Query Learning of Regular Languages over Large Ordered Alphabets

Kaizaburo Chubachi, Diptarama, Ryo Yoshinaka, Ayumi Shinohara
Graduate School of Information Sciences, Tohoku University, Sendai, Japan
Email: {kaizaburo_chubachi@shino., diptarama@shino., ry@, ayumi@}ecei.tohoku.ac.jp

***Abstract*—We propose an efficient query learning algorithm for regular languages over large ordered alphabets under the minimally adequate teacher (MAT) model. When neighbor symbols tend to cause the same transition in the underlying DFA, our algorithm learns the target language with much fewer queries than classical MAT learning algorithms require.**

1. Introduction

The learning of regular languages has been studied for a long time. In particular, since Angluin [1] has proposed the query learning model with a *minimally adequate teacher (MAT)*, different learning algorithms, including Angluin's, working under this model have been designed [2], [3], [4], [5]. A MAT replies to two types of queries. The first type is a *membership query (MQ)*, getting the learner to know whether a string selected by the learner is a member of the target language or not. The second type is an *equivalence query (EQ)*, getting the learner to know whether the hypothesis language accepted by an automaton presented by the learner is equivalent to the target language or not. If not equal, the learner gets a counterexample from the symmetric difference of the hypothesis and target languages.

On the other hand, more elaborated types of languages that are accepted by extensions of finite state automata are considered to be learning targets in recent years. *Event recording automata* [6] accept *timed words*, which are sequences of pairs of a symbol and a non-negative number. Learning those languages as efficiently as regular languages is a challenging task. Grinchtein et al. [7] have proposed a MAT learning algorithm for (subclasses of) *deterministic event-recording automata (DERAs)*. The required number of queries cannot be bounded polynomially unless we target only a kind of normal form of DERAs which can be exponentially bigger than equivalent compact DERAs. Lin et al. [8] have argued that assuming a much stronger teacher, learning DERAs with a polynomial number of queries is possible. Mens and Maler [9] have considered one of the simplest cases of DERAs where we have just a single (thus ignorable) untimed symbol. Their teacher is also stronger than a standard MAT, whose counterexamples to EQs are always the smallest with respect to the length-lexicographic order. Under this strong setting, their learner identifies learning targets with K EQs and $\mathrm{O}(Kn)$ MQs, where n is the number of the states and K is the total number of transition edges in a target DERA. Note that DERAs with a single untimed symbol are essentially equivalent to deterministic finite state automata (DFAs), except that each transition edge in such DERAs is labeled by an interval, specified by two numbers, whereas we need duplicated edges for all numbers in the interval in the corresponding DFAs.

Following Mens and Maler, this paper targets regular languages over integers in a certain interval $[1, \sigma]$, but our proposed algorithm works under the standard MAT model, where the teacher's counterexamples are arbitrary. Of course, classical algorithms like Angluin's [1] and Rivest and Schapire's [2] can be used for this task. However, those algorithms determine the destination of a transition from a state caused by each letter one by one, so the number of MQs they make grows linearly in the size of the alphabet σ. Rivest and Schapire's requires n EQs and $\mathrm{O}(\sigma n^2 + n \log m)$ MQs, where m is the length of the longest counterexample given to the learner.

Our proposed algorithm does not determine transition destinations in such a way. We presume that neighbor integers i and $i+1$ in $[1, \sigma]$ tend to cause the same transition, unless a witness falsifying this assumption is found. Based on this, we refrain from determining the transition by i using MQs for every integer i in $[1, \sigma]$. Consequently, our learner requires at most K EQs and $\mathrm{O}(K(n + \log m + \log \sigma))$ MQs. We have $n \leq K \leq \sigma n$. The extreme case $K = \sigma n$ happens when every transition edge of a target DERA is labeled by an interval of size 1. However, when K is much smaller than σn, our algorithm raises much fewer queries than existing MAT algorithms. The number of MQs that our learner makes appears bigger than that of Mens and Maler's, but it is a reasonable compensation for lifting the strong assumption on a teacher's answer.

