An Efficient Algorithm for Learning Event-Recording Automata

Shang-Wei LIN, Étienne André, Jin Song DONG, Jun SUN, and Yang LIU

Department of Computer Science
School of Computing
National University of Singapore

October 14, 2011

Motivation

Why is learning of models important?

- Automatic inference or construction of abstract models

Outline

Outline

The L* Algorithm

Timed Language and Event-Recording Automata

The TL* Algorithm

Conclusion and Future Work

The L* Algorithm

The L* algorithm is a formal method to learn a minimal DFA that accepts an unknown language U over an alphabet Σ.

- D. Angluin, "Learning regular sets from queries and counterexamples," *Information and Computation* 75 (1987), no. 2, 87-106.

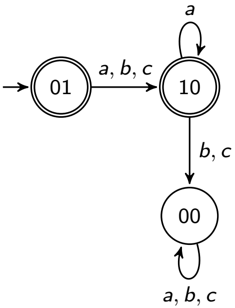
The L$*$ algorithm interacts with a Minimal Adequate *Teacher*

- *membership query*
  - Is a string in the unknown language U?
- *candidate query*
  - Does a DFA accept the unknown language U?

The L* Algorithm (cont.)

The unknown language $U = (a \mid b \mid c) \cdot a^*$

	λ	c
λ	0	1
a	1	0
b	1	0
c	1	0
$a \cdot a$	1	0
$a \cdot b$	0	0
$a \cdot c$	0	0
$ab \cdot a$	0	0
$ab \cdot b$	0	0
$ab \cdot c$	0	0



Outline

Timed Language

Let Σ be a finite alphabet.

For every symbol (event) $a \in \Sigma$, we use x_a to denote the *event-recording clock* of the symbol a

- x_a records the time elapsed since the last occurrence of the symbol a
- We use C_Σ to denote the set of event-recording clocks over Σ

An *atomic clock guard* τ is an inequation of the form $x_a \sim n$ for $x_a \in C_\Sigma$, $\sim \in \{<, \leq, >, \geq\}$, and $n \in N$.

- A *clock guard* g is a conjunction of atomic clock guards.
- We use G_Σ to denote the set of clock guards over C_Σ

A *guarded word* is a sequence $w_g = (a_1, g_1)(a_2, g_2) \cdots (a_n, g_n)$ where $a_i \in \Sigma$ for $i \in \{1, 2, \ldots, n\}$ and $g_i \in G_\Sigma$ is a clock guard.

Event-Recording Automata

An *event-recording automaton* (ERA) $D = (\Sigma, L, l_0, \delta, L^f)$ consists of

- a finite input *alphabet* Σ
- a finite set of *locations* L
- an *initial location* $l_0 \in L$
- a *transition function* $\delta : L \times \Sigma \times G_\Sigma \mapsto 2^L$
- a set of *accepting locations* $L^f \subseteq L$

Each event-recording clock $x_a \in C_\Sigma$ is implicitly and automatically *reset* when a transition with event a is taken.

Event-Recording Automata (cont.)

An event-recording automaton $D = (\Sigma, L, l_0, \delta, L^f)$ is *deterministic* if

- $\delta(l, a, g)$ is a singleton set when it is defined
- if both $\delta(l, a, g_1)$ and $\delta(l, a, g_2)$ are both defined, then $[[g_1]] \cap [[g_2]] = \emptyset$

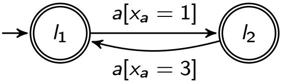
A guarded word $w_g = (a_1, g_1)(a_2, g_2) \cdots (a_n, g_n)$ is *accepted* by an ERA $D = (\Sigma, L, l_0, \delta, L^f)$ if

- $l_i = \delta(l_{i-1}, a_i, g_i)$ is defined for all $i \in \{1, 2, \ldots, n\}$
- $l_n \in L^f$

The *timed language* accepted by D, denoted by $\mathcal{L}(D)$, is the set of guarded words accepted by D.

Event-Recording Automata (cont.)

The following ERA $\mathcal{A}_1$ accepts the timed language $U_T = ((a, x_a = 1)(a, x_a = 3))^*$



$\mathcal{A}_1$

Outline

The TL* Algorithm

The TL* algorithm is a *timed* extension of the L* algorithm.

The TL* algorithm is a formal method to learn a minimal event-recording automaton (ERA) that accepts an unknown timed language U_T over an alphabet Σ

- We use U to denote the untimed language of U_T

The TL* Algorithm (cont.)

The TL* algorithm has to interact with a Minimal Adequate *Teacher*

- *untimed membership query* Q_m
  - Is an untimed word in the unknown untimed language U?
- *untimed candidate query* Q_c
  - Does a DFA accept the unknown untimed language U?
- *timed membership query* Q_m^T
  - Is a guarded word in the unkown timed language U_T?
- *timed candidate query* Q_c^T
  - Does an ERA accept the unknown timed language U_T?

The TL* Algorithm (cont.)

