathbb{T}_{i,j} = \sum_{k=i}^{j} \tau_k$*,* $\bowtie \in \{>, \geq, \leq, <\}$*, and* $d \in \mathbb{N}$*.*

A timed condition Λ is *simple*[2] if for each $\mathbb{T}_{i,j}$, Λ contains $d < \mathbb{T}_{i,j} < d+1$ or $d \leq \mathbb{T}_{i,j} \wedge \mathbb{T}_{i,j} \leq d$ for some $d \in \mathbb{N}$. A timed condition Λ is *canonical* if we cannot strengthen or add any inequality $\mathbb{T}_{i,j} \bowtie d$ to Λ without changing its semantics.

[1] We use $\xrightarrow{\varepsilon, \tau}$ to avoid the discussion with an arbitrary small dwell time in [21].
[2] The notion of simplicity is taken from [15].

Definition 7 (elementary language). *A timed language $\mathcal{L}$ is* elementary *if there are $u = a_1, a_2, \ldots, a_n \in \Sigma^*$ and a timed condition Λ over $\{\tau_0, \tau_1, \ldots, \tau_n\}$ satisfying $\mathcal{L} = \{\tau_0 a_1 \tau_1 a_2 \ldots a_n \tau_n \mid \tau_0, \tau_1, \ldots, \tau_n \models \Lambda\}$, and the set of valuations of $\{\tau_0, \tau_1, \ldots, \tau_n\}$ defined by Λ is bounded. We denote such $\mathcal{L}$ by (u, Λ). We let $\mathcal{E}(\Sigma)$ be the set of elementary languages over Σ.*

For $p, p' \in \mathcal{E}(\Sigma)$, p is a *prefix* of p' if for any $w' \in p'$, there is a prefix $w \in p$ of w', and for any $w \in p$, there is $w' \in p'$ such that w is a prefix of w'. For any elementary language, the number of its prefixes is finite. For a set of elementary languages, *prefix-closedness* is defined based on the above definition of prefixes.

An elementary language (u, Λ) is *simple* if there is a simple and canonical timed condition Λ' satisfying $(u, \Lambda) = (u, \Lambda')$. We let $\mathcal{SE}(\Sigma)$ be the set of simple elementary languages over Σ. Without loss of generality, we assume that for any $(u, \Lambda) \in \mathcal{SE}(\Sigma)$, Λ is simple and canonical. We remark that any DTA cannot distinguish timed words in a simple elementary language, i. e., for any $p \in \mathcal{SE}(\Sigma)$ and a DTA $\mathcal{A}$, we have either $p \subseteq \mathcal{L}(\mathcal{A})$ or $p \cap \mathcal{L}(\mathcal{A}) = \emptyset$. We can decide if $p \subseteq \mathcal{L}(\mathcal{A})$ or $p \cap \mathcal{L}(\mathcal{A}) = \emptyset$ by taking some $w \in p$ and checking if $w \in \mathcal{L}(\mathcal{A})$.

Definition 8 (immediate exterior). *Let $\mathcal{L} = (u, \Lambda)$ be an elementary language. For $a \in \Sigma$, the* discrete immediate exterior $\mathrm{ext}^a(\mathcal{L})$ *of $\mathcal{L}$ is $\mathrm{ext}^a(\mathcal{L}) = (u \cdot a, \Lambda \cup \{\tau_{|u|+1} = 0\})$. The* continuous immediate exterior $\mathrm{ext}^t(\mathcal{L})$ *of $\mathcal{L}$ is $\mathrm{ext}^t(\mathcal{L}) = (u, \Lambda^t)$, where Λ^t is the timed condition such that each inequality $\mathbb{T}_{i,|u|} = d$ in Λ is replaced with $\mathbb{T}_{i,|u|} > d$ if such an inequality exists, and otherwise, the inequality $\mathbb{T}_{i,|u|} < d$ in Λ with the smallest index i is replaced with $\mathbb{T}_{i,|u|} = d$. The* immediate exterior *of $\mathcal{L}$ is $\mathrm{ext}(\mathcal{L}) = \bigcup_{a \in \Sigma} \mathrm{ext}^a(\mathcal{L}) \cup \mathrm{ext}^t(\mathcal{L})$.*

Example 9. For a word $u = \mathrm{a} \cdot \mathrm{a}$ and a timed condition $\Lambda = \{\mathbb{T}_{0,0} \in (1,2) \wedge \mathbb{T}_{0,1} \in (1,2) \wedge \mathbb{T}_{0,2} \in (1,2) \wedge \mathbb{T}_{1,2} \in (0,1) \wedge \mathbb{T}_{2,2} = 0\}$, we have $1.3 \cdot \mathrm{a} \cdot 0.5 \cdot \mathrm{a} \cdot 0 \in (u, \Lambda)$ and $1.7 \cdot \mathrm{a} \cdot 0.5 \cdot \mathrm{a} \cdot 0 \notin (u, \Lambda)$. The discrete and continuous immediate exteriors of (u, Λ) are $\mathrm{ext}^{\mathrm{a}}((u, \Lambda)) = (u \cdot \mathrm{a}, \Lambda^{\mathrm{a}})$ and $\mathrm{ext}^t((u, \Lambda)) = (u, \Lambda^t)$, where $\Lambda^{\mathrm{a}} = \{\mathbb{T}_{0,0} \in (1,2) \wedge \mathbb{T}_{0,1} \in (1,2) \wedge \mathbb{T}_{0,2} \in (1,2) \wedge \mathbb{T}_{1,2} \in (0,1) \wedge \mathbb{T}_{2,2} = \mathbb{T}_{3,3} = 0\}$ and $\Lambda^t = \{\mathbb{T}_{0,0} \in (1,2) \wedge \mathbb{T}_{0,1} \in (1,2) \wedge \mathbb{T}_{0,2} \in (1,2) \wedge \mathbb{T}_{1,2} \in (0,1) \wedge \mathbb{T}_{2,2} > 0\}$.

Definition 10 (chronometric timed language). *A timed language $\mathcal{L}$ is* chronometric *if there is a finite set $\{(u_1, \Lambda_1), (u_2, \Lambda_2), \ldots, (u_m, \Lambda_m)\}$ of disjoint elementary languages satisfying $\mathcal{L} = \bigcup_{i \in \{1,2,\ldots,m\}} (u_i, \Lambda_i)$.*

For any elementary language $\mathcal{L}$, its immediate exterior $\mathrm{ext}(\mathcal{L})$ is chronometric. We naturally extend the notion of exterior to chronometric timed languages, i. e., for a chronometric timed language $\mathcal{L} = \bigcup_{i \in \{1,2,\ldots,m\}} (u_i, \Lambda_i)$, we let $\mathrm{ext}(\mathcal{L}) = \bigcup_{i \in \{1,2,\ldots,m\}} \mathrm{ext}((u_i, \Lambda_i))$, which is also chronometric. For a timed word $w = \tau_0 a_1 \tau_1 a_2 \ldots a_n \tau_n$, we denote the valuation of $\tau_0, \tau_1, \ldots, \tau_n$ by $\kappa(w)$.

Chronometric relational morphism [21] relates any timed word to a timed word in a certain set P, which is later used to define a timed language. Intuitively, the tuples in Φ specify a mapping from timed words immediately out of P to timed words in P. By inductively applying it, any timed word is mapped to P.

Definition 11 (chronometric relational morphism). *Let P be a chronometric and prefix-closed timed language. Let $(u, \Lambda, u', \Lambda', R)$ be a 5-tuple such that $(u, \Lambda) \subseteq \mathrm{ext}(P)$, $(u', \Lambda') \subseteq P$, and R is a finite conjunction of equations of the form $\mathbb{T}_{i,|u|} = \mathbb{T}'_{j,|u'|}$, where $i \leq |u|$ and $j \leq |u'|$. For such a tuple, we let $[\![(u, \Lambda, u', \Lambda', R)]\!] \subseteq (u, \Lambda) \times (u', \Lambda')$ be the relation such that $(w, w') \in [\![(u, \Lambda, u', \Lambda', R)]\!]$ if and only if $\kappa(w), \kappa(w') \models R$. For a finite set Φ of such tuples, the* chronometric relational morphism *$[\![\Phi]\!] \subseteq \mathcal{T}(\Sigma) \times P$ is the relation inductively defined as follows: 1) for $w \in P$, we have $(w, w) \in [\![\Phi]\!]$; 2) for $w \in \mathrm{ext}(P)$ and $w' \in P$, we have $(w, w') \in [\![\Phi]\!]$ if we have $(w, w') \in [\![(u, \Lambda, u', \Lambda', R)]\!]$ for one of the tuples $(u, \Lambda, u', \Lambda', R) \in \Phi$; 3) for $w \in \mathrm{ext}(P)$, $w' \in \mathcal{T}(\Sigma)$, and $w'' \in P$, we have $(w \cdot w', w'') \in [\![\Phi]\!]$ if there is $w''' \in \mathcal{T}(\Sigma)$ satisfying $(w, w''') \in [\![\Phi]\!]$ and $(w''' \cdot w', w'') \in [\![\Phi]\!]$. We also require that all (u, Λ) in the tuples in Φ must be disjoint and the union of each such (u, Λ) is $\mathrm{ext}(P) \setminus P$.*

A chronometric relational morphism $[\![\Phi]\!]$ is *compatible* with $F \subseteq P$ if for each tuple $(u, \Lambda, u', \Lambda', R)$ defining $[\![\Phi]\!]$, we have either $(u', \Lambda') \subseteq F$ or $(u', \Lambda') \cap F = \emptyset$.

Definition 12 (recognizable timed language). *A timed language $\mathcal{L}$ is* recognizable *if there is a chronometric prefix-closed set P, a chronometric subset F of P, and a chronometric relational morphism $[\![\Phi]\!] \subseteq \mathcal{T}(\Sigma) \times P$ compatible with F satisfying $\mathcal{L} = \{w \mid \exists w' \in F, (w, w') \in [\![\Phi]\!]\}$.*

It is known that for any recognizable timed language $\mathcal{L}$, we can construct a DTA $\mathcal{A}$ recognizing $\mathcal{L}$, and vice versa [21].

