Active Learning of One-Clock Timed Automata using Constraint Solving

Runqing Xu[1,2], Jie An[3] and Bohua Zhan[1,2]

ATVA · October 27, 2022

[1]State Key Lab. of Computer Science, Institute of Software, CAS, Beijing, China

[2]University of Chinese Academy of Sciences, Beijing, China

[3]Max Planck Institute for Software Systems, Kaiserslautern, Germany

Introduction and motivation

Timed automaton

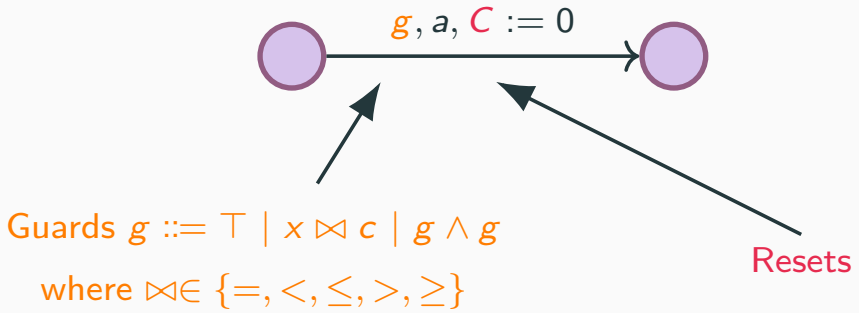
- Time plays a crucial role in real-world systems (e.g. embedded systems, protocols).

Timed automaton

- Time plays a crucial role in real-world systems (e.g. embedded systems, protocols).
- Timed automata (finite automata + clock variables) is a popular model of timed systems.

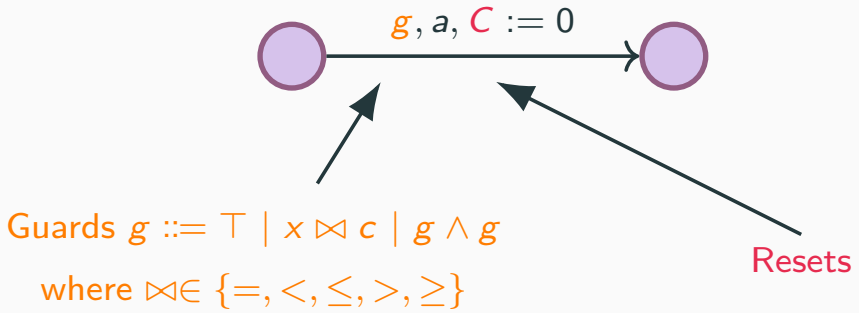
Timed automaton

- Time plays a crucial role in real-world systems (e.g. embedded systems, protocols).
- Timed automata (finite automata + clock variables) is a popular model of timed systems.
- As opposed to finite automata, timed automata adds guards and can reset clocks on transition



Timed automaton

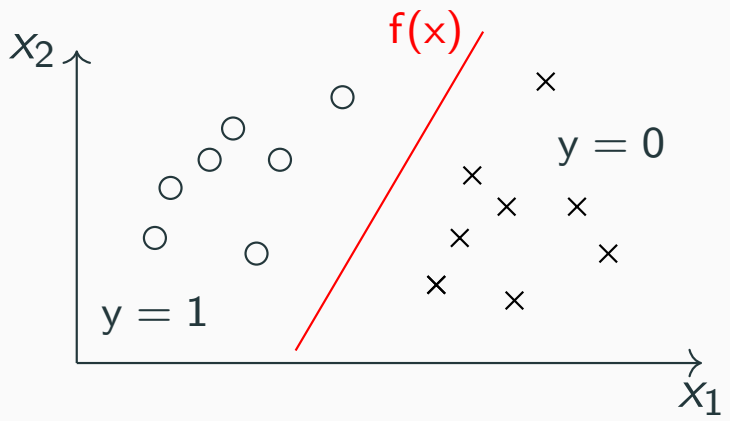
- Time plays a crucial role in real-world systems (e.g. embedded systems, protocols).
- Timed automata (finite automata + clock variables) is a popular model of timed systems.
- As opposed to finite automata, timed automata adds guards and can reset clocks on transition



- Timed automata can be used to model and analyze the timing behavior of computer systems.

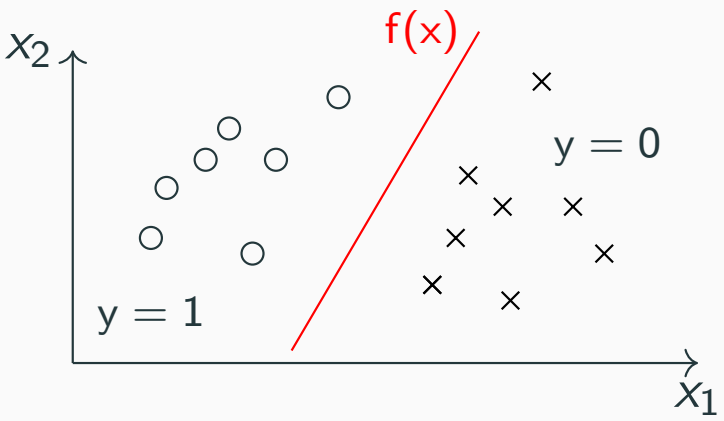
Model/Automata learning

- Machine Learning
  - a sample set $\mathcal{M} = \{(x, y) | x \in X, y \in Y\}$
  - learn an function $f : X \mapsto Y$ s.t. $\forall (x, y) \in \mathcal{M}, f(x) = y$
  - identify $f(x)$ for all $x \in X$

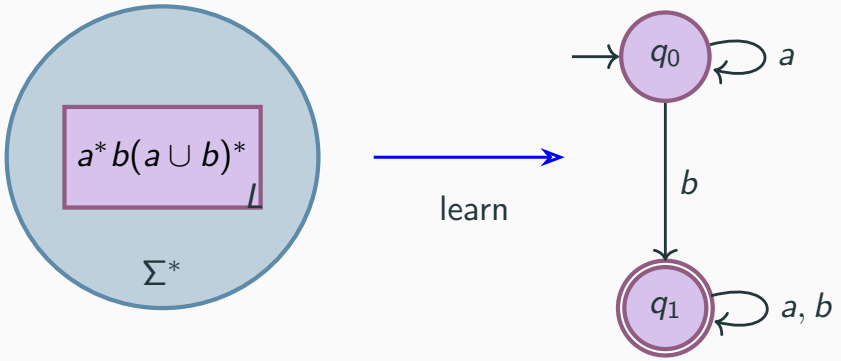


Model/Automata learning

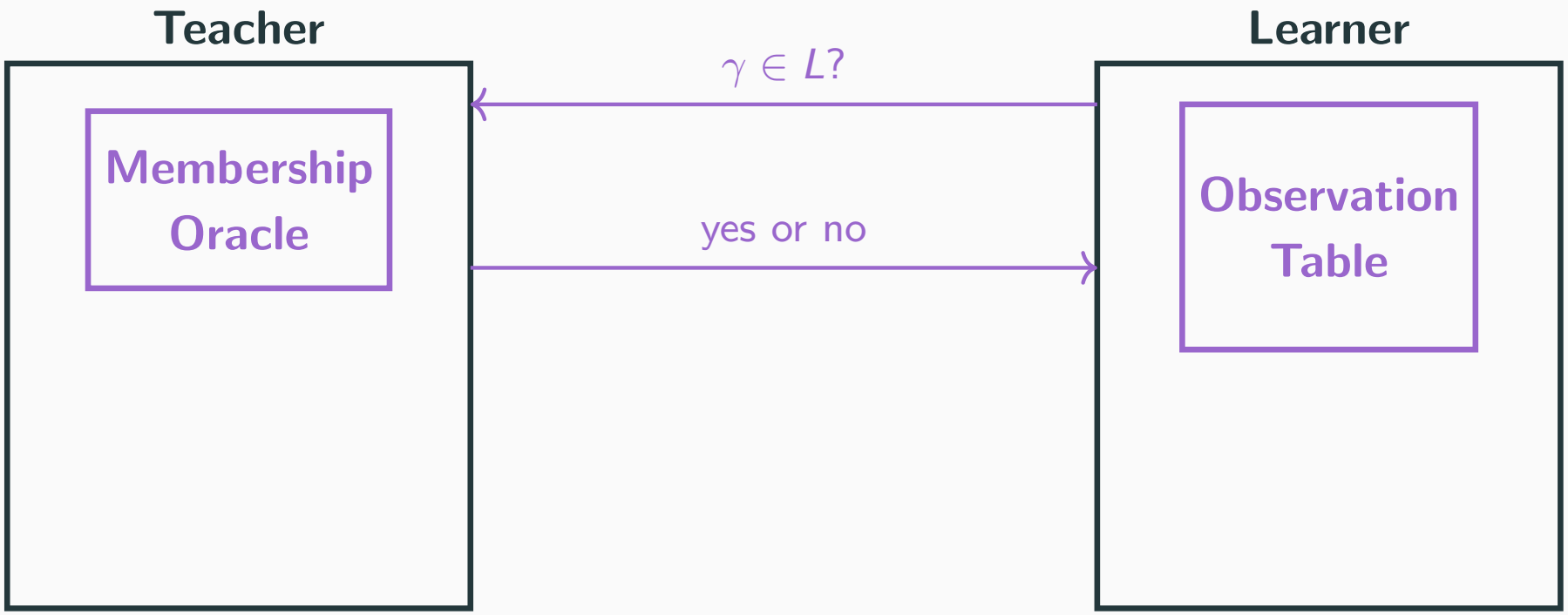
- Machine Learning
  - a sample set $\mathcal{M} = \{(x, y) | x \in X, y \in Y\}$
  - learn an function $f : X \mapsto Y$ s.t. $\forall (x, y) \in \mathcal{M}, f(x) = y$
  - identify $f(x)$ for all $x \in X$



