Active Learning of Symbolic Mealy Machines

Kengo Irie Masaki Waga Kohei Suenaga

Active automata learning is a learning framework for finite state machines using an oracle that correctly answers queries. Symbolic finite automata are one of the finite state machines in which trasntitions are labeled with predicates to operate over an infinite set. Λ^* is an active automata learning algorithm for symbolic finite automata. In this paper, we propose an active automata learning algorithm for finite state machines with an infinite set of inputs and a finite set of outputs. First, we give the formal definition of *symbolic Mealy machines* so that Mealy machines can work on an infinite set of inputs. Next, we give the extension of Λ^* for learning symbolic Mealy machines, called Λ^*_M. Then, we prove minimality of conjectured symbolic Mealy machines.

1 Introduction

The field of active automata learning has significantly impacted a wide area of software engineering. For example, learning techniques have been used to test the system [14], and extract automata from recurrent neural networks [19] [13].

L* [1] is one of the active automata learning algorithms. L* is an algorithm for learning deterministic finite automata with the minimum number of states. Variants of L* can be adapted to learn Mealy machines [12] [16].

Symbolic finite automata are finite state automata in which the alphabet is given by a Boolean algebra that may have an infinite domain, and transitions are labeled with predicates over such algebra.

Λ^* [7] is an algorithm for learning symbolic finite automata. Λ^* is an extenstion of L* and has an additional step for generalizing the concrete transition labels to predicates.

In this paper, we propose the algorithm to learn finite state machines with an infinite set of inputs and a finite set of outputs.

First, we give the formal definition of symbolic Mealy machines so that Mealy machines can operate over an infinite set. We then define the minimality of symbolic Mealy machines and prove that minimal symbolic Mealy machines have the minimum number of states.

Next, we give an extension of Λ^* for learning symbolic Mealy machines, called Λ^*_M. Finally, We prove that a symbolic Mealy machine learned by Λ^*_M has the minimum number of states.

In summary, our contributions are:

- the formal definition of symbolic Mealy machines
- the learning algorithm for symbolic Mealy machines
- a proof of minimality of learned machines

記号的ミーリ機械の能動的オートマトン学習

入江 堅吾, 和賀 正樹, 末永 幸平, 京都大学 大学院情報学研究科, Graduate School of Informatics, Kyoto University.

1.1 Related work

There are various active automata learning algorithms. L* [1] is the first learning algorithm of DFA using membership queries and equivalence queries. Rivest & Shapire [15] is a variant of L*. L* and Rivest & Shapire are observation table based algorithms. While TTT [9] is a tree-like data structure based algorithm for DFA.

Active automata learning is also adapted to Mealy machines. Niese [12] is the first adaptation of L* for Mealy machines. Shabaz & Groz [16] improves an algorithm for Mealy machines to provide a new method for processing counterexamples.

Symbolic finite automata is a extension of classical automata and transitions are labeled by predicates so that it can operate over infinite input alphabets. Determinization and completion of symbolic finite automata are studied in [6]. Minimization of symbolic finite automata is also studied in [6].

There are multiple learning algorithms for symbolic finite automata. Mens et al. [11] proposed an adaptation of L* for symbolic finite automata for totally ordered alphabet such as $\mathbb{N}$. Argyros et al. [3] proposed Shabaz & Groz [16] based algorithm for symbolic finite automata using guardgen algorithm which generalizes concrete symbols into predicates. Drews et al. [7] presented $\Lambda*$ which uses a partitioning function to get predicates for transition. They also defined the learnability of an underlying Boolean algebra and classified Boolean algebras with respect to complexity of learning. Argyros et al. [2] presented MAT*. MAT* is based on TTT [9] and takes as input a learning algorithm for predicates. And it uses a learning algorithm to infer the predicates appearing on the transitions in the target automaton. Fisman et al. [8] studied the learnability of symbolic finite automata under the paradigm of identification in the limit using polynomial time and data. They provide a necessary condition for the identification of symbolic finite automata in the limit using polynomial time and data and a sufficient condition for efficient learnability of symbolic finite automata.

Symbolic finite transducers [18] are extensions of classical transducers by allowing transitions to carry predicates and functions. There are also several learning algorithms [4] [3] [10].

In addition, there are learning algorithms for variants of symbolic finite automata such as residual symbolic automata [5] and symbolic weighted finite automata [17].

2 Preliminaries

For a set Σ, the set of words over Σ is Σ^*. The empty word of length 0 is denoted by ϵ. We also denote $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$.

2.1 Symbolic Finite Automaton

In symbolic automata, transitions carry predicates over a decidable Boolean algebra.

Definition 1 (Boolean Algebra)**.** *An effective Boolean algebra $\mathcal{A}$ is a tuple $\mathcal{A} = (\mathfrak{D}, \Psi, [\![\_]\!], \bot, \top, \vee, \wedge, \neg)$ where $\mathfrak{D}$ is a recursively enumerable set of* domain elements*; Ψ is a recursively enumerable set of predicates closed under the Boolean connectives, with $\bot, \top \in \Psi$; $[\![\_]\!]\colon \Psi \to 2^{\mathfrak{D}}$ is a* denotation function *such that (i) $[\![\bot]\!] = \varnothing$, (ii) $[\![\top]\!] = \mathfrak{D}$, (iii) for all $\varphi, \psi \in \Phi$, $[\![\varphi \vee \psi]\!] = [\![\varphi]\!] \cup [\![\psi]\!]$, $[\![\varphi \wedge \psi]\!] = [\![\varphi]\!] \cap [\![\psi]\!]$, $[\![\neg\varphi]\!] = \mathfrak{D} \setminus [\![\varphi]\!]$. For $\varphi \in \Psi$, we write $IsSat(\varphi)$ when $[\![\varphi]\!] \neq \emptyset$ and say that φ is satisfiable. $\mathcal{A}$ is decidable if IsSat is decidable.*

Definition 2 (Symbolic Finite Automaton)**.** *A symbolic finite automaton $\mathcal{M}$ is a tuple $\mathcal{M} = (\mathcal{A}, Q, q_{init}, F, \delta)$ where $\mathcal{A}$ is the Boolean algebra, Q is the non-empty finite set of states, q_{init} is the initial state, $F \subseteq Q$ is the set of final states, $\delta \subseteq Q \times \Psi_A \times Q$ is the transition relation.*

We denote a domain of elements of $\mathcal{A}$ by $\mathfrak{D}_{\mathcal{A}}$. Characters are elements of $\mathfrak{D}_{\mathcal{A}}$, and words are finite

sequences of characters, namely elements of $\mathfrak{D}^*_{\mathcal{A}}$. A move $\rho = (q_1, \varphi, q_2) \in \delta$, also denoted by $q_1 \xrightarrow{\varphi} q_2$, is a transition from the source state q_1 to the target state q_2, where φ is the guard or predicate of the move. For a character $a \in \mathfrak{D}_{\mathcal{A}}$, an a-move of $\mathcal{M}$, denoted $q_1 \xrightarrow{a} q_2$ is a move $q_1 \xrightarrow{\varphi} q_2$ such that $a \in [\![\varphi]\!]$.

