Active learning of One-Clock Timed Automata

Jie An

Joint work with Mingshuai Chen, Runqing Xu, Bohua Zhan, et al.

CyPhAI Workshop @ NII, Tokyo, Nov. 15, 2022

Outline

1. **Model learning and L^* algorithm**
2. **Active Learning of DOTAs [TACAS20]**
   - Learning from a smart teacher
   - Learning from a normal teacher
3. **Active learning of DOTAs using Constraint solving [ATVA22]**
   - Learning DOTAs using constraint solving
4. **Conclusion and future work**

Outline

Model/Automata learning

- Machine learning

a sample set

$M = \{(x, y) | x \in X, y \in Y\}$

$\xrightarrow{\text{learn}}$

Model

$f : X \to Y$

$f(x) = y, \forall x \in X$

predict or identify $f(x)$

for all $x \in X$

Model/Automata learning

- Machine learning

a sample set

$M = \{(x, y) | x \in X, y \in Y\}$

$\xrightarrow{\text{learn}}$

Model

$f : X \to Y$

$f(x) = y, \forall x \in X$

predict or identify $f(x)$ for all $x \in X$

- Model/Automata learning

Σ is an alphabet

$X = \Sigma^*$ set of words

$Y = \{+, -\}$ or other set of labels

$\xrightarrow{\text{learn}}$

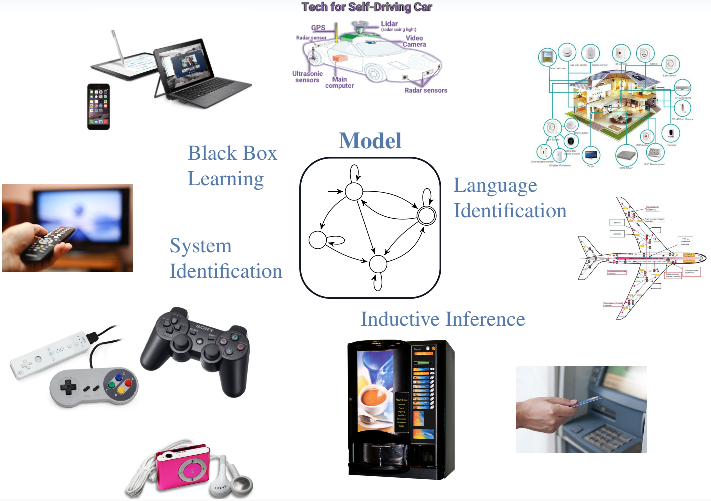
Model

f is a language

$L \subseteq \Sigma^*$

The model is a kind of Automaton

Model/Automata learning



© The figure comes from Irini-Eleftheria Mens.

A prehistory and history

1956 Edward Moore. **Gedanken-experiments on sequential machines.**

- Defines the problem as a *black box* model inference.

1972 E. Mark Gold. **System identification via state characterization.**

- Learning finite automata is possible in finite time. First use the basic idea on table-based methods.

1987 Dana Angluin. **Learning regular sets from queries and counter-examples.**

- The L^* active learning algorithm with *membership and equivalence queries*. Polynomial in the automaton size.

1993 Ronald Rivest and Robert Schapire. **Inference of finite automata using homing sequences.**

- An improved version of L^* by using the *breakpoint method* to treat counterexamples.

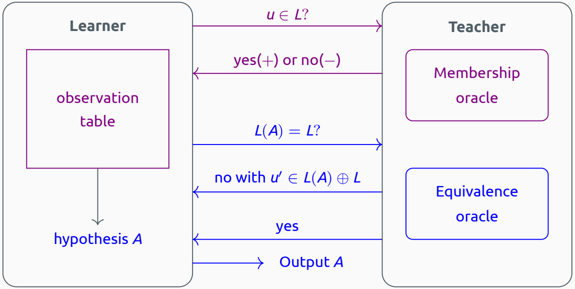
2014 Malte Isberner, Falk Howar, and Bernhard Steffen. **The TTT algorithm : a redundancy-free approach to active automata learning**.

