Query Learning Algorithm for Symbolic Weighted Finite Automata

Kaito Suzuki KAITO_SUZUKI@SHINO.ECEI.TOHOKU.AC.JP
Diptarama Hendrian DIPTARAMA@TOHOKU.AC.JP
Ryo Yoshinaka RYOSHINAKA@TOHOKU.AC.JP
Ayumi Shinohara AYUMIS@TOHOKU.AC.JP
Graduate School of Information Sciences, Tohoku University, Sendai, Japan

Editors: Jane Chandlee, Rémi Eyraud, Jeffrey Heinz, Adam Jardine, and Menno van Zaanen

Abstract

We propose a query learning algorithm for an extension of weighted finite automata (WFAs), named symbolic weighted finite automata (SWFAs), which can handle strings over infinite alphabets more efficiently. Based on the idea of symbolic finite automata, SWFAs generalize WFAs by allowing transitions to be functions from a possibly infinite alphabet to weights. Our algorithm can learn SWFAs if functions in transitions are also learnable by queries. We also investigate minimization and equivalence checking for SWFAs.

Keywords: query learning, exact concept learning, weighted finite automata, symbolic finite automata

1. Introduction

In this paper, we present a query learning algorithm for symbolic weighted finite automata (SWFAs), along with algorithms for minimizing and checking equivalence of them. SWFAs can be seen as the unification of two notable extensions of classical finite automata (FAs), weighted finite automata (WFAs) and symbolic finite automata (SFAs). WFAs are finite automata to represent *formal power series* on a semiring, which are functions from a set of strings over a finite alphabet to a semiring. SFAs are an extension of FAs, where each transition edge is labeled with a *predicate*, which represents a possibly infinite subset of the alphabet compactly. By combining them, SWFAs can represent power series as with WFAs while dealing with a large or infinite alphabet efficiently as well as SFAs. Transition edges of an SWFA are labeled by functions from the alphabet to a semiring instead of predicates. Because of their generality, SWFAs have been considered in several recent studies. Herrmann and Vogler (2016) introduced SWFAs with data storage and investigated their expressiveness for some types of data storage. Alur et al. (2017) considered SWFAs with nesting operations and parallel execution, and proposed an efficient evaluation algorithm for them. Jaksic et al. (2018a,b); Waga (2019) used SWFAs to develop underlying algorithms for runtime verification of cyber-physical systems. However, the learning of SWFAs has not been studied yet.

The problem of learning FAs has been studied for a long time. Angluin (1987) proposed the first algorithm L_* which learns deterministic FAs with an oracle, called a *minimally adequate teacher (MAT)*, which answers *membership queries (MQs)* and *equivalence queries (EQs)* from the learner. Bergadano and Varricchio (1996) extended L^* for learning WFAs

© 2021 K. Suzuki, D. Hendrian, R. Yoshinaka & A. Shinohara.

and Bisht et al. (2006) improved the complexity of the algorithm. The learning of SFAs under the MAT model has also been studied. Drews and D'Antoni (2017) showed that deterministic SFAs can be learned if partitions of Σ are inferable from finite examples. Argyros and D'Antoni (2018) proposed a more powerful algorithm that learns deterministic SFAs under the assumption that predicates on transition edges are also MAT learnable. This work was extended for learning non-deterministic SFAs by Chubachi et al. (2019).

This paper proposes a query learning algorithm for SWFAs by combining Bisht et al. and Argyros and D'Antoni. We prove that SWFAs are exactly learnable under the MAT model if functions on transition edges are also MAT learnable. The query complexity of our algorithm is polynomial in the minimal number of states to represent the target power series, the length of the longest counterexample against EQs, and the number of queries required to learn functions on transition edges. We also show that minimization and equivalence checking for SWFAs can be achieved in polynomial time in the number of states of given SWFAs.

After preparing required mathematical notions in Section 2, we give a formal definition and important properties of SWFAs in Section 3. Our proposed query learning algorithm is given in Section 4, whose correctness and query complexity are shown in Section 5. Section 6 concludes this paper.

2. Preliminaries

2.1. Notation

The set of all strings over an alphabet Σ is denoted by Σ^*. The empty string is denoted by ϵ. The set of all functions from X to Y is denoted by Y^X.

We consider possibly infinite matrices throughout this paper. A matrix over Y whose rows and columns are indexed by sets P and S, respectively, is identified with a function from $P \times S$ to Y. The element of a matrix $A \in Y^{P\times S}$ indexed by $p \in P$ and $s \in S$ is denoted by $A[p, s]$. A vector may be seen as a special case of a matrix whose column index set is a singleton $S = \{s\}$, where we often drop s and simply refer to the element $A[p, s]$ by $A[p]$ regarding $A \in Y^P$. A pair (P', S') of subsets $P' \subseteq P$ and $S' \subseteq S$ is called a *mask* and the (P', S')-sub-block of $A \in Y^{P\times S}$ with row index set P' and column index set S' is denoted as $A_{(P',S')} \in Y^{P'\times S'}$. Particularly when $P' = P$ and S' is a singleton set $\{s\}$, the matrix $A_{(P,\{s\})}$ is often denoted by $A[\cdot, s]$ and is called the *column vector* of A indexed by $s \in S$. The *row vector* of A indexed by $p \in P$ is symmetrically defined and denoted by $A[p, \cdot]$. The transpose of $A \in Y^{P\times S}$, denoted by $A^\mathsf{T} \in Y^{S\times P}$, exchanges the rows and columns of A. We often express a matrix as a table ordering the row and column indices arbitrarily.

When Y is a semiring, the multiplication $AB \in Y^{P\times S}$ of two matrices $A \in Y^{P\times Q}$ and $B \in Y^{Q\times S}$ is defined by $(AB)[p, s] = \sum_{q\in Q} A[p, q]B[q, s]$ for all $p \in P$ and $s \in S$ provided that Q is finite. When Y is a field, the rank of $A \in Y^{P\times S}$ is defined in the usual way and denoted by $\mathrm{rank}(A)$. A mask (P', S') is called a *basis* if $\mathrm{rank}(A_{(P',S')}) = \mathrm{rank}(A)$. The basis (P', S') is *minimal* if $|P'| = |S'| = \mathrm{rank}(A)$. A mask (P', S') is *non-singular* when $A_{(P',S')}$ is non-singular. That is, there is a unique matrix called the inverse $A^{-1}_{(P',S')}$ of $A_{(P',S')}$ such that $A_{(P',S')}A^{-1}_{(P',S')} = I$, where I is an identity matrix. A minimal basis is always non-

singular since a matrix $A_{(P',S')}$ is non-singular if and only if $|P'| = |S'| = \mathrm{rank}(A_{(P',S')})$. However, the reverse is not always true.

2.2. Learning under Minimally Adequate Teacher

Query learning is an active learning model, where an algorithm actively asks queries to the teacher and constructs a representation of the target function from a domain X to a codomain Y using the teacher's answers. The most famous setting of query learning is learning under a *minimally adequate teacher (MAT)*, proposed by Angluin (1987). In this model, the MAT can answer two types of queries concerning an unknown function f_*: *membership queries (MQs)* and *equivalence queries (EQs)*. For an MQ, the MAT receives $w \in X$ and returns $f_*(w)$. For an EQ, the MAT receives a hypothesis $\mathcal{H}$ that represents a function $f_\mathcal{H} \in Y^X$. The MAT returns "yes" if $f_* = f_\mathcal{H}$. Otherwise, the MAT returns $c \in X$ such that $f_*(c) \neq f_\mathcal{H}(c)$ as a counterexample.

