Learning Doubly Labeled Automata using Queries and Counterexamples
Unpublished Manuscript

Sagar Chaki
chaki@sei.cmu.edu

Software Engineering Institute, CMU

1 Background and Notation

In this section we present some basic definitions. We begin with doubly labeled automata (DLAs) which can be thought of as Kripke structres enhanced in two directions: (i) transitions are labeled with actions, and (ii) there is a notion of final or accepting states.

Definition 1 (DLA). *A doubly labeled automaton (DLA) is a 7-tuple* $(S, \mathit{Init}, AP, \mathcal{L}, \Sigma, \delta, F)$ *with (i)* S *a finite set of* states, *(ii)* $\mathit{Init} \subseteq S$ *a set of initial states, (iii)* AP *a finite set of (atomic) state propositions, (iv)* $\mathcal{L} : S \to 2^{AP}$ *a state-labeling function, (v)* Σ *a finite set of events or actions (alphabet), (vi)* $\delta \subseteq S \times \Sigma \times S$ *a transition relation, and (vii)* $F \subseteq S$ *is a set of final states.*

Definition 2 (Concatenation). *Given two sets* X *and* Y *we will denote the set* $\{x \bullet y \mid x \in X \wedge y \in Y\}$ *by* $X \bullet Y$ *where* $x \bullet y$ *denotes the concatenation of* x *and* y*.*

Definition 3 (Word). *Let* M *be any DLA with alphabet* Σ *and set of propositions* AP*. Let us denote the extended alphabet* $\Sigma \bullet 2^{AP}$ *by* $\widehat{\Sigma}$*. Let us denote the set* $2^{AP} \bullet \widehat{\Sigma}^*$ *by* W*. An element of* W *is called a word of* M*. Intuitively, a word of* M *is any non-empty finite alternating sequence of propositional labelings and actions of* M *begining and ending with a propositional labeling. In particular, note that if* x *is a word and* $y \in \widehat{\Sigma}^*$ *then* $x \bullet y$ *is again a word.*

In the rest of this article we will often omit the DLA M associated with a word whenever that DLA is clear from the context.

Definition 4 (Reachability). *Let* $M = (S, \mathit{Init}, AP, \mathcal{L}, \Sigma, \delta, F)$ *be any DLA and* w *be any word of* M*. We denote by* $\mathit{Reach}(M, w)$ *the set of states that* M *can reach by starting from an initial state and simulating* w*. More precisely, let* $w = \langle P_1, \alpha_1, P_2, \alpha_2, \ldots, P_{n-1}, \alpha_{n-1}, P_n \rangle$*. Then* $\mathit{Reach}(M, w)$ *is the set of all states* s *of* M *which satisfy the following criteria: there exists a sequence of states* $\langle s_1, \ldots, s_n \rangle \in S^*$ *such that: (i)* $s_1 \in \mathit{Init}$*, (ii) for* $1 \le i \le n$*,* $P_i = \mathcal{L}(s_i)$*, (iii)* $1 \le i < n$*,* $(s_i, \alpha_i, s_{i+1}) \in \delta$*, and (iii)* $s = s_n$*.*

Definition 5 (Trace and Language). *Let $M = (S, Init, AP, \mathcal{L}, \Sigma, \delta, F)$ be an DLA. A word w is said to be a trace of M iff $Reach(M, w) \cap F \neq \emptyset$. In other words, w is a trace of M iff M can reach an accepting state after simulating w from an initial state. The language of M, denoted by $L(M)$ is the set of all its traces. Thus:*

$$L(M) = \{w \mid Reach(M, w) \cap F \neq \emptyset\}$$

Definition 6 (Trace containment). *An DLA M_1 is said to be trace contained by an DLA M_2 if $L(M_1) \subseteq L(M_2)$. We denote this by $M_1 \preccurlyeq M_2$.*

Definition 7 (Deterministic DLA). *An DLA $M = (S, Init, AP, \mathcal{L}, \Sigma, \delta, F)$ is said to be a deterministic DLA (DDLA) iff the following conditions hold:*

