# Learning One-Clock Timed Automata

Jie An<sup>1</sup>, Mingshuai Chen<sup>2</sup>, **Bohua Zhan**<sup>3</sup>, Naijun Zhan<sup>3</sup>, Miaomiao Zhang<sup>1</sup>

School of Software Engineering, Tongji University, Shanghai, China
 Lehrstuhl für Informatik 2, RWTH Aachen University, Aachen, Germany
 SKLCS, Institute of Software, Chinese Academy of Sciences, Beijing, China

bzhan@ios.ac.cn

TACAS · March 30, 2021







### 1 Introduction and motivation

- Short introduction to model/automaton learning
- L\* : Classic automaton learning of DFA
- Motivation

### 2 Learning one-clock timed automata

- Basic idea
- Learning from a smart teacher
- Learning from a normal teacher

### **3** Conclusion and future work

### Outline

### **1** Introduction and motivation

- Short introduction to model/automaton learning
- L\* : Classic automaton learning of DFA
- Motivation

### 2 Learning one-clock timed automata

### Conclusion and future work

• Machine learning









• Model/Automaton learning





• Model/Automaton learning



### Model/Automaton learning



© The figure comes from Irini-Eleftheria Mens.

- Dana Angluin proposed an online, active, and exact learning framework *L*\* for Deterministic Finite Automata (DFA) in 1987 [2].
- Two kinds of queries : membership query and equivalence query.

- Dana Angluin proposed an online, active, and exact learning framework *L*\* for Deterministic Finite Automata (DFA) in 1987 [2].
- Two kinds of queries : membership query and equivalence query.

Learner	Teacher

- Dana Angluin proposed an online, active, and exact learning framework *L*\* for Deterministic Finite Automata (DFA) in 1987 [2].
- Two kinds of queries : membership query and equivalence query.



- Dana Angluin proposed an online, active, and exact learning framework *L*\* for Deterministic Finite Automata (DFA) in 1987 [2].
- Two kinds of queries : membership query and equivalence query.



- More recent work extends *L*\* algorithm to different models
  - Mealy machines [9], I/O automata [1], register automata [6], NFA [3], Büchi automata [7], symbolic automata [8, 4] and MDP [10], etc..

- More recent work extends  $L^*$  algorithm to different models
  - Mealy machines [9], I/O automata [1], register automata [6], NFA [3], Büchi automata [7], symbolic automata [8, 4] and MDP [10], etc..
- Motivation
  - How to actively learn a timed model for a real-time system?
- Related work
  - Active learning of event-recording automata [5].
  - Passive identification of timed automata in the limit via fitting a labelled sample  $S = (S_+, S_-)$  [12].
  - Passive learning of timed automata via Genetic Programming and testing [11].

### Introduction and motivation

### 2 Learning one-clock timed automata

- Basic idea
- Learning from a smart teacher
- Learning from a normal teacher

### 3 Conclusion and future work

- Learning (regular) timed-automata with a single clock.
- Challenges
  - State now includes both location and clock value.
  - Determining the guard condition on transitions.
  - Determining reset information on transitions.
  - (related to the previous points) Matching time observed from outside to internal clock used on the guards.
- Solutions of learning deterministic one-clock timed automata (DOTA).
  - © A normalization map from delay timed words (outside) to logical timed words (inside).
  - © Utilize a partition function to map *logical-timed* values to finite intervals (similar to learning symbolic automata).
  - © First consider the case of a smart teacher who can tell the learner reset informations. Then drop the assumption (i.e. reduction to a normal teacher) by guessing reset information.

- The DOTA  $\mathcal A$  recognizes the target language  $\mathcal L$ .
- $\Sigma = \{a, b\}$ ;  $B = \{\top, \bot\}$  where  $\top$  is for reset,  $\bot$  otherwise.



- The DOTA  ${\mathcal A}$  recognizes the target language  ${\mathcal L}.$
- $\Sigma = \{a, b\}$ ;  $\mathcal{B} = \{\top, \bot\}$  where  $\top$  is for reset,  $\bot$  otherwise.
- A is a *complete* DOTA of A. Timed language  $\mathcal{L}(A) = \mathcal{L}(A) = \mathcal{L}$ .