A symbolic finite automaton $\mathcal{M}$ is deterministic if for all (q, φ_1, q_1), $(q, \varphi_2, q_2) \in \delta$, if $q_1 \neq q_2$ then $[\![\varphi_1 \wedge \varphi_2]\!] = \emptyset$. A symbolic finite automaton $\mathcal{M}$ is complete if for all $q \in Q$, $\bigvee_{(q,\varphi_i,q_i)\in\delta} \varphi_i = \top$. Throughout the paper, we assume all symbolic finite automata are deterministic and complete.

Given a symbolic finite automaton $\mathcal{M} = (\mathcal{A}, Q, q_{init}, F, \delta)$ and a state $q \in Q$, we say a word $w = a_1 a_2 \dots a_k$ is accepted at state q if, for $1 \leq i \leq k$, there exist moves $q_{i-1} \xrightarrow{a_i} q_i$ such that $q_0 = q$ and $q_k \in F$. We refer to the set of words accepted at q as the language accepted at q, denoted as $L_q(\mathcal{M})$; the language accepted by $\mathcal{M}$ is $L(\mathcal{M}) = L_{q_{init}}(\mathcal{M})$.

2.2 Λ^* Algorithm

Λ^* [7] learns a symbolic finite automaton using two kinds of queries. A membership query submits an input word w to an oracle and obtains whether w is accepted by an automaton. An equivalence query submits a hypothesis automaton to an oracle and obtains whether a hypothesis recognizes the language of the target; if not it gets a counterexample.

Λ^* uses the data structure called the *observation table*.

Definition 3 (Observation Table)**.** *An observation table T for a symbolic finite automaton $\mathcal{M}$ is a tuple (Σ, S, R, E, f) where Σ is a potentially infinite set called the alphabet; $S, R, E \subseteq \Sigma^*$ are finite subsets of words; $f : (S \cup R) \cdot E \to \{0, 1\}$ is a classification function such that for a word $w \cdot e \in (S \cup R) \cdot E$, $f(w \cdot e) = 1$ if $w \cdot e \in L(\mathcal{M})$, and $f(w \cdot e) = 0$ if $w \cdot e \notin L(\mathcal{M})$. Additionally, (i) S and R are disjoint, (ii) $S \cup R$ is prefix-closed and $\epsilon \in S$, (iii) for all $s \in S$, there exists a character $a \in \Sigma$ such that $s \cdot a \in S \cup R$, (iv) E is suffix-closed.*

In order for Λ^* to infer an automaton without contradiction, an observation table must satisfy the condition *cohesive*. We use the notation $row(w)$ for $w \in S \cup R$ to denote the vector indexed by $e \in E$ of $f(w \cdot e)$.

Definition 4 (closed)**.** *An observation table is closed if for each $r \in R$ there exists $s \in S$ such that $row(s) = row(r)$.*

Definition 5 (consistent)**.** *An observation table is consistent if for all $w_1, w_2 \in S \cup R$, if $a \in \Sigma$ and $w_1 \cdot a, w_2 \cdot a \in S \cup R$ and $row(w_1) = row(w_2)$, then $row(w_1 \cdot a) = row(w_2 \cdot a)$.*

Definition 6 (evidence-closed)**.** *An observation table is evidence-closed if for all $e \in E$ and $s \in S$, $s \cdot e \in S \cup R$.*

Definition 7 (reduced)**.** *An observation table is reduced if for all $s_1, s_2 \in S, row(s_1) \neq row(s_2)$.*

An observation table is cohesive if it is closed, reduced, consistent, and evidence-closed. If an observation table is cohesive, then we can construct an *evidence automaton* in which transitions are still labeled with concrete symbols.

Given a cohesive observation table we produce a evidence automaton $A = (\Sigma, Q, q_{init}, F, \delta)$ as follows: For each $s \in S$, we introduce a state $q_s \in Q$. q_{init} is assigned to q_ϵ. The final state set F contains all q_s such that $s \in S$ and $f(s) = 1$. Since the observation table is closed and reduced, there exists a function $g : S \cup R \to S$ such that $g(w) = s$ if and only if $row(w) = row(s)$. This function allows us to define the transition relation: if $w \cdot a \in S \cup R$ for $w \in \Sigma^*$ and $a \in \Sigma$, then $(q_{g(w)}, a, q_{g(w \cdot a)}) \in \delta$.

To construct a symbolic finite automaton from an evidence automaton, we use a *partitioning function* to generalize concrete characters to predicates.

Definition 8 (Partitioning Function)**.** *A par-*

titioning function for a Boolean algebra $\mathcal{A} = (\mathfrak{D}, \Psi, [\![\_]\!], \bot, \top, \vee, \wedge, \neg)$ *is a function* $P : (2^{\mathfrak{D}})^* \to \Psi^*$ *that takes as input a list* $L_{\mathfrak{D}} = l_1 \dots l_k$ *of disjoint sets of elements in* $\mathfrak{D}$*, and returns a list* $L_\Psi = \varphi_1 \dots \varphi_k$ *of predicates in* Ψ *such that*

- $\bigvee_{\varphi_i \in L_\Psi} \varphi_{i,j} = \top$
- $\varphi_i \wedge \varphi_j = \bot$ *for all* $\varphi_i, \varphi_j \in L_\Psi$ *with* $i \neq j$
- *for each* $l_i \in L_{\mathfrak{D}}$ *corresponding to* $\varphi_i \in L_\Psi$ *all* $a \in l_i$ *are such that* $a \in [\![\varphi_i]\!]$

Given an evidence automaton $A = (\Sigma, Q, q_{init}, F, \delta)$, a Boolean algebra $\mathcal{A}$ with domain Σ, and an appropriate partitioning function P, we build a symbolic finite automaton $\mathcal{M} = (\mathcal{A}, Q, q_{init}, F, \delta_M)$ using a given Boolean algebra and that exact configuration of the state space Q, the initial state q_{init}, and the final states F of A. All that remains is the construction of the transition relation δ_M.

