THE LEARNABILITY OF SIMPLE DETERMINISTIC FINITE-MEMORY AUTOMATA VIA QUERIES

Sakamoto, Hiroshi
Graduate School of Information Science and Electrical Engineering, Department of Informatics, Kyushu University

https://doi.org/10.5109/13472

出版情報：Bulletin of informatics and cybernetics. 30 (1), pp.93-108, 1998-03. Research Association of Statistical Sciences
バージョン：
権利関係：

Bulletin of Informatics and Cybernetics, Vol. 30, No. 1, 1998

THE LEARNABILITY OF SIMPLE DETERMINISTIC FINITE-MEMORY AUTOMATA VIA QUERIES

By

Hiroshi SAKAMOTO *

Abstract

In this paper, we establish the learnability of simple deterministic finite-memory automata via membership and equivalence queries. Simple deterministic finite-memory automata are a subclass of finite-memory automata introduced by Kaminski and Francez (1994) as a model generalizing finite automata to infinite alphabets. Continuously, Sakamoto and Ikeda investigated several decision problems for finite-memory automata as well as for the deterministic class. By this result, we can arrive at a meaningful learning model for simple deterministic finite-memory automata. We provide the announced learning algorithm, show its correctness, and analyze its running time. The algorithm is partially based on Angluin's (1987) observation table. In particular, for every target and each finite alphabet Σ, the algorithm outputs a hypothesis that is consistent with the target over Σ. Finally, we obtain the main result of this paper, i.e., the class of simple deterministic finite-memory automata is exactly learnable via membership and equivalence queries.

1. Introduction

Active learning goes at least back to Shapiro (1983) who developed an algorithmic debugging system that uses variety of types of queries to the user to pinpoint errors in Prolog programs. Subsequently, Sammut and Banerji (1986) proposed a learning system using membership queries. Theoretical investigation of learning via queries started with Angluin's (1987) pioneering paper. In particular, Angluin showed the class of regular languages represented by deterministic finite automata (DFAs) to be learnable by using polynomially many membership and equivalence queries. During the last decade, various researchers continued along this line. Burago (1994) and Ishizaka (1989) extended Angluin's result, i.e., they showed that more powerful classes of languages including regular languages are learnable via membership and equivalence queries in polynomial time. Gavaldà (1993) investigated the learnability of concepts via equivalence queries only. Bergadano and Varricchio (1995) introduced new setting of learning regular languages,

* Graduate School of Information Science and Electrical Engineering, Department of Informatics, Kyushu University, Fukuoka 812-8581, Japan
Email: hiroshi@i.kyushu-u.ac.jp, **Phone:** +81-92-642-2697

and showed that regular languages are polynomially learnable via shortest counterexamples returned from equivalence queries. Sakakibara (1990) assumed a kind of data structure for context-free grammars. Consequently, the context-free grammars became polynomial time learnable via structural membership and structural equivalence queries. We aim to extend this line of research into a direction previously not considered, i.e., we study the identification of languages defined over an infinite alphabet. In actual applications of computational models, the data size is rapidly growing and the process of input data is controlled by natural numbers. Any practical device will have some bound on these numbers and such bound may be determined in some complicated, architecture-dependent way. Consequently, these restrictions have to be taken into account whenever the system on hand is used. Thus, more compact and useful model of computation is needed to ignore these bounds and to consider arbitrary natural numbers.