2.3 Distinguishing Extensions and Active DFA Learning

Most DFA learning algorithms are based on *Nerode's congruence* [18]. For a (not necessarily regular) language $\mathcal{L} \subseteq \Sigma^*$, Nerode's congruence $\equiv_{\mathcal{L}} \subseteq \Sigma^* \times \Sigma^*$ is the equivalence relation satisfying $w \equiv_{\mathcal{L}} w'$ if and only if for any $w'' \in \Sigma^*$, we have $w \cdot w'' \in \mathcal{L} \iff w' \cdot w'' \in \mathcal{L}$.

Generally, we cannot decide if $w \equiv_{\mathcal{L}} w'$ by testing because it requires infinitely many membership checking. However, if $\mathcal{L}$ is regular, there is a finite set of suffixes $S \subseteq \Sigma^*$ called *distinguishing extensions* satisfying $\equiv_{\mathcal{L}} = \sim^S_{\mathcal{L}}$, where $\sim^S_{\mathcal{L}}$ is the equivalence relation satisfying $w \sim^S_{\mathcal{L}} w'$ if and only if for any $w'' \in S$, we have $w \cdot w'' \in \mathcal{L} \iff w' \cdot w'' \in \mathcal{L}$. Thus, the minimum DFA recognizing $\mathcal{L}_{\mathrm{tgt}}$ can be learned by[3]: i) identifying distinguishing extensions S satisfying $\equiv_{\mathcal{L}_{\mathrm{tgt}}} = \sim^S_{\mathcal{L}_{\mathrm{tgt}}}$ and ii) constructing the minimum DFA $\mathcal{A}$ corresponding to $\sim^S_{\mathcal{L}_{\mathrm{tgt}}}$.

The *L^* algorithm* [8] is an algorithm to learn the minimum DFA $\mathcal{A}_{\mathrm{hyp}}$ recognizing the target regular language $\mathcal{L}_{\mathrm{tgt}}$ with finitely many *membership* and *equivalence* queries to the teacher. In a membership query, the learner asks if $w \in \Sigma^*$ belongs to the target language $\mathcal{L}_{\mathrm{tgt}}$ i. e., $w \in \mathcal{L}_{\mathrm{tgt}}$. In an equivalence query, the learner asks if the hypothesis DFA $\mathcal{A}_{\mathrm{hyp}}$ recognizes the target language

[3] The distinguishing extensions S can be defined locally. For example, the TTT algorithm [19] is optimized with *local* distinguishing extensions for some prefixes $w \in \Sigma^*$. Nevertheless, we use the global distinguishing extensions for simplicity.

Algorithm 1: Outline of an L*-style active DFA learning algorithm

```
P ← {ε}; S ← {ε}
while ⊤ do
    while the observation table is not closed or consistent do
        update P and S so that the observation table is closed and consistent
    A_hyp ← ConstructDFA(P, S, T)
    switch eq_{L_tgt}(A_hyp) do
        case ⊤ do
            return A_hyp
        case cex do
            Update P and/or S using cex
```

$\mathcal{L}_{\mathrm{tgt}}$ i. e., $\mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) = \mathcal{L}_{\mathrm{tgt}}$, where $\mathcal{L}(\mathcal{A}_{\mathrm{hyp}})$ is the language of the hypothesis DFA $\mathcal{A}_{\mathrm{hyp}}$. When we have $\mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) \neq \mathcal{L}_{\mathrm{tgt}}$, the teacher returns a counterexample $cex \in \mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) \triangle \mathcal{L}_{\mathrm{tgt}}$. The information obtained via queries is stored in a 2-dimensional array called an *observation table*. See Fig. 1b for an illustration. For finite index sets $P, S \subseteq \Sigma^*$, for each pair $(p, s) \in (P \cup P \cdot \Sigma) \times S$, the observation table stores whether $p \cdot s \in \mathcal{L}_{\mathrm{tgt}}$. S is the current candidate of the distinguishing extensions, and P represents $\Sigma^*/\sim^S_{\mathcal{L}_{\mathrm{tgt}}}$. Since P and S are finite, one can fill the observation table using finite membership queries.

Algorithm 1 outlines an L*-style algorithm. We start from $P = S = \{\varepsilon\}$ and incrementally increase them. To construct a hypothesis DFA $\mathcal{A}_{\mathrm{hyp}}$, the observation table must be *closed* and *consistent*. An observation table is *closed* if, for each $p \in P \cdot \Sigma$, there is $p' \in P$ satisfying $p \sim^S_{\mathcal{L}_{\mathrm{tgt}}} p'$. An observation table is *consistent* if, for any $p, p' \in P \cup P \cdot \Sigma$ and $a \in \Sigma$, $p \sim^S_{\mathcal{L}_{\mathrm{tgt}}} p'$ implies $p \cdot a \sim^S_{\mathcal{L}_{\mathrm{tgt}}} p' \cdot a$.

Once the observation table becomes closed and consistent, the learner constructs a hypothesis DFA $\mathcal{A}_{\mathrm{hyp}}$ and checks if $\mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) = \mathcal{L}_{\mathrm{tgt}}$ by an equivalence query. If $\mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) = \mathcal{L}_{\mathrm{tgt}}$ holds, $\mathcal{A}_{\mathrm{hyp}}$ is the resulting DFA. Otherwise, the teacher returns $cex \in \mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) \triangle \mathcal{L}_{\mathrm{tgt}}$, which is used to refine the observation table. There are several variants of the refinement. In the L* algorithm, all the prefixes of cex are added to P. In the Rivest-Schapire algorithm [20,25], an extension s strictly refining S is obtained by an analysis of cex, and such s is added to S.

3 A Myhill-Nerode Style Characterization of Recognizable Timed Languages with Elementary Languages

Unlike the case of regular languages, any finite set of timed words cannot correctly distinguish recognizable timed languages due to the infiniteness of dwell time in timed words. Instead, we use a finite set of *elementary languages* to define a Nerode-style congruence. To define the Nerode-style congruence, we extend the notion of membership to elementary languages.

Definition 13 (symbolic membership). *For a timed language $\mathcal{L} \subseteq \mathcal{T}(\Sigma)$ and an elementary language $(u, \Lambda) \in \mathcal{E}(\Sigma)$, the* symbolic membership $\mathtt{mem}^{\mathtt{sym}}_{\mathcal{L}}((u,\Lambda))$ *of (u, Λ) to $\mathcal{L}$ is the strongest constraint such that for any $w \in (u, \Lambda)$, we have $w \in \mathcal{L}$ if and only if $\kappa(w) \models \mathtt{mem}^{\mathtt{sym}}_{\mathcal{L}}(\mathcal{L})$.*

We discuss how to obtain symbolic membership in Sect. 4.5. We define a Nerode-style congruence using symbolic membership. A naive idea is to distinguish two elementary languages by the equivalence of their symbolic membership. However, this does not capture the semantics of TAs. For example, for the DTA $\mathcal{A}$ in Fig. 1c, for any timed word w, we have $1.3 \cdot \mathtt{a} \cdot 0.4 \cdot w \in \mathcal{L}(\mathcal{A}) \iff 0.3 \cdot \mathtt{a} \cdot 1.0 \cdot \mathtt{a} \cdot 0.4 \cdot w \in \mathcal{L}(\mathcal{A})$, while they have different symbolic membership. This is because symbolic membership distinguishes the *position* in timed words where each clock variable is reset, which must be ignored. We use *renaming equations* to abstract such positional information in symbolic membership. Note that $\mathbb{T}_{i,n} = \sum_{k=i}^{n} \tau_k$ corresponds to the value of the clock variable reset at τ_i.

Definition 14 (renaming equation). *Let $\mathbb{T} = \{\tau_0, \tau_1, \dots, \tau_n\}$ and $\mathbb{T}' = \{\tau_0', \tau_1', \dots, \tau_{n'}'\}$ be sets of ordered variables. A* renaming equation *R over $\mathbb{T}$ and $\mathbb{T}'$ is a finite conjunction of equations of the form $\mathbb{T}_{i,n} = \mathbb{T}'_{i',n'}$, where $i \in \{0, 1, \dots, n\}$, $i' \in \{0, 1, \dots, n'\}$, $\mathbb{T}_{i,n} = \sum_{k=i}^{n} \tau_k$ and $\mathbb{T}'_{i',n'} = \sum_{k=i'}^{n'} \tau'_k$.*

Definition 15 ($\sim^S_{\mathcal{L}}$). *Let $\mathcal{L} \subseteq \mathcal{T}(\Sigma)$ be a timed language, let $(u, \Lambda), (u', \Lambda'), (u'', \Lambda'') \in \mathcal{E}(\Sigma)$ be elementary languages, and let R be a renaming equation over $\mathbb{T}$ and $\mathbb{T}'$, where $\mathbb{T}$ and $\mathbb{T}'$ are the variables of Λ and Λ', respectively. We let $(u, \Lambda) \sqsubseteq^{(u'',\Lambda''),R}_{\mathcal{L}} (u', \Lambda')$ if we have the following: for any $w \in (u, \Lambda)$, there is $w' \in (u', \Lambda')$ satisfying $\kappa(w), \kappa(w') \models R$; $\mathtt{mem}^{\mathtt{sym}}_{\mathcal{L}}((u, \Lambda) \cdot (u'', \Lambda'')) \wedge R \wedge \Lambda'$ is equivalent to $\mathtt{mem}^{\mathtt{sym}}_{\mathcal{L}}((u', \Lambda') \cdot (u'', \Lambda'')) \wedge R \wedge \Lambda$. We let $(u, \Lambda) \sim^{(u'',\Lambda''),R}_{\mathcal{L}} (u', \Lambda')$ if we have $(u, \Lambda) \sqsubseteq^{(u'',\Lambda''),R}_{\mathcal{L}} (u', \Lambda')$ and $(u', \Lambda' \sqsubseteq^{(u'',\Lambda''),R}_{\mathcal{L}} (u, \Lambda)$. Let $S \subseteq \mathcal{E}(\Sigma)$. We let $(u, \Lambda) \sim^{S,R}_{\mathcal{L}} (u', \Lambda')$ if for any $(u'', \Lambda'') \in S$, we have $(u, \Lambda) \sim^{(u'',\Lambda''),R}_{\mathcal{L}} (u', \Lambda')$. We let $(u, \Lambda) \sim^{S}_{\mathcal{L}} (u', \Lambda')$ if $(u, \Lambda) \sim^{S,R}_{\mathcal{L}} (u', \Lambda')$ for some renaming equation R.*