For each $q \in Q$, we perform the following. We gather all transitions of the evidence automaton out of q into a set $\delta_q = \{(q, a, q') \in \delta\}$ and construct a list L_Σ indexed over the states $q_i \in Q$, where each set in L_Σ is $l_i = \{a \mid (q, a, q_i) \in \delta_q\}$. We apply the partitioning function to get the list of separating predicates $L_{\Psi_\mathcal{A}} = P(L_\Sigma)$ which is also indexed over $q_i \in Q$, and add (q, φ_i, q_i) to δ_M for each $\varphi_i \in L_{\Psi_\mathcal{A}}$.

We are now ready to describe the algorithm: Initially, we start with the observation table $S = \{\epsilon\}$, $R = \{a\}$ and $E = \{\epsilon\}$(a is an arbitrary character from Σ). f is initially undefined. The knowledge of the table is grown using the operations *fill*, *close*, *make-consistent* and *evidence-close*.

The operation fill asks a membership query for all $w \cdot e \in (S \cup R) \cdot E$ for which f is undefined and then adds those results to f; in this way, it ensures f is defined over the entire domain of the observation table.

The operation close checks the existence of an $r \in R$ such that for all $s \in S$, $row(r) \neq row(s)$. If such an r exists, r is moved from R to S, and $r \cdot a$ is added to R for some arbitrary $a \in \Sigma$.

The operation evidence-close ensures for all $s \in S$ and $e \in E$ that $s \cdot e \in S \cup R$ by adding to R all $s \cdot e$ that are not. It also adds to R any necessary prefixes so that $S \cup R$ is prefix-closed.

The operation make-consistent operates as follows: if there exist $w_1, w_2 \in S \cup R$ and $w_1 \cdot a, w_2 \cdot a \in S \cup R$ for some $a \in \Sigma$ such that $row(w_1) = row(w_2)$ but $row(w_1 \cdot a) \neq row(w_2 \cdot a)$, then w_1 and w_2 actually lead to different states; using the $e \in E$ such that $f(w_1 \cdot a \cdot e) \neq f(w_2 \cdot a \cdot e)$, it is clear $a \cdot e$ thus differentiates those states. Accordingly, $a \cdot e$ is added to E.

Upon receiving a counterexample $t \in \Sigma^*$ from an equivalence query sent to the oracle, all prefixes of t are added to R except those already present in S.

Algorithm 1 shows an overview of the learning algorithm: after the table is initialized, the operations make-consistent, evidence-close and close are applied until the table is cohesive. A symbolic finite automaton $\mathcal{M}_H$ is then conjectured from the table, and an equivalence query is performed: if $\mathcal{M}_H$ is equivalent to the target, then the algorithm terminates. Otherwise, a counterexample is produced and processed, and the procedure repeats.

3 Symbolic Mealy Machine

We now define symbolic Mealy machines. Intuitively, a symbolic Mealy machine is a Mealy machine over a symbolic alphabet, where edge labels are replaced by predicates. The predicates must form an effective Boolean algebra.

Definition 9 (Symbolic Mealy Machine)**.** *A symbolic Mealy machine* $\mathcal{M}$ *is a tuple* $\mathcal{M} = (\mathcal{A}, Q, q_{init}, O, \delta)$ *where* $\mathcal{A}$ *is a Boolean algebra,* Q *is the non-empty finite set of states,* q_{init} *is the initial state,* O *is the finite set of output symbols,* $\delta \subseteq Q \times \Psi_A \times Q \times O$ *is the transition-output relation.*

Here, we define various properties of a symbolic Mealy machine.

Algorithm 1 Active Learning of symbolic finite automata

Initialize $S = \{\epsilon\}$, $R = \{a\}$ and $E = \{\epsilon\}$ (a is an arbitrary character from Σ)
Construct the initial table $T = (\Sigma, S, R, E, f)$
while true **do**
  while T is not cohesive **do**
    if T is not closed **then**
      Apply close to T
    else if T is not consistent **then**
      Apply make-consistent to T
    else if T is not evidence-closed **then**
      Apply evidence-close to T
    end if
  end while
  Construct the evidence automaton $\mathcal{M}_H^e$
  Construct the symbolic finite automaton $\mathcal{M}_H$
  Send $\mathcal{M}_H$ to the equivalence oracle
  if the equivalence oracle returns a counterexample c **then**
    Add all prefixes of c to R
  else
    return $\mathcal{M}_H$
  end if
end while

Definition 10 (deterministic)**.** *A symbolic Mealy machine $\mathcal{M}$ is deterministic if, for all (q, φ_1, q_1, o_1), $(q, \varphi_2, q_2, o_2) \in \delta$, if $q_1 \neq q_2$ or $o_1 \neq o_2$ then $[\![\varphi_1 \wedge \varphi_2]\!] = \emptyset$.*

Definition 11 (complete)**.** *A symbolic Mealy machine $\mathcal{M}$ is complete if, for all $q \in Q$, $\bigvee_{(q,\varphi_i,q_i,o_i)\in\delta} \varphi_i = \top$.*

Definition 12. *$q \in Q$ is reachable if there exists $w \in \mathfrak{D}_{\mathcal{A}}^*$ such that $q = \delta_1(q_{init}, w)$.*

Definition 13 (clean)**.** *A symbolic Mealy machine $\mathcal{M}$ is clean if for all $(q, \varphi, q_1, o) \in \delta$, q is reachable from q_{init} and $[\![\varphi]\!] \neq \emptyset$.*

Definition 14 (normalized)**.** *A symbolic Mealy machine $\mathcal{M}$ is normalized if for all $p, q \in Q$ and for all $o \in O$, there is at most one φ satisfying $(p, \varphi, q, o) \in \delta$.*

Throughout the paper we assume all symbolic Mealy machines are deterministic and complete.

For the special case in which $\mathcal{M}$ is deterministic and complete, we define the transition function δ_1: $Q \times \mathfrak{D}_{\mathcal{A}} \to Q$ and output function δ_2: $Q \times \mathfrak{D}_{\mathcal{A}} \to O$ such that for all $q \in Q$ and $a \in \mathfrak{D}_{\mathcal{A}}$, $\delta_1(q, a) = q'$ and $\delta_2(q, a) = o$ where q' and o is the state and output such that $(q, \varphi, q', o) \in \delta$ and $a \in [\![\varphi]\!]$.

We denote the straightforward inductive extensions δ_1: $Q \times \mathfrak{D}_{\mathcal{A}}^* \to Q$ and δ_2: $Q \times \mathfrak{D}_{\mathcal{A}}^+ \to O$ such that $\delta_1(q, \epsilon) = q$ and $\delta_1(q, wa) = \delta_1(\delta_1(q, w), a)$ where $a \in \mathfrak{D}_{\mathcal{A}}$ and $w \in \mathfrak{D}_{\mathcal{A}}^*$; $\delta_2(q, aw) = \delta_2(\delta_1(q, a), w)$ where $a \in \mathfrak{D}_{\mathcal{A}}$ and $w \in \mathfrak{D}_{\mathcal{A}}^+$.