- Model/Automata Learning
  - an alphabet Σ
    an unknown language $L \subseteq \Sigma^*$
  - learn an automata $\mathcal{A}$ which can represent L



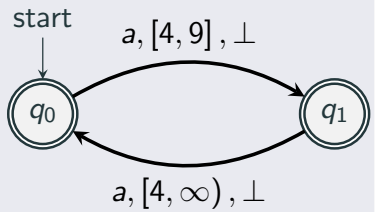
Active learning of deterministic one-clock timed automata [1]



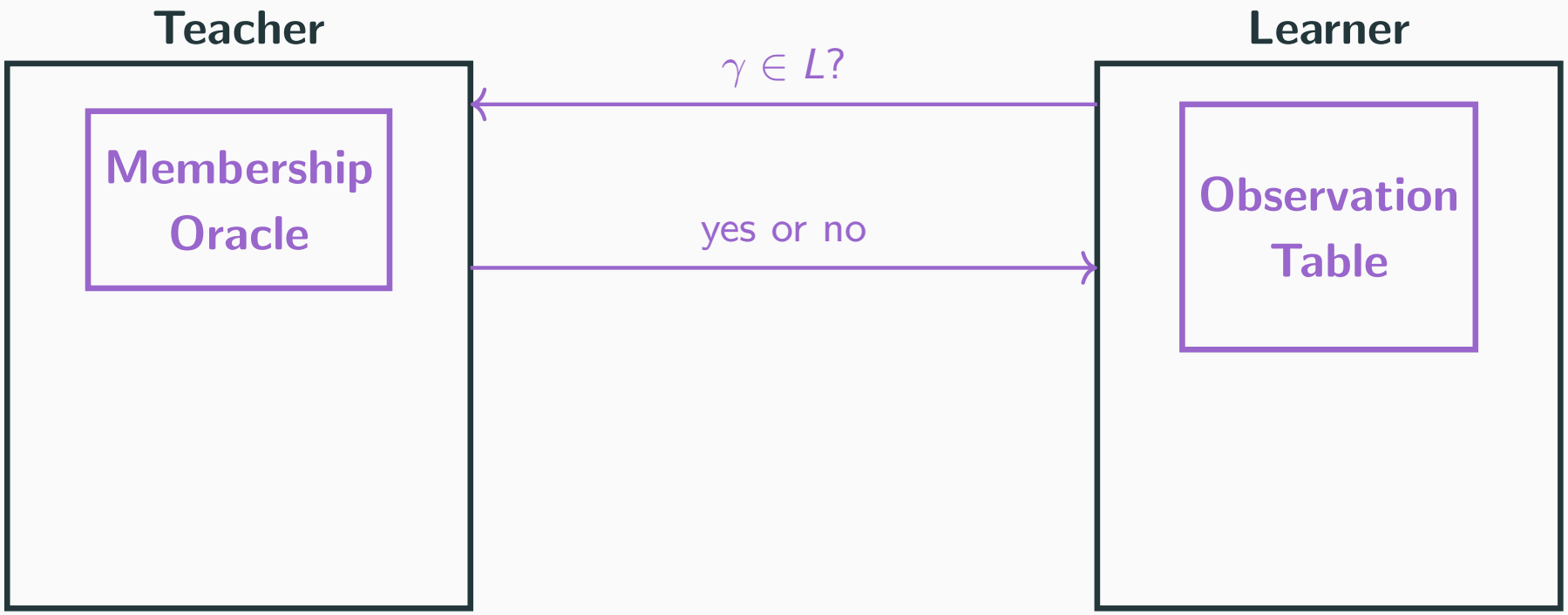
Learning DOTA without reset information

Example: membership query

$$\epsilon \to \text{yes} \quad (a,0) \to \text{no} \quad (a,0)(a,0) \to \text{no}$$



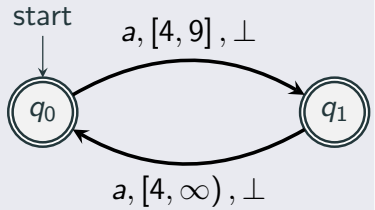
Active learning of deterministic one-clock timed automata [1]



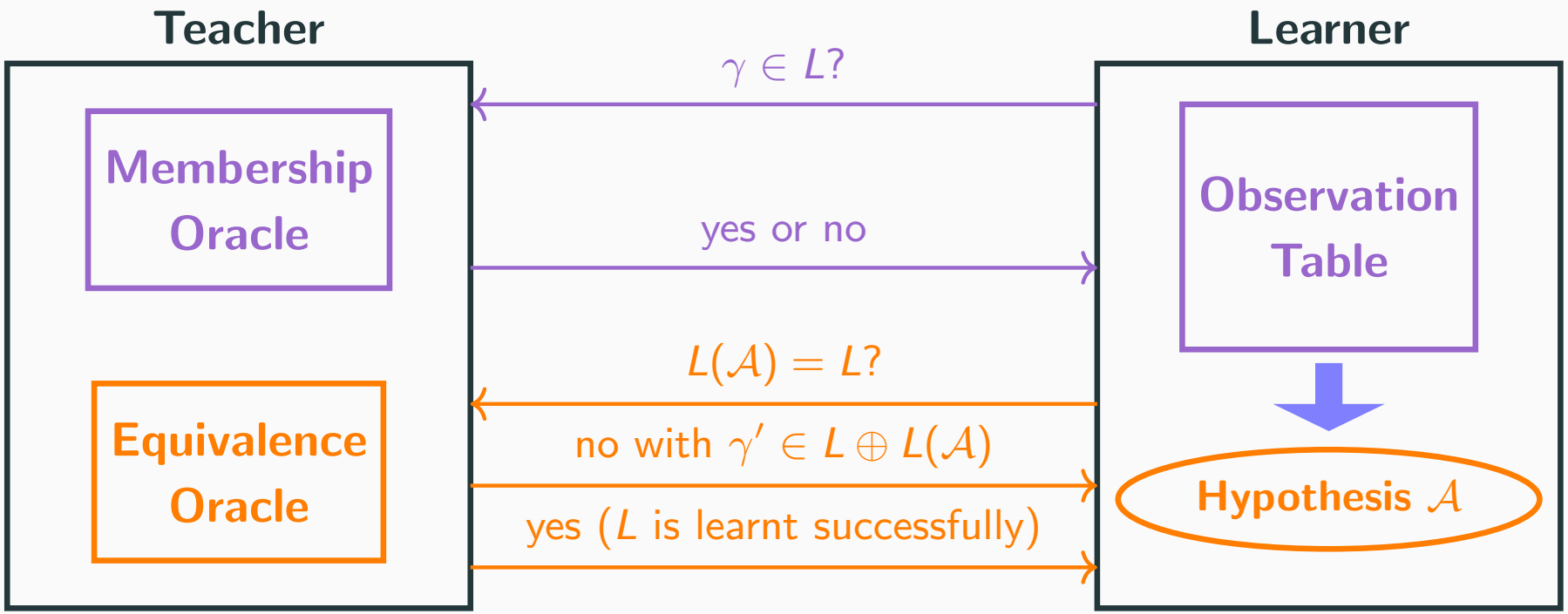
Learning DOTA without reset information

Example: observation table (one reset)

$\mathcal{O}$	ϵ
ϵ	yes
$(a, 0, \bot)$	no
$(a, 0, \bot)(a, 0, \bot)$	no

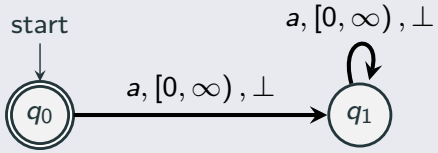


Active learning of deterministic one-clock timed automata [1]