Example 16. Let $\mathcal{A}$ be the DTA in Fig. 1c and let (u, Λ), (u', Λ'), and (u'', Λ'') be elementary languages, where $u = \mathtt{a}$, $\Lambda = \{\tau_0 \in (1,2) \wedge \tau_0 + \tau_1 \in (1,2) \wedge \tau_1 \in (0,1)\}$, $u' = \mathtt{a} \cdot \mathtt{a}$, $\Lambda' = \{\tau'_0 \in (0,1) \wedge \tau'_0 + \tau'_1 \in (1,2) \wedge \tau'_1 + \tau'_2 \in (1,2) \wedge \tau'_2 \in (0,1)\}$, $u'' = \mathtt{a}$, and $\Lambda'' = \{\tau_0 \in (0,1) \wedge \tau_1 = 0\}$. We have $\mathtt{mem}^{\mathtt{sym}}_{\mathcal{L}(\mathcal{A})}((u, \Lambda) \cdot (u'', \Lambda'')) = \Lambda \wedge \Lambda'' \wedge \tau_1 + \tau''_0 \leq 1$ and $\mathtt{mem}^{\mathtt{sym}}_{\mathcal{L}(\mathcal{A})}((u', \Lambda') \cdot (u'', \Lambda'')) = \Lambda' \wedge \Lambda'' \wedge \tau'_2 + \tau''_0 \leq 1$. Therefore, for the renaming equation $\mathbb{T}_{1,1} = \mathbb{T}'_{2,2}$, we have $(u, \Lambda) \sim^{(u'',\Lambda''),\mathbb{T}_{1,1}=\mathbb{T}'_{2,2}}_{\mathcal{L}} (u', \Lambda')$.

An algorithm to check if $(u, \Lambda) \sim^S_{\mathcal{L}} (u', \Lambda')$ is shown in Appendix B.2 of [29].

Intuitively, $(u, \Lambda) \sqsubseteq^{s,R}_{\mathcal{L}} (u', \Lambda')$ shows that any $w \in (u, \Lambda)$ can be "simulated" by some $w' \in (u', \Lambda')$ with respect to s and R. Such intuition is formalized as follows.

Theorem 17. *For any $\mathcal{L} \subseteq \mathcal{T}(\Sigma)$ and $(u,\Lambda),(u',\Lambda'),(u'',\Lambda'') \in \mathcal{E}(\Sigma)$ satisfying $(u,\Lambda) \sqsubseteq_{\mathcal{L}}^{(u'',\Lambda'')} (u',\Lambda')$, for any $w \in (u,\Lambda)$, there is $w' \in (u',\Lambda')$ such that for any $w'' \in (u'',\Lambda'')$, $w \cdot w'' \in \mathcal{L} \iff w' \cdot w'' \in \mathcal{L}$ holds.* □

By $\bigcup_{(u,\Lambda)\in\mathcal{E}(\Sigma)}(u,\Lambda) = \mathcal{T}(\Sigma)$, we have the following as a corollary.

Corollary 18. *For any timed language $\mathcal{L} \subseteq \mathcal{T}(\Sigma)$ and for any elementary languages $(u,\Lambda),(u',\Lambda') \in \mathcal{E}(\Sigma)$, $(u,\Lambda) \sim_{\mathcal{L}}^{\mathcal{E}(\Sigma)} (u',\Lambda')$ implies the following.*

- *For any $w \in (u,\Lambda)$, there is $w' \in (u',\Lambda')$ such that for any $w'' \in \mathcal{T}(\Sigma)$, we have $w \cdot w'' \in \mathcal{L} \iff w' \cdot w'' \in \mathcal{L}$.*
- *For any $w' \in (u',\Lambda')$, there is $w \in (u,\Lambda)$ such that for any $w'' \in \mathcal{T}(\Sigma)$, we have $w \cdot w'' \in \mathcal{L} \iff w' \cdot w'' \in \mathcal{L}$.* □

The following characterizes recognizable timed languages with $\sim_{\mathcal{L}}^{\mathcal{E}(\Sigma)}$.

Theorem 19. (Myhill-Nerode style characterization). *A timed language $\mathcal{L}$ is recognizable if and only if the quotient set $\mathcal{SE}(\Sigma)/\sim_{\mathcal{L}}^{\mathcal{E}(\Sigma)}$ is finite.* □

By Theorem 19, we always have a finite set S of distinguishing extensions.

Theorem 20. *For any recognizable timed language $\mathcal{L}$, there is a finite set S of elementary languages satisfying $\sim_{\mathcal{L}}^{\mathcal{E}(\Sigma)} = \sim_{\mathcal{L}}^{S}$.* □

4 Active Learning of Deterministic Timed Automata

We show our L*-style active learning algorithm for DTAs with the Nerode-style congruence in Sect. 3. We let $\mathcal{L}_{\mathrm{tgt}}$ be the target timed language in learning.

For simplicity, we first present our learning algorithm with a smart teacher answering the following three kinds of queries: membership query $\mathtt{mem}_{\mathcal{L}_{\mathrm{tgt}}}(w)$ asking whether $w \in \mathcal{L}_{\mathrm{tgt}}$, symbolic membership query asking $\mathtt{mem}^{\mathtt{sym}}_{\mathcal{L}_{\mathrm{tgt}}}((u,\Lambda))$, and equivalence query $\mathtt{eq}_{\mathcal{L}_{\mathrm{tgt}}}(\mathcal{A}_{\mathrm{hyp}})$ asking whether $\mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) = \mathcal{L}_{\mathrm{tgt}}$. If $\mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) = \mathcal{L}_{\mathrm{tgt}}$, $\mathtt{eq}_{\mathcal{L}_{\mathrm{tgt}}}(\mathcal{A}_{\mathrm{hyp}}) = \top$, and otherwise, $\mathtt{eq}_{\mathcal{L}_{\mathrm{tgt}}}(\mathcal{A}_{\mathrm{hyp}})$ is a timed word $cex \in \mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) \triangle \mathcal{L}_{\mathrm{tgt}}$. Later in Sect. 4.5, we show how to answer a symbolic membership query with finitely many membership queries. Our task is to construct a DTA $\mathcal{A}$ satisfying $\mathcal{L}(\mathcal{A}) = \mathcal{L}_{\mathrm{tgt}}$ with finitely many queries.

4.1 Successors of Simple Elementary Languages

Similarly to the L* algorithm in Sect. 2.3, we learn a DTA with an observation table. Reflecting the extension of the underlying congruence, we use sets of simple elementary languages for the indices. To define the extensions, $P \cup (P \cdot \Sigma)$ in the L* algorithm, we introduce *continuous* and *discrete successors* for simple elementary languages, which are inspired by *regions* [4]. We note that immediate exteriors do not work for this purpose. For example, for $(u,\Lambda) = (\mathrm{a}, \{\mathbb{T}_{0,1} < 2 \wedge \mathbb{T}_{1,1} < 1\})$ and $w = 0.7 \cdot \mathrm{a} \cdot 0.9$, we have $w \in (u,\Lambda)$ and $\mathrm{ext}^t((u,\Lambda)) = (\mathrm{a}, \{\mathbb{T}_{0,1} = 2 \wedge \mathbb{T}_{1,1} < 1\})$, but there is no $t > 0$ satisfying $w \cdot t \in \mathrm{ext}^t((u,\Lambda))$.

Algorithm 2: DTA construction from a timed observation table

Input : A cohesive timed observation table (P, S, T)
Output : A DTA $\mathcal{A}_{\mathrm{hyp}}$ row-faithful to the given timed observation table

Function `MakeDTA`(P, S, T)**:**
  $\Phi \leftarrow \emptyset$; $F \leftarrow \{(u, \Lambda) \in P \mid T((u, \Lambda), (\varepsilon, \tau'_0 = 0)) = \{\Lambda \wedge \tau'_0 = 0\}\}$
  for $p \in P$ *such that* $\mathrm{succ}^t(p) \notin P$ *(resp.* $\mathrm{succ}^a(p) \notin P$*)* **do**
    `// Construct` $(u, \Lambda, u', \Lambda', R)$ `for some` $p' \in P$ `and` R
    `// Such` R `is chosen using an exhaustive search`
    pick $p' \in P$ and R such that $\mathrm{succ}^t(p) \sim^{S,R}_{\mathcal{L}_{\mathrm{tgt}}} p'$ (resp. $\mathrm{succ}^a(p) \sim^{S,R}_{\mathcal{L}_{\mathrm{tgt}}} p'$)
    add $(u, \Lambda, u', \Lambda', R)$ **to** Φ, where $(u, \Lambda) = \mathrm{ext}^t(p)$ (resp. $\mathrm{ext}^a(p)$) and $(u', \Lambda') = p'$
  return the DTA $\mathcal{A}_{\mathrm{hyp}}$ obtained from (P, F, Φ) by the construction in [21]

For any $(u, \Lambda) \in \mathcal{SE}(\Sigma)$, we let $\Theta_{(u,\Lambda)}$ be the total order over 0 and the fractional parts $\mathrm{frac}(\mathbb{T}_{0,n}), \mathrm{frac}(\mathbb{T}_{1,n}), \ldots, \mathrm{frac}(\mathbb{T}_{n,n})$ of $\mathbb{T}_{0,n}, \mathbb{T}_{1,n}, \ldots, \mathbb{T}_{n,n}$. Such an order is uniquely defined because Λ is simple and canonical (Proposition 36 of [29]).