Additionally, we define the function $\mathcal{M}$: $Q \times \mathfrak{D}_{\mathcal{A}}^+ \to O$ such that $\mathcal{M}(q, w) = \delta_2(q, w)$.

Finally, we define minimality of symbolic Mealy machines. And we prove that a minimal symbolic Mealy machine has the minimum number of states.

Definition 15 (minimal)**.** *A symbolic Mealy machine $\mathcal{M}$ is minimal if all states of $\mathcal{M}$ are reachable and $\mathcal{M}$ is deterministic, complete, clean, normalized, and for all $p, q \in Q$ if $p \neq q$ then there exists $w \in \mathfrak{D}_{\mathcal{A}}^+$ such that $\mathcal{M}(p, w) \neq \mathcal{M}(q, w)$.*

Before proving the main theorem, we prove the following lemma.

Lemma 1. *For all $x \in \mathfrak{D}_{\mathcal{A}}^*$ and $w \in \mathfrak{D}_{\mathcal{A}}^+$, $\mathcal{M}(q_{init}, x \cdot w) = \mathcal{M}(\delta_1(q_{init}, x), w)$.*

Proof. We prove this lemma by induction on the length of x. For length 0, it is obviously true, as $\mathcal{M}(q_{init}, w) = \mathcal{M}(\delta_1(q_{init}, \epsilon), w)$

Let us assume that for all $x \in \mathfrak{D}_{\mathcal{A}}^*$ of length at most $k \geq 0$, and for all $w \in \mathfrak{D}_{\mathcal{A}}^+$ $\mathcal{M}(q_{init}, x \cdot w) = \mathcal{M}(\delta_1(q_{init}, x), w)$. Let $x' \in \mathfrak{D}_{\mathcal{A}}^*$ with length $k + 1$. Then we can find a decomposition of x' such that

$x \cdot a$ for some word of length k and $a \in \mathfrak{D}_{\mathcal{A}}$.

$$\begin{aligned}\mathcal{M}(\delta_1(q_{init}, x'), w) &= \mathcal{M}(\delta_1(q_{init}, x \cdot a), w) \\ &= \delta_2(\delta_1(q_{init}, x \cdot a), w) \\ &= \delta_2(\delta_1(\delta_1(q_{init}, x), a), w) \\ &= \delta_2(\delta_1(q_{init}, x), a \cdot w) \\ &= \mathcal{M}(\delta_1(q_{init}, x), a \cdot w) \\ &= \mathcal{M}(q_{init}, x \cdot a \cdot w) \\ &= \mathcal{M}(q_{init}, x' \cdot w)\end{aligned}$$

□

We are now ready to prove that a minimal symbolic Mealy machine has the minimum number of states.

Theorem 1. *A minimal symbolic Mealy machine $\mathcal{M}$ has the minimum number of states.*

Proof. Let $\mathcal{M} = (\mathcal{A}, Q, q_{init}, O, \delta)$ be a minimal symbolic Mealy machine. Suppose there exists another symbolic Mealy machine $\mathcal{M}' = (\mathcal{A}, Q', q'_{init}, O, \delta')$ with for all $w \in \mathfrak{D}^*_{\mathcal{A}}$ $\mathcal{M}(q_{init}, w) = \mathcal{M}'(q'_{init}, w)$ and $|Q| > |Q'|$.

Let $Q = \{q_1, q_2, \ldots, q_n\}$. Since all states of $\mathcal{M}$ are reachable, for each $i = 1, 2, \ldots, n$, we can choose a word $x_i \in \mathfrak{D}^*_{\mathcal{A}}$ such that $q_i = \delta_1(q_{init}, x_i)$. Let $q'_i = \delta'_1(q'_{init}, x_i)$, from the pigeonhole principle there exists an integer $1 \leq i < j \leq n$ such that $q'_i = q'_j$. For all $w \in \mathfrak{D}^+_{\mathcal{A}}$, we have

$$\begin{aligned}\mathcal{M}'(q'_i, w) = \mathcal{M}'(q'_j, w) &\iff \\ \mathcal{M}'(\delta'_1(q'_{init}, x_i), w) = \mathcal{M}'(\delta'_1(q'_{init}, x_j), w) &\iff \\ \mathcal{M}'(q'_{init}, x_i \cdot w) = \mathcal{M}'(q'_{init}, x_j \cdot w) &\iff \\ \mathcal{M}(q_{init}, x_i \cdot w) = \mathcal{M}(q_{init}, x_j \cdot w) &\iff \\ \mathcal{M}(\delta_1(q_{init}, x_i), w) = \mathcal{M}(\delta_1(q_{init}, x_j), w) &\iff \\ \mathcal{M}(q_i, w) = \mathcal{M}(q_j, w)\end{aligned}$$

This contradicts that $\mathcal{M}$ is minimal. Hence, $|Q| \leq |Q'|$. Therefore $\mathcal{M}$ has the minimum number of states. □

4 Learning Algorithm

Here we present our algorithm, Λ^*_M, for learning symbolic Mealy machines. The premise is that the symbolic Mealy machine to be learned called the target is hidden in a black box, so knowledge of it comes from some oracle that admits two kinds of queries: output queries that ask input from $\mathfrak{D}_{\mathcal{A}}$ and obtain output from the machine, and equivalence queries that ask whether a conjectured machine is equivalent to the target—if not, a counterexample is provided.

4.1 Observation Table

The observation table consists of rows of prefixes and columns of suffixes. Each entry keeps information on the output of the word formed by concatenating the prefix and suffix, which is obtained by output queries.

Definition 16 (Observation Table)**.** *An observation table T for a symbolic Mealy machine $\mathcal{M}$ is a tuple $(\Sigma, S, R, \Sigma_E, E, f)$ where Σ is a potentially infinite set called the alphabet; $S, R, E \subset \Sigma^*$ are finite subsets of words and $\Sigma_E \subset \Sigma$ is a finite subset of characters; $f : (S \cup R) \times (\Sigma_E \cup E) \to O$ is a finite function such that $f(w, e) = \mathcal{M}(q_{init}, w \cdot e)$. Additionally, (i) S and R are disjoint, (ii) $S \cup R$ is prefix-closed and $\epsilon \in S$, (iii) for all $s \in S$, there exists a character $a \in \Sigma$ such that $s \cdot a \in S \cup R$, (iv) $\Sigma_E \cup E$ is suffix-closed, (v) Σ_E is non-empty.*

Λ^*_M manipulates the observation table and eventually conjectures a symbolic Mealy machine. For this to happen, the table must first satisfy certain properties. We call such a table *cohesive*.