- The DOTA  $\mathcal A$  recognizes the target language  $\mathcal L$ .
- $\Sigma = \{a, b\}$ ;  $\mathcal{B} = \{\top, \bot\}$  where  $\top$  is for reset,  $\bot$  otherwise.
- A is a *complete* DOTA of A. Timed language  $\mathcal{L}(A) = \mathcal{L}(A) = \mathcal{L}$ .
- Delay timed words (Σ × ℝ<sub>≥0</sub>)\* : outside observations;

e.g.  $\omega = (b, 0)(a, 1.1)(b, 1)$  is an accepting timed words.



- The DOTA  ${\mathcal A}$  recognizes the target language  ${\mathcal L}.$
- $\Sigma = \{a, b\}$ ;  $B = \{\top, \bot\}$  where  $\top$  is for reset,  $\bot$  otherwise.
- A is a *complete* DOTA of A. Timed language  $\mathcal{L}(A) = \mathcal{L}(A) = \mathcal{L}$ .
- Delay timed words (Σ × ℝ≥0)\*: outside observations;
   e.g. ω = (b,0)(a,1.1)(b,1) is an accepting timed words.
- Reset-logical timed words (Σ × ℝ<sub>≥0</sub> × B)\* : inside logical actions;
   e.g. γ<sub>r</sub> = (b, 0, T)(a, 1.1, ⊥)(b, 2.1, T) is the reset-logical counterpart of ω.
   Logical counterpart γ = (b, 0)(a, 1.1)(b, 2.1).

#### Example



• Given a DOTA A, *L<sub>r</sub>*(A) represents the recognized reset-logical timed language of A; *L*(A) represents the logical timed language.

#### Theorem

Given two DOTAs  $\mathbb{A}$  and  $\mathcal{H}$ , if  $L_r(\mathbb{A}) = L_r(\mathcal{H})$ , then  $\mathcal{L}(\mathbb{A}) = \mathcal{L}(\mathcal{H})$ .

- Given a DOTA A, *L<sub>r</sub>*(A) represents the recognized reset-logical timed language of A; *L*(A) represents the logical timed language.
- Guiding principle : learning the (delayed) timed language of a DOTA A can be reduced to learning the reset-logical timed language of A.

#### Theorem

Given two DOTAs  $\mathbb{A}$  and  $\mathcal{H}$ , if  $L_r(\mathbb{A}) = L_r(\mathcal{H})$ , then  $\mathcal{L}(\mathbb{A}) = \mathcal{L}(\mathcal{H})$ .

- Given a DOTA A, *L<sub>r</sub>*(A) represents the recognized reset-logical timed language of A; *L*(A) represents the logical timed language.
- Guiding principle : learning the (delayed) timed language of a DOTA A can be reduced to learning the reset-logical timed language of A.
- Smart teacher setting : membership queries are logical timed words, teacher responds with reset information.

#### Theorem

Given two DOTAs  $\mathbb{A}$  and  $\mathcal{H}$ , if  $L_r(\mathbb{A}) = L_r(\mathcal{H})$ , then  $\mathcal{L}(\mathbb{A}) = \mathcal{L}(\mathcal{H})$ .

#### Definition (Reset-logical-timed observation table)

A reset-logical-timed observation table for a DOTA  $\mathbb{A}$  is a 7-tuple  $\mathbf{T} = (\Sigma, \Sigma, \Sigma_r, S, R, E, f)$ where  $\Sigma$  is the finite alphabet;  $\Sigma = \Sigma \times \mathbb{R}_{\geq 0}$  is the infinite set of logical-timed actions;  $\Sigma_r = \Sigma \times \mathbb{R}_{\geq 0} \times \mathcal{B}$  is the infinite set of reset-logical-timed actions;  $S, R \subset \Sigma_r^*$  and  $E \subset \Sigma^*$ are finite sets of words, where S is called the set of prefixes, R the boundary, and E the set of suffixes. Specifically,