- A *redundancy-free* organization of observations based on Discrimination Trees.

......

Minimally adequate teacher (MAT)

- Dana Angluin proposed an online, active, and exact learning framework L^* for Deterministic Finite Automata (DFA) in 1987.
- Two kinds of queries : membership query and equivalence query.
- Table conditions : **closed** and **consistent**.

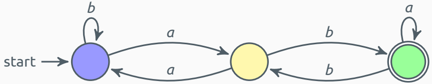


Myhill-Nerode theorem

- The Myhill-Nerode theorem : a language L is **regular** iff $\sim_L$ has a **finite** number of equivalence classes, and moreover, that this number is equal to the number of states in the minimal DFA.
  - Right congruence relation $\sim_L$: For $u, v \in \Sigma^*$, $u \sim_L v$ iff $\forall w \in \Sigma^*$, $uw \in L \iff vw \in L$.

Myhill-Nerode theorem

- The Myhill-Nerode theorem : a language L is **regular** iff $\sim_L$ has a **finite** number of equivalence classes, and moreover, that this number is equal to the number of states in the minimal DFA.
  - Right congruence relation $\sim_L$: For $u, v \in \Sigma^*$, $u \sim_L v$ iff $\forall w \in \Sigma^*$, $uw \in L \iff vw \in L$.
- Intuitively, L^* algorithm aims at finding the suffixes(cols) w to **distinguish** the prefixes(rows) u, v. Each prefix can reach a state in the underlying minimal DFA.



	ϵ	b
ϵ	$-$	$-$
ab	$+$	$-$
a	$-$	$+$
b	$-$	$-$
aba	$+$	$-$
abb	$-$	$+$
aa	$-$	$-$

- $\epsilon \sim_L b \sim_L aa$,
  $ab \sim_L aba$,
  $a \sim_L abb$

- Closed : For every row from $U\Sigma$, there is a equal row in U. If not, move the prefix to U.
- Consistent : For every two rows u, v from U, if their rows are equal, then the rows of $u\sigma$ and $v\sigma$ are equal. If not, extend V.
- Counterexample process : add all prefixes of a counterexample to U.

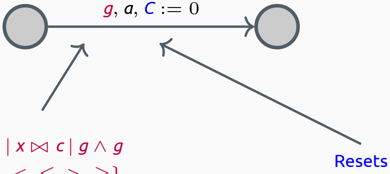
Motivation

- More recent work extends L^* to different models
  - ☺ e.g., Mealy machines [9], I/O automata [1], register automata [5], NFA [2], Büchi automata [6], symbolic automata [7, 3] 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 [4].
  - Passive identification of deterministic one-clock 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].

Outline

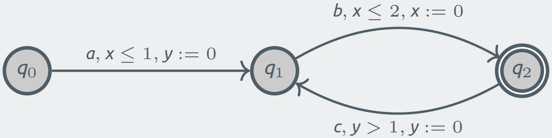
Timed automata

- As opposed to finite automata, timed automata are equipped with clocks and add guards and can reset clocks on transitions



Guards $g := \top \mid x \bowtie c \mid g \wedge g$
where $\bowtie \in \{=, <, \leq, >, \geq\}$

Example : Timed automata



- Clocks : x, y, Alphabet $\Sigma := \{a, b, c\}$
- (Delay) Timed words $(\Sigma \times \mathbb{R}_{\geq 0})^*$: eg. $(a, 1)(b, 0.5)$

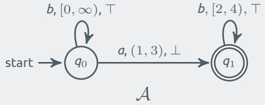
Challenges on active learning of timed automata

- Fundamental obstacle : Do not have a Myhill-Nerode-like theorem for (regular) timed languages...
- Tech. challenges (continuous-time, black-box)
  - Determining how many clocks.
  - Infinite states (pair of a location and a clock valuation).
  - Determining timing constraints on transitions.
  - Determining clock resets on transitions.
  - (Related to the previews points) Mapping observable delay timed behaviours from outside to internal logical clock valuations.
- Active learning of deterministic timed automata with a single clock (DOTAs).
- Solution of learning DOTAs [TACAS 20]
  - A connection between learning from delay timed words (outside) and learning from logical timed words (inside).
  - Utilize a partition function to map *logical-timed* valuations to finite intervals.
  - First consider a smart teacher who can tell the learner reset information. Then drop the assumption (i.e. reduction to a normal teacher) by guessing the resets.