The TL* algorithm consists of two phases

- *Untimed Learning* Phase
  - The L* algorithm is used to learn a DFA M accepting the untimed language U
- *Timed Refinement* Phase
  - The DFA M is refined into an event-recording automaton (ERA) by adding time constraints

The TL* Algorithm (cont.)

input : Σ: alphabet, C_Σ: the set of event-recording clocks
output: a deterministic ERA accepting the unknown timed language U_T

Use L^* to learn a DFA M accepting $Untime(U_T)$;
Let (S, E, T) be the observation table during the L* learning process ;
Replace α by $(\alpha, true)$, s by $(s, true)$, and e by $(e, true)$ for each $\alpha \in \Sigma$, $s \in S$ and $e \in E$;
while *true* **do**
    if $Q_c^T(M) = 1$ **then** **return** M ;
    else
        Let $(a_1, g_1)(a_2, g_2) \cdots (a_n, g_n)$ be the counterexample given by Teacher ;
        foreach (a_i, g_i), $i \in \{1, 2, \ldots, n\}$ **do**
            if (a_i, g) *is a substring of* p *or* e *for some* $p \in S \cup (S \cdot \Sigma)$ *and* $e \in E$ *such that* $[[g_i]] \subset [[g]]$ **then**
                Let $G = \{\hat{g_1}, \hat{g_2}, \ldots, \hat{g_m}\}$ obtained by $[[g]] - [[g_i]]$;
                $\Sigma = \Sigma \setminus \{(a_i, g)\} \cup \{(a_i, g_i), (a_i, \hat{g_1}), (a_i, \hat{g_2}), \ldots, (a_i, \hat{g_m})\}$;
                Split p into $\{\hat{p_0}, \hat{p_1}, \hat{p_2}, \ldots, \hat{p_m}\}$ where (a_i, g_i) is a substring of $\hat{p_0}$ and $(a_i, \hat{g_j})$ is a substring of $\hat{p_j}$ for all $j \in \{1, 2, \ldots, m\}$;
                Split e into $\{\hat{e_0}, \hat{e_1}, \hat{e_2}, \ldots, \hat{e_m}\}$ where (a_i, g_i) is a substring of $\hat{e_0}$ and $(a_i, \hat{g_j})$ is a substring of $\hat{e_j}$ for all $j \in \{1, 2, \ldots, m\}$;
                Update T by $Q_{m_T}(\hat{p_j} \cdot \hat{e_j})$ for all $j \in \{0, 1, 2, \ldots, m\}$;

        while *there exists* $(s \cdot \alpha)$ *such that* $s \cdot \alpha \not\equiv s'$ *for all* $s' \in S$ **do**
            $S \longleftarrow S \cup \{s \cdot \alpha\}$;
            Update T by $Q_{m_T}((s \cdot \alpha) \cdot \beta)$ for all $\beta \in \Sigma$;

        $v \longleftarrow WS((a_1, g_1)(a_2, g_2) \cdots (a_n, g_n))$;
        if $|v| > 0$ **then**
            $E \longleftarrow E \cup \{v\}$;
            Update T by $Q_{m_T}(s \cdot v)$ and $Q_{m_T}(s \cdot \alpha \cdot v)$ for all $s \in S$ and $\alpha \in \Sigma$;

        Construct candidate M from (S, E, T) ;

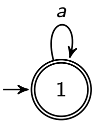
An Example

Suppose $U_T = ((a, x_a = 1)(a, x_a = 3))^*$ is the timed language to be learned.

Untimed Learning Phase

	λ
λ	1 (s_0)
a	1

(a) T_1



(b) M_1

	λ
λ	1 (s_0)
(a, $true$)	1

(c) T_2

$\mathcal{L}(M_1) = U = a^*$

An Example (cont.)

$Q_c^T(M_1) = 0$ with a negative counterexample $(a, x_a < 1)$

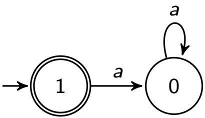
Timed Refinement 1

	λ
λ	1 (s_0)
$(a, x_a < 1)$	0
$(a, x_a \geq 1)$	0

(a) T_3

	λ
λ	1 (s_0)
$(a, x_a < 1)$	0 (s_1)
$(a, x_a \geq 1)$	0
$(a, x_a < 1)(a, x_a < 1)$	0
$(a, x_a < 1)(a, x_a \geq 1)$	0

(b) T_4



(c) M_2

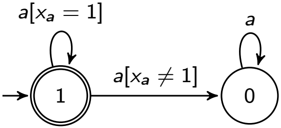
An Example (cont.)

$Q_c^T(M_2) = 0$ with a positive counterexample $(a, x_a = 1)$

Timed Refinement 2

	λ
λ	$1\ (s_0)$
$(a, x_a < 1)$	$0\ (s_1)$
$(a, x_a = 1)$	1
$(a, x_a > 1)$	0
$(a, x_a < 1)(a, x_a < 1)$	0
$(a, x_a < 1)(a, x_a = 1)$	0
$(a, x_a < 1)(a, x_a > 1)$	0

(a) T_5



(b) M_3

An Example (cont.)