– *For every possible propositional labeling P, M has exactly one initial state labeled with P.*

$$\forall P \subseteq AP \centerdot \exists ! s \in Init \centerdot \mathcal{L}(s) = P$$

– *Every state must have exactly one α-successor labeled with P for each α in the alphabet and each possible propositional labeling P.*

$$\forall s_1 \in S \centerdot \forall \alpha \in \Sigma \centerdot \forall P \subseteq AP \centerdot \exists ! s_2 \in S \centerdot s_1 \xrightarrow{\alpha} s_2 \wedge \mathcal{L}(s_2) = P$$

It is obvious from the above definition that a deterministic DLA M can always simulate any word w. In addition, it reaches a unique state after simulating w. This is expressed by the following theorem.

Theorem 1. *For any deterministic DLA M and any word w of M, $|Reach(M, w)| = 1$.*

The following theorem expresses the equivalence of non-deterministic and deterministic DLAs as far as languages are concerned.

Theorem 2. *For every non-deterministic DLA M_1 there exists a deterministic DLA M_2 such that $L(M_1) = L(M_2)$.*

Proof. The proof is constructive. Given a non-deterministic DLA M_1 we can convert it to a deterministic DLA via subset construction. This will be slightly different from the subset construction for ordinary finite automata since we will only group together successors that can be reached via the same action and also agree on their propositional labelings.

□

Definition 8 (Composition). *Let $M_1 = (S_1, Init_1, AP_1, \mathcal{L}_1, \Sigma_1, \delta_1, F_1)$ and $M_2 = (S_2, Init_2, AP_2, \mathcal{L}_2, \Sigma_2, \delta_2, F_2)$ be two DLAs. The parallel composition of M_1 and M_2, denoted by $M_1 \parallel M_2$, is the DLA $(S_1 \times S_2, Init_1 \times Init_2, AP_1 \cup AP_2, \mathcal{L}, \Sigma_1 \cup \Sigma_2, \delta, F_1 \times F_2)$, where: (i) $\mathcal{L}(s_1, s_2) = \mathcal{L}_1(s_1) \cup \mathcal{L}_2(s_2)$, and (ii) δ is such that $(s_1, s_2) \xrightarrow{\alpha} (s'_1, s'_2)$ iff:*

$$\forall i \in \{1, 2\} \centerdot (\alpha \notin \Sigma_i \wedge s_i = s'_i) \bigvee (\alpha \in \Sigma_i \wedge s_i \xrightarrow{\alpha} s'_i)$$

In other words, DLAs must synchronize on shared actions and proceed independently on local actions. This notion of parallel composition is derived from CSP [3].

2 DLA Learning

Our algorithm, called $SE - L^*$, is based on the L^* algorithm which developed by Angluin [1] and later improved by Rivest et. al. [2]. In the rest of this article, we will denote the symmetric difference of two sets X and Y by $X \triangle Y$.

2.1 Preliminaries

Let U be an unknown DLA over some alphabet Σ and set of propsotions AP. Recall from Definition 3 that we denote the set $\Sigma \bullet 2^{AP}$ by $\widehat{\Sigma}$. Also recall the definition of a word and that the set of all words is denoted by W.

Definition 9 (Prefix Closed). *A set $X \subseteq W$ is said to prefix-closed if for each $x \in X$, it is the case that every element of W that is a prefix of x is also in X. Note that we are not interested in every prefix of x but only in those prefixes of x which are also words.*

Definition 10 (Last Label). *For any word $w = \langle P_1, \alpha_1, \ldots, P_n \rangle$ we write $\mathcal{P}(w)$ to mean P_n. Thus $\mathcal{P}(w)$ is the propositional labeling at the end of w.*

In the rest of this chapter we present the core $SE - L^*$ algorithm. We begin with the notion of a minimally adequate teacher along the lines of the L^* algorithm.

2.2 Minimally Adequate Teacher

In order to learn U, $SE - L^*$ interacts with a *minimally adequate teacher* MAT for U, which can provide Boolean answers to the following two kinds of queries:

1. *Membership*. Given a word $\rho \in W$, MAT returns TRUE iff $\rho \in L(U)$.
2. *Candidate*. Given an DLA D, MAT returns TRUE iff $L(D) = L(U)$. If MAT returns FALSE, it also returns a counterexample word $w \in L(D) \triangle L(U)$.