Deterministic One-clock timed automata

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

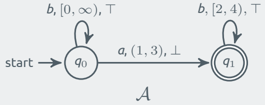
An example of DOTA



Deterministic One-clock timed automata

- The DOTA $\mathcal{A}$ recognizes the target language $\mathcal{L}$.
- $\Sigma = \{a, b\}$; $\mathcal{B} = \{\top, \bot\}$ where $\top$ is for reset, $\bot$ otherwise.
- **Timed words** $(\Sigma \times \mathbb{R}_{\geq 0})^*$: outside observations;
  e.g. $\omega = (b, 1)(a, 1.1)(b, 1)$ is an accepting timed words.
- **Reset-logical timed words** $(\Sigma \times \mathbb{R}_{\geq 0} \times \mathcal{B})^*$: inside logical actions;
  e.g. $\gamma_r = (b, 1, \top)(a, 1.1, \bot)(b, 2.1, \top)$ is the reset-logical counterpart of ω.
  Logical counterpart $\gamma = (b, 1)(a, 1.1)(b, 2.1)$.

An example of DOTA



Learning from a smart teacher

- Given a DOTA $\mathcal{A}$, $L_r(\mathcal{A})$ represents the recognized reset-logical timed language of $\mathcal{A}$; $\mathcal{L}(\mathcal{A})$ represents the (delay) timed language.

Theorem

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

Learning from a smart teacher

- Given a DOTA $\mathcal{A}$, $L_r(\mathcal{A})$ represents the recognized reset-logical timed language of $\mathcal{A}$; $\mathcal{L}(\mathcal{A})$ represents the (delay) timed language.
- Guiding principle : learning the timed language of a DOTA $\mathcal{A}$ can be reduced to learning the reset-logical timed language of $\mathcal{A}$
- Smart teacher setting
  - Membership queries are logical timed words, teacher responds with reset information.
  - For equivalence queries, instead of checking directly whether $L_r(\mathcal{H}) = L_r(\mathcal{A})$, the contraposition of the theorem guarantees to perform equivalence queries over their timed counterparts (checking $\mathcal{L}(\mathcal{H}) = \mathcal{L}(\mathcal{A})$).

Theorem

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

Learning from a smart teacher

$\mathcal{T}$	ϵ	$(a, 1.1)$
ϵ	$-$	$+$
$(a, 1.1, \bot)$	$+$	$-$
$(a, 0, \top)$	$-$	$-$
$(b, 0, \top)$	$-$	$+$
$(a, 1.1, \bot)(a, 0, \top)$	$-$	$-$
$(a, 1.1, \bot)(b, 0, \top)$	$-$	$-$
$(a, 1.1, \bot)(b, 2, \top)$	$+$	$-$
$(a, 3, \top)$	$-$	$-$
$(a, 0, \top)(a, 1.1, \top)$	$-$	$-$

Learning from a smart teacher

	$\mathcal{T}$	ϵ	$(a, 1.1)$
S	ϵ	$-$	$+$
	$(a, 1.1, \bot)$	$+$	$-$
R	$(a, 0, \top)$	$-$	$-$
	$(b, 0, \top)$	$-$	$+$
	$(a, 1.1, \bot)(a, 0, \top)$	$-$	$-$
	$(a, 1.1, \bot)(b, 0, \top)$	$-$	$-$
	$(a, 1.1, \bot)(b, 2, \top)$	$+$	$-$
	$(a, 3, \top)$	$-$	$-$
	$(a, 0, \top)(a, 1.1, \top)$	$-$	$-$

- Observation table $\mathcal{T} = (\Sigma, S, R, E, f)$
  - The prefixes set S indicates the different locations
  - The extended prefixes R indicates the transitions
  - The suffixes set E distinguishes the locations.
  - The mapping $f(\omega \cdot e) = +$ iff $MQ(\omega \cdot e) = +$ for $\forall \omega \in S \cup R, e \in E$
- Function $val : (S \cup R) \to (E \to \{+, -\})$ helps to label the locations, e.g., q_{-+}, q_{+-}, q_{--}.