Definition 21 (successor). *Let $p = (u, \Lambda) \in \mathcal{SE}(\Sigma)$ be a simple elementary language. The* discrete successor $\mathrm{succ}^a(p)$ *of p is* $\mathrm{succ}^a(p) = (u \cdot a, \Lambda \wedge \tau_{n+1} = 0)$. *The* continuous successor $\mathrm{succ}^t(p)$ *of p is* $\mathrm{succ}^t(p) = (u, \Lambda^t)$, *where Λ^t is defined as follows: if there is an equation $\mathbb{T}_{i,n} = d$ in Λ, all such equations are replaced with $\mathbb{T}_{i,n} \in (d, d+1)$; otherwise, for each greatest $\mathbb{T}_{i,n}$ in terms of $\Theta_{(u,\Lambda)}$, we replace $\mathbb{T}_{i,n} \in (d, d+1)$ with $\mathbb{T}_{i,n} = d+1$. We let* $\mathrm{succ}(p) = \bigcup_{a \in \Sigma} \mathrm{succ}^a(p) \cup \mathrm{succ}^t(p)$. *For $P \subseteq \mathcal{SE}(\Sigma)$, we let* $\mathrm{succ}(P) = \bigcup_{p \in P} \mathrm{succ}(p)$.

Example 22. Let $u = \mathrm{aa}$, $\Lambda = \{\mathbb{T}_{0,0} \in (1,2) \wedge \mathbb{T}_{0,1} \in (1,2) \wedge \mathbb{T}_{0,2} \in (1,2) \wedge \mathbb{T}_{1,1} \in (0,1) \wedge \mathbb{T}_{1,2} \in (0,1) \wedge \mathbb{T}_{2,2} = 0\}$. The order $\Theta_{(u,\Lambda)}$ is such that $0 = \mathrm{frac}(\mathbb{T}_{2,2}) < \mathrm{frac}(\mathbb{T}_{1,2}) < \mathrm{frac}(\mathbb{T}_{0,2})$. The continuous successor of (u, Λ) is $\mathrm{succ}^t((u, \Lambda)) = (u, \Lambda^t)$, where $\Lambda^t = \{\mathbb{T}_{0,0} \in (1,2) \wedge \mathbb{T}_{0,1} \in (1,2) \wedge \mathbb{T}_{0,2} \in (1,2) \wedge \mathbb{T}_{1,1} \in (0,1) \wedge \mathbb{T}_{1,2} \in (0,1) \wedge \mathbb{T}_{2,2} \in (0,1)\}$. The continuous successor of (u, Λ^t) is $\mathrm{succ}^t((u, \Lambda^t)) = (u, \Lambda^{tt})$, where $\Lambda^{tt} = \{\mathbb{T}_{0,0} \in (1,2) \wedge \mathbb{T}_{0,1} \in (1,2) \wedge \mathbb{T}_{0,2} = 2 \wedge \mathbb{T}_{1,1} \in (0,1) \wedge \mathbb{T}_{1,2} \in (0,1) \wedge \mathbb{T}_{2,2} \in (0,1)\}$.

4.2 Timed Observation Table for Active DTA Learning

We extend the observation table with (simple) elementary languages and symbolic membership to learn a *recognizable timed language*.

Definition 23 (timed observation table). *A* timed observation table *is a 3-tuple (P, S, T) such that: P is a prefix-closed finite set of simple elementary languages, S is a finite set of elementary languages, and T is a function mapping $(p, s) \in (P \cup \mathrm{succ}(P)) \times S$ to the symbolic membership $\mathtt{mem}^{\mathtt{sym}}_{\mathcal{L}_{\mathrm{tgt}}}(p \cdot s)$.*

Figure 2 illustrates timed observation tables: each cell indexed by (p, s) show the symbolic membership $\mathtt{mem}^{\mathtt{sym}}_{\mathcal{L}_{\mathrm{tgt}}}(p \cdot s)$. For timed observation tables, we extend the notion of closedness and consistency with $\sim^{S}_{\mathcal{L}_{\mathrm{tgt}}}$ we introduced in Sect. 3.

Algorithm 3: Counterexample analysis in our DTA learning algorithm

Function `AnalyzeCEX`(cex)**:**
  $i \leftarrow 1$; $w_0 \leftarrow cex$
  while $\nexists p \in P. w_i \in p$ **do**
    $i \leftarrow i + 1$
    split w_{i-1} **into** $w'_i \cdot w''_i$ such that $w'_i \in p'_i$ for some $p'_i \in \mathrm{succ}(P) \setminus P$
    let $p_i \in P$ and R_i **be** such that $p'_i \sim^{S,R_i}_{\mathcal{L}_{\mathrm{tgt}}} p_i$
    let $\overline{w}_i \in p_i$ **be** such that $\kappa(w'_i), \kappa(\overline{w}_i) \models R_i$
    $w_i \leftarrow \overline{w}_i \cdot w''_i$
  find $j \in \{1, 2, \ldots, i\}$ such that $w_{j-1} \in \mathcal{L}_{\mathrm{tgt}} \triangle \mathcal{L}(\mathcal{A}_{\mathrm{hyp}})$ and $w_j \notin \mathcal{L}_{\mathrm{tgt}} \triangle \mathcal{L}(\mathcal{A}_{\mathrm{hyp}})$
  `// We use a binary search with membership queries for` $\lceil \log(i) \rceil$ `times.`
  return the simple elementary language including w''_j

We note that consistency is defined only for discrete successors. This is because a timed observation table does not always become "consistent" for continuous successors. See Appendix C of [29] for a detailed discussion. We also require *exterior-consistency* since we construct an exterior from a successor.

Definition 24 (closedness, consistency, exterior-consistency, cohesion). *Let $O = (P, S, T)$ be a timed observation table. O is* closed *if, for each $p \in \mathrm{succ}(P) \setminus P$, there is $p' \in P$ satisfying $p \sim^S_{\mathcal{L}_{\mathrm{tgt}}} p'$. O is* consistent *if, for each $p, p' \in P$ and for each $a \in \Sigma$, $p \sim^S_{\mathcal{L}_{\mathrm{tgt}}} p'$ implies $\mathrm{succ}^a(p) \sim^S_{\mathcal{L}_{\mathrm{tgt}}} \mathrm{succ}^a(p')$. O is* exterior-consistent *if for any $p \in P$, $\mathrm{succ}^t(p) \notin P$ implies $\mathrm{succ}^t(p) \subseteq \mathrm{ext}^t(p)$. O is* cohesive *if it is closed, consistent, and exterior-consistent.*

From a cohesive timed observation table (P, S, T), we can construct a DTA as outlined in Algorithm 2. We construct a DTA via a recognizable timed language. The main ideas are as follows: 1) we approximate $\sim^{\mathcal{E}(\Sigma),R}_{\mathcal{L}_{\mathrm{tgt}}}$ by $\sim^{S,R}_{\mathcal{L}_{\mathrm{tgt}}}$; 2) we decide the equivalence class of $\mathrm{ext}(p) \in \mathrm{ext}(P) \setminus P$ in $\mathcal{E}(\Sigma)$ from successors. Notice that there can be multiple renaming equations R showing $\sim^{S,R}_{\mathcal{L}_{\mathrm{tgt}}}$. We use one of them found by an exhaustive search in Appendix B.2 of [29].

The DTA obtained by `MakeDTA` is faithful to the timed observation table in *rows*, i. e., for any $p \in P \cup \mathrm{succ}(P)$, $\mathcal{L}_{\mathrm{tgt}} \cap p = \mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) \cap p$. However, unlike the L* algorithm, this does not hold for each *cell*, i. e., there may be $p \in P \cup \mathrm{succ}(P)$ and $s \in S$ satisfying $\mathcal{L}_{\mathrm{tgt}} \cap (p \cdot s) \neq \mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) \cap (p \cdot s)$. This is because we do not (and actually cannot) enforce consistency for continuous successors. See Appendix C of [29] for a discussion. Nevertheless, this does not affect the correctness of our algorithm partly by Theorem 26.