3. Symbolic Weighted Finite Automata

Symbolic weighted finite automata (SWFAs) combine symbolic finite automata (SFAs) and weighted finite automata (WFAs) to represent formal power series over a possibly infinite alphabet. SFAs are defined over a Boolean algebra, which has *predicates* (or *guards*), finite descriptions to represent possibly infinite subsets of the alphabet. SFAs use predicates as labels of transition edges. Likewise, SWFAs have edges labeled by finitely describable functions, called *guard functions*, in a fixed class $\mathcal{G}$ from the alphabet Σ to a semiring $\mathbb{S}$.

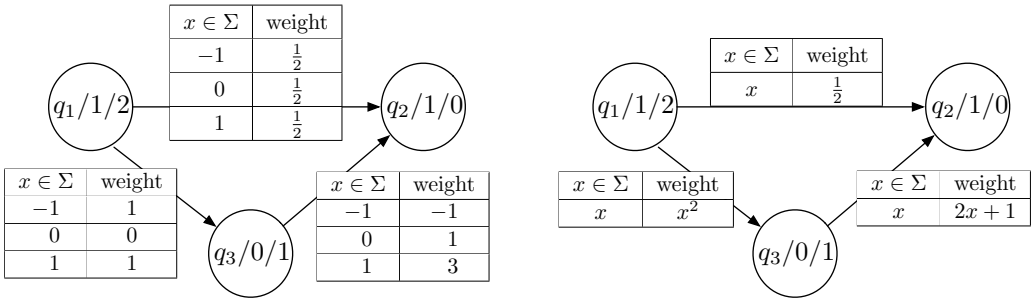
Definition 1 (Symbolic Weighted Finite Automata, SWFAs) A *symbolic weighted finite automaton (SWFA)* $\mathcal{A}$ over $\mathcal{G}$ is a tuple $(\mathcal{G}, Q, \boldsymbol{\alpha}, \boldsymbol{\beta}, \Delta)$, where $\mathcal{G} \subseteq \mathbb{S}^\Sigma$ is a class of guard functions from Σ to $\mathbb{S}$, Q is a finite set of states, $\boldsymbol{\alpha} \in \mathbb{S}^Q$ is a vector of initial weights, $\boldsymbol{\beta} \in \mathbb{S}^Q$ is a vector of final weights, and $\Delta \in \mathcal{G}^{Q \times Q}$ is a transition relation. For any $x \in \Sigma$, the transition relation for x is represented by a matrix $\Delta_x \in \mathbb{S}^{Q \times Q}$, i.e., $\Delta_x[q, q'] = \Delta[q, q'](x)$ for all $(q, q') \in Q^2$. Δ_x can be extended for strings $w \in \Sigma^*$ as $\Delta_w = \Delta_{x_1} \Delta_{x_2} \dots \Delta_{x_l}$, where $w = x_1 x_2 \dots x_l$ with $x_i \in \Sigma$. The formal power series $f_\mathcal{A}$ represented by $\mathcal{A}$ is defined by $f_\mathcal{A}(w) = \boldsymbol{\alpha}^\mathsf{T} \Delta_w \boldsymbol{\beta}$ for all $w \in \Sigma^*$ and called *$\mathcal{G}$-recognizable*. We allow a special SWFA with no states, which represents the zero constant series, called the *zero SWFA*.

If $\mathbb{S}$ is the two-element Boolean algebra, SWFAs coincide with SFAs. For the class $\mathcal{G}^{\mathrm{wfa}}$ of all guard functions from a *finite* alphabet Σ to a semiring $\mathbb{S}$, SWFAs over $\mathcal{G}^{\mathrm{wfa}}$ are exactly WFAs, except that usually the transition relations of WFAs are described by enumeration. That is, the transition relation of a WFA is $|\Sigma|$ matrices in $\mathbb{S}^{Q \times Q}$.

Figure 1 shows toy examples of a WFA and an SWFA representing the same power series $f \colon \{-1, 0, 1\}^* \to \mathbb{Q}$, where $\mathbb{Q}$ is the set of rational numbers. The SWFA is readily applicable for extending the alphabet to the integer set with no change.

3.1. Basic Properties of SWFAs

In the following of this paper, we assume the codomain of the power series in concern is a field $\mathbb{F}$. This restriction has brought fruitful positive results on WFAs including their efficient learnability (Bergadano and Varricchio, 1996; Bisht et al., 2006). In addition, we



(a) WFA (b) SWFA

Figure 1: (a) An example WFA on $\Sigma = \{-1, 0, 1\}$. Each circle represents a state q, in which the triple shows q, $\boldsymbol{\alpha}[q]$, and $\boldsymbol{\beta}[q]$. The table labeling each edge from q to q' shows $\Delta_x[q, q']$ for each $x \in \Sigma$. (b) An example SWFA representing the same power series as (a). The label on each edge from q to q' is the assigned guard function from Σ to $\mathbb{Q}$. We omit edges whose functions always return zero.

assume that any guard function class $\mathcal{G}$ is closed under linear combination throughout this paper. This assumption may be seen as the weighted counterpart of the Boolean closure property of predicates in the context of SFAs. Under the assumption, we observe that some basic properties of WFAs over fields also hold for SWFAs.

Definition 2 The matrix $\mathbf{H}_f$ associated to a formal power series $f\colon \Sigma^* \to \mathbb{F}$ is an infinite matrix with rows and columns indexed by strings in Σ^* such that $\mathbf{H}_f[p, s] = f(ps)$ for all $p, s \in \Sigma^*$.

Fliess's theorem (1974) can be seen as a weighted counterpart of the Myhill-Nerode theorem (Nerode, 1958), which also holds for SWFAs.

Theorem 3 *The rank of the matrix $\mathbf{H}_f$ associated to $f\colon \Sigma^* \to \mathbb{F}$ is finite if and only if f is $\mathcal{G}$-recognizable for some $\mathcal{G}$. In that case, there exists an SWFA $\mathcal{A}$ over $\mathcal{G}$ with $\operatorname{rank}(\mathbf{H}_f)$ states representing f and no SWFA with fewer states can represent f.*

The proof in Balle and Mohri (2015) works regardless of whether Σ is finite or infinite.

Hereafter, we fix the $\mathcal{G}$-recognizable power series f in concern and we often drop the subscript f from $\mathbf{H}_f$ if there is no risk of confusion.

When (P, S) is a minimal basis of $\mathbf{H}$ and the alphabet is finite, one can construct a WFA representing f from $\mathbf{H}_{(P,S)}$ and $\mathbf{H}_{(P,x,S)}$ for all $x \in \Sigma$, where $\mathbf{H}_{(P,x,S)}[p, s] = f(pxs)$ for all $p \in P$ and $s \in S$ (Bisht et al., 2006). We will establish the parallel result on SWFAs in Proposition 5.