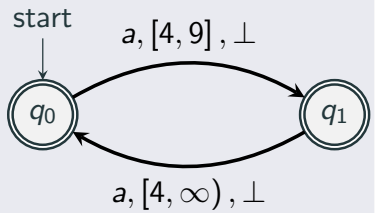


Learning DOTA without reset information

Example: equivalence query with reset assignment $\perp$ for both transitions

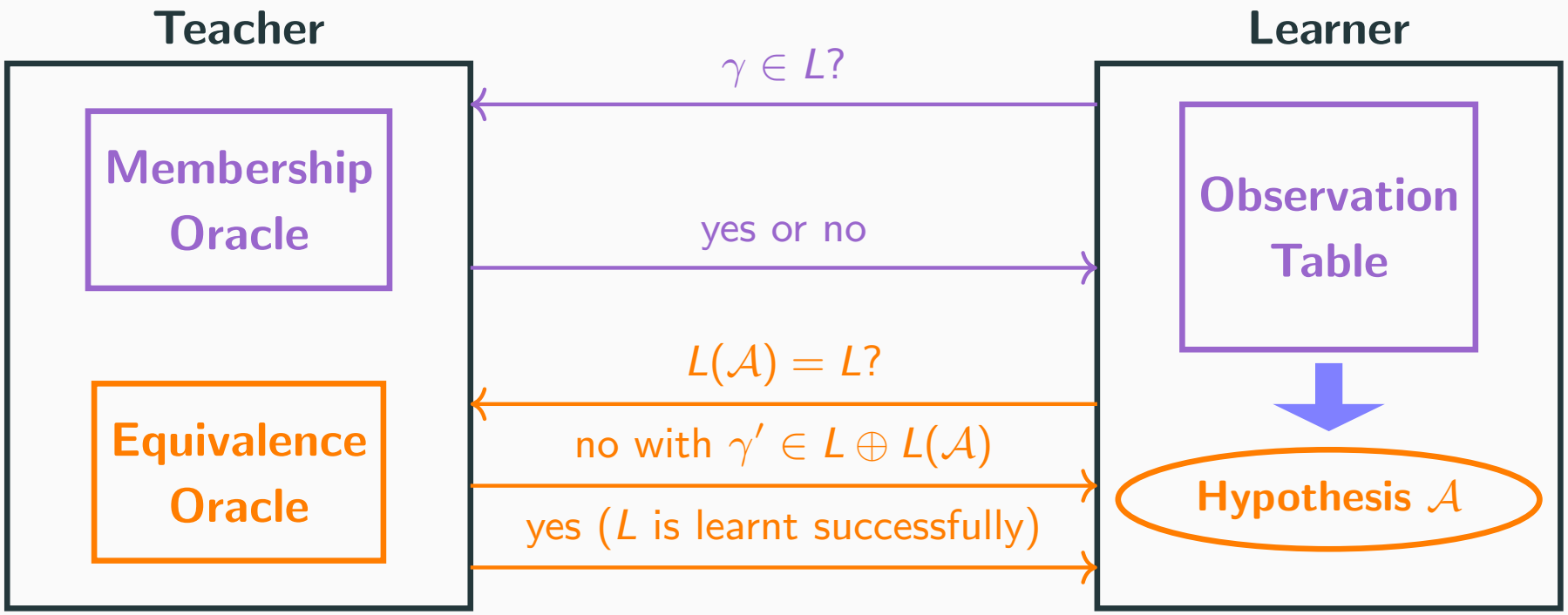
Hypothesis $\mathcal{A}$ 

$\rightarrow$ counterexample $\gamma' = (a, 4.0)$.



There are at least $2^{|S|+|R|-1}$ choices of resets.

Active learning of deterministic one-clock timed automata [1]



Learning DOTA without reset information

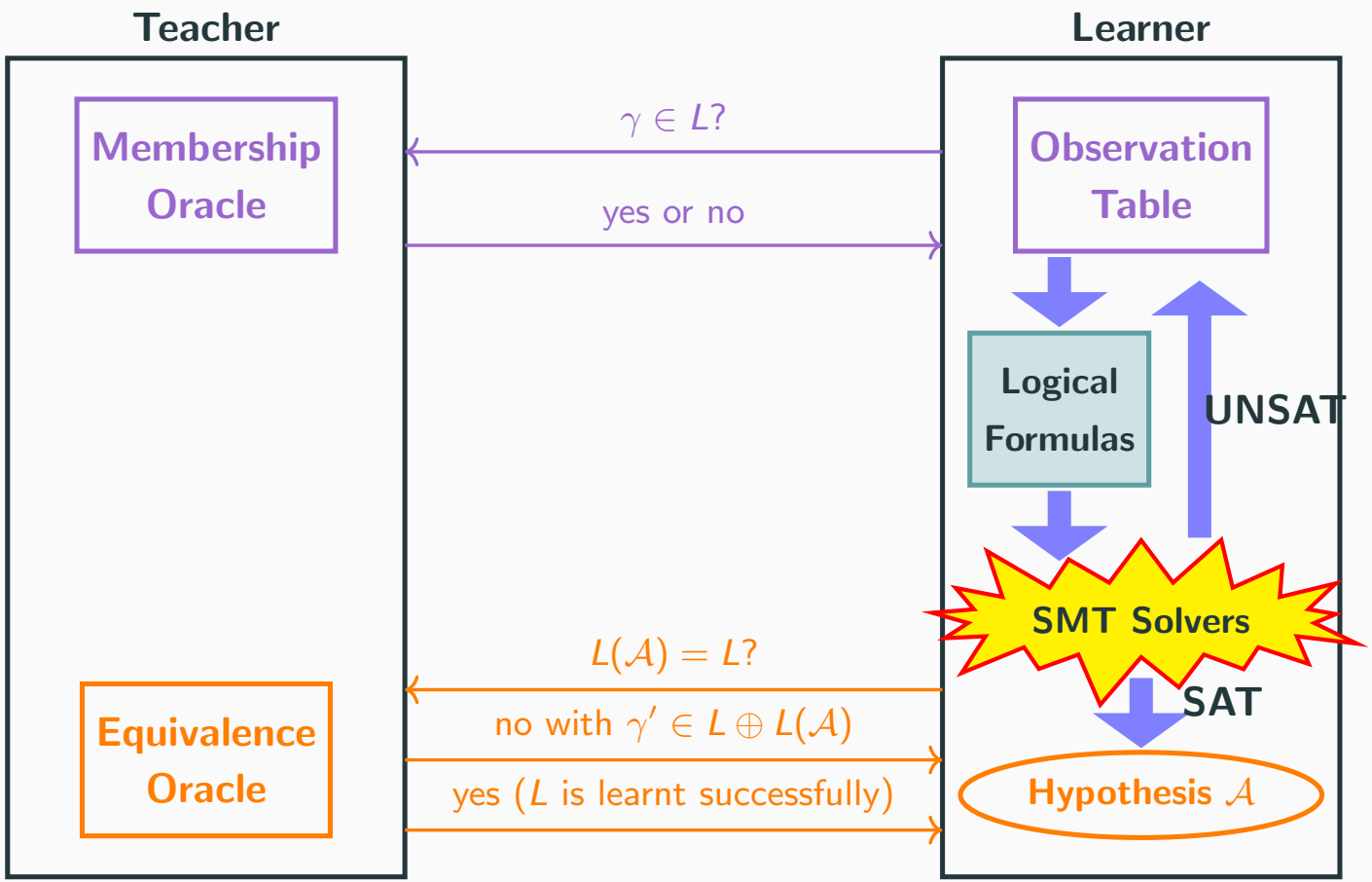
Challenge: need to try each choice of reset, which blow up exponentially w.r.t the number of transitions [1].

Motivation

- Motivation
  - How to learn larger timed automata models efficiently?
- Main idea
  - Rather than trying different choices of resets one-by-one, use constraint solving to determine reset assignments.
- Related work
  - Active learning algorithms for deterministic [2] and non-deterministic [3] real-time automata.
  - Active learning algorithm for a new kind of timed models named Mealy Machine with one timer [7].
  - Passive learning of DFAs, Mealy machines, register automata [5] and timed automata [6] via constraint solving.

Learning OTA via constraint solving

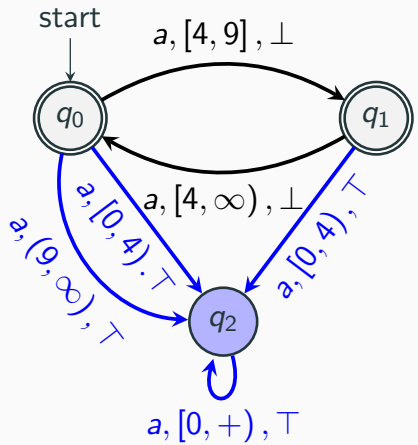
Basic idea



- Maintaining all available observations in a single observation table.
- Encoding readiness condition as logical formulas on reset assignments.
- Utilizing SMT solvers to determine feasible reset assignments.