Theorem 25 (row faithfulness). *For any cohesive timed observation table (P, S, T), for any $p \in P \cup \mathrm{succ}(P)$, $\mathcal{L}_{\mathrm{tgt}} \cap p = \mathcal{L}(\texttt{MakeDTA}(P, S, T)) \cap p$ holds.* □

Theorem 26. *For any cohesive timed observation table (P, S, T), $\sim^S_{\mathcal{L}_{\mathrm{tgt}}} = \sim^{\mathcal{E}(\Sigma)}_{\mathcal{L}_{\mathrm{tgt}}}$ implies $\mathcal{L}_{\mathrm{tgt}} = \mathcal{L}(\texttt{MakeDTA}(P, S, T))$.* □

Algorithm 4: Outline of our L*-style algorithm for DTA learning

1 $P \leftarrow \{(\varepsilon, \tau_0 = 0)\}$; $S \leftarrow \{(\varepsilon, \tau_0' = 0)\}$
2 **while** $\top$ **do**
3 **while** (P, S, T) *is not cohesive* **do**
4 **if** $\exists p \in \mathrm{succ}(P) \setminus P. \nexists p' \in P. p \sim^S_{\mathcal{L}_{\mathrm{tgt}}} p'$ **then** // (P, S, T) is not closed
5 $P \leftarrow P \cup \{p\}$ // Add such p to P
6 **else if** $\exists p, p' \in P, a \in \Sigma. p \sim^S_{\mathcal{L}_{\mathrm{tgt}}} p' \wedge \mathrm{succ}^a(p) \not\sim^S_{\mathcal{L}_{\mathrm{tgt}}} \mathrm{succ}^a(p')$ **then**
 // (P, S, T) is inconsistent due to a
7 **let** $S' \subseteq S$ **be** a minimal set such that $p \not\sim^{S \cup \{\{a \cdot w \mid w \in s\} \mid s \in S'\}}_{\mathcal{L}_{\mathrm{tgt}}} p'$
8 $S \leftarrow S \cup \{\{a \cdot w \mid w \in s\} \mid s \in S'\}$
9 **else** // (P, S, T) is not exterior-consistent
10 $P \leftarrow P \cup \{p' \in \mathrm{succ}^t(P) \setminus P \mid \exists p \in P. p' = \mathrm{succ}^t(p) \wedge p' \notin \mathrm{ext}^t(p)\}$
11 fill T using symbolic membership queries
12 $\mathcal{A}_{\mathrm{hyp}} \leftarrow$ `MakeDTA`(P, S, T)
13 **if** $cex = \mathtt{eq}_{\mathcal{L}_{\mathrm{tgt}}}(\mathcal{A}_{\mathrm{hyp}})$ **then**
14 **add** `AnalyzeCEX`(cex) **to** S
15 **else return** $\mathcal{A}_{\mathrm{hyp}}$ // It returns $\mathcal{A}_{\mathrm{hyp}}$ if $\mathtt{eq}_{\mathcal{L}_{\mathrm{tgt}}}(\mathcal{A}_{\mathrm{hyp}}) = \top$.

4.3 Counterexample Analysis

We analyze the counterexample *cex* obtained by an equivalence query to refine the set S of suffixes in a timed observation table. Our analysis, outlined in Algorithm 3, is inspired by the Rivest-Schapire algorithm [20,25]. The idea is to reduce the counterexample *cex* using the mapping defined by the congruence $\sim^S_{\mathcal{L}_{\mathrm{tgt}}}$ (lines 5–7), much like Φ in recognizable timed languages, and to find a suffix s strictly refining S (line 9), i. e., satisfying $p \sim^S_{\mathcal{L}_{\mathrm{tgt}}} p'$ and $p \not\sim^{S \cup \{s\}}_{\mathcal{L}_{\mathrm{tgt}}} p'$ for some $p \in \mathrm{succ}(P)$ and $p' \in P$.

By definition of *cex*, we have $cex = w_0 \in \mathcal{L}_{\mathrm{tgt}} \triangle \mathcal{L}(\mathcal{A}_{\mathrm{hyp}})$. By Theorem 25, $w_n \notin \mathcal{L}_{\mathrm{tgt}} \triangle \mathcal{L}(\mathcal{A}_{\mathrm{hyp}})$ holds, where n is the final value of i. By construction of $\mathcal{A}_{\mathrm{hyp}}$ and w_i, for any $i \in \{1, 2, \ldots, n\}$, we have $w_0 \in \mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) \iff w_i \in \mathcal{L}(\mathcal{A}_{\mathrm{hyp}})$. Therefore, there is $i \in \{1, 2, \ldots, n\}$ satisfying $w_{i-1} \in \mathcal{L}_{\mathrm{tgt}} \triangle \mathcal{L}(\mathcal{A}_{\mathrm{hyp}})$ and $w_i \notin \mathcal{L}_{\mathrm{tgt}} \triangle \mathcal{L}(\mathcal{A}_{\mathrm{hyp}})$. For such i, since we have $w_{i-1} = w_i' \cdot w_i'' \in \mathcal{L}_{\mathrm{tgt}} \triangle \mathcal{L}(\mathcal{A}_{\mathrm{hyp}})$, $w_i = \overline{w}_i \cdot w_i'' \notin \mathcal{L}_{\mathrm{tgt}} \triangle \mathcal{L}(\mathcal{A}_{\mathrm{hyp}})$, and $\kappa(w_i'), \kappa(\overline{w}_i) \models R_i$, such w_i'' is a witness of $p_i' \not\sim^{\mathcal{E}(\Sigma), R_i}_{\mathcal{L}_{\mathrm{tgt}}} p_i$. Therefore, S can be refined by the simple elementary language $s \in \mathcal{SE}(\Sigma)$ including w_i''.

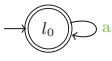
4.4 L*-Style Learning Algorithm for DTAs

Algorithm 4 outlines our active DTA learning algorithm. At line 1, we initialize the timed observation table with $P = \{(\varepsilon, \tau_0 = 0)\}$ and $S = \{(\varepsilon, \tau_0' = 0)\}$. In the loop in lines 2–15, we refine the timed observation table until the hypothesis DTA $\mathcal{A}_{\mathrm{hyp}}$ recognizes the target language $\mathcal{L}_{\mathrm{tgt}}$, which is checked by equivalence queries. The refinement finishes when the equivalence relation $\sim^S_{\mathcal{L}_{\mathrm{tgt}}}$ defined by the suffixes S converges to $\sim^{\mathcal{E}(\Sigma)}_{\mathcal{L}_{\mathrm{tgt}}}$, and the prefixes P covers $\mathcal{SE}(\Sigma)/\sim^{\mathcal{E}(\Sigma)}_{\mathcal{L}_{\mathrm{tgt}}}$.

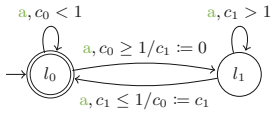
In the loop in lines 3–11, we make the timed observation table cohesive. If the timed observation table is not closed, we move the incompatible row in $\mathrm{succ}(P) \setminus P$ to P (line 5). If the timed observation table is inconsistent, we concatenate an

	$(\varepsilon, \tau_0' = 0)$
$(\varepsilon, \tau_0 = 0)$	$\tau_0 = \tau_0' = 0$
$(\varepsilon, \tau_0 \in (0,1))$	$0 = \tau_0' < \tau_0 < 1$
$(\mathrm{a}, \tau_0 = \tau_1 = 0)$	$\tau_0 = \tau_1 = \tau_0' = 0$

(a) The initial timed observation table $O_1 = (P_1, S_1, T_1)$



(b) DTA $\mathcal{A}^1_{\mathrm{hyp}}$ constructed from O_1



(c) DTA $\mathcal{A}^3_{\mathrm{hyp}}$ constructed from O_3

	$s_0 = (\varepsilon, \tau_0' = 0)$	$s_1 = (\mathrm{a}, \tau_1' = 0 < \tau_0' < 1)$
$p_0 = (\varepsilon, \tau_0 = 0)$	$\top$	$\top$
$p_1 = (\varepsilon, \tau_0 \in (0,1))$	$\top$	$\tau_0 + \tau_0' \in (0,1)$
$p_2 = (\mathrm{a}, \tau_0 = \tau_0 + \tau_1 = \tau_1 = 0)$	$\top$	$\top$

(d) Timed observation table $O_2 = (P_2, S_2, T_2)$ after processing *cex*

	$(\varepsilon, \tau_0' = 0)$	$(\mathrm{a}, \tau_1' = 0 < \tau_0' < 1)$
$(\varepsilon, \tau_0 = 0)$	$\top$	$\top$
$(\varepsilon, \tau_0 \in (0,1))$	$\top$	$\tau_0 + \tau_0' \in (0,1)$
$(\varepsilon, \tau_0 = 1)$	$\top$	$\bot$
$(\mathrm{a}, \tau_0 = \tau_0 + \tau_1 = 1 \wedge \tau_1 = 0)$	$\bot$	$\top$
$(\mathrm{a}, \tau_0 = 1 \wedge \tau_1 \in (0,1))$	$\bot$	$\tau_1 + \tau_0' \in (0,1]$
$(\mathrm{a}, \tau_0 = \tau_1 = 1 \wedge \tau_0 + \tau_1 = 2)$	$\bot$	$\bot$
$(\mathrm{a}, \tau_0 = \tau_0 + \tau_1 = \tau_1 = 0)$	$\top$	$\top$
$(\mathrm{a}, \tau_0 = \tau_0 + \tau_1 \in (0,1) \wedge \tau_1 = 0)$	$\top$	$\tau_0 + \tau_1 + \tau_0' \in (0,1)$
$(\varepsilon, \tau_0 \in (1,2))$	$\top$	$\bot$
$(\mathrm{aa}, \tau_0 = \tau_0 + \tau_1 = \tau_0 + \tau_1 + \tau_2 = 1 \wedge \tau_1 = \tau_2 = \tau_1 + \tau_2 = 0)$	$\top$	$\top$
$(\mathrm{aa}, \tau_0 = 1 \wedge \tau_1 = \tau_1 + \tau_2 \in (0,1) \wedge \tau_0 + \tau_1 = \tau_0 + \tau_1 + \tau_2 \in (1,2) \wedge \tau_2 = 0)$	$\top$	$\tau_1 + \tau_2 + \tau_0' \in (0,1)$
$(\mathrm{a}, \tau_0 = 1 < \tau_1 < 2 < \tau_0 + \tau_1 < 3)$	$\bot$	$\bot$
$(\mathrm{aa}, \tau_0 = \tau_1 = \tau_1 + \tau_2 = 1 \wedge \tau_0 + \tau_1 = \tau_0 + \tau_1 + \tau_2 = 2 \wedge \tau_2 = 0)$	$\top$	$\bot$

(e) The final observation table $O_3 = (P_3, S_3, T_3)$

Fig. 2. Timed observation tables O_1, O_2, O_3, and the DTAs $\mathcal{A}^1_{\mathrm{hyp}}$ and $\mathcal{A}^3_{\mathrm{hyp}}$ made from O_1 and O_3, respectively. In O_2 and O_3, we only show the constraints non-trivial from p and s. The DTAs are simplified without changing the language. The use of clock assignments, which does not change the expressiveness, is from [21].

event $a \in \Sigma$ in front of some of the suffixes in S (line 8). If the timed observation table is not exterior-consistent, we move the boundary $\mathrm{succ}^t(p) \in \mathrm{succ}^t(P) \setminus P$ satisfying $\mathrm{succ}^t(p) \nsubseteq \mathrm{ext}^t(p)$ to P (line 10). Once we obtain a cohesive timed observation table, we construct a DTA $\mathcal{A}_{\mathrm{hyp}} = \texttt{MakeDTA}(P, S, T)$ and make an equivalence query (line 12). If we have $\mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) = \mathcal{L}_{\mathrm{tgt}}$, we return $\mathcal{A}_{\mathrm{hyp}}$. Otherwise, we have a timed word *cex* witnessing the difference between the language of the hypothesis DTA $\mathcal{A}_{\mathrm{hyp}}$ and the target language $\mathcal{L}_{\mathrm{tgt}}$. We refine the timed observation table using Algorithm 3.