2.3 Observation Table

The $SE - L^*$ algorithm constructs iteratively a minimal DDLA D such that $L(D) = L(U)$. It maintains an observational table $\mathcal{T} = (S, E, T)$ where:

- $S \subseteq W$ is a prefix-closed set of words
- $E \subseteq \widehat{\Sigma}^*$ is a set of experiments
- $T : (S \cup S \bullet \widehat{\Sigma}) \bullet E \to \{0, 1\}$ is a function such that:

$$\forall s \in S \cup S \bullet \widehat{\Sigma} \centerdot \forall e \in E \centerdot T[s \bullet e] = 1 \iff s \bullet e \in L(U)$$

Intuitively, one can think of $\mathcal{T}$ as a two dimensinal table. The rows of $\mathcal{T}$ are labeled with the elements of $S \cup S \bullet \widehat{\Sigma}$ while the columns are labeled with elements of E. Finally T denotes the table entries. In other words the entry corresponding to row s and column e is simply $T(s \bullet e)$. Carrying on with this intuition, let us define, for any $s \in S \cup S \bullet \widehat{\Sigma}$, a function $row(s)$ as follows:

$$\forall e \in E \centerdot row(s)(e) = T[s \bullet e]$$

Intuitively, $row(s)$ denotes the sequence of table entries corresponding to the row s. For any two elements s_1 and s_2 of $S \cup S \bullet \widehat{\Sigma}$, let us write $\mathit{Diff}(s_1, s_2)$ to mean that s_1 and s_2 either have different propositional labelings at the end or have different corresponding row entries. In other words:

$$\mathit{Diff}(s_1, s_2) \equiv \mathcal{P}(s_1) \neq \mathcal{P}(s_2) \vee row(s_1) \neq row(s_2)$$

Then the following invariant will always hold on the table maintained by $SE - L^*$:

$$\textbf{CONSISTENCY}: \quad \forall s_1, s_2 \in S \centerdot s_1 \neq s_2 \implies \mathit{Diff}(s_1, s_2)$$

In other words, any two distinct entries in S must differ either in their end propositional labelings or in their corresponding row entries. A table $\mathcal{T} = (S, E, T)$ is said to be *closed* if the following condition hold:

$$\forall t \in S \bullet \widehat{\Sigma} \centerdot \exists s \in S \centerdot \neg \mathit{Diff}(s, t)$$

The following theorem is crucial for proving termination of $SE - L^*$. It essentially provides an upper bound on the size of S.

Theorem 3. *Let n be the number of states in a minimal DDLA M such that $L(M) = L(U)$. Then the size of S cannot exceed n.*

Proof. The proof is by contradiction. Suppose that the size of S exceeds n. Then by the pigeon-hole principle, there exists two elements s_1 and s_2 of S such that $s_1 \neq s_2$ and $\mathit{Reach}(M, s_1) = \mathit{Reach}(M, s_2)$. From this it follows that $\mathcal{P}(s_1) = \mathcal{P}(s_2)$ and $row(s_1) = row(s_2)$. In other words $\neg \mathit{Diff}(s_1, s_2)$. However this contradicts the **CONSISTENCY** invariant since we now have two distinct elements of S which do not differ. This concludes the proof.

□

2.4 Candidate Construction

Given a closed table $\mathcal{T} = (S, E, T)$, we denote by $M_{\mathcal{T}} = (S_{\mathcal{T}}, \mathit{Init}_{\mathcal{T}}, AP_{\mathcal{T}}, \mathcal{L}_{\mathcal{T}}, \Sigma_{\mathcal{T}}, \delta_{\mathcal{T}}, F_{\mathcal{T}})$ the DDLA constructed from $\mathcal{T}$ as follows:

- The states of $M_{\mathcal{T}}$ are the rows corresponding to S: $S_{\mathcal{T}} = \{row(s) \mid s \in S\}$.
- The initial states of $M_{\mathcal{T}}$ are defined as follows: $\mathit{Init}_{\mathcal{T}} = \{row(s) \mid s \in 2^{AP}\}$.
- The atomic propositions and alphabet of $M_{\mathcal{T}}$ are the same as that of U.