- S and R are disjoint, i.e., S ∪ R = S ⊎ R;
- The empty word is by default both a prefix and a suffix, i.e.,  $\epsilon \in \mathbf{E}$  and  $\epsilon \in \mathbf{S}$ ;
- $f: (\mathbf{S} \cup \mathbf{R}) \cdot \mathbf{E} \mapsto \{-, +\}$  is a classification function such that for a reset-logical-timed word  $\gamma_r, \gamma_r \cdot \mathbf{e} \in (\mathbf{S} \cup \mathbf{R}) \cdot \mathbf{E}, f(\gamma_r \cdot \mathbf{e}) = -i \operatorname{fl} \Pi_{\{1,2\}} \gamma_r \cdot \mathbf{e}$  is invalid<sup>1</sup>, otherwise if  $\Pi_{\{1,2\}} \gamma_r \cdot \mathbf{e} \notin L(\mathbb{A}), f(\gamma_r \cdot \mathbf{e}) = -$ , and  $f(\gamma_r \cdot \mathbf{e}) = +i \operatorname{fl} \Pi_{\{1,2\}} \gamma_r \cdot \mathbf{e} \in L(\mathbb{A});$

1. The projection of an *n*-tuple x onto its first two components is denoted by  $\Pi_{\{1,2\}}x$ , which extends to a sequence

of tuples as 
$$\Pi_{\{1,2\}}(\mathbf{x}_1,\ldots,\mathbf{x}_k) = (\Pi_{\{1,2\}}\mathbf{x}_1,\ldots,\Pi_{\{1,2\}}\mathbf{x}_k).$$

- Reduced
  - $\forall s, s' \in \mathbf{S} : s \neq s' \text{ implies } row(s) \neq row(s');$
- Closed
  - $\forall r \in \mathbf{R}, \exists s \in \mathbf{S} : row(s) = row(r);$
- Consistent
  - $\forall \gamma_r, \gamma_r' \in \mathbf{S} \cup \mathbf{R}, row(\gamma_r) = row(\gamma_r') \text{ implies } row(\gamma_r \cdot \sigma_r) = row(\gamma_r' \cdot \sigma_r'), \text{ for all } \sigma_r, \sigma_r' \in \Sigma_r \text{ satisfying } \gamma_r \cdot \sigma_r, \gamma_r' \cdot \sigma_r' \in \mathbf{S} \cup \mathbf{R} \text{ and } \Pi_{\{1,2\}}\sigma_r = \Pi_{\{1,2\}}\sigma_r';$
- Evidence-closed
  - $\forall s \in S$  and  $\forall e \in E$ , the reset-logical-timed word  $\pi(\Pi_{\{1,2\}}s \cdot e)$  belongs to  $S \cup R^2$ .

2. For the sake of simplicity, we define a function  $\pi$  that maps a logical-timed word to its unique reset-logical-timed counterpart in membership queries.

Τ	$\epsilon$	• • •
$\epsilon$	-	
$(a, 1.1, \bot)$	+	
(a,0, op)	_	
$(b, 0, \top)$	_	
$(a, 1.1, \bot)(a, 0, \top)$	_	
$(\boldsymbol{a}, 1.1, \bot)(\boldsymbol{b}, 0, \top)$	_	

S	Т	$\epsilon$	•••
	$\epsilon$	_	
( <i>a</i> , 1.	$1, \perp)$	+	
( <i>a</i> ,	$(0, \top)$	_	
( <i>b</i> ,	$(0, \top)$	_	
$(a, 1.1, \perp)(a,$	$0, \top)$	_	
$(\boldsymbol{a}, 1.1, \bot)(\boldsymbol{b},$	$0, \top)$	_	

The prefixes set **S** indicates the locations

<b>S</b> <sub>T</sub>	$\epsilon$	•••
E	-	
$(a, 1.1, \perp)$	+	
(a,0, op)	-	
(b,0, op)	-	
$(a, 1.1, \bot)(a, 0, \top)$	-	
$(a, 1.1, \bot)(b, 0, \top)$	_	

The prefixes set **S** indicates the locations

The boundary **R** indicates the transitions





The prefixes set **S** indicates the locations

The boundary **R** indicates the transitions

The suffixes set *E* distinguishes the locations





The prefixes set **S** indicates the locations

The boundary *R* indicates the transitions

The suffixes set *E* distinguishes the locations

**Body** records whether automaton accepts logical timed words

D

Т	$\epsilon$	• • •
$\epsilon$	-	
( <i>a</i> , 1.1, ⊥)	+	
( <i>a</i> , 0, ⊤)	_	]
$(b, 0, \top)$	_	
$(a, 1.1, \perp)(a, 0, \top)$	_	
$(\boldsymbol{a}, 1.1, \bot)(\boldsymbol{b}, 0, \top)$	_	