Definition 17 (closed)**.** *An observation table is closed if for each $r \in R$ there exists $s \in S$ such that $row(s) = row(r)$.*

Definition 18 (consistent)**.** *An observation table is consistent if for all $w_1, w_2 \in S \cup R$, if $a \in \Sigma$ and $w_1 \cdot a, w_2 \cdot a \in S \cup R$ and $row(w_1) = row(w_2)$, then $row(w_1 \cdot a) = row(w_2 \cdot a)$.*

Definition 19 (evidence-closed)**.** *An observation table is evidence-closed if for all $e \in (\Sigma_E \cup E)$ and $s \in S$, $s \cdot e \in S \cup R$.*

Definition 20 (reduced)**.** *An observation table is*

reduced if for all $s_1, s_2 \in S, row(s_1) \neq row(s_2)$.

Definition 21 (output-closed)**.** *An observation table is output-closed if for all* $w \in \Sigma^*$*, for all* $a \in \Sigma$ *if* $w \cdot a \in S \cup R$ *then* $a \in \Sigma_E$.

An observation table is cohesive if it is closed, reduced, consistent, evidence-closed, and *output-closed*.

Output-closed is a new condition. Output-closed guarantees that all characters appearing in an observation table are in Σ_E. In other words, Σ_E will record all characters that appear in an observation table.

If an observation table is cohesive, then it admits the construction of an *evidence Mealy machine*. We build an evidence Mealy machine as follows.

Definition 22 (Evidence Mealy Machine)**.** *Let* $(\Sigma, S, R, \Sigma_E, E, f)$ *be a cohesive observation table, then the evidence Mealy machine conjecture* $\mathcal{M}^e = (\Sigma_E, Q, q_{init}, O, \delta^e)$ *is defined, where*

- $Q = \{row(s) \mid s \in S\}$
- $q_{init} = row(\epsilon)$
- $(row(w), a, row(w \cdot a), f(w, a)) \in \delta^e$ *(*$w \in S$*,* $a \in \Sigma_E$*)*

Here, we should make sure that an evidence Mealy machine is deterministic and complete.

Lemma 2. *Given a cohesive observation Table* $T = (\Sigma, S, R, \Sigma_E, E, f)$*, if* $\mathcal{M}^e = (\Sigma_E, Q, q_{init}, O, \delta^e)$ *is the evidence mealy machine construction of* T*, then* $\mathcal{M}^e$ *is deterministic and complete.*

Proof. For all $w \in S$, for all $a \in \Sigma_E$, there exists exactly one transition $(row(w), a, row(w \cdot a), f(w, a)) \in \delta^e$. So $\mathcal{M}^e$ is obviously deterministic. Since T is evidence-closed, for all $w \in S$, for all $a \in \Sigma_E$, $w \cdot a \in S \cup R$ holds. So $row(w \cdot a)$ always exists and $\mathcal{M}^e$ is obviously complete. □

Since $\mathcal{M}^e$ is deterministic and complete, we define the transition function δ_1^e: $Q \times \Sigma_E \to Q$ and output function δ_2^e: $Q \times \Sigma_E \to O$ such that for all $q \in Q$ and $a \in \Sigma_E$, $\delta_1^e(q, a) = q'$ and $\delta_2^e(q, a) = o$ where q' and o is the state and output such that $(q, a, q', o) \in \delta^e$.

We denote the straightforward inductive extensions δ_1^e: $Q \times \Sigma_E^* \to Q$ and δ_2^e: $Q \times \Sigma_E^+ \to O$ such that $\delta_1^e(q, \epsilon) = q$ and $\delta_1^e(q, wa) = \delta_1^e(\delta_1^e(q, w), a)$ where $a \in \Sigma_E$ and $w \in \Sigma_E^*$; $\delta_2^e(q, aw) = \delta_2^e(\delta_1^e(q, a), w)$ where $a \in \Sigma_E$ and $w \in \Sigma_E^+$.

Additionally, we define the function $\mathcal{M}^e$: $Q \times \Sigma_E^+ \to O$ such that $\mathcal{M}^e(q, w) = \delta_2^e(q, w)$.

From here, we prove that an evidence mealy machine is minimal. First, we check the simple lemma that holds from the conditions of an observation table.

Lemma 3. *Given a cohesive observation Table* $T = (\Sigma, S, R, \Sigma_E, E, f)$*, if* $\mathcal{M}^e = (\Sigma_E, Q, q_{init}, O, \delta^e)$ *is the evidence mealy machine construction of* T*, for all* $s_1, s_2 \in S \cup R$*, for all* $a \in \Sigma_E$ *if* $row(s_1) = row(s_2)$ *then* $row(s_1 \cdot a) = row(s_2 \cdot a)$ *and* $f(s_1, a) = f(s_2, a)$.

Proof. For all $s_1, s_2 \in S \cup R$ such that $row(s_1) = row(s_2)$,then since the observation Table T is consistent, for all $a \in \Sigma_E$ $row(s_1 \cdot a) = row(s_2 \cdot a)$ holds. Since the table is closed, there exists $s \in S$ such that $row(s) = row(s_1 \cdot a) = row(s_2 \cdot a)$ holds. Σ_E is non-empty. So for all $s_1, s_2 \in S \cup R$ such that $row(s_1) = row(s_2)$, then for all $a \in \Sigma_E$ $f(s_1, a) = f(s_2, a)$. □

Next, we prove the following lemma which states that all words in S of the observation table are represented by valid states in $\mathcal{M}^e$.

Lemma 4. *Given a cohesive observation Table* $T = (\Sigma, S, R, \Sigma_E, E, f)$*, if* $\mathcal{M}^e = (\Sigma_E, Q, q_{init}, O, \delta^e)$ *is the evidence mealy machine construction of* T*, for all* $s \in S \cup R$ $\delta_1^e(q_{init}, s) = row(s)$.

Proof. We prove this lemma by induction on the length of s. For length 0, it is obviously true, as $\sigma(q_{init}, \epsilon) = q_{init} = row(\epsilon)$.