Learning from a smart teacher

	$\mathcal{T}$	ϵ	$(a, 1.1)$
S	ϵ	$-$	$+$
	$(a, 1.1, \bot)$	$+$	$-$
R	$(a, 0, \top)$	$-$	$-$
	$(b, 0, \top)$	$-$	$+$
	$(a, 1.1, \bot)(a, 0, \top)$	$-$	$-$
	$(a, 1.1, \bot)(b, 0, \top)$	$-$	$-$
	$(a, 1.1, \bot)(b, 2, \top)$	$+$	$-$
	$(a, 3, \top)$	$-$	$-$
	$(a, 0, \top)(a, 1.1, \top)$	$-$	$-$

- Observation table $\mathcal{T} = (\Sigma, S, R, E, f)$
  - The prefixes set S indicates the different locations
  - The extended prefixes R indicates the transitions
  - The suffixes set E distinguishes the locations.
  - The mapping $f(\omega \cdot e) = +$ iff $MQ(\omega \cdot e) = +$ for $\forall \omega \in S \cup R, e \in E$
- Function $val : (S \cup R) \to (E \to \{+, -\})$ helps to label the locations, e.g., q_{-+}, q_{+-}, q_{--}.

- Table conditions :
  - Reduced : $\forall s, s' \in S : s \neq s'$ implies $val(s) \neq val(s')$;
  - closed : $\forall r \in R, \exists s \in S : val(s) = val(r)$;
  - Consistent :
    $\forall \gamma_r, \gamma_r' \in S \cup R, val(\gamma_r) = val(\gamma_r')$ implies $val(\gamma_r \cdot \sigma_r) = val(\gamma_r' \cdot \sigma_r')$, for all $\sigma_r, \sigma_r' \in \Sigma_r$ satisfying $\gamma_r \cdot \sigma_r, \gamma_r' \cdot \sigma_r' \in S \cup R$ and $\Pi_{\{1,2\}}\sigma_r = \Pi_{\{1,2\}}\sigma_r'$

$\mathcal{T}$	ϵ
ϵ	$-$
$(a, 1.1, \bot)$	$+$
$(a, 0, \top)$	$-$
$(b, 0, \top)$	$-$
$(a, 1.1, \bot)(a, 0, \top)$	$-$
$(a, 1.1, \bot)(b, 0, \top)$	$-$
$(a, 1.1, \bot)(b, 2, \top)$	$+$
$(a, 3, \top)$	$-$

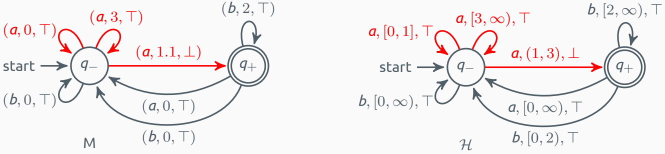


Figure 1 – The prepared timed observation table $\mathcal{T}$, the corresponding DFA M and hypothesis $\mathcal{H}$.

- Partition function maps a list of clock valuations $\ell = \tau_0, \tau_1, \cdots, \tau_n$ with $\lfloor \tau_i \rfloor \neq \lfloor \tau_j \rfloor$ to $\{I_0, I_1, \ldots, I_n\}$ with $\bigcup I_i = \mathbb{R}_{\geq 0}$,

$$I_i = \begin{cases} [\tau_i, \tau_{i+1}), & \text{if } \tau_i \in \mathbb{N} \wedge \tau_{i+1} \in \mathbb{N}; \\ (\lfloor \tau_i \rfloor, \tau_{i+1}), & \text{if } \tau_i \in \mathbb{R}_{\geq 0} \setminus \mathbb{N} \wedge \tau_{i+1} \in \mathbb{N}; \\ [\tau_i, \lfloor \tau_{i+1} \rfloor], & \text{if } \tau_i \in \mathbb{N} \wedge \tau_{i+1} \in \mathbb{R}_{\geq 0} \setminus \mathbb{N}; \\ (\lfloor \tau_i \rfloor, \lfloor \tau_{i+1} \rfloor], & \text{if } \tau_i \in \mathbb{R}_{\geq 0} \setminus \mathbb{N} \wedge \tau_{i+1} \in \mathbb{R}_{\geq 0} \setminus \mathbb{N}. \end{cases}$$