Definition 4 For a non-singular mask (P, S), the SWFA based on (P, S) is defined to be $\mathcal{A}_{(P,S)} = (\mathcal{G}, Q, \boldsymbol{\alpha}, \boldsymbol{\beta}, \Delta)$, where $Q = P$, $\boldsymbol{\alpha}^\mathsf{T} = \mathbf{H}_{(\{\epsilon\},S)}\mathbf{H}^{-1}_{(P,S)}$, $\boldsymbol{\beta} = \mathbf{H}_{(P,\{\epsilon\})}$ and $\Delta_x = \mathbf{H}_{(P,x,S)}\mathbf{H}^{-1}_{(P,S)}$ for all $x \in \Sigma$.

One may wonder whether functions in Δ defined above belong to $\mathcal{G}$. We will show it is indeed the case in the following propositions.

Proposition 5 *Let (P, S) be a minimal basis. Then, $\mathcal{A}_{(P,S)}$ correctly represents f. Moreover, all guard functions in Δ of $\mathcal{A}_{(P,S)}$ are in $\mathcal{G}$.*

Proof By Theorem 3, there is an SWFA $\mathcal{A}' = (\mathcal{G}, Q', \boldsymbol{\alpha}', \boldsymbol{\beta}', \Delta')$ with $|P|$ $(= |S|)$ states representing f, i.e., $f(w) = \boldsymbol{\alpha}'^{\mathsf{T}} \Delta'_w \boldsymbol{\beta}'$ for all $w \in \Sigma^*$. Define matrices $U_P \in \mathbb{F}^{P \times Q'}$ and $V_S \in \mathbb{F}^{Q' \times S}$ by $U_P[p, \cdot] = \boldsymbol{\alpha}'^{\mathsf{T}} \Delta'_p$ for all $p \in P$ and $V_S[\cdot, s] = \Delta'_s \boldsymbol{\beta}'$ for all $s \in S$, respectively. Then we have $\mathbf{H}_{(P,S)} = U_P V_S$ and $\mathbf{H}_{(P,x,S)} = U_P \Delta'_x V_S$ for $x \in \Sigma$. Since $\mathbf{H}_{(P,S)}$ is non-singular, U_P and V_S are also non-singular. From the definition of $\mathcal{A}_{(P,S)}$, $\mathbf{H}_{(P,x,S)} = \Delta_x \mathbf{H}_{(P,S)} = \Delta_x U_P V_S$. Thus, $\Delta'_x = U_P^{-1} \Delta_x U_P$. This fact can be extended to strings $w \in \Sigma^*$ as $\Delta'_w = U_P^{-1} \Delta_w U_P$. Since $\mathbf{H}_{(\{\epsilon\},S)} = \boldsymbol{\alpha}'^{\mathsf{T}} V_S$ and $\mathbf{H}_{(P,\{\epsilon\})} = U_P \boldsymbol{\beta}'$, we have $\boldsymbol{\alpha}'^{\mathsf{T}} = \boldsymbol{\alpha}^{\mathsf{T}} U_P$ and $\boldsymbol{\beta}' = U_P^{-1} \boldsymbol{\beta}$. Therefore, $f(w) = \boldsymbol{\alpha}'^{\mathsf{T}} \Delta'_w \boldsymbol{\beta}' = \boldsymbol{\alpha}^{\mathsf{T}} U_P U_P^{-1} \Delta_w U_P U_P^{-1} \boldsymbol{\beta} = \boldsymbol{\alpha}^{\mathsf{T}} \Delta_w \boldsymbol{\beta}$. Additionally, $\Delta'_x = U_P^{-1} \Delta_x U_P$ shows that every guard function in Δ can be represented by a linear transformation of those in Δ'. Thus, all guard functions in Δ are in $\mathcal{G}$. ■

Proposition 6 guarantees that the guard functions in the transition relation constructed in Definition 4 indeed belong to $\mathcal{G}$, by extending the property shown in Proposition 5 to the case that (P, S) is a non-singular mask.

Proposition 6 *If (P, S) is non-singular, all guard functions of $\mathcal{A}_{(P,S)}$ are in $\mathcal{G}$.*

Proof Let (P', S') be a minimal basis that expands (P, S), i.e., $P' \supseteq P$ and $S' \supseteq S$. For the SWFA $\mathcal{A}_{(P',S')} = (\mathcal{G}, Q', \boldsymbol{\alpha}', \boldsymbol{\beta}', \Delta')$, we have for all $x \in \Sigma$,

$$\Delta'_x \mathbf{H}_{(P',S')} = \mathbf{H}_{(P',x,S')} ,$$

which can be rewritten as

$$\begin{pmatrix} \Delta'_{(P,P)_x} & \Delta'_{(P,\tilde{P})_x} \\ \Delta'_{(\tilde{P},P)_x} & \Delta'_{(\tilde{P},\tilde{P})_x} \end{pmatrix} \begin{pmatrix} \mathbf{H}_{(P,S)} & \mathbf{H}_{(P,\tilde{S})} \\ \mathbf{H}_{(\tilde{P},S)} & \mathbf{H}_{(\tilde{P},\tilde{S})} \end{pmatrix} = \begin{pmatrix} \mathbf{H}_{(P,x,S)} & \mathbf{H}_{(P,x,\tilde{S})} \\ \mathbf{H}_{(\tilde{P},x,S)} & \mathbf{H}_{(\tilde{P},x,\tilde{S})} \end{pmatrix}$$

where $\tilde{P} = P' \setminus P$ and $\tilde{S} = S' \setminus S$. Thus $\Delta'_{(P,P)_x} \mathbf{H}_{(P,S)} + \Delta'_{(P,\tilde{P})_x} \mathbf{H}_{(\tilde{P},S)} = \mathbf{H}_{(P,x,S)}$ and

$$\Delta_x = \mathbf{H}_{(P,x,S)} \mathbf{H}_{(P,S)}^{-1} = (\Delta'_{(P,P)_x} \mathbf{H}_{(P,S)} + \Delta'_{(P,\tilde{P})_x} \mathbf{H}_{(\tilde{P},S)}) \mathbf{H}_{(P,S)}^{-1}$$

for all $x \in \Sigma$. Therefore, all guard functions in Δ can be represented by a linear transformation of those of Δ'. That is, they belong to $\mathcal{G}$. ■

3.2. Minimization and Equivalence Checking for SWFAs

In this subsection, we consider *minimization* and *equivalence checking* for SWFAs. Minimization (also called *standardization*) is the problem to minimize the number of states of a given SWFA. Equivalence checking is the problem to check whether given two SWFAs are equivalent or not. Our algorithms assume that $\mathcal{G}$ admits a zero-checking procedure.