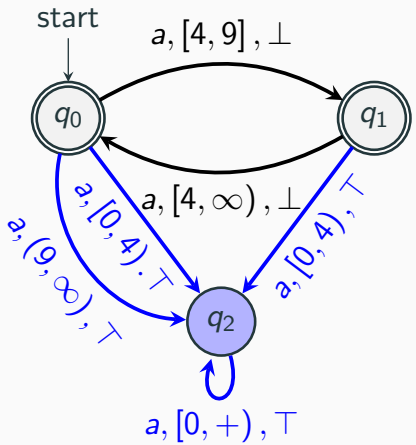
Preliminaries

- A one-clock timed automata is a sextuple $(\Sigma, Q, q_0, F, c, \Delta)$.



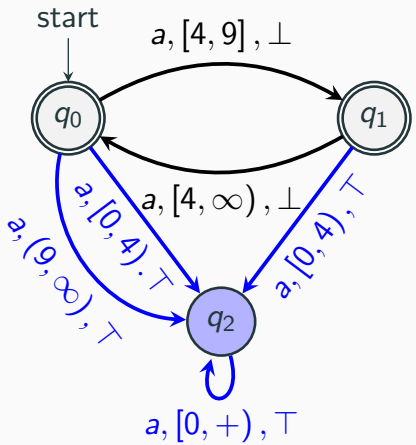
Preliminaries

- A one-clock timed automata is a sextuple $(\Sigma, Q, q_0, F, c, \Delta)$.
- **Timed words** $(\Sigma \times \mathbb{R}_{\geq 0})^*$: outside observations; e.g. $\omega = \epsilon$ and $\omega' = (a, 4.0)(a, 4.0)$ are both accepting timed words.



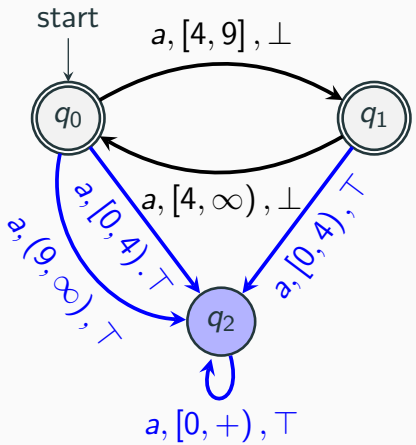
Preliminaries

- A one-clock timed automata is a sextuple $(\Sigma, Q, q_0, F, c, \Delta)$.
- **Timed words** $(\Sigma \times \mathbb{R}_{\geq 0})^*$: outside observations; e.g. $\omega = \epsilon$ and $\omega' = (a, 4.0)(a, 4.0)$ are both accepting timed words.
- **Reset timed words** $(\Sigma \times \mathbb{R}_{\geq 0} \times \mathbb{B})^*$: record reset information for each step.



Preliminaries

- A one-clock timed automata is a sextuple $(\Sigma, Q, q_0, F, c, \Delta)$.
- **Timed words** $(\Sigma \times \mathbb{R}_{\geq 0})^*$: outside observations; e.g. $\omega = \epsilon$ and $\omega' = (a, 4.0)(a, 4.0)$ are both accepting timed words.
- **Reset timed words** $(\Sigma \times \mathbb{R}_{\geq 0} \times \mathbb{B})^*$: record reset information for each step.

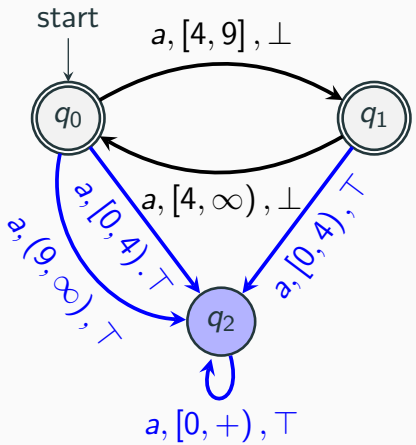


Definition (Last reset of timed word)

Given a timed word $\omega = (\sigma_1, t_1)(\sigma_2, t_2) \cdots (\sigma_n, t_n)$, and DOTA $\mathcal{A}$. Let $\omega_r = (\sigma_1, t_1, b_1)(\sigma_2, t_2, b_2) \cdots (\sigma_n, t_n, b_n)$ be the reset timed word that results from running ω on $\mathcal{A}$. The **last reset** $k_{\mathcal{A}}(\omega)$ is defined to be 0 if $b_i = \bot$ for all $1 \leq i \leq n$, and $k_{\mathcal{A}}(\omega) = i$ if $b_i = \top$ and $b_j = \bot$ for all $j > i$.

Preliminaries

- A one-clock timed automata is a sextuple $(\Sigma, Q, q_0, F, c, \Delta)$.
- **Timed words** $(\Sigma \times \mathbb{R}_{\geq 0})^*$: outside observations; e.g. $\omega = \epsilon$ and $\omega' = (a, 4.0)(a, 4.0)$ are both accepting timed words.
- **Reset timed words** $(\Sigma \times \mathbb{R}_{\geq 0} \times \mathbb{B})^*$: record reset information for each step.



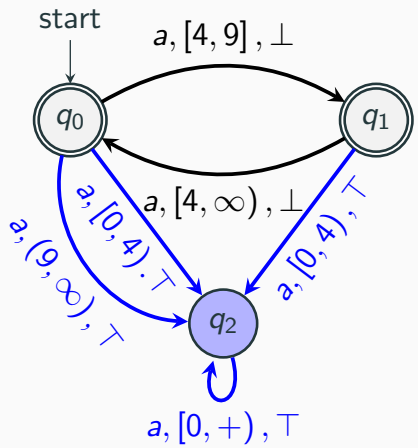
Definition (Last reset of timed word)

Given a timed word $\omega = (\sigma_1, t_1)(\sigma_2, t_2) \cdots (\sigma_n, t_n)$, and DOTA $\mathcal{A}$. Let $\omega_r = (\sigma_1, t_1, b_1)(\sigma_2, t_2, b_2) \cdots (\sigma_n, t_n, b_n)$ be the reset timed word that results from running ω on $\mathcal{A}$. The **last reset** $k_{\mathcal{A}}(\omega)$ is defined to be 0 if $b_i = \bot$ for all $1 \leq i \leq n$, and $k_{\mathcal{A}}(\omega) = i$ if $b_i = \top$ and $b_j = \bot$ for all $j > i$.

- Define $\nu_c(w, i)$ be the value of clock after executing ω if the last reset equals i, e.g. $\nu_c((a, 4.0)(a, 5.5), 1) = 5.5$.

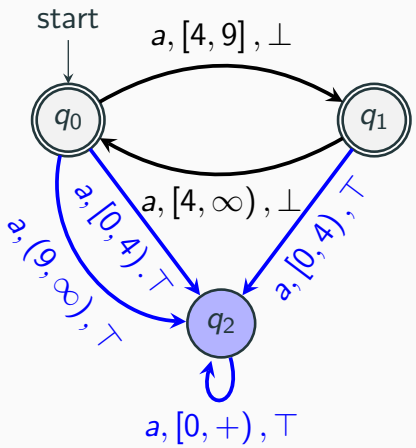
Alignment and comparison of timed words

- One key step of L^*-style framework: determine if two words ω_1 and ω_2 end in different locations.



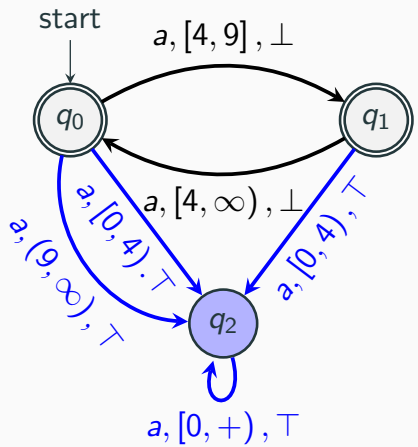
Alignment and comparison of timed words

- One key step of L^*-style framework: determine if two words ω_1 and ω_2 end in different locations.
- To compare two timed words with some suffix e, we need to **align** the values of clock before executing e. This requires knowing the (last) resets for each timed word.



Alignment and comparison of timed words

- One key step of L^*-style framework: determine if two words ω_1 and ω_2 end in different locations.
- To compare two timed words with some suffix e, we need to **align** the values of clock before executing e. This requires knowing the (last) resets for each timed word.



Example (Alignment for testing on a suffix)

Consider timed words $\omega_1 = (a, 4)$, $\omega_2 = \epsilon$, suffix $e = (a, 5.5)$, there are two cases of alignment based on different choices of last resets i_1, i_2:

1. $i_1 = i_2 = 0$, compute $\nu_c(\omega_1, 0) = 4, \nu_c(\omega_2, 0) = 0$, then $e_1 = (a, 5.5)$, $e_2 = (a, 9.5)$. $\mathrm{MQ}(\omega_1 \cdot e_1) = +, \mathrm{MQ}(\omega_1 \cdot e_2) = - \Longrightarrow$ ω_1 and ω_2 are distinguishable.
2. $i_1 = 1$, $i_2 = 0$, compute $\nu_c(\omega_1, 1) = \nu_c(\omega_2, 0) = 0$, then $e_1 = e_2 = (a, 5.5)$. $\mathrm{MQ}(\omega_1 \cdot e_1) = \mathrm{MQ}(\omega_1 \cdot e_2) = + \Longrightarrow \omega_1$ and ω_2 are indistinguishable.