2. Preliminaries

The interval between two integers a and b with $a \leq b$ is denoted by $[a, b] = \{\, c \mid a \leq c \leq b \,\}$. For an alphabet Σ, Σ^* denotes the set of strings over Σ. The empty string is ε. A *language* is any subset of Σ^*. The concatenation of two strings u and v is denoted by $u \cdot v$, which is straightforwardly generalized for sets S and T as $S \cdot T = \{\, s \cdot t \mid s \in S,\, t \in T \,\}$. A string set S is *prefix-closed* if every prefix of every member of S is also a member of S. We denote the membership of a string w to a language L by a function $\mathrm{MEM}_L : \Sigma^* \to \{+, -\}$, i.e., $\mathrm{MEM}_L(w) = +$ if $w \in L$ and $\mathrm{MEM}_L(w) = -$ otherwise.

2.1. DFA on an Integer Alphabet

For the technical simplicity of the formalism, we will work on DFAs rather than DERAs. However, this does not affect the discussions on the query complexity of our learner.

Definition 1. A *deterministic finite automaton* (DFA) is a quintuple $M = (Q, \Sigma, \delta, q_0, F)$, where Q is a non-empty finite set of *states*, Σ is a finite alphabet, $\delta : Q \times \Sigma \to Q$ is the *transition function*, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of *accepting states*.

As usual, δ can be naturally extended to $\delta : Q \times \Sigma^* \to Q$, by $\delta(q, \varepsilon) = q$ and $\delta(q, u \cdot a) = \delta(\delta(q, u), a)$ for all $q \in Q$, $a \in \Sigma$ and $u \in \Sigma^*$. The *language accepted by* M, denoted by $L(M)$, is the set $\{\, w \in \Sigma^* \mid \delta(q_0, w) \in F \,\}$.

We are interested in automata on an alphabet consisting of integers of a big interval, namely $\Sigma = [1, \sigma]$ for some big integer σ, where the transitions from a state by close numbers are likely the same. In other words, each state has much fewer *transition boundaries* than σ.

Definition 2. We call a symbol $a \in \Sigma - \{1\}$ a *transition boundary of a state* $q \in Q$ if $\delta(q, a) \neq \delta(q, a-1)$.

Through the well-known correspondence between the quotients of a regular language by prefixes and the states of its minimum DFA representation, a language theoretic characterization of boundaries is given as follows.

Definition 3. For a regular language $L \subseteq \Sigma^*$ and a string $s \in \Sigma^*$, we say that $a \in \Sigma - \{1\}$ is an *extension boundary of s with respect to L* if $\mathrm{MEM}_L(s \cdot (a-1) \cdot y) \neq \mathrm{MEM}_L(s \cdot a \cdot y)$ for some $y \in \Sigma^*$.

Lemma 1. A symbol $a \in \Sigma$ is an extension boundary of a string $s \in \Sigma^*$ with respect to L if and only if a is a transition boundary of the state $\delta(q_0, s)$ of the minimum DFA accepting L.

2.2. MAT Learning

The learning model this paper works on is query learning with a minimally adequate teacher (MAT), proposed by Angluin [1]. The learner identifies a language through two kinds of queries on the target: membership queries (MQs) and equivalence queries (EQs). Suppose that the target regular language is $L_* \subseteq \Sigma^*$. In an MQ, a string $w \in \Sigma^*$ is selected by the learner, for which the teacher replies $\mathrm{MEM}_{L_*}(w)$. In an EQ, a DFA M is selected as a hypothesis by the learner, for which the teacher replies *yes* if $L_* = L(M)$, and replies a *counterexample*, that is a string in the symmetric difference $(L_* - L(M)) \cup (L(M) - L_*)$ if $L_* \neq L(M)$. Note that any of the possible counterexamples may be chosen by the teacher.

3. Learning Regular Languages on Integer Alphabets

In the sequel, we fix an alphabet $\Sigma = [1, \sigma]$ and a regular language $L_* \subseteq \Sigma^*$ as the learning target. We simply write MEM for MEM_{L_*}.

Our learning algorithm largely follows Rivest and Schapire [2]. They have improved Angluin's learning algorithm [1] in the number of MQs required by using a kind of binary search for an evidence for that two strings should lead to distinct states in the target DFA. Our proposed learner employs their idea also for finding transition boundaries.

3.1. Observation Table

Following the tradition of the query learning of regular languages, we use an observation table organizing the answers towards MQs to construct a hypothesis DFA.

Definition 4. An *observation table* (OT) is a quadruple $\mathcal{T} = (U, S, E, T)$, where U and S are non-empty finite prefix-closed sets of strings with $S \cup (S \cdot \{1\}) \subseteq U \subseteq S \cup (S \cdot \Sigma)$, E is a finite set of strings containing ε, and T is a sub-function of MEM whose domain is restricted to $(U \cdot E)$.