Let us assume that for all $s \in S \cup R$ of length

at most $k \geq 0$, $\delta_1(q_{init}, s) = row(s)$ holds. Let $t \in S \cup R$ with length $k+1$. Then we can find a decomposition of t such that $s \cdot a$ for some word of length k and $a \in \Sigma_E$. In fact, s is in $S \cup R$. Since T is closed, there exists $s_1 \in S$ such that $row(s) = row(s_1)$. We can conclude

$$\begin{aligned}
\delta_1^e(q_{init}, t) &= \delta_1^e(q_{init}, s \cdot a) \\
&= \delta_1^e(\delta_1^e(q_{init}, s), a) \\
&= \delta_1^e(row(s), a) \quad \text{induction hypothesis} \\
&= \delta_1^e(row(s_1), a) \\
&= row(s_1 \cdot a) \\
&= row(s \cdot a) \\
&= row(t)
\end{aligned}$$

□

Note that from Lemma 4 we can also conclude that all states in $\mathcal{M}^e$ are reachable from within the initial state.

Finally, we prove evidence compatibility which denotes intuitively that every entry of the observation table can be observed in the evidence Mealy machine.

Theorem 2 (Evidence compatibility)**.** *Given a cohesive observation Table $T = (\Sigma, S, R, \Sigma_E, E, f)$, if $\mathcal{M}^e = (\Sigma_E, Q, q_{init}, O, \delta^e)$ is the evidence mealy machine construction of T, then for all $s \in S \cup R$, for all $e \in \Sigma_E \cup E$, $\delta_2^e(\delta_1^e(q_{init}, s), e) = f(s, e)$.*

Proof. We prove this theorem by induction on the length of e. We begin with length 1, $a \in \Sigma_E$. Since T is closed, there exists $s_1 \in S$ such that $row(s) = row(s_1)$.

$$\begin{aligned}
\delta_2^e(\delta_1^e(q_{init}, s), a) &= \delta_2^e(row(s), a) \quad \text{Lemma 4} \\
&= \delta_2^e(row(s_1), a) \\
&= f(s_1, a) \\
&= f(s, a)
\end{aligned}$$

Suppose that $\delta_2^e(\delta_1^e(q_{init}, s), e) = f(s, e)$ holds for all $e \in \Sigma_E \cup E$ of length at $k \geq 1$, and let $e' \in \Sigma_E \cup E$ be of length $k+1$. Then we can find a decomposition of e' such that $e' = a \cdot e$ where e is of length k and $a \in \Sigma_E$. In fact e is in $\Sigma_E \cup E$ because $\Sigma_E \cup E$ is suffix-closed.

Let furthermore $s \in S \cup R$. Since T is closed, there exists a word $s_1 \in S$ such that $row(s) = row(s_1)$. Thus we can conclude

$$\begin{aligned}
\delta_2^e(\delta_1^e(q_{init}, s), e') &= \delta_2^e(\delta_1^e(q_{init}, s), a \cdot e) \\
&= \delta_2^e(row(s), a \cdot e) \quad \text{Lemma 4} \\
&= \delta_2^e(row(s_1), a \cdot e) \\
&= \delta_2^e(\delta_1^e(row(s_1), a), e) \\
&= \delta_2^e(row(s_1 \cdot a), e) \\
&= \delta_2^e(\delta_1^e(q_{init}, s_1 \cdot a), e) \quad \text{Lemma4} \\
&= f(s_1 \cdot a, e) \, \text{induction hypothesis} \\
&= \mathcal{M}(q_{init}, s_1 \cdot a \cdot e) \\
&= f(s_1, a \cdot e) \\
&= f(s_1, e') \\
&= f(s, e')
\end{aligned}$$

□

Lemma 5. $E \subseteq \Sigma_E^*$.

Proof. For all $e \in E$, from evidence-closed and $\epsilon \in S$, $e \in S \cup R$ holds. Let $e = e_1 e_2 \dots e_n$. Since $S \cup R$ is prefix-closed, all prefixes of e are in $S \cup R$. From output-closed, last characters of all prefixes of e are in Σ_E, that is $e_1, e_2, \dots, e_n$ are in Σ_E. Hence, $e \in \Sigma_E^*$. □

From Theorem 2 and Lemma 5, we can prove that an evidence Mealy machine is minimal.

Theorem 3 (Minimality of Evidence Mealy Machine)**.** *Given a cohesive observation Table $T = (\Sigma, S, R, \Sigma_E, E, f)$, if $\mathcal{M}^e = (\Sigma_E, Q, q_{init}, O, \delta^e)$ is a Mealy machine constructed from T, then $\mathcal{M}^e$ is minimal.*

Proof. Assume $\mathcal{M}^e$ is compatible with its observation table. For all $s \in S$ and $e \in \Sigma_E \cup E$,

$$\begin{aligned}
\mathcal{M}^e(row(s))(e) &= \mathcal{M}^e(\delta_1^e(q_{init}, s))(e) \quad \text{Lemma 4} \\
&= \delta_2^e(\delta_1^e(q_{init}, s), e) \\
&= f(s, e) \quad \text{Theorem 2}
\end{aligned}$$

For all $s_1, s_2 \in S$, if $row(s_1) \neq row(s_2)$ then there

exists $e \in \Sigma_E \cup E$ such that $f(s_1, e) \neq f(s_2, e)$ because $row(s_1) \neq row(s_2)$.

Then for all $s_1, s_2 \in S$, if $row(s_1) \neq row(s_2)$ then there exists $e \in \Sigma_E \cup E$ such that $\mathcal{M}^e(row(s_1))(e) \neq \mathcal{M}^e(row(s_2))(e)$. From Lemma 5, for all $s_1, s_2 \in S$, if $row(s_1) \neq row(s_2)$ then there exists $e \in \Sigma_E^*$ such that $\mathcal{M}^e(row(s_1))(e) \neq \mathcal{M}^e(row(s_2))(e)$. Therefore, $\mathcal{M}^e$ is minimal. □

4.2 Separating Predicates

Given an evidence Mealy machine with an alphabet Σ, we require two pieces to build a symbolic Mealy machine: (i) a Boolean algebra $\mathcal{A}$ with $\mathfrak{D}_\mathcal{A} = \Sigma$, and (ii) a partitioning function P for $\mathcal{A}$.

We have already defined a partitioning function in Definition 8.

Given an evidence Mealy machine $\mathcal{M}^e = (\Sigma_E, Q, q_{init}, O, \delta^e)$, a Boolean algebra $\mathcal{A}$ with domain Σ, and an appropriate partitioning function P, we build a symbolic Mealy machine $\mathcal{M} = (\mathcal{A}, Q, q_{init}, O, \delta)$ using that Boolean algebra and that exact configuration of the state space Q, the initial state q_{init}. All that remains is the construction of the transition relation δ.