Alignment and comparison of timed words (cont.)

Conside two timed words ω_1, ω_2, suppose the longest common prefixes of ω_1 and ω_2 is m, let $\mathcal{C}(\omega_1, \omega_2)$ be the set of valid combinations of last resets.

$$\mathcal{C}(\omega_1, \omega_2) = \{(i_1, i_2) \mid 0 \leq i_1 \leq |\omega_1| \wedge 0 \leq i_2 \leq |\omega_2| \wedge (i_1 \leq m \wedge i_2 \leq m \Rightarrow i_1 = i_2)\}.$$

Alignment and comparison of timed words (cont.)

Conside two timed words ω_1, ω_2, suppose the longest common prefixes of ω_1 and ω_2 is m, let $\mathcal{C}(\omega_1, \omega_2)$ be the set of valid combinations of last resets.

$$\mathcal{C}(\omega_1, \omega_2) = \{(i_1, i_2) \mid 0 \leq i_1 \leq |\omega_1| \wedge 0 \leq i_2 \leq |\omega_2| \wedge (i_1 \leq m \wedge i_2 \leq m \Rightarrow i_1 = i_2)\}.$$

Example

Suppose $\omega_1 = (a, 4)$ and $\omega_2 = (a, 4)(a, 5.5)$, then

$$\mathcal{C}(\omega_1, \omega_2) = \{(0, 0), (0, 2), (1, 1), (1, 2)\}$$

But $(1, 0)$ and $(0, 1)$ are not allowed, since the reset choices of $(a, 4)$ are contradictory.

Timed observation table

Definition (Observation table)

An observation table $\mathcal{O} = (\boldsymbol{S}, \boldsymbol{S}_+, \boldsymbol{R}, \boldsymbol{E}, f, N)$ is a sextuple, satisfying the following conditions:

- $\boldsymbol{S}, \boldsymbol{S}_+, \boldsymbol{R}$ are disjoint finite sets of timed words called prefixes. $\boldsymbol{S} \cup \boldsymbol{S}_+ \cup \boldsymbol{R}$ is prefix-closed and $\epsilon \in \boldsymbol{S}$. If $\omega \in \boldsymbol{S} \cup \boldsymbol{S}_+$ and $\sigma \in \Sigma$, then $\omega \cdot (\sigma, 0) \in \boldsymbol{S} \cup \boldsymbol{S}_+ \cup \boldsymbol{R}$.
- $\boldsymbol{E}$ is a finite set of timed words called suffixes, with $\epsilon \in \boldsymbol{E}$.
- f is a function mapping pairs $\omega_1, \omega_2 \in \boldsymbol{S} \cup \boldsymbol{S}_+ \cup \boldsymbol{R}$ and $(i_1, i_2) \in \mathcal{C}(\omega_1, \omega_2)$ to $\mathbb{B}$, indicating whether ω_1 and ω_2 are currently distinguished under last resets i_1, i_2.
- N is the current limit on the number of locations in the candidate automaton.

Timed observation table (cont.)

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

Timed observation table (cont.)

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

Prefixes set ***S*** indicates certainly distinct timed words (locations in timed automata);

Timed observation table (cont.)

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

Prefixes set $\boldsymbol{S}$ indicates certainly distinct timed words (locations in timed automata);

Auxiliary prefixes set $\boldsymbol{S}_+$ indicates timed words that are certainly distinct from rows in $\boldsymbol{S}$ under some choice of resets (possible locations in timed automata);

Timed observation table (cont.)

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

Prefixes set $\boldsymbol{S}$ indicates certainly distinct timed words (locations in timed automata);

Auxiliary prefixes set $\boldsymbol{S_+}$ indicates timed words that are certainly distinct from rows in $\boldsymbol{S}$ under some choice of resets (possible locations in timed automata);

Boundary $\boldsymbol{R}$ indicates the transitions;

Timed observation table (cont.)

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

Prefixes set $\boldsymbol{S}$ indicates certainly distinct timed words (locations in timed automata);

Auxiliary prefixes set $\boldsymbol{S_+}$ indicates timed words that are certainly distinct from rows in $\boldsymbol{S}$ under some choice of resets (possible locations in timed automata);

Boundary $\boldsymbol{R}$ indicates the transitions;

Sufffixes set $\boldsymbol{E}$ distinguishes the locations;

Timed observation table (cont.)

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

Reset variables $\mathbb{b}_i$ denotes whether clock resets after running ω_i.

Timed observation table (cont.)

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

(**Innovation**) Cells record all membership queries (different from [1]) by comparing each pair of timed words in $\boldsymbol{S} \cup \boldsymbol{S}_+ \cup \boldsymbol{R}$ under all valid combinations of last resets. Example: Given suffix $e = (a, 5.5)$, we have

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

this can be summarized as $\mathbb{b}_4$.

Encoding of readiness constraints

Definition (Encoding of last reset)

Given $\omega = (\sigma_1, t_1)...(\sigma_n, t_n) \in \boldsymbol{S} \cup \boldsymbol{S}_+ \cup \boldsymbol{R}$. Let $\omega|_i$ for $0 \leq i \leq n$ be the prefix of ω with length i. Since $\boldsymbol{S} \cup \boldsymbol{S}_+ \cup \boldsymbol{R}$ is prefix-closed, we have each $\omega|_i \in \boldsymbol{S} \cup \boldsymbol{S}_+ \cup \boldsymbol{R}$ as well. Let $lr(\omega, i)$, encoding the condition that the last reset of ω equals i, be defined as follows.

$$lr(\omega, i) \triangleq \mathbb{b}_{\omega|_i} \wedge \bigwedge_{i<j\leq n} \neg \mathbb{b}_{\omega|_j}.$$

Let $LR(\omega_1, \omega_2, i, j) = lr(\omega_1, i) \wedge lr(\omega_2, j)$.

Let $\mathbb{q}_\omega \in \{1, ..., N\}$ represent the location after running timed word ω in the candidate automation.

Encoding of readiness constraints (cont.)

1. **Distinctness of rows**: for some choices of resets, if two rows can be distinguished, then they must be assigned to different locations.

$$C_1(\omega_1, \omega_2, i, j) \triangleq LR(\omega_1, \omega_2, i, j) \Rightarrow \mathbb{q}_{\omega_1} \neq \mathbb{q}_{\omega_2}.$$

$$C_1 \triangleq \bigwedge_{\substack{\omega_1,\omega_2 \in \boldsymbol{S} \cup \boldsymbol{S}_+ \cup \boldsymbol{R}, \\ (i,j) \in \mathcal{C}(\omega_1,\omega_2), \\ f(\omega_1,\omega_2,i,j) = \bot}} C_1(\omega_1, \omega_2, i, j).$$

Encoding of readiness constraints (cont.)

1. **Distinctness of rows**: for some choices of resets, if two rows can be distinguished, then they must be assigned to different locations.

$$C_1(\omega_1, \omega_2, i, j) \triangleq LR(\omega_1, \omega_2, i, j) \Rightarrow \mathbb{q}_{\omega_1} \neq \mathbb{q}_{\omega_2}.$$

$$C_1 \triangleq \bigwedge_{\substack{\omega_1,\omega_2 \in \boldsymbol{S} \cup \boldsymbol{S}_+ \cup \boldsymbol{R}, \\ (i,j) \in \mathcal{C}(\omega_1,\omega_2), \\ f(\omega_1,\omega_2,i,j) = \bot}} C_1(\omega_1, \omega_2, i, j).$$

2. **Consistency**: if starting location, action, and region of time is the same, then reset and ending location are the same.

$$C_2(\omega_1, \omega_2, i, j, \sigma, t_1, t_2) \triangleq \mathbb{q}_{\omega_1} = \mathbb{q}_{\omega_2} \wedge LR(\omega_1, \omega_2, i, j) \Rightarrow \mathbb{b}_{\omega_1'} = \mathbb{b}_{\omega_2'} \wedge \mathbb{q}_{\omega_1'} = \mathbb{q}_{\omega_2'}.$$

$$C_2 \triangleq \bigwedge_{\substack{\omega_1,\omega_2 \in \boldsymbol{S} \cup \boldsymbol{S}_+ \cup \boldsymbol{R}, \\ \omega_1 \cdot (\sigma,t_1), \omega_2 \cdot (\sigma,t_2) \in \boldsymbol{S} \cup \boldsymbol{S}_+ \cup \boldsymbol{R}, \\ (i,j) \in \mathcal{C}(\omega_1,\omega_2), f(\omega_1,\omega_2,i,j) = \top, \\ [\![\nu_c(\omega_1,i) + t_1]\!] = [\![\nu_c(\omega_2,j) + t_2]\!]}} C_2(\omega_1, \omega_2, i, j, \sigma, t_1, t_2).$$

Encoding of readiness constraints (cont.)