Example 27. Let $\mathcal{L}_{\mathrm{tgt}}$ be the timed language recognized by the DTA in Fig. 1c. We start from $P = \{(\varepsilon, \tau_0 = 0)\}$ and $S = \{(\varepsilon, \tau_0' = 0\}$. Figure 2a shows the initial timed observation table O_1. Since the timed observation table O_1 in Fig. 2a is cohesive, we construct a hypothesis DTA $\mathcal{A}^1_{\mathrm{hyp}}$. The hypothesis recognizable timed language is (P_1, F_1, Φ_1) is such that $P_1 = F_1 = \{(\varepsilon, \tau_0 = 0)\}$ and $\Phi_1 = \{(\varepsilon, \tau_0 > 0, \varepsilon, \tau_0, \top), (\mathrm{a}, \tau_0 = \tau_0 + \tau_1 = \tau_1 = 0, \varepsilon, \tau_0, \top)\}$. Figure 2b shows the first hypothesis DTA $\mathcal{A}^1_{\mathrm{hyp}}$.

We have $\mathcal{L}(\mathcal{A}^1_{\mathrm{hyp}}) \neq \mathcal{L}_{\mathrm{tgt}}$, and the learner obtains a counterexample, e. g., $cex = 1.0 \cdot \mathrm{a} \cdot 0$, with an equivalence query. In Algorithm 3, we have $w_0 = cex$, $w_1 = 0.5 \cdot \mathrm{a} \cdot 0$, $w_2 = 0 \cdot \mathrm{a} \cdot 0$, and $w_3 = 0$. We have $w_0 \notin \mathcal{L}(\mathcal{A}^1_{\mathrm{hyp}}) \triangle \mathcal{L}_{\mathrm{tgt}}$ and $w_1 \in \mathcal{L}(\mathcal{A}^1_{\mathrm{hyp}}) \triangle \mathcal{L}_{\mathrm{tgt}}$, and the suffix to distinguish w_0 and w_1 is $0.5 \cdot \mathrm{a} \cdot 0$. Thus, we add $s_1 = (\mathrm{a}, \tau_1' = 0 < \tau_0' = \tau_0' + \tau_1' < 1)$ to S_1 (Fig. 2d).

In Fig. 2d, we observe that $T_2(p_1, s_1)$ is more strict than $T_2(p_0, s_1)$, and we have $p_1 \not\sim^{S_2}_{\mathcal{L}_{\mathrm{tgt}}} p_0$. To make (P_2, S_2, T_2) closed, we add p_1 to P_2. By repeating similar operations, we obtain the timed observation table $O_3 = (P_3, S_3, T_3)$ in Fig. 2e, which is cohesive. Figure 2c shows the DTA $\mathcal{A}^3_{\mathrm{hyp}}$ constructed from O_3. Since $\mathcal{L}(\mathcal{A}^3_{\mathrm{hyp}}) = \mathcal{L}_{\mathrm{tgt}}$ holds, Algorithm 4 finishes returning $\mathcal{A}^3_{\mathrm{hyp}}$.

By the use of equivalence queries, Algorithm 4 returns a DTA recognizing the target language if it terminates, which is formally as follows.

Theorem 28 (correctness). *For any target timed language $\mathcal{L}_{\mathrm{tgt}}$, if Algorithm 4 terminates, for the resulting DTA $\mathcal{A}_{\mathrm{hyp}}$, $\mathcal{L}(\mathcal{A}_{\mathrm{hyp}}) = \mathcal{L}_{\mathrm{tgt}}$ holds.* □

Moreover, Algorithm 4 terminates for any recognizable timed language $\mathcal{L}_{\mathrm{tgt}}$ essentially because of the finiteness of $\mathcal{SE}(\Sigma)/\sim^{\mathcal{E}(\Sigma)}_{\mathcal{L}_{\mathrm{tgt}}}$.

Theorem 29 (termination). *For any recognizable timed language $\mathcal{L}_{\mathrm{tgt}}$, Algorithm 4 terminates and returns a DTA $\mathcal{A}$ satisfying $\mathcal{L}(\mathcal{A}) = \mathcal{L}_{\mathrm{tgt}}$.*

Proof (Theorem 29). By the recognizability of $\mathcal{L}_{\mathrm{tgt}}$ and Theorem 19, $\mathcal{SE}(\Sigma)/\sim^{\mathcal{E}(\Sigma)}_{\mathcal{L}_{\mathrm{tgt}}}$ is finite. Let $N = |\mathcal{SE}(\Sigma)/\sim^{\mathcal{E}(\Sigma)}_{\mathcal{L}_{\mathrm{tgt}}}|$. Since each execution of line 5 adds p to P, where p is such that for any $p' \in P$, $p \not\sim^{\mathcal{E}(\Sigma)}_{\mathcal{L}_{\mathrm{tgt}}} p'$ holds, it is executed at most N times. Since each execution of line 8 refines S, i. e., it increases $|\mathcal{SE}(\Sigma)/\sim^{S}_{\mathcal{L}_{\mathrm{tgt}}}|$, line 8 is executed at most N times. For any $(u, \Lambda) \in \mathcal{SE}(\Sigma)$, if Λ contains $\mathbb{T}_{i,|u|} = d$ for some $i \in \{0, 1, \ldots, |u|\}$ and $d \in \mathbb{N}$, we have $\mathrm{succ}^t((u, \Lambda)) \subseteq \mathrm{ext}^t((u, \Lambda))$. Therefore, line 10 is executed at most N times. Since S is strictly refined in line 14, i. e., it increases $|\mathcal{SE}(\Sigma)/\sim^{S}_{\mathcal{L}_{\mathrm{tgt}}}|$, line 14 is executed at most N times. By Theorem 26, once $\sim^{S}_{\mathcal{L}_{\mathrm{tgt}}}$ saturates to $\sim^{\mathcal{E}(\Sigma)}_{\mathcal{L}_{\mathrm{tgt}}}$, `MakeDTA` returns the correct DTA. Overall, Algorithm 4 terminates. □

4.5 Learning with a Normal Teacher

We briefly show how to learn a DTA only with membership and equivalence queries. We reduce a symbolic membership query to finitely many membership queries, answerable by a normal teacher. See Appendix B.1 of [29] for detail.

Let (u, Λ) be the elementary language given in a symbolic membership query. Since Λ is bounded, we can construct a finite and disjoint set of simple and canonical timed conditions $\Lambda'_1, \Lambda'_2, \ldots, \Lambda'_n$ satisfying $\bigvee_{1 \le i \le n} \Lambda'_i = \Lambda$ by a simple enumeration. For any simple elementary language $(u', \Lambda') \in \mathcal{SE}(\Sigma)$ and timed words $w, w' \in (u', \Lambda')$, we have $w \in \mathcal{L} \iff w' \in \mathcal{L}$. Thus, we can construct $\mathtt{mem}^{\mathtt{sym}}_{\mathcal{L}}((u, \Lambda))$ by making a membership query $\mathtt{mem}_{\mathcal{L}}(w)$ for each such $(u', \Lambda') \subseteq (u, \Lambda)$ and for some $w \in (u', \Lambda')$. We need such an exhaustive search, instead of a binary search, because $\mathtt{mem}^{\mathtt{sym}}_{\mathcal{L}}((u, \Lambda))$ may be non-convex.

Assume Λ is a canonical timed condition. Let M be the size of the variables in Λ and I be the largest difference between the upper bound and the lower bound for some $\mathbb{T}_{i,j}$ in Λ. The size n of the above decomposition is bounded by $(2 \times I + 1)^{1/2 \times M \times (M+1)}$, which exponentially blows up with respect to M.

In our algorithm, we only make symbolic membership queries with elementary languages of the form $p \cdot s$, where p and s are simple elementary languages. Therefore, I is at most 2. However, even with such an assumption, the number of the necessary membership queries blows up exponentially to the size of the variables in Λ.

4.6 Complexity Analysis

After each equivalence query, our DTA learning algorithm strictly refines S or terminates. Thus, the number of equivalence queries is at most N. In the proof of Theorem 29, we observe that the size of P is at most $2N$. Therefore, the number $(|P| + |\mathrm{succ}(P)|) \times |S|$ of the cells in the timed observation table is at most $(2N + 2N \times (|\Sigma| + 1)) \times N = 2N^2|\Sigma| + 2$. Let J be the upper bound of i in the analysis of *cex* returned by equivalence queries (Algorithm 3). For each equivalence query, the number of membership queries in Algorithm 3 is bounded by $\lceil \log J \rceil$, and thus, it is, in total, bounded by $N \times \lceil \log J \rceil$. Therefore, if the learner can use symbolic membership queries, the total number of queries is bounded by a polynomial of N and J. In Sect. 4.5, we observe that the number of membership queries to implement a symbolic membership query is at most exponential to M. Since P is prefix-closed, M is at most N. Overall, if the learner cannot use symbolic membership queries, the total number of queries is at most exponential to N.