The prefixes set **S** indicates the locations

The boundary **R** indicates the transitions

The suffixes set **E** distinguishes the locations

**Body** records whether automaton accepts logical timed words

 $\begin{array}{l} \operatorname{accepts}\left(\textit{a},1.1\right)\cdot\epsilon \text{ and}\\ \\ \operatorname{gives the reset information}\bot\end{array}$ 

does not accept  $(a, 0) \cdot \epsilon$  and gives the reset information  $\top$ 

 Given a target timed language *L* which is recognized by a DOTA A, let
 n = |Q| be the number of locations of A, m = |Σ| the size of the alphabet,
 and κ the maximal constant appearing in the clock constraints of A.

#### Theorem

The learning process with a smart teacher terminates and returns a DOTA which recognizes the target timed language  $\mathcal{L}$ .

#### Theorem

The complexity of the algorithm is  $\mathcal{O}(mn^5\kappa^4)$  for number of membership queries, and  $\mathcal{O}(mn^2\kappa^3)$  for number of equivalence queries.

- In the normal teacher setting, the teacher responds to delay timed words, and no longer returns reset information in answers to membership and equivalence queries.
- The learner guesses the resets in order to convert between delay and logical timed words.

- In the normal teacher setting, the teacher responds to delay timed words, and no longer returns reset information in answers to membership and equivalence queries.
- The learner **guesses the resets** in order to convert between delay and logical timed words.
- Basic process
  - At every round, guess all needed resets and put all resulting table candidates into a set *ToExplore*.
  - Take out one table instance from the set *ToExplore*.
  - The operations on the table are same to those in the situation with a smart teacher.

### • Termination and complexity

- At every iteration, the learner selects the table instance which requires the least number of guesses.
- The learner keeps the correct table instance of each iteration in *ToExplore* since he guesses all reset informations.
- If  $\mathbf{T} = (\Sigma, \Sigma, \Sigma, \mathbf{r}, \mathbf{S}, \mathbf{R}, \mathbf{E}, f)$  is the final observation table for the correct candidate in the situation with a smart teacher, the learner can find it after checking  $\mathcal{O}(2^{(|\mathbf{S}|+|\mathbf{R}|) \times (1+\sum_{e_i \in \mathbf{E} \setminus \{e_i\}} (|e_i|-1))})$  table instances in the worst situation with a normal teacher.
- The process also may terminate and return a DOTA which is different to the one in the smart teacher situation.

#### Theorem

The learning process with a normal teacher terminates and returns a DOTA which recognizes the target timed language  $\mathcal{L}$ . It has exponential complexity in the number of membership and equivalence queries.

Case ID		#Membership			#Equivalence				Omana	tman
	— mean	N <sub>min</sub>	N <sub>mean</sub>	N <sub>max</sub>	 N <sub>min</sub>	N <sub>mean</sub>	N <sub>max</sub>		can	-mean
4_4_20	16.3	118	245.0	650	20	30.1	42	4	.5	24.7
7_2_10	16.9	568	920.8	1393	23	31.3	37	9	.1	14.6
7_4_10	25.7	348	921.7	1296	34	50.9	64	9	3	38.0
7 6 10	26.0	351	634.5	1050	35	44.7	70	7	.8	49.6
7_4_20	34.3	411	1183.4	1890	52	70.5	93	9	.5	101.7
10_4_20	39.1	920	1580.9	2160	61	73.1	88	11	.7	186.7
12_4_20	47.6	1090	2731.6	5733	66	97.4	125	16	0.0	521.8
14_4_20	58.4	1390	2238.6	4430	79	107.7	135	16	.0	515.5

Table 1 – Experimental results on random examples for the smart teacher situation.

Case ID :  $n_m\kappa$ , consisting of the number of locations, the size of the alphabet and the maximum constant appearing in the clock constraints, respectively, of the corresponding group of A's.

 $|\Delta|_{mean}$  : the average number of transitions in the corresponding group.

#Membership & #Equivalence : the number of conducted membership and equivalence queries, respectively.  $N_{\min}$  : the minimal,  $N_{mean}$  : the mean,  $N_{max}$  : the maximum.