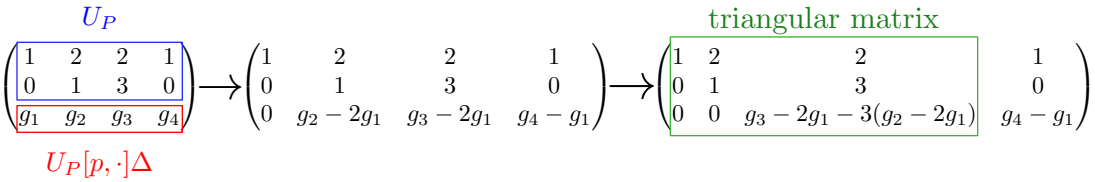


Figure 2: Gaussian elimination dealing with guard functions

Assumption 7 *For a given $g \in \mathcal{G}$, there is a procedure that decides whether $g(x) = 0$ for all $x \in \Sigma$. If not, it finds a witness $x \in \Sigma$ such that $g(x) \neq 0$.*

This requirement corresponds to the emptiness checking in the context of SFAs (Argyros and D'Antoni, 2018; Chubachi et al., 2019). Under the assumption, we present a minimization procedure for SWFAs inspired by Schützenberger's algorithm for WFAs (1961). The minimization procedure for SWFAs consists of two phases: First we obtain a minimal basis (P, S) for a given SWFA; Second, we construct a minimized SWFA based on (P, S) and $\mathbf{H}_{(P,S)}$.

Suppose an SWFA $\mathcal{A} = (\mathcal{G}, Q, \boldsymbol{\alpha}, \boldsymbol{\beta}, \Delta)$ representing $f_{\mathcal{A}}$ is given as an input. The first phase is shown in Algorithm 1. For two sets P and S of strings, define matrices $U_P \in \mathbb{F}^{P\times Q}$ and $V_S \in \mathbb{F}^{Q\times S}$ by $U_P[p,\cdot] = \boldsymbol{\alpha}^\mathsf{T}\Delta_p$ for all $p \in P$ and $V_S[\cdot, s] = \Delta_s\boldsymbol{\beta}$ for all $s \in S$, respectively. When U_P and V_S are bases of U_{Σ^*} and V_{Σ^*}, respectively, (P, S) will be a basis of $\mathbf{H}_{f_{\mathcal{A}}}$, since $U_{\Sigma^*}V_{\Sigma^*} = \mathbf{H}_{f_{\mathcal{A}}}$. Starting from $P = \{\epsilon\}$, our procedure constructs such P by expanding P and increasing the rank of U_P in the loop of Line 2. Of course when $|P| = |Q|$, we cannot expand P any further. Otherwise, to execute Line 4, we first consider the matrix $[U_P^\mathsf{T}\ (U_P[p,\cdot]\Delta)^\mathsf{T}]^\mathsf{T}$, which concatenates U_P and $U_P[p,\cdot]\Delta$, for each $p \in P$. As shown in the example in Figure 2, Gaussian elimination process dealing with guard functions makes the first $|P| + 1$ columns of the concatenated matrix a triangular matrix, which includes a guard function only at the lower-right corner. Then, we check whether the guard function g at the lower-right corner of the triangular matrix always outputs 0 or not. If $g(x) = 0$ for all $x \in \Sigma$, the finding in Line 4 fails. If not, we can get $x \in \Sigma$ such that $g(x) \neq 0$ by Assumption 7. It means we have succeeded in finding $p \in P$ and $x \in \Sigma$ such that $U_P[p,\cdot]\Delta_x$ is linearly independent of U_P. Then, we extend P by px. We construct S and V_S with the desired property by the symmetric procedure. The obtained mask (P, S) is a basis, but not necessarily minimal. By removing elements of P and S as much as possible while keeping the rank of the matrix $\mathbf{H}_{(P,S)}$, we obtain a minimal basis.

The second phase constructs a minimized SWFA from the minimal basis. By Proposition 5, it is enough to show that the SWFA $\mathcal{A}_{(P,S)}$ in Definition 4 is computable. The only possible difficulty is in the construction of the transition relation, named $\Delta^{\min}$ here, for which $\Delta^{\min}_x = \mathbf{H}_{(P,x,S)}\mathbf{H}^{-1}_{(P,S)}$ must hold for all $x \in \Sigma$. One can compute the matrix $\mathbf{H}_{(P,\Delta,S)} \in \mathcal{G}^{P\times S}$ defined by $\mathbf{H}_{(P,\Delta,S)}[p, s] = \boldsymbol{\alpha}^\mathsf{T}\Delta_p\Delta\Delta_s\boldsymbol{\beta}$. We then have $\mathbf{H}_{(P,\Delta,S)}[p, s](x) = \mathbf{H}_{(P,x,S)}$ for all $x \in \Sigma$, $p \in P$ and $s \in S$. Thus, $\Delta^{\min} = \mathbf{H}_{(P,\Delta,S)}\mathbf{H}^{-1}_{(P,S)}$ has the desired property[1].

1. The matrices Δ and $\mathbf{H}_{(P,\Delta,S)}$ are over $\mathcal{G}$, which is not necessarily a semiring, whereas the others are over $\mathbb{F}$. We can naturally apply the definition of multiplication of two matrices over a semiring to these cases, since $\mathcal{G}$ is closed under linear combination.

Algorithm 1: Computing a minimal basis from an SWFA

Input: an SWFA $\mathcal{A} = (\mathcal{G}, Q, \boldsymbol{\alpha}, \boldsymbol{\beta}, \Delta)$
Output: a minimal basis (P, S)

```
1  initialize P ← {ϵ}, S ← {ϵ};
2  repeat
3  |  U_P[p, ·] ← α^⊤ Δ_p for all p ∈ P;
4  |  find p ∈ P and x ∈ Σ such that U_P[p, ·]Δ_x is linearly independent of U_P;
5  |  P ← P ∪ {px};
6  until finding in Line 4 fails;
7  repeat
8  |  V_S[·, s] ← Δ_s β for all s ∈ S;
9  |  find s ∈ S and x ∈ Σ such that Δ_x V_S[·, s] is linearly independent of V_S;
10 |  S ← S ∪ {xs};
11 until finding in Line 9 fails;
12 H_(P,S) ← U_P V_S;
13 P ← P \ P' for a maximal P' ⊆ P such that rank(H_(P,S)) = rank(H_(P\P',S));
14 S ← S \ S' for a maximal S' ⊆ S such that rank(H_(P,S)) = rank(H_(P,S\S'));
15 return (P, S);
```

Theorem 8 *Let L and E be the time complexities of computing a linear combination and zero-checking of a guard function, respectively. One can minimize an arbitrary SWFA in $O(|Q|(L|Q|^2 + E))$ time, where $|Q|$ is the number of states of the SWFA.*

Proof The finding at Lines 4 and 9 of Algorithm 1 involves Gaussian elimination dealing with guard functions and a zero-checking on $\mathcal{G}$. The former requires $O(L|Q|^2)$ and the latter requires E time. Thus, the entire procedure requires $O(|Q|(L|Q|^2 + E))$ time since these lines are executed at most $|Q|$ times. ■

When $\mathcal{G}$ is $\mathcal{G}^{\mathrm{wfa}}$, we have $L, E \in O(|\Sigma|)$ and the time complexity $O(|\Sigma||Q|^3)$ coincides with that of Schützenberger's algorithm for minimizing WFAs (1961).

The equivalence checking between SWFAs is immediately derived.