Table 1. Summary of the results for Random. Each row index $|L|\_|\Sigma|\_K_C$ shows the number of locations, the alphabet size, and the upper bound of the maximum constant in the guards, respectively. The row "count" shows the number of instances finished in 3 h. Cells with the best results are highlighted.

		# of Mem. queries			# of Eq. queries			Exec. time [sec.]			count
		max	mean	min	max	mean	min	max	mean	min	
3_2_10	LearnTA	35,268	14,241	2,830	**11**	**6**	**4**	2.32e+00	6.68e-01	**4.50e-02**	**10/10**
	OneSMT	**468**	**205**	**32**	13	8	5	**9.58e-01**	**2.89e-01**	6.58e-02	**10/10**
4_2_10	LearnTA	194,442	55,996	10,619	**14**	**7**	**4**	2.65e+01	7.98e+00	4.88e-01	**10/10**
	OneSMT	**985**	**451**	**255**	16	12	7	**3.53e-01**	**2.09e-01**	**1.27e-01**	**10/10**
4_4_20	LearnTA	1,681,769	858,759	248,399	**21**	**15**	**10**	8.34e+03	1.41e+03	3.23e+01	8/10
	OneSMT	**5,329**	**3,497**	**1,740**	42	32	26	**2.19e+00**	**1.42e+00**	**8.27e-01**	**10/10**
5_2_10	LearnTA	627,980	119,906	8,121	**19**	**8**	**5**	1.67e+02	2.28e+01	**1.96e-01**	**10/10**
	OneSMT	**1,332**	**876**	**359**	22	16	12	**5.20e-01**	**3.66e-01**	2.58e-01	**10/10**
6_2_10	LearnTA	555,939	106,478	2,912	**14**	**9**	**6**	2.44e+02	2.81e+01	**4.40e-02**	**10/10**
	OneSMT	**3,929**	**1,894**	**104**	35	20	11	**1.72e+00**	**8.01e-01**	1.73e-01	**10/10**

Let $\mathcal{A}_{\mathrm{tgt}} = (\Sigma, L, l_0, C, I, \Delta, F)$ be a DTA recognizing $\mathcal{L}_{\mathrm{tgt}}$. As we observe in the proof of Lemma 33 of [29], N is bounded by the size of the state space of the region automaton [4] of $\mathcal{A}_{\mathrm{tgt}}$, N is at most $|C|! \times 2^{|C|} \times \prod_{c \in C}(2K_c + 2) \times |L|$, where K_c is the largest constant compared with $c \in C$ in $\mathcal{A}_{\mathrm{tgt}}$. Thus, without symbolic

membership queries, the total number of queries is at most doubly-exponential to $|C|$ and singly exponential to $|L|$. We remark that when $|C| = 1$, the total number of queries is at most singly exponential to $|L|$ and K_c, which coincides with the worst-case complexity of the one-clock DTA learning algorithm in [30].

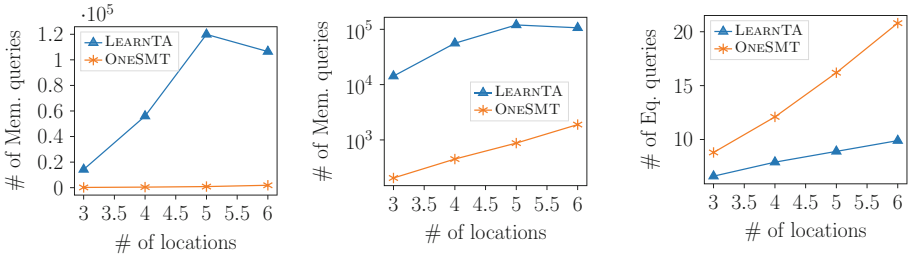
5 Experiments

We experimentally evaluated our DTA learning algorithm using our prototype library LearnTA[4] implemented in C++. In LearnTA, the equivalence queries are answered by a zone-based reachability analysis using the fact that DTAs are closed under complement [4]. We pose the following research questions.

RQ1 How is the scalability of LearnTA to the language complexity?
RQ2 How is the efficiency of LearnTA for practical benchmarks?

For the benchmarks with one clock variable, we compared LearnTA with one of the latest one-clock DTA learning algorithms [1,30], which we call OneSMT. OneSMT is implemented in Python with Z3 [23] for constraint solving.

For each execution, we measured the number of queries and the total execution time, including the time to answer the queries. For the number of queries, we report the number with memoization, i. e., we count the number of the queried timed words (for membership queries) and the counterexamples (for equivalence queries). We conducted all the experiments on a computing server with Intel Core i9-10980XE 125 GiB RAM that runs Ubuntu 20.04.5 LTS. We used 3 h as the timeout.



(a) Membership queries (b) Membership queries (log scale) (c) Equivalence queries

Fig. 3. The number of locations and the number of queries for $|L|$_2_10 in Random, where $|L| \in \{3, 4, 5, 6\}$

[4] LearnTA is publicly available at https://github.com/masWag/LearnTA. The artifact of the experiments is available at https://doi.org/10.5281/zenodo.7875383.

Table 2. Summary of the target DTAs and the results for Unbalanced. $|L|$ is the number of locations, $|\Sigma|$ is the alphabet size, $|C|$ is the number of clock variables, and K_C is the maximum constant in the guards in the DTA.

		$\|L\|$	$\|\Sigma\|$	$\|C\|$	K_C	# of Mem. queries	# of Eq. queries	Exec. time [sec.]
Unbalanced:1	LearnTA	5	1	1	2	51	2	2.00e-03
Unbalanced:2	LearnTA	5	1	2	4	576,142	3	3.64e+01
Unbalanced:3	LearnTA	5	1	3	4	403,336	4	2.24e+01
Unbalanced:4	LearnTA	5	1	4	6	4,142,835	5	2.40e+02
Unbalanced:5	LearnTA	5	1	5	6	10,691,400	5	8.68e+02

5.1 RQ1: Scalability with Respect to the Language Complexity

To evaluate the scalability of LearnTA, we used randomly generated DTAs from [5] (denoted as Random) and our original DTAs (denoted as Unbalanced). Random consists of five classes: 3_2_10, 4_2_10, 4_4_20, 5_2_10, and 6_2_10, where each value of $|L|$_$|\Sigma|$_K_C is the number of locations, the alphabet size, and the upper bound of the maximum constant in the guards in the DTAs, respectively. Each class consists of 10 randomly generated DTAs. Unbalanced is our original benchmark inspired by the "unbalanced parentheses" timed language from [10]. Unbalanced consists of five DTAs with different complexity of timing constraints. Table 2 summarizes their complexity.

Table 1 and 3 summarize the results for Random, and Table 2 summarizes the results for Unbalanced. Table 1 shows that LearnTA requires more membership queries than OneSMT. This is likely because of the difference in the definition of prefixes and successors: OneSMT's definitions are discrete (e. g., prefixes are only with respect to events with time elapse), whereas ours are both continuous and discrete (e. g., we also consider prefixes by trimming the dwell time in the end); Since our definition makes significantly more prefixes, LearnTA tends to require much more membership queries. Another, more high-level reason is that LearnTA learns a DTA without knowing the number of the clock variables, and many more timed words are potentially helpful for learning. Table 1 shows that LearnTA requires significantly many membership queries for 4_4_20. This is likely because of the exponential blowup with respect to K_C, as discussed in Sect. 4.6. In Fig. 3, we observe that for both LearnTA and OneSMT, the number of membership queries increases nearly exponentially to the number of locations. This coincides with the discussion in Sect. 4.6.

In contrast, Table 1 shows that LearnTA requires fewer equivalence queries than OneSMT. This suggests that the cohesion in Definition 24 successfully detected contradictions in observation before generating a hypothesis, whereas OneSMT mines timing constraints mainly by equivalence queries and tends to require more equivalence queries. In Fig. 3c, we observe that for both LearnTA and OneSMT, the number of equivalence queries increases nearly linearly to the number of locations. This also coincides with the complexity analysis in Sect. 4.6. Figure 3c also shows that the number of equivalence queries increases faster in OneSMT than in LearnTA.

Table 3. Summary of the target DTA and the results for practical benchmarks. The columns are the same as Table 2. Cells with the best results are highlighted.

		$\|L\|$	$\|\Sigma\|$	$\|C\|$	K_C	# of Mem. queries	# of Eq. queries	Exec. time [sec.]
AKM	LearnTA	17	12	1	5	12,263	**11**	**5.85e-01**
	OneSMT	17	12	1	5	**3,453**	49	7.97e+00
CAS	LearnTA	14	10	1	27	66,067	**17**	**4.65e+00**
	OneSMT	14	10	1	27	**4,769**	18	9.58e+01
Light	LearnTA	5	5	1	10	3,057	**7**	**3.30e-02**
	OneSMT	5	5	1	10	**210**	7	9.32e-01
PC	LearnTA	26	17	1	10	245,134	**23**	**6.49e+01**
	OneSMT	26	17	1	10	**10,390**	29	1.24e+02
TCP	LearnTA	22	13	1	2	11,300	**15**	**3.82e-01**
	OneSMT	22	13	1	2	**4,713**	32	2.20e+01
Train	LearnTA	6	6	1	10	13,487	**8**	**1.72e-01**
	OneSMT	6	6	1	10	**838**	13	1.13e+00
FDDI	LearnTA	16	5	7	6	**9,986,271**	**43**	**3.00e+03**

Table 2 also suggests a similar tendency: the number of membership queries rapidly increases to the complexity of the timing constraints; In contrast, the number of equivalence queries increases rather slowly. Moreover, LearnTA is scalable enough to learn a DTA with five clock variables within 15 min.

Table 1 also suggests that LearnTA does not scale well to the maximum constant in the guards, as observed in Sect. 4.6. However, we still observe that LearnTA requires fewer equivalence queries than OneSMT. Overall, compared with OneSMT, LearnTA has better scalability in the number of equivalence queries and worse scalability in the number of membership queries.