3. **Closedness**: each row is assigned a location between 1 and N (optional for C_3: each location between 1 and N is represented by a row in $S \cup S_+$)

$$C_3 \triangleq \bigwedge_{1 \le i \le N} \bigvee_{\omega \in S \cup S_+} \mathbb{q}_\omega = i \ \wedge \ C_3' \text{ where } C_3' \triangleq \bigwedge_{\omega \in \boldsymbol{S} \cup \boldsymbol{S}_+ \cup \boldsymbol{R}} 1 \le \mathbb{q}_\omega \le N.$$

Encoding of readiness constraints (cont.)

3. **Closedness**: each row is assigned a location between 1 and N (optional for C_3: each location between 1 and N is represented by a row in $S \cup S_+$)

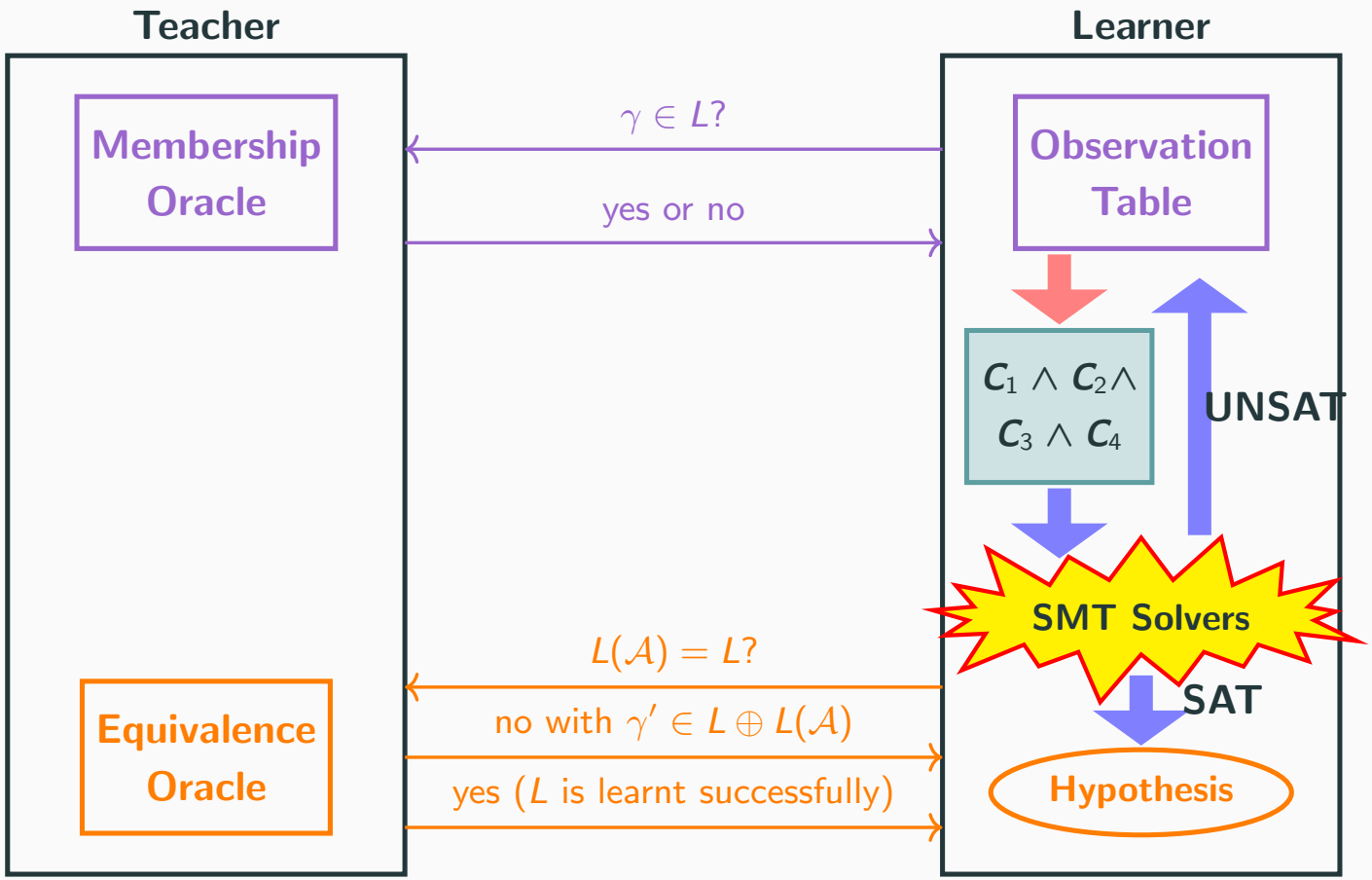
$$C_3 \triangleq \bigwedge_{1 \le i \le N} \bigvee_{\omega \in S \cup S_+} \mathbb{q}_\omega = i \ \wedge \ C_3' \text{ where } C_3' \triangleq \bigwedge_{\omega \in \boldsymbol{S} \cup \boldsymbol{S}_+ \cup \boldsymbol{R}} 1 \le \mathbb{q}_\omega \le N.$$

4. **Special assignments** (for speeding-up): rows in S are assigned to fixed, distinct locations

$$C_4 \triangleq \bigwedge_{1 \le i \le |S|} \mathbb{q}_{\omega_i} = i.$$

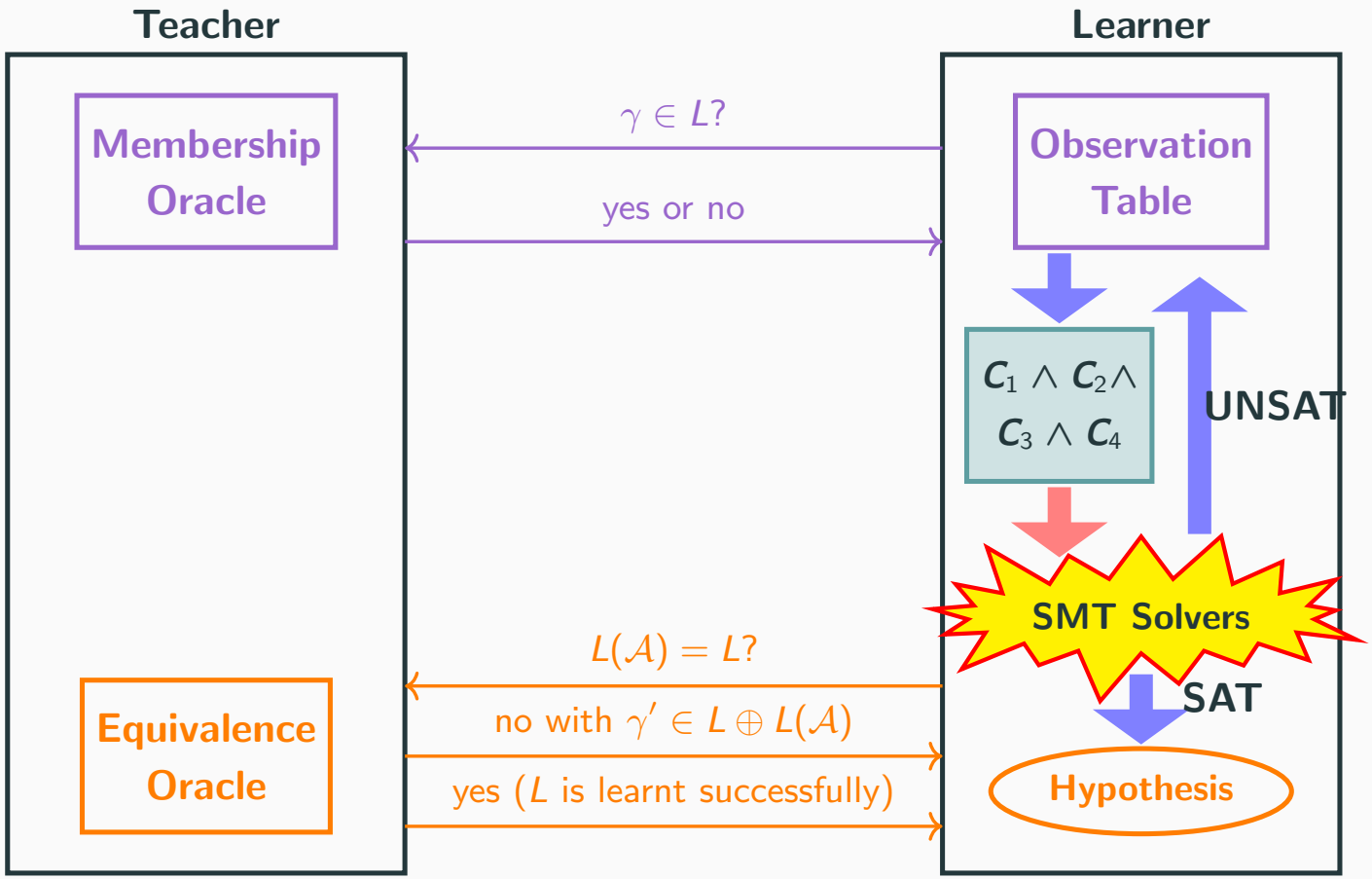
Smart learner

Build readiness condition $C = C_1 \wedge C_2 \wedge C_3 \wedge C_4$.