 $n_{\text{mean}}$ : the average number of locations of the learned automata in the corresponding group.

 $t_{mean}$  : the average wall-clock time in seconds, including that taken by the learner and by the teacher.



**Figure 1** – Left : The functional specification of the TCP protocol with timing settings. Right : The learnt functional specification of the TCP protocol. Colors indicate the splitting of locations.

#### Table 2 – Experimental results on random examples for the normal teacher situation.

Case ID $ \Delta _{mean}$		#Membership			#Equivalence			Nmean	tmean	#Texplored	#Learnt
	i incon	N <sub>min</sub>	N <sub>mean</sub>	N <sub>max</sub>	N <sub>min</sub>	N <sub>mean</sub>	N <sub>max</sub>	mean	mean	explored	
3_2_10	4.8	43	83.7	167	5	8.8	14	3.0	0.9	149.1	10/10
4_2_10	6.8	67	134.0	345	6	13.3	24	4.0	7.4	563.0	10/10
5_2_10	8.8	75	223.9	375	9	15.2	24	5.0	35.5	2811.6	10/10
6_2_10	11.9	73	348.3	708	10	16.7	30	5.6	59.8	5077.6	7/10
4_4_20	16.3	231	371.0	564	27	30.9	40	4.0	137.5	8590.0	6/10

#Membership & #Equivalence : the number of conducted membership and equivalence queries with the cached methods, respectively. *N*<sub>min</sub> : the minimal, *N*<sub>mean</sub> : the mean, *N*<sub>max</sub> : the maximum.

 $T_{explored}$ : the average number of the explored table instances.

#Learnt : the number of the learnt DOTAs in the group (learnt/total).





### • Contributions

- Give an active learning algorithm with a smart teacher for DOTAs. It is an efficient (polynomial) algorithm. (white-box or gray-box)
- Give an active learning algorithm with a normal teacher for DOTAs. It has an exponential complexity increase. (black-box)

### • Contributions

- Give an active learning algorithm with a smart teacher for DOTAs. It is an efficient (polynomial) algorithm. (white-box or gray-box)
- Give an active learning algorithm with a normal teacher for DOTAs. It has an exponential complexity increase. (black-box)
- Future work
  - Extension to non-deterministic and multi-clock timed automata.
  - Improvements to efficiency of the algorithms.

### **Reference** I

- [1] F. Aarts and F. W. Vaandrager. Learning I/O automata. In CONCUR'10, pages 71–85, 2010.
- [2] D. Angluin. Learning regular sets from queries and counterexamples. Inf. Comput., 75(2):87–106, 1987.
- [3] B. Bollig, P. Habermehl, C. Kern, and M. Leucker. Angluin-style learning of NFA. In IJCAI'09, pages 1004–1009, 2009.
- S. Drews and L. D'Antoni.
   Learning symbolic automata.
   In TACAS'17, pages 173–189, 2017.
- [5] O. Grinchtein, B. Jonsson, and M. Leucker. Learning of event-recording automata. *Theor. Comput. Sci.*, 411(47):4029–4054, 2010.
- [6] F. Howar, B. Steffen, B. Jonsson, and S. Cassel. Inferring canonical register automata. In VMCAI'12, pages 251–266, 2012.

### **Reference II**

- Y. Li, Y. Chen, L. Zhang, and D. Liu.
   A novel learning algorithm for Büchi automata based on family of DFAs and classification trees.
   In *TACAS'17*, pages 208–226, 2017.
- [8] O. Maler and I. Mens. Learning regular languages over large alphabets. In TACAS'14, pages 485–499, 2014.
- [9] M. Shahbaz and R. Groz. Inferring Mealy machines. In *FM'09*, pages 207–222, 2009.
- [10] M. Tappler, B. K. Aichernig, G. Bacci, M. Eichlseder, and K. G. Larsen. L\*-based learning of Markov decision processes. In *FM*'19, pages 651–669, 2019.
- [11] M. Tappler, B. K. Aichernig, K. G. Larsen, and F. Lorber. Time to learn - learning timed automata from tests. In FORMATS'19, pages 216–235, 2019.
- [12] S. Verwer, M. de Weerdt, and C. Witteveen. The efficiency of identifying timed automata and the power of clocks. Inf. Comput., 209(3) :606–625, 2011.

# Thanks.