Corollary 9 *One can decide whether two SWFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ represent the same power series in $O((|Q_{\mathcal{A}_1}| + |Q_{\mathcal{A}_2}|)(L(|Q_{\mathcal{A}_1}| + |Q_{\mathcal{A}_2}|)^2 + E))$ time.*

Proof One can construct in linear time the "difference SWFA" $\mathcal{A}$ with $|Q_{\mathcal{A}_1}| + |Q_{\mathcal{A}_2}|$ states that represents $f_{\mathcal{A}} = f_{\mathcal{A}_1} - f_{\mathcal{A}_2}$. Two SWFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are equivalent if and only if the basis of $\mathbf{H}_{f_{\mathcal{A}}}$ obtained by Algorithm 1 is empty. ■

Again, when $\mathcal{G}$ is $\mathcal{G}^{\mathrm{wfa}}$, the time complexity becomes $O(|\Sigma|(|Q_{\mathcal{A}_1}| + |Q_{\mathcal{A}_2}|)^3)$, which coincides with the time complexity of Cortes et al.'s algorithm for equivalence checking for WFAs (2007). Note that if $\mathcal{A}_1$ and $\mathcal{A}_2$ are not equivalent, there must be strings $p \in P$ and $s \in S$ of the obtained minimal basis (P, S) such that $\mathbf{H}_{(P,S)}[p, s] \neq 0$, for which ps witnesses the difference of $f_{\mathcal{A}_1}$ and $f_{\mathcal{A}_2}$. This gives a counterexample to an EQ on SWFAs.

4. Learning Algorithm for Symbolic Weighted Finite Automata

This section presents a learning algorithm under the MAT model for $\mathcal{G}$-recognizable power series using SWFAs when $\mathcal{G}$ is closed under linear combination and satisfies Assumption 7. We extend the existing learning algorithm for WFAs (Bisht et al., 2006) by embedding the key idea of the SFA learning algorithm of Argyros and D'Antoni (2018).

Argyros and D'Antoni's algorithm takes as input a MAT learning algorithm Λ for predicates and uses instances $\Lambda^{q,q'}$ to infer a predicate on the edge between two states q and q'. The learning algorithm plays the role of a MAT for those instances and answers queries from them. Through communication with predicate learners, the learning algorithm constructs its hypothesis. Following this idea, our algorithm also assumes that the class of guard functions $\mathcal{G}$ admits a MAT learner Λ and uses its instances to construct the transition relation of a hypothesis SWFA.

The pseudo-code of our algorithm is shown in Algorithms 2–5. Now, our learning target power series is $f_*\colon \Sigma^* \to \mathbb{F}$. Our goal is to find a minimal basis (P, S) of $\mathbf{H}_{f_*}$ and to construct the SWFA $\mathcal{A}_{(P,S)}$, for which $f_{\mathcal{A}_{(P,S)}} = f_*$ holds by Proposition 5. We call a triple $\mathcal{T} = (P, S, \mathbf{H}_{(P,S)})$ an *observation table*.

Algorithm 2 shows the overall picture of our algorithm. First, our algorithm asks an EQ to the MAT on the zero SWFA. If the MAT answers with a counterexample $c \in \Sigma^*$, our algorithm initializes the observation table $\mathcal{T}$ with $P = \{c\}$ and $S = \{\epsilon\}$. We will expand the mask to obtain a minimal basis, while keeping it non-singular.

To build a hypothesis, we follow the construction shown in Definition 4. Among components of $\mathcal{A}_{(P,S)} = (\mathcal{G}, Q, \boldsymbol{\alpha}, \boldsymbol{\beta}, \Delta)$, one can easily construct $Q = P$, $\boldsymbol{\alpha}^\mathsf{T} = \mathbf{H}_{(\{\epsilon\},S)}\mathbf{H}^{-1}_{(P,S)}$, and $\boldsymbol{\beta} = \mathbf{H}_{(P,\{\epsilon\})}$, since any finite sub-block of $\mathbf{H}_{f_*}$ can be obtained by an appropriate number of MQs. The remaining issue is how to compute Δ. We can get $\Delta_x = \mathbf{H}_{(P,x,S)}\mathbf{H}^{-1}_{(P,S)}$ for concrete $x \in \Sigma$ using MQs, but computing guard functions in Δ is not trivial. For this sake, we need the help of the guard function learner Λ. Algorithm 3 initializes all entries of the transition relation $\Delta^{\mathcal{H}}$ of our hypothesis $\mathcal{H}$ by null and creates $|Q|^2$ instances $\Lambda^{(q,q')}$ of Λ for all state pairs $(q, q') \in Q^2$, and then calls Algorithm 4.

To bring $\Delta^{\mathcal{H}}$ closer to Δ, Algorithm 4 lets $\Lambda^{(q,q')}$ learn $\Delta[q, q']$ by pretending to be a MAT for them, though we do not know the exact goal Δ itself. Proposition 6 guarantees that Λ is capable of learning $\Delta[q, q'] \in \mathcal{G}$. To avoid confusion between queries by $\Lambda^{(q,q')}$ to our algorithm and those by our algorithm to the MAT, we use $\mathcal{G}$-EQs and $\mathcal{G}$-MQs for equivalence queries and membership queries from the instances of Λ, respectively. The answer to a $\mathcal{G}$-MQ on $x \in \Sigma$ from $\Lambda^{(q,q')}$ is just $\Delta_x[q, q']$. Since $\Delta_x[q, \cdot] = \mathbf{H}_{(\{q\},x,S)}\mathbf{H}^{-1}_{(P,S)}$, we require only $|S|$ extra MQs given the observation table.

When $\Lambda^{(q,q')}$ asks a $\mathcal{G}$-EQ on a hypothesis guard function g, our algorithm takes g as the (q, q')-element of $\Delta^{\mathcal{H}}$. To answer the $\mathcal{G}$-EQ from $\Lambda^{(q,q')}$, we need to find a counterexample. This will be done by processing the MAT's answer to our EQ by Algorithm 5. Until then, answering the $\mathcal{G}$-EQ of $\Lambda^{(q,q')}$ is suspended.

After building a hypothesis SWFA, our algorithm asks an EQ to the MAT on the hypothesis. When the MAT replies with a counterexample c, Algorithm 5 starts a procedure to fix the hypothesis. Following Rivest and Schapire (1993) and Bisht et al. (2006), our algorithm finds a prefix ux of c, with $u \in \Sigma^*$ and $x \in \Sigma$, that satisfies the following

equations, where $S = \{s_1, s_2, \ldots, s_k\}$ (Line 1 of Algorithm 5):

$$(\mathrm{MQ}(us_1), \mathrm{MQ}(us_2), \ldots, \mathrm{MQ}(us_k)) = \boldsymbol{\alpha}^\mathsf{T} \Delta_u^{\mathcal{H}} \mathbf{H}_{(P,S)}\,, \tag{1}$$

$$(\mathrm{MQ}(uxs_1), \mathrm{MQ}(uxs_2), \ldots, \mathrm{MQ}(uxs_k)) \neq \boldsymbol{\alpha}^\mathsf{T} \Delta_{ux}^{\mathcal{H}} \mathbf{H}_{(P,S)}\,, \tag{2}$$

where $\mathrm{MQ}(w)$ is just $f_*(w)$, but we emphasize the learner obtains the values by MQs. Such a prefix ux always exists and can be found by binary search about the length $|c|$ of c, which requires $O(|S| \log |c|)$ MQs. This suggests that the transition relation $\Delta^{\mathcal{H}}$ of our current hypothesis has something wrong with the transition by x.