– The propositional labeling of $M_{\mathcal{T}}$ is defined as: $\mathcal{L}_{\mathcal{T}}(row(s)) = \mathcal{P}(s)$.
– The set of final states is defined as: $F_{\mathcal{T}} = \{row(s) \mid T(s) = 1\}$.
– The transition relation $\delta_{\mathcal{T}}$ is defined as follows:

$$\forall s, s' \in S \centerdot \forall a \in \widehat{\Sigma} \centerdot (s, a, s') \in \delta_{\mathcal{T}} \iff \neg \mathit{Diff}(s \bullet a, s')$$

Note that any DLA derived from a close observation table by the above procedure obeys the conditions specified in Definition 7.

2.5 $SE - L^*$

Let us denote the empty string by λ. Then $SE - L^*$ starts with a table $\mathcal{T} = (S, E, T)$ such that $S = 2^{AP}$ and $E = \{\lambda\}$ and in each iteration proceeds as follows.

1. It first updates $\mathcal{T}$ using membership queries till it is closed.
2. Next $SE - L^*$ builds a candidate DDLA $M_{\mathcal{T}}$ from the table and makes a candidate query with $M_{\mathcal{T}}$.
3. If the MAT returns TRUE to the candidate query, $SE - L^*$ returns $M_{\mathcal{T}}$ and stops.
4. Otherwise, $SE - L^*$ updates E by adding to it a single new element and repeats from step 1.

```
Algorithm SE − L*
1: $S := 2^{AP}$; $E := \{\lambda\}$;
2: forever do
3:    CloseTable();
4:    $M$ := candidate DLA constructed from $\mathcal{T}$;
5:    if (IsCandidate($M$)) return $M$;
6:    let $CE$ be the counterexample returned by IsCandidate;
7:    $E := E \cup \{$NewExperiment($CE$)$\}$;
```

Fig. 1. Pseudo-code for algorithm $SE - L^*$.

The new element (let us call it e) added to E in step 4 is derived from the counterexample to the candidate query made in step 2. It is such that when e is added to E and the table made closed again (in the next iteration of $SE - L^*$), the size of S is guaranteed to increase by at least one. Since the size of S cannot increase indefinitely (by Theorem 3) there can only be a finite number of failed candidate queries. The process of constructing e will be presented later. The pseudo-code for $SE - L^*$ is presented in Figure 1.

At line 3, the function **CloseTable** create a closed table $\mathcal{T}$ using membership queries. At line 4, a candidate M is derived from $\mathcal{T}$ as described in Section 2.4. At line 5 the function **IsCandidate** performs a candidate query using M. If the candidate query passes then M is returned as the final result. Otherwise, at line 7, a new experiment is constructed from the counterexample CE by function **NewExperiment**. This new experiment is added to E and the entire loop is repeated. We now describe functions **CloseTable** and **NewExperiment** is more detail.

Algorithm CloseTable. The pseudo-code for algorithm **CloseTable** is presented in Figure 2. It iteratively identifies elements in $S \bullet \widehat{\Sigma}$ which cause the table to be not closed and adds these elements to the set S. In each iteration the size of S increases by one. Also, the size of S cannot increase indefinitely because of Theorem 3. Hence **CloseTable** must always terminate with a closed table.

```
Algorithm CloseTable
1: forever do
2:   if ($\forall t \in S \bullet \widehat{\Sigma} \centerdot \exists s \in S \centerdot \neg Diff(s,t)$) return;
3:     //table is already closed
4:   find $t \in S \bullet \widehat{\Sigma}$ such that $\forall s \in S \centerdot Diff(s,t)$;
5:   $S := S \cup \{t\}$;    //note that this maintains CONSISTENCY
```

Fig. 2. Pseudo-code for algorithm **CloseTable**.