For each $q \in Q$, we gather all transitions of an evidence Mealy machine out of q into a set $\Delta = \{(q, a, q', o\}$ and counstruct a list L_Σ indexed over the states $q_i \in Q$ and the outputs $o_j \in O$ such that each set in l_Σ is $l_{i,j} = \{a \mid (q, a, q_i, o_j) \in \Delta_q\}$ We apply the partitioning function to get a list of separating predicates $L_{\Psi_A} = P(L_\Sigma)$ which is also indexed over $q_i \in Q$ and $o_j \in O$, and add $(q, \varphi_{i,j}, q_i, o_j)$ to δ for each $\varphi_{i,j} \in L_{\Psi_A}$.

We are now ready to prove the final result that a symbolic Mealy machine constructed from an observation table is minimal.

Theorem 4 (Minimality of Symbolic Mealy Machine)**.** *Given a cohesive observation Table $T = (\Sigma, S, R, \Sigma_E, E, f)$, if $\mathcal{M}^e = (\Sigma, Q, q_{init}, O, \delta^e)$ is minimal and compatible with T, a symbolic Mealy machine $\mathcal{M} = (\mathcal{A}, Q, q_{init}, O, \delta)$ constructed from $\mathcal{M}^e$ and a partitioning function P is minimal.*

Proof. $\mathcal{M}$ is obviously deterministic, complete, clean, and normalized from the condition of a partitioning function. From Lemma 4, all states of $\mathcal{M}$ are reachable. Finally, we show for all $p, q \in Q$ if $p \neq q$ then there exists $w \in \Sigma^+ \mathcal{M}(p, w) \neq \mathcal{M}(q, w)$. Before that, we have to prove for all $w \in \Sigma_E^+$ and $q \in Q$, $\mathcal{M}(q, w) = \mathcal{M}^e(q, w)$. We prove this by induction on the length of w. For length 1, by the definition of a partitioning function, $\delta_2(q, a) = \delta_2^e(q, a)$ holds.

Let us assume that for all $w \in \Sigma_E^+$ of length at most $k \geq 0$, $\delta_2(q, w) = \delta_2^e(q, w)$ holds. Let $w' \in \Sigma_E^+$ with length $k + 1$. we can find a decomposition of w' such that $w' = a \cdot w$ for some word of length k and $a \in \Sigma_E$ We can conclude

$$\begin{aligned}
\delta_2(q, w') &= \delta_2(q, a \cdot w) \\
&= \delta_2(q', w) \quad q' = \delta_1(q, a) \\
&= \delta_2^e(q', w) \quad \text{induction hypothesis} \\
&= \delta_2^e(q, a \cdot w) \\
&= \delta_2^e(q, w')
\end{aligned}$$

Therefore, for all $w \in \Sigma_E^+$ and $q \in Q$, $\mathcal{M}(q, w) = \mathcal{M}^e(q, w)$ holds, from theorem 3, for all $p, q \in Q$ if $p \neq q$ then there exists $w \in \Sigma_E^+ \mathcal{M}(p, w) \neq \mathcal{M}(q, w)$ also holds. Here, from Lemma 5, for all $p, q \in Q$ if $p \neq q$ then there exists $w \in \Sigma^+ \mathcal{M}(p, w) \neq \mathcal{M}(q, w)$ holds. Thus $\mathcal{M}$ is minimal. □

The results of Theorem 1 and 4 prove that a symbolic mealy machine constructed from an observation table has the minimum number of states.

Corollary 1. *Given a cohesive observation table $T = (\Sigma, S, R, \Sigma_E, E, f)$ and a partitioning function P, a symbolic Mealy machine $\mathcal{M} = (\mathcal{A}, Q, q_{init}, O, \delta)$ constructed from T and P has a minimum number of states.*

Algorithm 2 Active Learning of symbolic Mealy machine

1: Initialize $S = \{\epsilon\}$, $R = \{a\}$, $\Sigma_E = \{a\}$, and $E = \emptyset$ (a is an arbitrary character from Σ)
2: Construct the initial table $T = (\Sigma, S, R, \Sigma_E, E, f)$
3: **while** true **do**
4: **while** T is not cohesive **do**
5: **if** T is not closed **then**
6: Apply close to T
7: **else if** T is not consistent **then**
8: Apply make-consistent to T
9: **else if** T is not evidence-closed **then**
10: Apply evidence-close to T
11: **else if** T is not output-closed **then**
12: Apply output-close to T
13: **end if**
14: **end while**
15: Construct the evidence mealy machine $\mathcal{M}_H^e$
16: Construct the symbolic mealy machine $\mathcal{M}_H$
17: Send $\mathcal{M}_H$ to the equivalence oracle
18: **if** the equivalence oracle returns a counterexample c **then**
19: Add all prefixes of c to R
20: **else**
21: **return** $\mathcal{M}_H$
22: **end if**
23: **end while**

4.3 Algorithm Description

We present a description of Λ_M^*. The algorithm begins by initializing an observation table with $S = \{\epsilon\}$, $R = \{a\}$, $\Sigma_E = \{a\}$ for an arbitrary $a \in \Sigma$, and $E = \emptyset$. f is initially undefined. The knowledge of the table is grown using the operations fill, close, make-consistent, evidence-close, and *output-close*.

The operation fill asks a membership query for all $w \in S \cup R$ and $e \in \Sigma_E \cup E$ for which f is undefined and then adds those results to f.

The operation close checks the existence of $r \in R$ such that for all $s \in S$ $row(s) \neq row(r)$ and adds to r to S such an r.

The operation make-consistent checks the existence of $w_1, w_2 \in S \cup R$, and $a \in \Sigma$ such that $w_1 \cdot a, w_2 \cdot a \in S \cup R$ and $row(w_1) = row(w_2)$ but $row(w_1 \cdot a) \neq row(w_2 \cdot a)$. Then it adds a $a \cdot e$ to E using the $e \in E$ such that $f(w_1 \cdot a \cdot e) \neq f(w_2 \cdot a \cdot e)$.

The operation evidence-close checks the existence of $s \in S$ and a $e \in E$ such that $s \cdot e \notin S \cup R$ and adds all prefixes of such a $s \cdot e$ to R except those already present in R.

The operation output-close checks the existence of $w \in \Sigma^*, a \in \Sigma$ such that $w \cdot a \in S \cup R$ but $a \notin \Sigma_E$. Then it adds such a to Σ_E.

Upon receiving a counterexample $t \in \Sigma^*$ from an equivalence query sent to the oracle, all prefixes of t are added to R except those already present in S.