So, our algorithm computes $\Delta_x = \mathbf{H}_{(P,x,S)} \mathbf{H}_{(P,S)}^{-1}$ and compares it with $\Delta_x^{\mathcal{H}}$. If Δ_x and $\Delta_x^{\mathcal{H}}$ disagree on the (q, q')-element, we give x to $\Lambda^{(q,q')}$ as a counterexample to the $\mathcal{G}$-EQ from $\Lambda^{(q,q')}$. With the counterexample, our algorithm calls Algorithm 4 and restarts learning the guard function of $\Delta[q, q']$ using $\Lambda^{(q,q')}$. If $\Delta_x^{\mathcal{H}} = \Delta_x$, this means that the target power series f_* requires an SWFA with more states (Lemma 10). By processing the counterexample properly, our algorithm expands the observation table $\mathcal{T}$ (Lines 9-11 of Algorithm 5). Now, our algorithm decides to reconstruct the hypothesis from scratch because the goal hypothesis $\mathcal{A}_{(P,S)}$ has changed. Along with this, all Λ instances are discarded.

Algorithm 2: SWFA Learning Algorithm

Input: a MAT learning algorithm Λ
Output: an SWFA representing the target power series

```
1  P ← {c} and S ← {ε} for the counterexample c to the EQ on the zero SWFA;
2  initialize the observation table T = (P, S, H_(P,S)) using the MQ on c;
3  H ← null;
4  loop
5  |  if H is null then H ← build_hypothesis(T);   // Algorithm 3
6  |  ask an EQ on H;
7  |  if the MAT replies with a counterexample c then
8  |  |  H, T ← process_counterexample(H, T, c);   // Algorithm 5
9  |  else return H and terminate;
```

Algorithm 3: build_hypothesis

Input: $\mathcal{T} = (P, S, \mathbf{H}_{(P,S)})$
Output: a hypothesis SWFA

```
1  Q ← P; α^T ← H_({ε},S) H_(P,S)^{-1}; β ← H_(P,{ε});
2  H ← (G, Q, α, β, Δ^H), where Δ^H[q, q'] = null for all (q, q') ∈ Q^2;
3  for (q, q') ∈ Q^2 do
4  |  initialize the algorithm Λ^(q,q');
5  |  H ← update_transition(q, q', Λ^(q,q'), H, T);   // Algorithm 4
6  return H;
```

Algorithm 4: update_transition

Input: $q, q', \Lambda^{(q,q')}, \mathcal{H} = (\mathcal{G}, Q, \boldsymbol{\alpha}, \boldsymbol{\beta}, \Delta^{\mathcal{H}}), \mathcal{T} = (P, S, \mathbf{H}_{(P,S)})$
Output: the updated hypothesis

repeat
  $\Lambda^{(q,q')}$ asks a $\mathcal{G}$-MQ on $x \in \Sigma$;
  get $\mathbf{H}_{(\{q\},x,S)}[q, \cdot]$ by MQs on qxs for all $s \in S$;
  $\Delta_x[q, \cdot] \leftarrow \mathbf{H}_{(\{q\},x,S)}\mathbf{H}^{-1}_{(P,S)}$;
  answer the $\mathcal{G}$-MQ by $\Delta_x[q, q']$;
until $\Lambda^{(q,q')}$ asks a $\mathcal{G}$-EQ on a hypothesis guard function g;
$\Delta^{\mathcal{H}}[q, q'] \leftarrow g$;
return $(\mathcal{G}, Q, \boldsymbol{\alpha}, \boldsymbol{\beta}, \Delta^{\mathcal{H}})$;

Algorithm 5: process_counterexample

Input: $\mathcal{H} = (\mathcal{G}, Q, \boldsymbol{\alpha}, \boldsymbol{\beta}, \Delta^{\mathcal{H}}), \mathcal{T} = (P, S, \mathbf{H}_{(P,S)})$, a counterexample c
Output: a hypothesis SWFA (or null if $\mathcal{T}$ is updated), the observation table

1 find a prefix ux of c, where $u \in \Sigma^*, x \in \Sigma$, that satisfies Eqs. (1) and (2);
2 get $\mathbf{H}_{(P,x,S)}$ by MQs on pxs for all $p \in P$ and $s \in S$;
3 $\Delta_x \leftarrow \mathbf{H}_{(P,x,S)}\mathbf{H}^{-1}_{(P,S)}$;
4 **for** $(q, q') \in Q^2$ **do**
5 **if** $\Delta_x[q, q'] \neq \Delta^{\mathcal{H}}_x[q, q']$ **then**
6 give x to $\Lambda^{(q,q')}$ as a counterexample to the $\mathcal{G}$-EQ;
7 $\mathcal{H} \leftarrow$update_transition$(q, q', \Lambda^{(q,q')}, \mathcal{H}, \mathcal{T})$; *// Algorithm 4*
8 **if** $\mathcal{H}$ is updated **then return** $\mathcal{H}, \mathcal{T}$;
9 $P' \leftarrow P \cup \{u\}$;
10 $S' \leftarrow S \cup \{xs_i\}$ where $s_i \in S$ satisfies $\mathrm{MQ}(uxs_i) \neq \boldsymbol{\alpha}^{\mathsf{T}}\Delta^{\mathcal{H}}_{ux}\mathbf{H}_{(P,S)}[s_i]$;
11 expand $\mathbf{H}_{(P,S)}$ to $\mathbf{H}_{(P',S')}$;
12 **return** null, $(P', S', \mathbf{H}_{(P',S')})$;

5. Correctness and Query Complexity

The following lemma ensures that our mask (P, S) is always kept non-singular and $\mathrm{rank}(\mathbf{H}_{(P,S)})$ is strictly increasing at Line 11 of Algorithm 5.

Lemma 10 (Bisht et al. (2006)) *Suppose ux with $u \in \Sigma^*$ and $x \in \Sigma$ satisfies Eqs. (1) and (2). If $\Delta^{\mathcal{H}}_x = \Delta_x$, then $\mathrm{rank}(\mathbf{H}_{(P',S')}) = \mathrm{rank}(\mathbf{H}_{(P,S)}) + 1$ for $P' = P \cup \{u\}$ and $S' = S \cup \{xs_i\}$, where $\mathrm{MQ}(uxs_i) \neq \boldsymbol{\alpha}^{\mathsf{T}}\Delta^{\mathcal{H}}_{ux}\mathbf{H}_{(P,S)}[s_i]$.*

The proof by Bisht et al. requires no change for the case of SWFAs, since the lemma involves only finitely many elements of Σ.

By Lemma 10, our mask will finally converge to a minimal basis (P, S), for which we have $f_{\mathcal{A}_{(P,S)}} = f_*$. Then each $\Lambda^{(q,q')}$ will finally output the correct guard function.