- e.g., $\ell_{q_-, a} = \{0, 1.1, 3\}$ and then get the intervals $[0, 1]$,$(1, 3)$ and $[3, \infty)$.

Learning from a smart teacher

- Given a target timed language $\mathcal{L}$ which is recognized by a DOTA $\mathcal{A}$, let $n = |Q|$ be the number of locations of $\mathcal{A}$, $m = |\Sigma|$ the size of the alphabet, and κ the maximal constant appearing in the clock constraints of $\mathcal{A}$, and h be the length of the longest counterexample returned by the teacher.

Theorem

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

Theorem

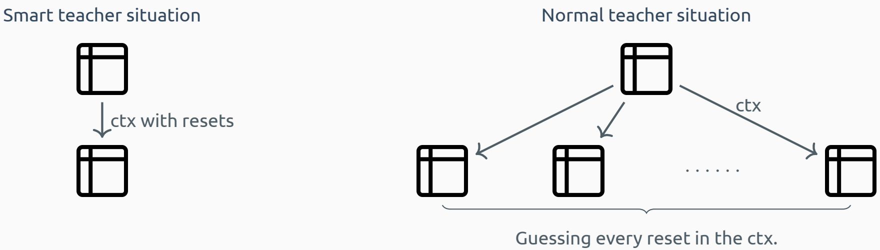
The complexity is $\mathcal{O}(hmn^2\kappa^3)$.

Learning from a normal teacher

- 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.

Learning from a normal teacher

- 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.



Learning from a normal 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 information.
  - If $\mathbf{T} = (\Sigma, S, R, E, f)$ is the final observation table for the correct candidate in the situation with a smart teacher, the learner can find it after checking $\mathcal{O}(2^{(|S|+|R|)\times(1+\sum_{e_i \in E \setminus \{\epsilon\}}(|e_i|-1))})$ 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 $\mathcal{L}$.

Experiment 1

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

Case ID	$\lvert\Delta\rvert_{\mathrm{mean}}$	#Membership			#Equivalence			n_{mean}	t_{mean}
		N_{min}	N_{mean}	N_{max}	N_{min}	N_{mean}	N_{max}		
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	521.8
14_4_20	58.4	1390	2238.6	4430	79	107.7	135	16.0	515.5

Case ID : n_m_κ, 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 $\mathcal{A}$'s.

$\lvert\Delta\rvert_{\mathrm{mean}}$: 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_{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.

Experiment 2

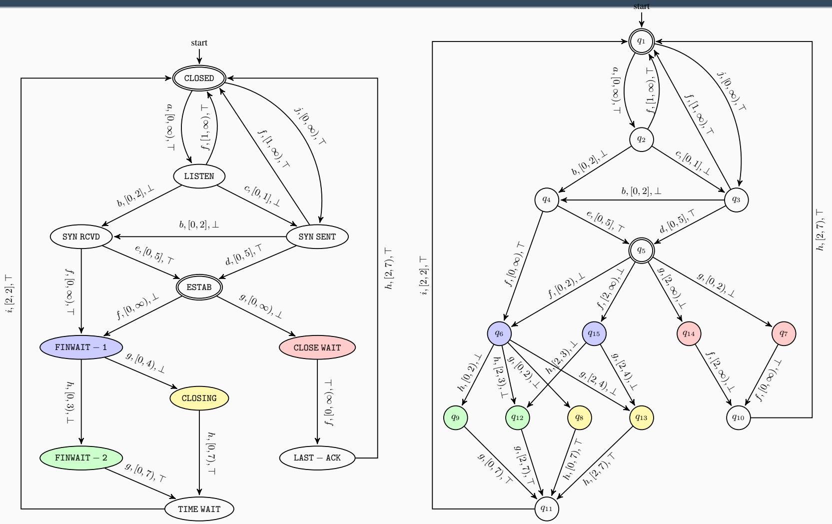


Figure 2 – Left : The functional specification of the TCP protocol with more complex timing constraints. Right : The learned functional specification of the TCP protocol. Colors indicate the splitting of locations.