Algorithm NewExperiment. We now describe the algorithm **NewExperiment**. Recall that **NewExperiment** returns a new experiment e based on a counterexample CE to a candidate query. The experiment e should be such that when it is added to E and the table closed, the size of S will increase by at least one. Let M be the candidate for which CE is the counterexample. In other words $CE \in L(M) \triangle L(U)$. Let CE be of the form $\langle P_1, \alpha_1, \ldots, \alpha_{n-1}, P_n \rangle$.

For $1 \leq i \leq n$, let p_i be the prefix of CE up to P_i and let r_i be the suffix of CE after P_i. For example, let $n = 3$. Then $p_1 = \langle P_1 \rangle$ and $r_1 = \langle \alpha_1, P_2, \alpha_2, P_3 \rangle$. Similary $p_2 = \langle P_1, \alpha_1, P_2 \rangle$ and $r_2 = \langle \alpha_2, P_3 \rangle$.

For $1 \leq i \leq n$, let $row(s_i)$ be the state reached by simulating p_i on M. In other words, let $row(s_i)$ be the unique element (cf. Theorem 1) of $Reach(M, p_i)$. Recall that every state of M is a row corresponding to some element of S. Then let b_i be 1 if $s_i \bullet r_i \in L(U)$ and 0 otherwise.

It easy to see that we can evaluate b_i using a membership query on $s_i \bullet r_i$ for any $i \in \{1, \ldots, n\}$. Since $CE \in L(M) \triangle L(U)$, we know that $b_1 \neq b_n$. Hence we can find a $k \in \{1, \ldots, n\}$ such that $b_k \neq b_{k+1}$ by doing a binary search along CE. Then the experiment e to be returned by **NewExperiment** is simply r_{k+1}. Clearly **NewExperiment** always terminates.

Correctness of NewExperiment. Let CE be of the form $\langle P_1, \alpha_1, \ldots, \alpha_{n-1}, P_n \rangle$ and suppose **NewExperiment** returns r_{k+1}. Let p_k and p_{k+1} be the prefixes of CE up to P_k and P_{k+1} respectively. Let $row(s_k)$ and $row(s_{k+1})$ be the states of M reached by simulating p_k and p_{k+1} respectively.

Now consider the word $w = s_k \bullet \alpha_k \bullet \mathcal{P}(s_{k+1})$. Since s_k belongs to S, w belongs to $S \bullet \widehat{\Sigma}$. Also since there is a transition in M of the form $row(s_k) \xrightarrow{\alpha_k} row(s_{k+1})$ we know that $\neg \mathit{Diff}(w, s_{k+1})$.

Since s_{k+1} differs from every other element of S so does w. We now show that after adding r_{k+1} to E, w must also differ s_{k+1}. Hence to make the table closed, w must be added to S. This will lead to the size of S increasing by at least one.

It is easy to see that w must differ from s_{k+1} on the experiment r_{k+1}. This is because the table entry $T[w \bullet r_{k+1}]$ must be the same as b_k. On the other hand the table entry $T[s_{k+1} \bullet r_{k+1}]$ must be the same as b_{k+1}. Since we already know that $b_k \neq b_{k+1}$, it must be the case that $T[w \bullet r_{k+1}] \neq T[s_{k+1} \bullet r_{k+1}]$. This completes our proof.

□

Theorem 4. *Algorithm $SE - L^*$ always terminates with the correct result.*

Proof. That $SE - L^*$ terminates with the correct result is obvious since it stops only after a candidate query has passed. To prove that it always terminates it suffices to note that every step in the top-level loop (line 2) terminates. Also as we showed earlier there can only be a finite number of failed candidate queries and hence a finite number of iterations of the top-level loop.

□

References

1. D. Angluin. Learning regular sets from queries and counterexamples. In *Information and Computation*, volume 75(2), pages 87–106, November 1987.
2. R. L. Rivest and R. E. Schapire. Inference of finite automata using homing sequences. In *Information and Computation*, volume 103(2), pages 299–347, April 1993.
3. A. W. Roscoe. *The Theory and Practice of Concurrency.* Prentice-Hall International, London, 1997.