For $u \in U$, $\mathrm{row}(u)$ is the finite function from E to $\{+, -\}$ defined by $\mathrm{row}(u)(e) = T(u \cdot e)$ for each $e \in E$.

An OT can be interpreted as a two-dimensional table whose rows and columns are indexed by elements of U and E, respectively, where the entry indexed by $(u, e) \in U \times E$ is $T(u \cdot e)$. To fulfill an OT, the learner asks MQs. See Fig. 1(a) for an example. A DFA is constructed from an OT which is *closed* and *reduced*.

Definition 5 ([2])**.** An OT $\mathcal{T} = (U, S, E, T)$ is said to be *closed* if for any $u \in U$, the exists $s \in S$ satisfying $\mathrm{row}(u) = \mathrm{row}(s)$, and *reduced* if for any $s_1, s_2 \in S$, $s_1 \neq s_2$ implies that $\mathrm{row}(s_1) \neq \mathrm{row}(s_2)$.

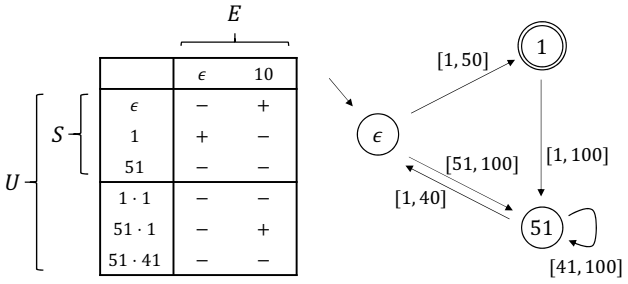
Similarly to Rivest and Schapire [2], we use S as the state set of the learner's hypothesis, while the rows indexed by elements of $S \cdot \Sigma$ are used to determine the transitions among those states. Their OT has rows indexed by *all* the elements of $S \cdot \Sigma$. The transition from a state $s \in S$ by a symbol $a \in \Sigma$ is defined according to the value of $\mathrm{row}(s \cdot a)$. This is why the number of MQs required by their algorithms grows linearly with respect to $|\Sigma|$ in order to fulfill the OT. A key idea to reduce the number of MQs in this paper is to make MQs on $s \cdot a$ for a limited number of symbols $a \in \Sigma$ for each $s \in S$. If a row indexed by $s \cdot a$ is missing in our OT for $s \in S$ and $a \in \Sigma$, we tentatively assume that $\mathrm{row}(s \cdot a)$ would equal to $\mathrm{row}(s \cdot b)$ for $b = \mu(s, a) = \max\{\, c \leq a \mid s \cdot c \in U \,\}$, until a witness falsifying the assumption is found. Since $S \cdot \{1\} \subseteq U$ by definition, $\mu(s, a)$ is always well-defined.

Now we are ready to define a DFA derived from an OT.

Definition 6. Let $\mathcal{T} = (U, S, E, T)$ be a closed and reduced OT. The *DFA derived from* $\mathcal{T}$ is $M_{\mathcal{T}} = (S, \Sigma, \delta, \varepsilon, F)$, where $F = \{\, s \in S \mid T(s) = + \,\}$ and $\delta(s, a) = s'$ such that $\mathrm{row}(s') = \mathrm{row}(s \cdot \mu(s, a))$ for each $s \in S$ and $a \in \Sigma$.

Example 1. Figure 1 shows an OT and the DFA derived from it, where $\Sigma = [1, 100]$.

If s has an extension boundary in $[\mu(s, a) + 1, a]$ with respect to L_*, then defining δ so that $\delta(s, a) = \delta(s, \mu(s, a))$ is not reasonable. We must find all the extension boundaries of every $s \in S$.



(a) Observation table (b) Derived DFA

Figure 1: An OT and the DFA derived from it.

Definition 7. We say that an extension boundary b of s is *identified* by an OT (U, S, E, T) if both $s \in S$ and $s \cdot b \in U$ hold.

Now we show when the DFA derived from an OT will be correct.

Lemma 2. Let n be the number of states in the the minimum DFA M_* accepting the target language L_*. If an OT $\mathcal{T} = (U, S, E, T)$ is closed and reduced, $|S| = n$, and all the extension boundaries of s are identified by $\mathcal{T}$ for all $s \in S$, then $M_{\mathcal{T}}$ is isomorphic to M_*.