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

Case ID	$\lvert\Delta\rvert_{\text{mean}}$	#Membership			#Equivalence			n_{mean}	t_{mean}	$\#\mathrm{T}_{explored}$	#Learnt
		N_{min}	N_{mean}	N_{max}	N_{min}	N_{mean}	N_{max}				
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_{min} : the minimal, N_{mean} : the mean, N_{max} : the maximum.

$\#\mathrm{T}_{explored}$: the average number of the explored table instances.

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

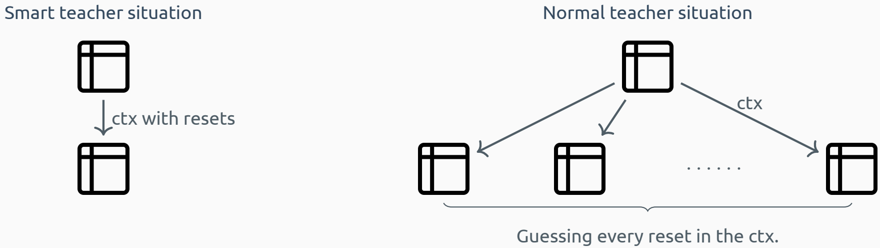
Short conclusion

- 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)
- DOTAs can be actively learned.

Outline

Learning DOTAs using constraint solving

- Brute-force guessing leads to exponential number of table instances, which limits the scalability of the algorithm in practical applications.



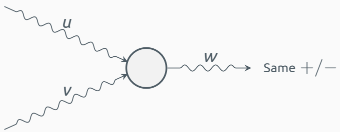
- Basic ideas : maintain a single observation table that collects all results from previous membership queries, rather than one observation table for each possible choice of resets.
  1. Associate each row with a boolean variable representing reset information after running the timed word.
  2. Encode the table conditions into the SMT formulas with the variables.
  3. Utilize the SMT solver to obtain a feasible choice of resets that make the table prepared.

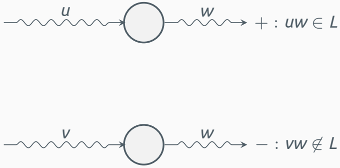
Go back to Nerode's right congruence

- Right congruence relation $\sim_L$: For $u, v \in \Sigma^*$, $u \sim_L v$ iff $\forall w \in \Sigma^*$, $uw \in L \iff vw \in L$.
- One key step of L^*-style framework : determine if two words ω_1 and ω_2 end in different locations using some suffixes.

Go back to Nerode's right congruence

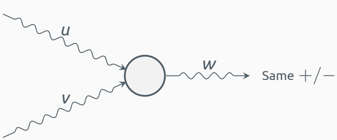
- Right congruence relation $\sim_L$: For $u, v \in \Sigma^*$, $u \sim_L v$ iff $\forall w \in \Sigma^*$, $uw \in L \iff vw \in L$.
- One key step of L^*-style framework : determine if two words ω_1 and ω_2 end in different locations using some suffixes.
  - ☺ If u and v reach the same location, then uw and vw should reach some same location.

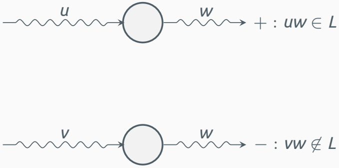




Go back to Nerode's right congruence

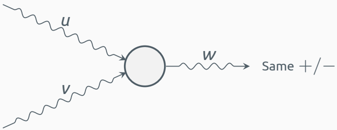
- Right congruence relation $\sim_L$: For $u, v \in \Sigma^*$, $u \sim_L v$ iff $\forall w \in \Sigma^*$, $uw \in L \iff vw \in L$.
- One **key step** of L^*-style framework : determine if two words ω_1 and ω_2 end in different locations using some suffixes.
  - ☺ If u and v reach the same location, then uw and vw should reach some same location.