5.2 RQ2: Performance on Practical Benchmarks

To evaluate the practicality of LearnTA, we used seven benchmarks: AKM, CAS, Light, PC, TCP, Train, and FDDI. Table 3 summarizes their complexity. All the benchmarks other than FDDI are taken from [30] (or its implementation [1]). FDDI is taken from TChecker [2]. We use the instance of FDDI with two processes.

Table 3 summarizes the results for the benchmarks from practical applications. We observe, again, that LearnTA requires more membership queries and fewer equivalence queries than OneSMT. However, for these benchmarks, the difference in the number of membership queries tends to be much smaller than in Random. This is because these benchmarks have simpler timing constraints than Random for the exploration by LearnTA. In AKM, Light, PC, TCP, and Train, the clock variable can be reset at every edge without changing the language. For such a DTA, all simple elementary languages are equivalent in terms of the Nerode-style congruence if we have the same edge at their last event and the same dwell time after it. If two simple elementary languages are equivalent, LearnTA explores the successors of only one of them, and the exploration is

relatively efficient. We have a similar situation in CAS. Moreover, in many of these DTAs, only a few edges have guards. Overall, despite the large number of locations and alphabets, these languages' complexities are mild for LearnTA.

We also observe that, surprisingly, for all of these benchmarks, LearnTA took a shorter time for DTA learning than OneSMT. This is partly because of the difference in the implementation language (i. e., C++ vs. Python) but also because of the small number of equivalence queries and the mild number of membership queries. Moreover, although it requires significantly more queries, LearnTA successfully learned FDDI with seven clock variables. Overall, such efficiency on benchmarks from practical applications suggests the potential usefulness of LearnTA in some realistic scenarios.

6 Conclusions and Future Work

Extending the L* algorithm, we proposed an active learning algorithm for DTAs. Our extension is by our Nerode-style congruence for recognizable timed languages. We proved the termination and the correctness of our algorithm. We also proved that our learning algorithm requires a polynomial number of queries with a smart teacher and an exponential number of queries with a normal teacher. Our experiment results also suggest the practical relevance of our algorithm.

One of the future directions is to extend more recent automata learning algorithms (e. g., TTT algorithm [19] to improve the efficiency) to DTA learning. Another direction is constructing a *passive* DTA learning algorithm based on our congruence and an existing passive DFA learning algorithm. It is also a future direction to apply our learning algorithm for practical usage, e. g., identification of black-box systems and testing black-box systems with black-box checking [22, 24, 28]. Optimization of the algorithm, e. g., by incorporating clock information is also a future direction.

Acknowledgements. This work is partially supported by JST ACT-X Grant No. JPMJAX200U, JST PRESTO Grant No. JPMJPR22CA, JST CREST Grant No. JPMJCR2012, and JSPS KAKENHI Grant No. 22K17873.

References

1. GitHub: Leslieaj/DOTALearningSMT. https://github.com/Leslieaj/DOTALearningSMT, (Accessed 10 Jan 2023)
2. Github: ticktac-project/tchecker. https://github.com/ticktac-project/tchecker, (Accessed 20 Jan 2023)
3. Aichernig, B.K., Pferscher, A., Tappler, M.: From passive to active: learning timed automata efficiently. In: Lee, R., Jha, S., Mavridou, A., Giannakopoulou, D. (eds.) NFM 2020. LNCS, vol. 12229, pp. 1–19. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-55754-6_1
4. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**(2), 183–235 (1994). https://doi.org/10.1016/0304-3975(94)90010-8

5. An, J., Chen, M., Zhan, B., Zhan, N., Zhang, M.: Learning one-clock timed automata. In: TACAS 2020. LNCS, vol. 12078, pp. 444–462. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45190-5_25
6. An, J., Wang, L., Zhan, B., Zhan, N., Zhang, M.: Learning real-time automata. Science China Inf. Sci. **64**(9), 1–17 (2021). https://doi.org/10.1007/s11432-019-2767-4
7. An, J., Zhan, B., Zhan, N., Zhang, M.: Learning nondeterministic real-time automata. ACM Trans. Embed. Comput. Syst. **20**(5s), 99:1–99:26 (2021). https://doi.org/10.1145/3477030,
8. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75**(2), 87–106 (1987). https://doi.org/10.1016/0890-5401(87)90052-6
9. Argyros, G., D'Antoni, L.: The learnability of symbolic automata. In: Chockler, H., Weissenbacher, G. (eds.) CAV 2018. LNCS, vol. 10981, pp. 427–445. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96145-3_23
10. Asarin, E., Caspi, P., Maler, O.: Timed regular expressions. J. ACM **49**(2), 172–206 (2002). https://doi.org/10.1145/506147.506151
11. Bersani, M.M., Rossi, M., San Pietro, P.: A logical characterization of timed regular languages. Theor. Comput. Sci. **658**, 46–59 (2017). https://doi.org/10.1016/j.tcs.2016.07.020
12. Bojańczyk, M., Lasota, S.: A machine-independent characterization of timed languages. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012. LNCS, vol. 7392, pp. 92–103. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31585-5_12
13. Bouyer, P., Petit, A., Thérien, D.: An algebraic characterization of data and timed languages. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 248–261. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44685-0_17
14. Drews, S., D'Antoni, L.: Learning symbolic automata. In: Legay, A., Margaria, T. (eds.) TACAS 2017. LNCS, vol. 10205, pp. 173–189. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54577-5_10
15. Grinchtein, O., Jonsson, B., Leucker, M.: Learning of event-recording automata. Theor. Comput. Sci. **411**(47), 4029–4054 (2010). https://doi.org/10.1016/j.tcs.2010.07.008
16. Grinchtein, O., Jonsson, B., Pettersson, P.: Inference of event-recording automata using timed decision trees. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 435–449. Springer, Heidelberg (2006). https://doi.org/10.1007/11817949_29
17. Henry, L., Jéron, T., Markey, N.: Active learning of timed automata with unobservable resets. In: Bertrand, N., Jansen, N. (eds.) FORMATS 2020. LNCS, vol. 12288, pp. 144–160. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-57628-8_9
18. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to automata theory, languages, and computation, 3rd edn. Addison-Wesley, Pearson international edition (2007)
19. Isberner, M., Howar, F., Steffen, B.: The TTT algorithm: a redundancy-free approach to active automata learning. In: Bonakdarpour, B., Smolka, S.A. (eds.) RV 2014. LNCS, vol. 8734, pp. 307–322. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11164-3_26
20. Isberner, M., Steffen, B.: An abstract framework for counterexample analysis in active automata learning. In: Clark, A., Kanazawa, M., Yoshinaka, R. (eds.) Proceedings of the 12th International Conference on Grammatical Inference, ICGI

2014, Kyoto, Japan, 17–19 September 2014. JMLR Workshop and Conference Proceedings, vol. 34, pp. 79–93. JMLR.org (2014). http://proceedings.mlr.press/v34/isberner14a.html
21. Maler, O., Pnueli, A.: On recognizable timed languages. In: Walukiewicz, I. (ed.) FoSSaCS 2004. LNCS, vol. 2987, pp. 348–362. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24727-2_25
22. Meijer, J., van de Pol, J.: Sound black-box checking in the learnlib. Innov. Syst. Softw. Eng. **15**(3–4), 267–287 (2019). https://doi.org/10.1007/s11334-019-00342-6
23. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_24
24. Peled, D.A., Vardi, M.Y., Yannakakis, M.: Black box checking. In: Wu, J., Chanson, S.T., Gao, Q. (eds.) Formal Methods for Protocol Engineering and Distributed Systems, FORTE XII / PSTV XIX 1999, IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XII) and Protocol Specification, Testing and Verification (PSTV XIX), 5–8 October 1999, Beijing, China. IFIP Conference Proceedings, vol. 156, pp. 225–240. Kluwer (1999)
25. Rivest, R.L., Schapire, R.E.: Inference of finite automata using homing sequences. Inf. Comput. **103**(2), 299–347 (1993). https://doi.org/10.1006/inco.1993.1021
26. Shijubo, J., Waga, M., Suenaga, K.: Efficient black-box checking via model checking with strengthened specifications. In: Feng, L., Fisman, D. (eds.) RV 2021. LNCS, vol. 12974, pp. 100–120. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88494-9_6
27. Tappler, M., Aichernig, B.K., Larsen, K.G., Lorber, F.: Time to learn – learning timed automata from tests. In: André, É., Stoelinga, M. (eds.) FORMATS 2019. LNCS, vol. 11750, pp. 216–235. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29662-9_13
28. Waga, M.: Falsification of cyber-physical systems with robustness-guided black-box checking. In: Ames, A.D., Seshia, S.A., Deshmukh, J. (eds.) HSCC 2020: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, 21–24 April 2020, pp. 11:1–11:13. ACM (2020). https://doi.org/10.1145/3365365.3382193
29. Waga, M.: Active learning of deterministic timed automata with myhill-nerode style characterization. CoRR abs/ arXiv: 2305.17742 (2023). http://arxiv.org/abs/2305.17742
30. Xu, R., An, J., Zhan, B.: Active learning of one-clock timed automata using constraint solving. In: Bouajjani, A., Holík, L., Wu, Z. (eds.) Automated Technology for Verification and Analysis - 20th International Symposium, ATVA 2022, Virtual Event, 25–28 October 2022, Proceedings. LNCS, vol. 13505, pp. 249–265. Springer (2022). https://doi.org/10.1007/978-3-031-19992-9_16
31. Zhang, H., Feng, L., Li, Z.: Control of black-box embedded systems by integrating automaton learning and supervisory control theory of discrete-event systems. IEEE Trans. Autom. Sci. Eng. **17**(1), 361–374 (2020). https://doi.org/10.1109/TASE.2019.2929563