Proof. Let $M_{\mathcal{T}} = (S, \Sigma, \delta_{\mathcal{T}}, q_0, F_{\mathcal{T}})$ and $M_* = (Q_*, \Sigma, \delta_*, q_{*0}, F_*)$. Define a function $\phi : S \to Q_*$ to be $\phi(s) = \delta_*(q_{*0}, s)$ for each $s \in S$.

Since $\mathcal{T}$ is reduced and $|S| = |Q_*|$, for any $q \in Q_*$,

$$\delta_*(q, y) \in F_* \iff \mathrm{row}(s)(y) = +$$

for all $y \in E$ if and only if $q = \phi(s)$. Therefore, $\mathrm{row}(s) = \mathrm{row}(u)$ implies $\delta_*(q_{0*}, u) = \phi(s)$ for all $u \in U$.

At first we show ϕ is bijective. Suppose $\phi(s_1) = \phi(s_2)$ for some $s_1, s_2 \in S$, that is, $\delta_*(q_{*0}, s_1) = \delta_*(q_{*0}, s_2)$. Then we have $\mathrm{MEM}(s_1 \cdot y) = \mathrm{MEM}(s_2 \cdot y)$ for all $y \in \Sigma^*$, which implies $\mathrm{row}(s_1) = \mathrm{row}(s_2)$. Since S is reduced, it holds that $s_1 = s_2$. Therefore, ϕ is injective. By $|S| = |Q_*|$, the function ϕ is bijective.

For the initial state, we have

$$\phi(q_0) = \phi(\varepsilon) = \delta_*(q_{*0}, \varepsilon) = q_{*0}.$$

Concerning accepting states, for any $s \in S$,

$$\begin{aligned} s \in F_{\mathcal{T}} &\iff T(s) = \mathrm{MEM}(s) = + \\ &\iff \phi(s) = \delta_*(q_{*0}, s) \in F_*. \end{aligned}$$

It remains to show $\phi(\delta_{\mathcal{T}}(s, a)) = \delta_*(\phi(s), a)$. We note that for any $s' \in S$ and $t \in Q_*$, if

$$\mathrm{row}(s')(y) = + \iff \delta_*(t, y) \in F_* \text{ for all } y \in E,$$

then $t = \phi(s')$, since ϕ is bijective and $\mathcal{T}$ is reduced. Let $b = \mu(s, a)$ and $s' = \delta_{\mathcal{T}}(s, a) = \delta_{\mathcal{T}}(s, b)$. For all $y \in E$,

$$\begin{aligned} \delta_*(q_{0*}, s \cdot b \cdot y) = \delta_*(\delta_*(\phi(s), b), y) \in F_* \\ \iff \mathrm{row}(s \cdot b)(y) = \mathrm{row}(s')(y) = +, \end{aligned}$$

Algorithm 1 Proposed Algorithm

Input: target language L_* over an alphabet Σ
Output: minimum DFA that accepts L_*

let $U = \{\varepsilon, 1\}, S = \{\varepsilon\}, E = \{\varepsilon\}$;
construct $\mathcal{T} = (U, S, E, T)$ using MQs;
loop
  while $\exists u \in U, \forall s \in S, \mathrm{row}(u) \neq \mathrm{row}(s)$ **do**
    / $\mathcal{T}$ is not closed */*
    add u to S and $u \cdot 1$ to U;
    update $\mathcal{T}$ using MQs;
  end while
  ask an EQ on $M_{\mathcal{T}}$;
  if the teacher replies a counterexample w **then**
    for $0 \leq i \leq |w|$ **do**
      let x_i and y_i be such that $w = x_i \cdot y_i$ and $|x_i| = i$;
      let $s_i = \delta(q_0, x_i)$;
    end for
    find $i \in [0, |w| - 1]$ such that $\mathrm{MEM}(s_i \cdot y_i) \neq \mathrm{MEM}(s_{i+1} \cdot y_{i+1})$ by binary search;
    let $a \in \Sigma$ be such that $y_i = a \cdot y_{i+1}$;
    let $b = \mu(s_i, a)$;
    if $\mathrm{MEM}(s_i \cdot b \cdot y_{i+1}) = \mathrm{MEM}(s_i \cdot a \cdot y_{i+1})$ **then**
      add y_{i+1} to E;
    else
      find $c \in [b + 1, a]$ such that $\mathrm{MEM}(s_i \cdot (c-1) \cdot y_{i+1}) \neq \mathrm{MEM}(s_i \cdot c \cdot y_{i+1})$ by binary search;
      add $s_i \cdot c$ to U;
    end if
    update $\mathcal{T}$ using MQs;
  else
    return $M_{\mathcal{T}}$ and terminate;
  end if
end loop