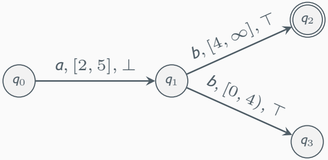
☕ How to compare two timed words ω_1 and ω_2 using some suffixes e?

- Right congruence relation $\sim_L$: For $u, v \in \Sigma^*$, $u \sim_L v$ iff $\forall w \in \Sigma^*, uw \in L \iff vw \in L$.
- One **key step** of L^*-style framework : determine if two words ω_1 and ω_2 end in different locations using some suffixes.
  - ☺ If u and v reach the same location, then uw and vw should reach some same location.



☕ How to compare two timed words ω_1 and ω_2 using some suffixes e?

- Even if ω_1 and ω_2 reach the same location, $\omega_1 e$ and $\omega_2 e$ may reach two different locations.



- For example, $\omega_1 = (a, 2)$ and $\omega_2 = (a, 3)$, $e = (b, 1)$

Learning DOTAs using constraint solving

- Two timed words reach the same location, however may reach different locations after appending the same suffix. (Belong to different regions)
  - Alignment and comparison : a method to determine whether two timed words are distinguished using membership queries with unknown reset information.
  - A method to encode table conditions into the SMT formulas using the variables.

- Basic ideas : maintain a single observation table that collects all results from previous membership queries, rather than one observation table for each possible choice of resets.
  1. Associate each row with a boolean variable representing reset information after running the timed word.
  2. Encode the table conditions into the SMT formulas with the variables.
  3. Utilize the SMT solver to obtain a feasible choice of resets that make the table prepared.

$\mathcal{O}$			$S \cup S_+ \cup R$						E
			ϵ	$(a,0)$	$(a,4)\,(a,5.5)$	$(a,0)\,(a,0)$	$(a,4)$	$(a,9.5)$	
S	ϵ	$\mathbb{b}_0$	$\top$	$\bot$	$\neg\mathbb{b}_5$	$\bot$	$\mathbb{b}_3$	$\bot$	ϵ
	$(a,0)$	$\mathbb{b}_1$	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\top$	$(a,5.5)$
S_+	$(a,4)\,(a,5.5)$	$\mathbb{b}_5$	$\neg\mathbb{b}_5$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	
	$(a,0)\,(a,0)$	$\mathbb{b}_2$	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\top$	
R	$(a,4)$	$\mathbb{b}_3$	$\mathbb{b}_3$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	
	$(a,9.5)$	$\mathbb{b}_4$	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\top$	

☺ $\mathcal{O} = \{\Sigma, S, S_+, R, E, f, N\}$

- S contains timed words that are certainly *distinct* from each other;
- S_+ : additional rows in the observation table that are distinct from rows in S under some choices of resets.
- R collects all current membership queries under all different choice of resets.
- f summarizes when two corresponding timed words are distinguished, using formulas in terms of ending reset variables $\mathbb{b}$.
- N is the current limit on the number of locations in the candidate automaton.

- Reset variables $\mathbb{b}_i$ denotes whether clock resets after running ω_i.
- (**Innovation**) Cells record all membership queries by comparing each pair of timed words in $S \cup S_+ \cup R$ under all valid combinations of last resets.
  Example : Given suffix $e = (a, 5.5)$, we have

$$f((a,4)\,,\epsilon,0,0) = \bot, f((a,4)\,,\epsilon,1,0) = \top$$

this can be summarized as $\mathbb{b}_3$.

Experiment 4

Table 3 – Experimental results on learning DOTAs using constraint solving.