Smart learner

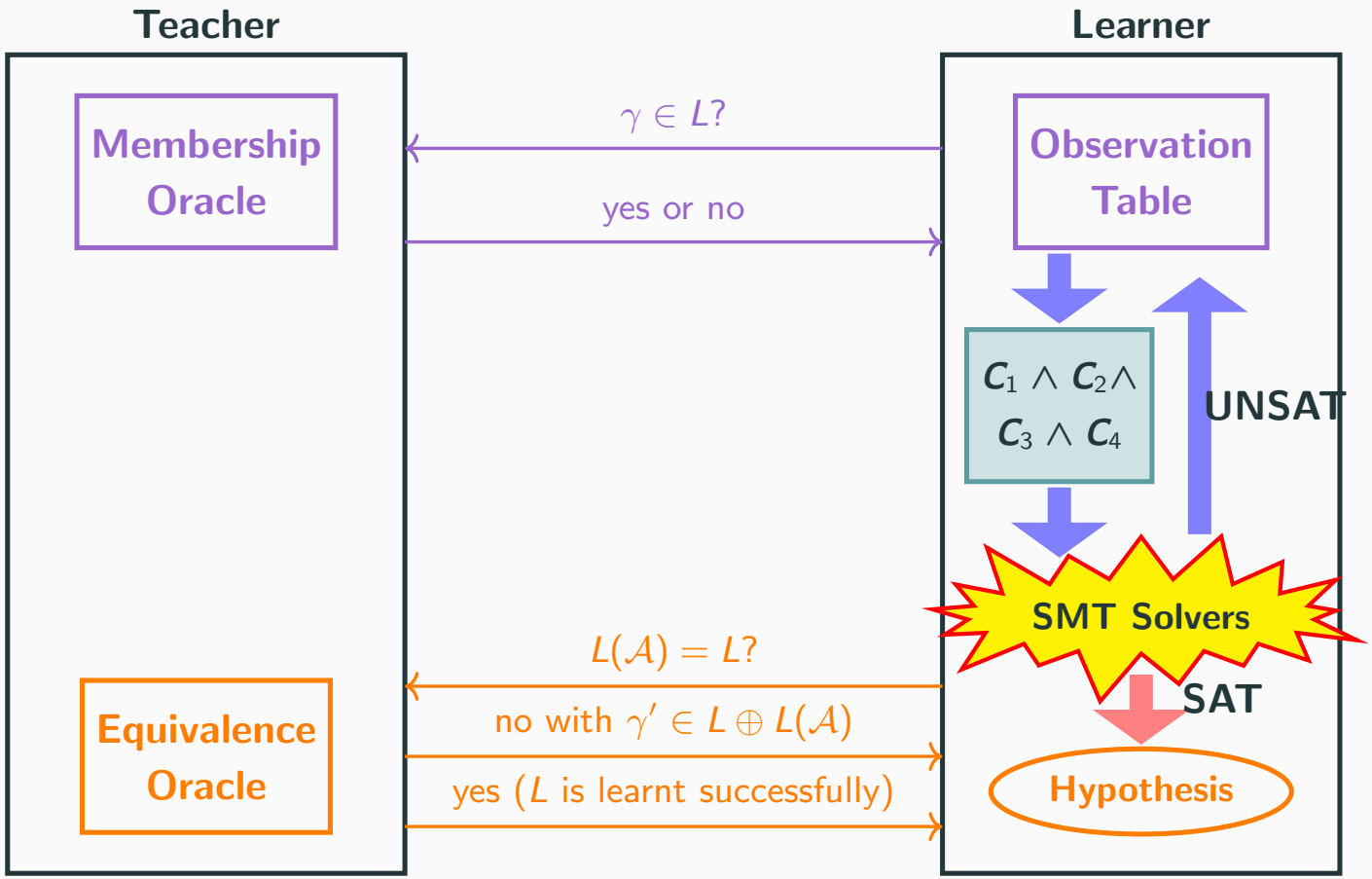
Send C to SMT solver.



Smart learner

SAT

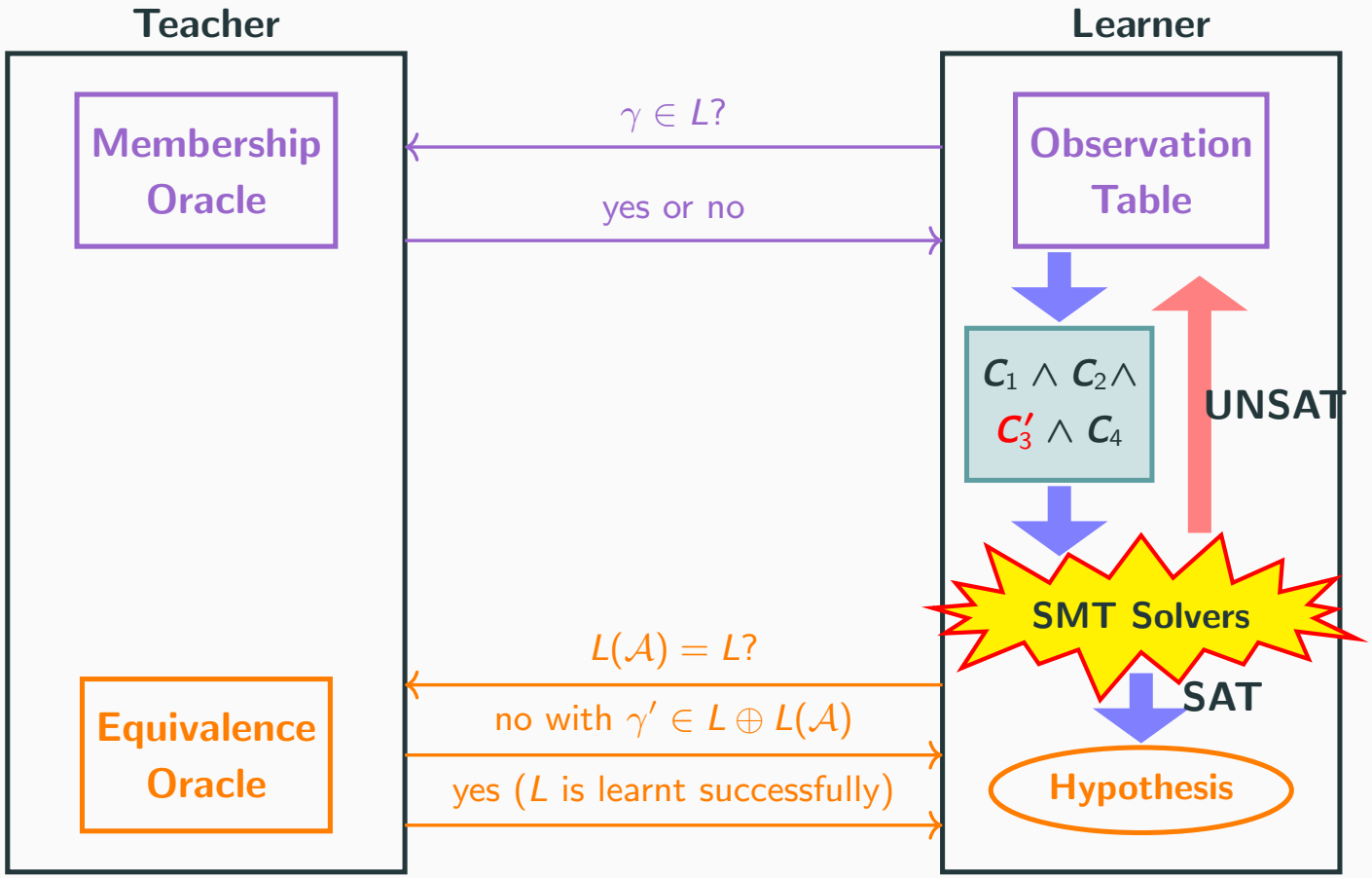
Construct hypothesis model with the assignments of resets and locations.