which implies that $\delta_*(\phi(s), b) = \phi(s')$. Since $\mathcal{T}$ has identified every boundary of s, the fact $b = \mu(s, a)$ means that s has no extension boundary in $[b + 1, a]$. Thus $\delta_*(\phi(s), b) = \delta_*(\phi(s), a)$. Therefore,

$$\begin{aligned} \phi(\delta_{\mathcal{T}}(s, a)) &= \phi(\delta_{\mathcal{T}}(s, b)) \\ &= \phi(s') = \delta_*(\phi(s), b) = \delta_*(\phi(s), a). \end{aligned}$$

This has shown that $M_{\mathcal{T}}$ is isomorphic to M_*. □

3.2. Proposed Algorithm

By Lemma 2, our goal is to collect $n = |Q_*|$ strings in S so that $\mathrm{row}(s_1) \neq \mathrm{row}(s_2)$ for any distinct $s_1, s_2 \in S$ and to identify every extension boundary of each $s \in S$. Our learner is shown in Algorithm 1.

If the OT $\mathcal{T}$ is not closed, the learner finds $u \in U$ such that $\mathrm{row}(u)$ is different from $\mathrm{row}(s)$ for all $s \in S$. Then u is added to S and becomes a new state. Transition edges from u are directed where the newly added row $u \cdot 1$ indicates.

It is guaranteed that $\mathcal{T}$ is always reduced in this algorithm. So the learner can construct a DFA $M_{\mathcal{T}}$ if $\mathcal{T}$ is closed.

The learner asks an EQ on $M_{\mathcal{T}}$. If the teacher replies *yes*, the learner outputs $M_{\mathcal{T}}$ and terminates.

We will consider the case that the teacher replies a counterexample w. Let x_i, y_i, and s_i be defined as in the algorithm. Following Rivest and Schapire [2], at first the learner finds $i \in [0, |w| - 1]$ such that $\mathrm{MEM}(s_i \cdot y_i) \neq \mathrm{MEM}(s_{i+1} \cdot y_{i+1})$. Since

$$\begin{aligned}&\mathrm{MEM}_{L_*}(s_0 \cdot y_0) = \mathrm{MEM}_{L_*}(w)\\ &\neq \mathrm{MEM}_{L(M_{\mathcal{T}})}(w) = \mathrm{MEM}_{L_*}(s_{|w|}) = \mathrm{MEM}_{L_*}(s_{|w|} \cdot y_{|w|}),\end{aligned}$$

certainly such an i exists and can be found by $\lceil \log |w| \rceil$ MQs using a kind of binary search.

For $a \in \Sigma$ with $y_i = a \cdot y_{i+1}$ and $b = \mu(s_i, a)$, either

$$\mathrm{MEM}(s_i \cdot a \cdot y_{i+1}) = \mathrm{MEM}(s_i \cdot b \cdot y_{i+1}) \neq \mathrm{MEM}(s_{i+1} \cdot y_{i+1})$$

or

$$\mathrm{MEM}(s_i \cdot a \cdot y_{i+1}) \neq \mathrm{MEM}(s_i \cdot b \cdot y_{i+1}) = \mathrm{MEM}(s_{i+1} \cdot y_{i+1})$$

holds. The former case can be treated as Rivest and Schapire's algorithm does. The string y_{i+1} witnesses that we need a new state to which we can reach by reading $s_i \cdot b$. The latter case is unique to our setting. In this case, there must be an extension boundary c of s_i in $[b + 1, a]$, which has not been identified by our OT $\mathcal{T}$ so far. We will find such a boundary c by a binary search with $\lceil \log (a - b) \rceil$ MQs, and add $s_i \cdot c$ to U so that $\mu(s_i, a)$ will not be b anymore.

3.3. Query Complexity

Let n be the number of states in the minimum DFA accepting the target language L_*, k the total number of transition boundaries of all states of M_*, m the maximum length of counterexamples replied by the teacher.