In order to evaluate the query complexity of our algorithm, we first discuss how many $\mathcal{G}$-MQs and $\mathcal{G}$-EQs each MAT learner $\Lambda^{(q,q')}$ may make. We write the class of all linear

combinations of guard functions in the transition relation Δ of a minimal SWFA representing f_* by

$$\mathcal{G}_{f_*} = \left\{ \sum_{(q,q')\in Q^2} a_{qq'}\Delta[q,q'] \;\middle|\; a_{qq'} \in \mathbb{F} \text{ for each } (q,q') \in Q^2 \right\} \subseteq \mathcal{G}\,.$$

Note that the class $\mathcal{G}_{f_*}$ does not depend on the choice of the minimal SWFA, since any minimal SWFAs representing the same power series can be converted into each other by linear transformation as shown in the proof of Proposition 5. Let $\mathcal{M}$ and $\mathcal{E}$ be the number of $\mathcal{G}$-MQs and $\mathcal{G}$-EQs that Λ makes to learn guard functions in $\mathcal{G}_{f_*}$, respectively.

Theorem 11 *Let $n = \mathrm{rank}(\mathbf{H}_{f_*})$ and m be the length of the longest counterexample to EQs returned by the MAT. Then, the proposed algorithm returns an SWFA representing f_* using Λ after raising at most $O(n^3\mathcal{E})$ EQs and $O(n^4\mathcal{M} + n^4\mathcal{E}(n + \log m))$ MQs.*

Proof At first, we claim that the observation table $\mathcal{T} = (P, S, \mathbf{H}_{(P,S)})$ cannot be extended beyond n times. The table is extended on Line 11 of Algorithm 5 only when the condition of Lemma 10 is satisfied, and hence the rank of the sub-block $\mathbf{H}_{(P,S)}$ is definitely increased. By $\mathbf{H}_{(P,S)} \leq \mathrm{rank}(\mathbf{H}_{f_*}) = n$, the claim holds.

Whenever a counterexample to an EQ is given, we call Algorithm 5, except the first EQ for initializing $\mathcal{T}$. To count the number of EQs, we count how many times Algorithm 5 is performed. Each call of Algorithm 5 ends in either Line 8 or Line 12. The former happens only when a counterexample to an instance of Λ is found, which can happen at most $|Q|^2\mathcal{E} \leq n^2\mathcal{E}$ times without extending $\mathcal{T}$. The latter extends $\mathcal{T}$ and it is performed at most n times as shown before. Therefore, the algorithm builds a correct hypothesis after making at most $O(n^3\mathcal{E})$ EQs.

After the first MQ for initializing $\mathcal{T}$, our algorithm makes MQs for (a) constructing $\boldsymbol{\alpha}$ at Line 1 of Algorithm 3, (b) answering $\mathcal{G}$-MQs from instances of Λ at Line 3 of Algorithm 4, (c) finding a critical prefix of a counterexample at Line 1 of Algorithm 5, (d) answering $\mathcal{G}$-EQs by instances of Λ at Line 2 of Algorithm 5. Note that computing $\boldsymbol{\beta} = \mathbf{H}_{(P,\{\epsilon\})}$ at Line 1 of Algorithm 3 requires no extra MQs, since $\mathbf{H}_{(P,\{\epsilon\})}$ is a sub-block of $\mathbf{H}_{(P,S)}$. We can construct $\mathbf{H}_{(P',S')}$ at Line 11 of Algorithm 5 using results of MQs in (c) and (d).

The total number of MQs for (a) is at most n. Concerning (b), we make $|S|$ MQs for each $\mathcal{G}$-MQ from an instance of Λ. Thus, the number of MQs per an instance of Λ is at most $O(n\mathcal{M})$ by $|S| \leq n$. Each time $\mathcal{T}$ is extended, we make $|Q|^2 \leq n^2$ new instances of Λ, which happens n times. Thus, $O(n^4\mathcal{M})$ MQs are asked for (b) till the algorithm outputs a correct SWFA. For (c), we use $O(n \log m)$ MQs to find a critical prefix of each counterexample with binary search in Algorithm 5, which is called at most $O(n^3\mathcal{E})$ times, as we have argued for counting EQs. Thus, in total $O(n^4\mathcal{E} \log m)$ MQs are asked for (c). The total number of MQs for (d) is at most $O(n^5\mathcal{E})$, since we need $|P| \times |S| \leq n^2$ MQs for constructing $\mathbf{H}_{(P,x,S)}$ each time Algorithm 5 is called. All in all, the total number of MQs that our algorithm makes is at most $O(n^4\mathcal{M} + n^4\mathcal{E}(n + \log m))$. ■

We compare the query complexities of representative algorithms to learn relevant classes of automata under the MAT model in Table 1[2]. The query complexity of the proposed

2. $\hat{\mathcal{M}}$ and $\hat{\mathcal{E}}$ denote the numbers of $\mathcal{G}$-MQs and $\mathcal{G}$-EQs that predicate learners make in Argyros and D'Antoni's algorithm, respectively.

Table 1: Query complexities of representative learning algorithms for relevant automata

		Deterministic	Weighted
FA	EQ	$O(n)$	$O(n)$
	MQ	$O(n^2\lvert\Sigma\rvert + n \log m)$ (Rivest and Schapire, 1993)	$O(n^2\lvert\Sigma\rvert + n^2 \log m)$ (Bisht et al., 2006)
SFA	EQ	$O(n^3\hat{\mathcal{E}})$	$O(n^3\mathcal{E})$
	MQ	$O(n^4\hat{\mathcal{M}} + n^4\hat{\mathcal{E}} \log m)$ (Argyros and D'Antoni, 2018)	$O(n^4\mathcal{M} + n^4\mathcal{E}(n + \log m))$ (Ours)

algorithm is higher than that of the one for learning WFAs with respect to n, which is also true when extending FA to SFA. The important point is that the if $\mathcal{M}$ and $\mathcal{E}$ do not depend on alphabet size $|\Sigma|$, the number of MQs of our algorithm also does not depend $|\Sigma|$ unlike that of WFAs. Therefore, our algorithm is suitable for learning power series over an extremely large or infinite alphabet.

We show some examples of $\mathcal{G}$ and guard function learners.

Example 1 Guard functions in $\mathcal{G}^{\mathrm{wfa}}$ can be learned by $|\Sigma|$ $\mathcal{G}$-MQs and a $\mathcal{G}$-EQ, i.e., $\mathcal{M} = |\Sigma|$ and $\mathcal{E} = 1$. When learning SWFAs over $\mathcal{G}^{\mathrm{wfa}}$ with such a guard function learner, Algorithm 5 never calls Algorithm 4, because guard learners always raise the correct hypotheses. Then the number of required EQs goes down to $O(n)$. Still our algorithm makes more MQs than the WFA learner, where the same MQs are repeatedly made many times. We can remove such redundant MQs using memorization and then the number of MQs of our algorithm becomes as few as the WFA learner's.

Example 2 Let $\mathcal{G}^{\mathrm{poly}}$ be the class of all polynomials over the set of rational numbers $\mathbb{Q}$. Then a trivial learning strategy for $\mathcal{G}^{\mathrm{poly}}$ requires $k+1$ $\mathcal{G}$-MQs and $k+1$ $\mathcal{G}$-EQs for a target polynomial of degree k. Suppose that an SWFA representing f_* has guard functions of polynomials of degree at most k. Since the maximum degree of polynomials does not increase by linear transformation, we have $\mathcal{M} = \mathcal{E} = k + 1$.