Smart learner

UNSAT

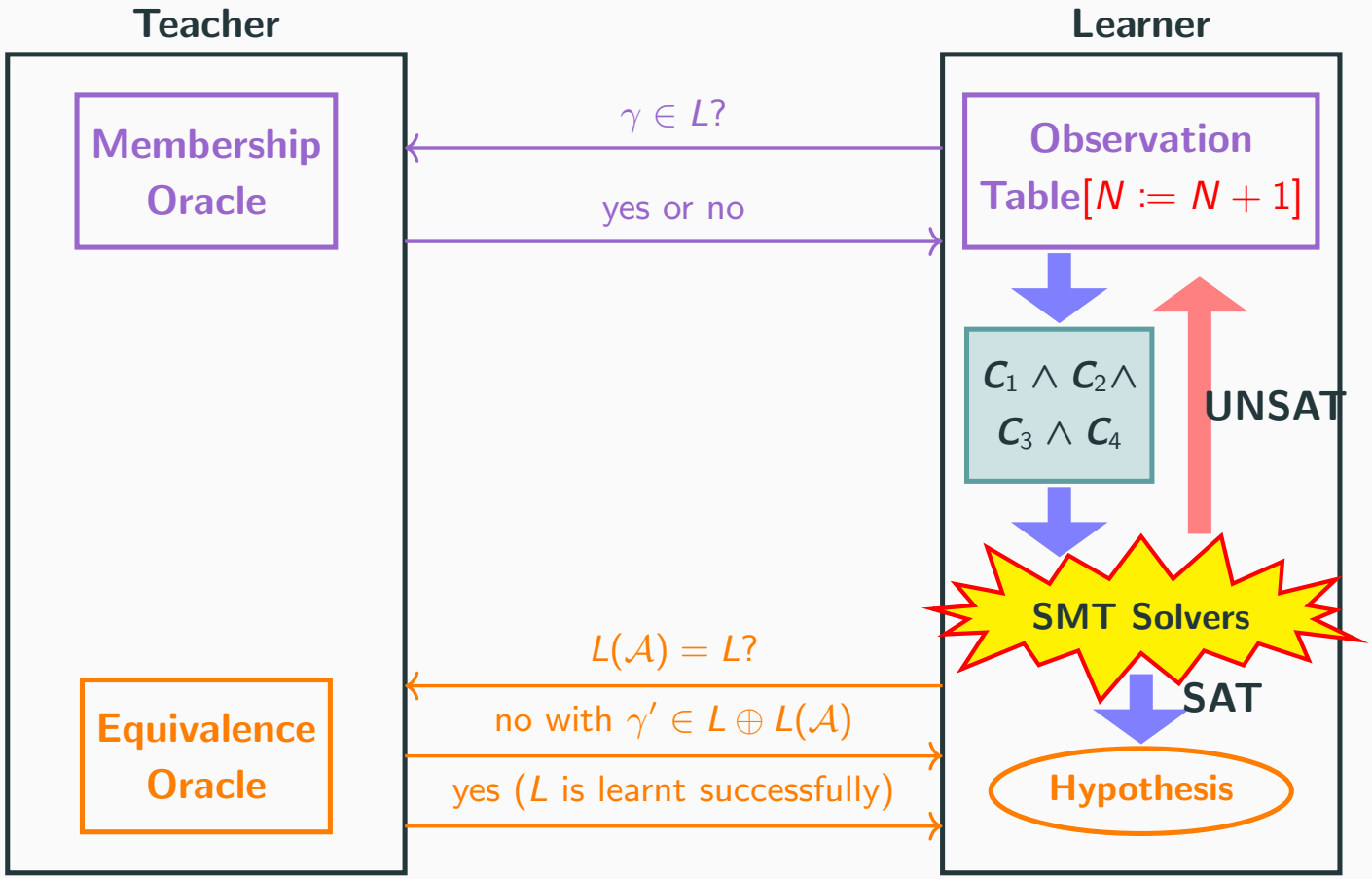
Case 1: C_3 is too stringent, no reset assignment can fulfil the closedness condition $\Longrightarrow C := C[C_3'/C_3]$.



Smart learner

UNSAT

Case 2 ($C_1 \wedge C_2 \wedge C_3' \wedge C_4$ is still UNSAT): The number of states in SUL is larger than $N \Longrightarrow$ increase N in observation table.



Correctness and termination

Theorem (Correctness and termination)

The learning process always terminates and returns a correct DOTA recognizing the underlying target timed language.

Outline of proof: by comparison with a brute-force of the algorithm that searches through reset assignments.

Experiment

Table 1: Experimental results on DOTAs.

Group	$\lvert\Delta\rvert$	Method	#Membership			#Equivalence			$\lvert Q_{\mathcal{H}}\rvert$	#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	100.22
		SL	1740	3497.7	5329	24	32.8	42	4.0	10/10	1.42
7_4_20	26.0	SL	6092	9393.3	15216	44	51.5	69	7.0	10/10	2.90
10_4_20	39.1	SL	8579	16322.3	23726	59	76.5	93	10.0	10/10	5.89
12_4_20	47.6	SL	13780	20345.5	29011	70	88.0	102	12.0	10/10	10.05
14_4_20	58.4	SL	18915	28569.0	40693	92	110.6	126	14.0	10/10	14.69
AKM (17_12_5)	40.0	SL	3453	3453.0	3453	49	49.0	49	12	1/1	7.19
TCP (22_13_2)	22.0	SL	4713	4713.0	4713	32	32.0	32	20	1/1	19.04
CAS (14_10_27)	23.0	SL	4769	4769.0	4769	18	18.0	18	14	1/1	126.30
PC (26_17_10)	42.0	SL	10854	10854.0	10854	28	28.0	28	25	1/1	109.01

Group: each group has ID of the form $\lvert Q\rvert\_\lvert\Sigma\rvert\_\kappa$, where $\lvert Q\rvert$ is the number of locations, $\lvert\Sigma\rvert$ is the size of the alphabet, and κ is the maximum constant appearing in the clock constraints.
$\lvert\Delta\rvert$: average number of transitions of a DOTA in the corresponding group.
Method: DOTAL and SL represent the method in [1] and our method respectively.
#Membership & **#Equivalence**: number of membership and equivalence queries, respectively. $N_{\min}$: minimal, N_{mean}: mean, $N_{\max}$: maximum. $\lvert Q_{\mathcal{H}}\rvert$: average number of locations of the learned automata for each group.
#Learnt: the number of the learnt DOTAs (learnt/total).
t: average wall-clock time in seconds.

Experiment

- Adapted the algorithm to Deterministic Timed Mealy Machines.
- Each transition also contain input/output. Membership query returns the sequence of outputs after giving the sequence of inputs.
- Evaluated the algorithm on DTMM versions of AKM, TCP, CAS, PC examples.

Case	$\|Q\|$	$\|I\|$	$\|\Delta\|$	#M	#E	$t(s)$
AKM	5	5	28	691	34	2.6
TCP	11	8	19	751	10	1.9
CAS	8	4	17	1654	21	17.1
PC	8	8	24	1194	27	6.8

Table 2: Results on the deterministic timed Mealy machines. $|Q|$: number of locations. $|I|$: number of input actions. $|\Delta|$: number of transitions. #M: number of membershp queries. #E: number of equivalence queries. $t(s)$: wall-clock time in seconds.

Conclusion and future work

Conclusion and future work

- Contribution
  - Propose a new algorithm for active learning of one clock timed automata and timed Mealy machines with constraint solving. Taking advantage of the ability SMT solvers to solve large constraints, this algorithm can scale up to larger timed automata models.
  - Incorporate constraint solving into active learning, for determining unknown data in the observation table.

Conclusion and future work

- Contribution
  - Propose a new algorithm for active learning of one clock timed automata and timed Mealy machines with constraint solving. Taking advantage of the ability SMT solvers to solve large constraints, this algorithm can scale up to larger timed automata models.
  - Incorporate constraint solving into active learning, for determining unknown data in the observation table.
- Future work
  - Extension to multi-clock timed automata and other types of automata.
  - Improvements to efficiency of the algorithm (e.g. reducing the number of membership queries).

References i

- Jie An, Mingshuai Chen, Bohua Zhan, Naijun Zhan, and Miaomiao Zhang.
  Learning one-clock timed automata.
  In TACAS 2020, pages 444–462. Springer, 2020.
- Jie An, Lingtai Wang, Bohua Zhan, Naijun Zhan, and Miaomiao Zhang.
  Learning real-time automata.
  Sci. China Inf. Sci., 64(9), 2021.
- Jie An, Bohua Zhan, Naijun Zhan, and Miaomiao Zhang.
  Learning nondeterministic real-time automata.
  ACM Trans. Embed. Comput. Syst., 20(5s):99:1–99:26, 2021.
- Dana Angluin.
  Learning regular sets from queries and counterexamples.
  Inf. Comput., 75(2):87–106, 1987.

References ii

- Rick Smetsers, Paul Fiterau-Brostean, and Frits W. Vaandrager.
  Model learning as a satisfiability modulo theories problem.
  In LATA 2018, pages 182–194. Springer, 2018.
- Martin Tappler, Bernhard K. Aichernig, and Florian Lorber.
  Timed automata learning via SMT solving.
  In NFM 2022, pages 489–507. Springer, 2022.
- Frits W. Vaandrager, Roderick Bloem, and Masoud Ebrahimi.
  Learning Mealy machines with one timer.
  In LATA 2021, pages 157–170. Springer, 2021.