$Q_c^T(M_3) = 0$ with a negative counterexample $(a, x_a = 1)(a, x_a = 1)$

A suffix $(a, x_a = 1)$ shows that λ and $(a, x_a = 1)$ should not be in the same class

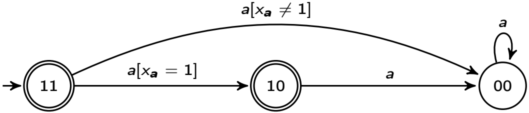
Timed Refinement 3

	λ	$(a, x_a = 1)$
λ	1	1 (s_0)
$(a, x_a < 1)$	0	0 (s_1)
$(a, x_a = 1)$	1	0
$(a, x_a > 1)$	0	0
$(a, x_a < 1)(a, x_a < 1)$	0	0
$(a, x_a < 1)(a, x_a = 1)$	0	0
$(a, x_a < 1)(a, x_a > 1)$	0	0

(a) T_6

	λ	$(a, x_a = 1)$
λ	1	1 (s_0)
$(a, x_a < 1)$	0	0 (s_1)
$(a, x_a = 1)$	1	0 (s_2)
$(a, x_a > 1)$	0	0
$(a, x_a < 1)(a, x_a < 1)$	0	0
$(a, x_a < 1)(a, x_a = 1)$	0	0
$(a, x_a < 1)(a, x_a > 1)$	0	0
$(a, x_a = 1)(a, x_a < 1)$	0	0
$(a, x_a = 1)(a, x_a = 1)$	0	0
$(a, x_a = 1)(a, x_a > 1)$	0	0

(b) T_7



(c) M_4

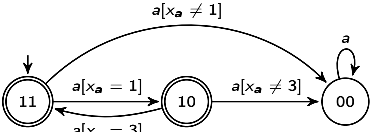
An Example (cont.)

$Q_c^T(M_4) = 0$ with a positive counterexample $(a, x_a = 1)(a, x_a = 3)$

Timed Refinement 4

	λ	$(a, x_a = 1)$
λ	1	1 (s_0)
$(a, x_a < 1)$	0	0 (s_1)
$(a, x_a = 1)$	1	0 (s_2)
$(a, 1 < x_a < 3)$	0	0
$(a, x_a = 3)$	0	0
$(a, x_a > 3)$	0	0
$(a, x_a < 1)(a, x_a < 1)$	0	0
$(a, x_a < 1)(a, x_a = 1)$	0	0
$(a, x_a < 1)(a, 1 < x_a < 3)$	0	0
$(a, x_a < 1)(a, x_a = 3)$	0	0
$(a, x_a < 1)(a, x_a > 3)$	0	0
$(a, x_a = 1)(a, x_a < 1)$	0	0
$(a, x_a = 1)(a, x_a = 1)$	0	0
$(a, x_a = 1)(a, 1 < x_a < 3)$	0	0
$(a, x_a = 1)(a, x_a = 3)$	1	1
$(a, x_a = 1)(a, x_a > 3)$	0	0

(a) T_8



(b) M_5

An Example (cont.)

$Q_c^T(M_5) = 1$, i.e., $\mathcal{L}(M_5) = U_T$

The learning process of TL* is finished

Analysis of TL*

Given a timed language U_T accepted by an ERA $\mathcal{A} = (\Sigma, L, l_0, \delta, L^f)$, the TL* algorithm needs to perform

- $O(|\Sigma| \cdot |G_\mathcal{A}| \cdot |L|^2 + |L| \log |\pi|)$ timed membership queries
  - π is the counterexample given by Teacher
  - $G_\mathcal{A}$ is the set of clock zones partitioned by the clock guards appearing in $\mathcal{A}$
- $O(|L| + |\Sigma| \cdot |G_\mathcal{A}|)$ timed candidate queries

Grinchtein's TL^*_{sg} needs $O(|\Sigma \times G_\Sigma| \cdot n^2 |\pi| \cdot |w| \binom{|\Sigma|+K}{|\Sigma|})$ timed membership queries

- n is the number of locations of the learned ERA
- w is the longest guarded word queried
- K is the largest constant appearing in the clock guards

Analysis of TL* (cont.)

Theorem

The TL algorithm is correct.*

Theorem

The TL algorithm terminates.*

Theorem

Assume the observation table (S, E, T) is closed and consistent and $M = (\Sigma, L, l^0, \delta, L^f)$ is the ERA constructed from the observation table (S, E, T). If $M' = (\Sigma, L', l^{0'}, \delta', L^{f'})$ is any other ERA consistent with T, then M' has at least $|L|$ locations.

Outline

Conclusion and Future Work

Conclusion

- We proposed an efficient polynomial time algorithm, TL$*$, for learning event-recording automata (ERA).
- The TL* algorithm has been implemented in the PAT model checker.

Future Work

- To extend the TL* algorithm to learn other subclasses of timed automata
- To automate assume-guarantee reasoning (AGR) for timed systems, we plan to use TL* to automatically generate timed assumptions needed for AGR.