Group	$\|\Delta\|$	Method	#Membership			#Equivalence			$\|Q_{\mathcal{H}}\|$	#Learnt	$t(s)$
			N_{min}	N_{mean}	N_{max}	N_{min}	N_{mean}	N_{max}			
6_2_10	11.9	DOTAL	73	348.3	708	10	16.7	30	5.6	7/10	39.88
		SL	104	1894.8	3929	11	20.8	35	5.6	10/10	0.78
4_4_20	16.3	DOTAL	231	317.0	564	27	30.8	40	4.0	6/10	100.223
		SL	1740	3497.7	5329	24	32.8	42	4.0	10/10	1.42
7_4_20	26.0	DOTAL				—				0/10	TO
		SL	6092	9393.3	15216	44	51.5	69	7.0	10/10	2.90
10_4_20	39.1	DOTAL				—				0/10	TO
		SL	8579	16322.3	23726	59	76.5	93	10.0	10/10	5.89
12_4_20	47.6	DOTAL				—				0/10	TO
		SL	13780	20345.5	29011	70	88.0	102	12.0	10/10	10.052
14_4_20	58.4	DOTAL				—				0/10	TO
		SL	18915	28569.0	40693	92	110.6	126	14.0	10/10	14.692
AKM (17_12_5)	40	DOTAL				—				0/1	TO
		SL	3453	3453.0	3453	49	49.0	49	12	1/1	7.19
TCP (22_13_2)	22	DOTAL				—				0/1	TO
		SL	4713	4713.0	4713	32	32.0	32	20	1/1	19.04
CAS (14_10_27)	23	DOTAL				—				0/1	TO
		SL	4769	4769.0	4769	18	18.0	18	14	1/1	126.30
PC (26_17_10)	42	DOTAL				—				0/1	TO
		SL	10854	10854.0	10854	28	28.0	28	25	1/1	109.01

Outline

Conclusion

- Current results
  - Learning DOTAs from a smart teacher (gray-box or white-box, efficient) and from a normal teacher [1] (black-box, inefficient);
  - Learning DOTAs using constraint solving [2] (black-box, scalable);
  - Extending in the PAC learning scheme when the exact equivalence oracle is not available [3];
  - Adapting to learning real-time automata [4].

- Ongoing work
  - Active learning of multi-clocks timed automata avoiding just mimicking region graphs.
    - Which kind of Myhill-Nerode Theorem for deterministic timed automata we can have. [8]
    - How compact the congruence relation can be.
  - Passive learning from observations.
    - Robust learning.
    - Multi-objects learning from demonstrations. (Involving heuristic methods)

1. Jie An, Mingshuai Chen, Bohua Zhan, et al.. Learning One-Clock Timed Automata. TACAS'20.
2. Runqing Xu, Jie An*, Bohua Zhan. Active Learning of One-Clock Timed Automata using Constraint Solving. ATVA'22.
3. Wei Shen, Jie An*, et al. PAC Learning of Deterministic One-Clock Timed Automata. ICFEM'20
4. Jie An, Bohua Zhan, et al.. Learning nondeterministic real-time automata. IEEE-TECS (EMSOFT'21).

Reference I

[1] F. Aarts and F. W. Vaandrager.
Learning I/O automata.
In *CONCUR'10*, pages 71–85, 2010.

[2] B. Bollig, P. Habermehl, C. Kern, and M. Leucker.
Angluin-style learning of NFA.
In *IJCAI'09*, pages 1004–1009, 2009.

[3] S. Drews and L. D'Antoni.
Learning symbolic automata.
In *TACAS'17*, pages 173–189, 2017.

[4] O. Grinchtein, B. Jonsson, and M. Leucker.
Learning of event-recording automata.
Theor. Comput. Sci., 411(47) :4029–4054, 2010.

[5] F. Howar, B. Steffen, B. Jonsson, and S. Cassel.
Inferring canonical register automata.
In *VMCAI'12*, pages 251–266, 2012.

[6] 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.

Reference II

[7] O. Maler and I. Mens.
Learning regular languages over large alphabets.
In *TACAS'14*, pages 485–499, 2014.

[8] O. Maler and A. Pnueli.
On recognizable timed languages.
In *FOSSACS 2004*, pages 348–362, 2004.

[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.