Suppose that a counterexample is given against an EQ. When the **if** condition of Line 18 is satisfied, a string y_{i+1} is added to E, which makes $\mathrm{row}(s_i \cdot b)$ with $s_i \cdot b \in U$ different from $\mathrm{row}(s)$ for any $s \in S$. By closing the table, the cardinality of S will be increased, which can happen at most $n - 1$ times. Otherwise, a string $s_i \cdot c$ is added to U where c is a newly identified extension boundary of s_i, which can happen at most k times. Therefore, the algorithm does not make more than $n + k$ EQs by Lemma 2.

We use MQs for fulfilling the table in Lines 2, 7 and 24, finding i on Line 15, checking the condition on Line 18, and finding c on Line 21. Since $|U| \leq n + k + 1$ and $|E| \leq n$, $(n + k + 1)n$ MQs will suffice to fulfill the table. Since at most $n + k$ counterexamples are given, Line 15 is executed at most $n + k$ times, each of which asks at most $\lceil \log m \rceil$ MQs. Line 18 is executed at most $n + k$ times. Line 21 is executed at most k times, each of which asks at most $\lceil \log \sigma \rceil$ MQs.

We establish the following theorem.

Theorem 1. Algorithm 1 terminates after at most $n+k$ EQs and $\mathrm{O}((n + k)(n + \log m) + k \log \sigma)$ MQs are raised.

We note that if we label transition edges with an interval rather than a single integer, the total number of edges in an automaton will be $K = k + n$. By this parameter, the numbers of required EQs and MQs are K and $\mathrm{O}(K(n + \log m + \log \sigma))$, respectively.

4. Discussion

We have proposed a MAT learning algorithm for regular languages over integers in $[1, \sigma]$, which uses a compact observation table and finds transition boundaries using binary search. It requires fewer queries than classical algorithms if there are few boundaries compared to the alphabet size. We can use the proposed algorithm for integers in $[1, \infty)$, since our algorithm does not rely on the prior knowledge on the upper bound on integers. The number of required MQs is then bounded using the maximum character σ in counterexamples.

The automata discussed in this paper can be seen as one of the simplest special cases of timed automata [10]. There are interesting subclasses of timed automata with respect to their learnability: Verwer et al. [11] have shown that *deterministic real-time automata* are efficiently learnable from positive and negative examples. Grinchtein et al. [7] have proposed a MAT learner for *deterministic event-recording automata*. We expect that our technique could be generalized so that those automata and/or other interesting variants would be able to learn efficiently. Another direction of future work is to target languages over two or higher-dimensional vectors of integers, like Mens and Maler [9] discussed.

References

[1] D. Angluin, "Learning regular sets from queries and counterexamples," *Information and Computation*, vol. 75, no. 2, pp. 87–106, 1987.

[2] R. L. Rivest and R. E. Schapire, "Inference of finite automata using homing sequences," *Information and Computation*, vol. 103, no. 2, pp. 299–347, 1993.

[3] M. J. Kearns and L. G. Valiant, "Cryptographic limitations on learning boolean formulae and finite automata," *Journal of ACM*, vol. 41, no. 1, pp. 67–95, 1994.

[4] B. Bollig, P. Habermehl, C. Kern, and M. Leucker, "Angluin-style learning of NFA," in *IJCAI*, 2009, pp. 1004–1009.

[5] J. Björklund, H. Fernau, and A. Kasprzik, "Polynomial inference of universal automata from membership and equivalence queries," *Information and Computation*, vol. 246, pp. 3–19, 2016.

[6] R. Alur, L. Fix, and T. A. Henzinger, "Event-clock automata: A determinizable class of timed automata," *Theoretical Computer Science*, vol. 211, no. 1-2, pp. 253–273, 1999.

[7] O. Grinchtein, B. Jonsson, and M. Leucker, "Learning of event-recording automata," *Theoretical Computer Science*, vol. 411, no. 47, pp. 4029–4054, 2010.

[8] S. Lin, É. André, Y. Liu, J. Sun, and J. S. Dong, "Learning assumptions for compositional verification of timed systems," *IEEE Transactions on Software Engineering*, vol. 40, no. 2, pp. 137–153, 2014.

[9] I. Mens and O. Maler, "Learning regular languages over large ordered alphabets," *Logical Methods in Computer Science*, vol. 11, no. 3, 2015.

[10] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[11] S. Verwer, M. de Weerdt, and C. Witteveen, "Efficiently identifying deterministic real-time automata from labeled data," *Machine Learning*, vol. 86, no. 3, pp. 295–333, 2012.