Example 3 Let $\mathcal{G}^{\mathrm{div}}$ be the class of all guard functions $g\colon \mathbb{Z} \to \mathbb{F}$ which partition the integer set $\mathbb{Z}$ into finitely many intervals and assign an entity of $\mathbb{F}$ to each interval. That is, $g \in \mathcal{G}^{\mathrm{div}}$ has a partition number $k \in \mathbb{N}$, k borders $a_1, a_2, \ldots, a_k \in \mathbb{Z}$ and $k+1$ values $b_0, b_1, b_2, \ldots, b_k \in \mathbb{F}$ such that $a_1 < a_2 < \cdots < a_k$ and $g(x) = b_i$ if $a_i \le x < a_{i+1}$, assuming that $a_0 = -\infty$ and $a_{k+1} = \infty$. Since identifying each border a_i requires $O(\log |a_i|)$ $\mathcal{G}$-MQs, to learn g requires in total $O(\sum_{i=1}^{k} \log |a_i|)$ $\mathcal{G}$-MQs and $O(k)$ $\mathcal{G}$-EQs. Let A be the set of all borders a_i used in any of guard functions of an SWFA representing the target power series. Then a linear combination of those guard functions may produce the function $h \in \mathcal{G}^{\mathrm{div}}$ with borders A. Such h requires $\mathcal{M} = O(|A| \log \hat{a})$ $\mathcal{G}$-MQs and $\mathcal{E} = O(|A|)$ $\mathcal{G}$-EQs to learn, where $\hat{a} = \max\{\, |a| \mid a \in A \,\}$.

Example 4 We have shown that if $\mathcal{G} \subseteq \mathbb{F}^{\Sigma}$ is learnable, the class $\mathcal{G}'$ of $\mathcal{G}$-recognizable series is also learnable. Here, we can consider SWFAs over $\mathcal{G}' \subseteq \mathbb{F}^{\Sigma^*}$, whose edges have SWFAs over $\mathcal{G}$ as representations of guard functions. In this way, one can recursively obtain learnable classes of more complex formal power series.

6. Conclusion and Future Work

We considered symbolic weighted finite automata (SWFAs), which unify SFAs and WFAs. SWFAs can deal with a possibly infinite alphabet efficiently taking advantage of the structure of the alphabet. We proposed a new query learning algorithm for SWFAs by combining the ones for SFAs and WFAs with a correctness proof and an upper bound for the number of queries. We also confirmed that the minimization and equivalence checking for SWFAs can be achieved efficiently, as with WFAs under some additional assumptions.

There are many potential applications of our algorithm as extensions of applications using WFAs. An interesting application is extracting an SWFA from a recurrent neural network (RNN) to obtain a faster surrogate. A prior work (Okudono et al., 2020) uses WFAs for this purpose, but the range of applicable RNNs is limited by the size of the alphabet. SWFAs and our algorithm are suitable to extract automata from RNNs over a large or infinite alphabet. It is useful since RNNs with real-valued inputs are very common for time series analysis. Such an application requires some devices for our algorithm to work well with numerical errors. Even though our algorithm is in the context of exact concept learning, we observed that using a small threshold to relax the equivalence checking actually worked to some extent in small preliminary experiments. However, some additional devices would be required for larger experiments.

As shown in some recent papers (Bollig et al., 2009; Chubachi et al., 2019), the theoretical worst-case query complexity and the empirical performance are often different. Evaluating the practical efficiency of the proposed algorithm by experiments is left for future work. We estimate that depending on $\mathcal{G}$ and Λ, our algorithm can share a counterexample for multiple $\mathcal{G}$-EQs in Line 6 of Algorithm 5, which will improve the practical performance well.

Learning WFAs has been studied under different learning schemes, including *spectral learning* (see Balle and Mohri, 2015, and references therein). Learning SWFAs under those learning schemes is a hopeful research direction. On the other hand, there is a number of automata models related to or extending SWFAs, including those mentioned in the introduction of this paper. Designing algorithms for learning them will also be an interesting future work.

Acknowledgments

The work is supported in part by JSPS KAKENHI Grant Numbers 18H04091, 18K11449, 18K11150, 20H05703, and 21K11745.

References

Rajeev Alur, Konstantinos Mamouras, and Caleb Stanford. Automata-based stream processing. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*, volume 80 of *LIPIcs*, pages 112:1–112:15, 2017.

Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

George Argyros and Loris D'Antoni. The learnability of symbolic automata. In *30th International Conference on Computer Aided Verification, CAV 2018*, pages 427–445, 2018.

Borja Balle and Mehryar Mohri. Learning weighted automata. In *6th International Conference on Algebraic Informatics, CAI 2015*, pages 1–21, 2015.

Francesco Bergadano and Stefano Varricchio. Learning behaviors of automata from multiplicity and equivalence queries. *SIAM Journal on Computing*, 25(6):1268–1280, 1996.

Laurence Bisht, Nader H. Bshouty, and Hanna Mazzawi. On optimal learning algorithms for multiplicity automata. In *19th Annual Conference on Learning Theory, COLT 2006*, pages 184–198, 2006.

Benedikt Bollig, Peter Habermehl, Carsten Kern, and Martin Leucker. Angluin-style learning of NFA. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009*, pages 1004–1009, 2009.

Kaizaburo Chubachi, Diptarama Hendrian, Ryo Yoshinaka, and Ayumi Shinohara. Query learning algorithm for residual symbolic finite automata. In *Proceedings Tenth International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2019*, pages 140–153, 2019.

Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. L_p distance and equivalence of probabilistic automata. *International Journal of Foundations of Computer Science*, 18(4): 761–779, 2007.

Samuel Drews and Loris D'Antoni. Learning symbolic automata. In *23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2017*, pages 173–189, 2017.

Michel Fliess. Matrices de Hankel. *Journal de Mathématiques Pures et Appliquées*, 53, 1974.

Luisa Herrmann and Heiko Vogler. Weighted symbolic automata with data storage. In *20th International Conference on Developments in Language Theory , DLT 2016*, volume 9840 of *Lecture Notes in Computer Science*, pages 203–215, 2016.

Stefan Jaksic, Ezio Bartocci, Radu Grosu, Thang Nguyen, and Dejan Nickovic. Quantitative monitoring of STL with edit distance. *Formal Methods in System Design*, 53(1):83–112, 2018a.

Stefan Jaksic, Ezio Bartocci, Radu Grosu, and Dejan Nickovic. An algebraic framework for runtime verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2233–2243, 2018b.

Anil Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9:541–544, 1958.

Takamasa Okudono, Masaki Waga, Taro Sekiyama, and Ichiro Hasuo. Weighted automata extraction from recurrent neural networks via regression on state spaces. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 5306–5314, 2020.

Ronald L. Rivest and Robert E. Schapire. Inference of finite automata using homing sequences. *Information and Computation*, 103(2):299–347, 1993.

Marcel Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2-3):245–270, 1961.

Masaki Waga. Online quantitative timed pattern matching with semiring-valued weighted automata. In *17th International Conference on Formal Modeling and Analysis of Timed Systems, FORMATS 2019*, volume 11750 of *Lecture Notes in Computer Science*, pages 3–22, 2019.