Algorithm 2 shows an overview of the learning algorithm: after the table is initialized, the operations make-consistent, evidence-close, close, and output-close are applied until the table is cohesive. A symbolic Mealy machine $\mathcal{M}_H$ is then conjectured from the table, and an equivalence query is performed: if $\mathcal{M}_H$ is equivalent to the target, then the algorithm terminates. Otherwise, a counterexample is produced and processed, and the procedure repeats.

5 Conclusion

We give the formal definition of symbolic Mealy machines so that Mealy machines can operate over an infinite set. We then define the minimality of symbolic Mealy machines and prove that minimal symbolic Mealy machines have the minimum number of states. Next, we give an extension of Λ^* for learning symbolic Mealy machines, called Λ_M^*.

Finally, We prove that a symbolic Melay machine learned by Λ_M^* has the minimum number of states.

Termination and complexity analysis of this algorithm is future work. Implementation is also future work.

参 考 文 献

[1] Angluin, D.: Learning Regular Sets from Queries and Counterexamples, *Inf. Comput.*, Vol. 75, No. 2(1987), pp. 87–106.

[2] Argyros, G. and D'Antoni, L.: The Learnability of Symbolic Automata, *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I*, Chockler, H. and Weissenbacher, G.(eds.), Lecture Notes in Computer Science, Vol. 10981, Springer, 2018, pp. 427–445.

[3] Argyros, G., Stais, I., Kiayias, A., and Keromytis, A. D.: Back in Black: Towards Formal, Black Box Analysis of Sanitizers and Filters, *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, IEEE Computer Society, 2016, pp. 91–109.

[4] Botincan, M. and Babic, D.: Sigma*: symbolic learning of input-output specifications, *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, Giacobazzi, R. and Cousot, R.(eds.), ACM, 2013, pp. 443–456.

[5] Chubachi, K., Hendrian, D., Yoshinaka, R., and Shinohara, A.: Query Learning Algorithm for Residual Symbolic Finite Automata, *Proceedings Tenth International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2019, Bordeaux, France, 2-3rd September 2019*, Leroux, J. and Raskin, J.(eds.), EPTCS, Vol. 305, 2019, pp. 140–153.

[6] D'Antoni, L. and Veanes, M.: Minimization of symbolic automata, *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, Jagannathan, S. and Sewell, P.(eds.), ACM, 2014, pp. 541–554.

[7] Drews, S. and D'Antoni, L.: Learning Symbolic Automata, *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I*, Legay, A. and Margaria, T.(eds.), Lecture Notes in Computer Science, Vol. 10205, 2017, pp. 173–189.

[8] Fisman, D., Frenkel, H., and Zilles, S.: Inferring Symbolic Automata, *Log. Methods Comput. Sci.*, Vol. 19, No. 2(2023).

[9] Isberner, M., Howar, F., and Steffen, B.: The TTT Algorithm: A Redundancy-Free Approach to Active Automata Learning, *Runtime Verification - 5th International Conference, RV 2014, Toronto, ON, Canada, September 22-25, 2014. Proceedings*, Bonakdarpour, B. and Smolka, S. A.(eds.), Lecture Notes in Computer Science, Vol. 8734, Springer, 2014, pp. 307–322.

[10] Lathouwers, S., Everts, M. H., and Huisman, M.: Verifying Sanitizer Correctness through Black-Box Learning: A Symbolic Finite Transducer Approach, *Proceedings of the 6th International Conference on Information Systems Security and Privacy, ICISSP 2020, Valletta, Malta, February 25-27, 2020*, Furnell, S., Mori, P., Weippl, E. R., and Camp, O.(eds.), SCITEPRESS, 2020, pp. 784–795.

[11] Maler, O. and Mens, I.: A Generic Algorithm for Learning Symbolic Automata from Membership Queries, *Models, Algorithms, Logics and Tools - Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday*, Aceto, L., Bacci, G., Bacci, G., Ingólfsdóttir, A., Legay, A., and Mardare, R.(eds.), Lecture Notes in Computer Science, Vol. 10460, Springer, 2017, pp. 146–169.

[12] Niese, O.: *An integrated approach to testing complex systems*, PhD Thesis, Technical University of Dortmund, Germany, 2003.

[13] Okudono, T., Waga, M., Sekiyama, T., and Hasuo, I.: Weighted Automata Extraction from Recurrent Neural Networks via Regression on State Spaces, *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, AAAI Press, 2020, pp. 5306–5314.

[14] Peled, D. A., Vardi, M. Y., and Yannakakis, M.: Black Box Checking, *Formal Methods for Protocol Engineering and Distributed Systems, FORTE XII / PSTV XIX'99, IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XII) and Protocol Specification, Testing and Verification (PSTV XIX), October 5-8, 1999, Beijing, China*, Wu, J., Chanson, S. T., and Gao, Q.(eds.), IFIP Conference Proceedings, Vol. 156, Kluwer, 1999, pp. 225–240.

[15] Rivest, R. L. and Schapire, R. E.: Inference of Finite Automata Using Homing Sequences, *Machine Learning: From Theory to Applications - Cooperative Research at Siemens and MIT*, Hanson, S. J., Remmele, W., and Rivest, R. L.(eds.), Lecture Notes in Computer Science, Vol. 661, Springer, 1993, pp. 51–73.

[16] Shahbaz, M. and Groz, R.: Inferring Mealy Ma-

chines, *FM 2009: Formal Methods, Second World Congress, Eindhoven, The Netherlands, November 2-6, 2009. Proceedings*, Cavalcanti, A. and Dams, D.(eds.), Lecture Notes in Computer Science, Vol. 5850, Springer, 2009, pp. 207–222.

[17] Suzuki, K., Hendrian, D., Yoshinaka, R., and Shinohara, A.: Query Learning Algorithm for Symbolic Weighted Finite Automata, *Proceedings of the 15th International Conference on Grammatical Inference, 23-27 August 2021, Virtual Event*, Chandlee, J., Eyraud, R., Heinz, J., Jardine, A., and van Zaanen, M.(eds.), Proceedings of Machine Learning Research, Vol. 153, PMLR, 2021, pp. 202–216.

[18] Veanes, M., Hooimeijer, P., Livshits, B., Molnar, D., and Bjørner, N. S.: Symbolic finite state transducers: algorithms and applications, *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, January 22-28, 2012*, Field, J. and Hicks, M.(eds.), ACM, 2012, pp. 137–150.

[19] Weiss, G., Goldberg, Y., and Yahav, E.: Extracting Automata from Recurrent Neural Networks Using Queries and Counterexamples, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, Dy, J. G. and Krause, A.(eds.), Proceedings of Machine Learning Research, Vol. 80, PMLR, 2018, pp. 5244–5253.