Recently, Kaminski and Francez (1994) introduced such a new model of computation, the so-called *finite-memory automata*, with the intention of expanding DFAs to infinite alphabets. Using a very restricted memory structure, finite-memory automata can deal with languages defined over infinite alphabets. However, Kaminski et al (1994) showed that when restricted to any finite alphabet, the computational power of finite-memory automata equals that of DFAs. Intuitively, this means the following. Consider an infinite sequence of finite alphabets, denoted by $\Sigma_1, \Sigma_2, \ldots$, such that $\Sigma_i \subset \Sigma_{i+1}$ and $||\Sigma_i|| = ||\Sigma_{i+1}|| + 1$. Then, corresponding to the sequence of alphabets, we can easily imagine an infinite sequence of regular languages, denoted by $L_1, L_2, \ldots$, such that $L_i \subset L_{i+1}$. Generally, each DFA M_i for L_i is greater than M_j for all $j < i$. This is caused by the fact that the size of an alphabet for a DFA is fixed in advance. However, the model of finite-memory automata can successfully describe the infinite number of regular languages by just one acceptor. More precisely, if all L_i have the same property, then they can be represented by a finite-memory automaton A such that $L(M_i) = \Sigma_i \cap L(A)$. Then, the most interesting feature of the model is the ability to represent DFAs in a more compact way, in case the alphabet is too large to be treated as if it is finite, in other words, by hiding the information what symbols construct an alphabet, the computational model of finite-memory automata becomes independent of the size of alphabets. Moreover, we should note that the representation of finite-memory automata is finite similarly to other natural computation models (e.g., DFAs, pushdown automata, and Turing machines).

In this paper, we introduce a subclass of finite-memory automata called *simple deterministic* finite-memory automata (simple DFMAs). The aim of this paper is to investigate the principle of learnability of languages defined over an infinite alphabet within Angluin's (1988) MAT-model, i.e., the learning problem of simple DFMAs via membership and equivalence queries. In order to solve this problem, first we establish reasonableness of the setting considered. Several decision problems for finite-memory automata were investigated by Sakamoto and Ikeda (1998). According to their results, the membership problem for finite-memory automata is NP-complete and it becomes P-complete in deterministic case. On the other hand, Sakamoto et al (1998) showed that the equivalence problem of deterministic case is decidable in PSPACE. Thus, it seems

to be reasonable to require the equivalence oracle, too. Next, we discuss the learnability of simple DFMAs using membership and equivalence queries. Our learning algorithm is partially based on Angluin's (1987) *observation table* technique. However, since the size of an alphabet is potentially infinite in our setting, it is impossible to directly apply her notion. But as it was mentioned above, the model of finite-memory automata is a natural extension of DFAs, i.e., for every finite alphabet and every finite-memory automaton, there exists a DFA consistent with the automaton over that alphabet. For the class of simple DFMAs, we show such a DFA to be computable using membership and equivalence queries by using an observation table. Furthermore, we also provide the method to transform the DFA to a simple DFMA, thereby preserving consistency. Consequently, for each finite alphabet and each target, the algorithm can compute a simple DFMA consistent with the target over that alphabet. As was shown by Sakamoto et al (1998), two finite-memory automata A and B are equivalent iff A and B are consistent over a finite alphabet of cardinality k, where k is a number dependent on A and B. Then, we conclude that our algorithm can learn every simple DFMA using membership and equivalence queries.

This paper is organized as follows. In the next section, basic definitions and notations are provided. The complete description and the correctness of the learning algorithm are provided in Section 3. In this section, we also estimate the running time of the algorithm. Open problems are discussed in the final section.

2. Preliminaries

Let $\mathbb{N}$ be the set of all natural numbers. An *alphabet* is a set of symbols. In particular, by $\Omega = \{a_i | i \in \mathbb{N}\}$, we denote an infinite, countable alphabet. Let Ω^* be the free monoid over Ω (Hopcroft and Ullman (1979)). The elements of Ω^* are called strings. Let $\alpha \in \Omega^*$ be a string; we use $|\alpha|$, $[\alpha]$ and $\alpha[i]$ to denote the *length*, the *range* and the i-th symbol of α, respectively, where the range of α is the set of symbols appearing in α. The unique string ε of length 0 is called the empty string. Let $w = \alpha\beta\gamma \in \Omega^*$. Then, β is said to be a *substring* of w. In particular, β is said to be a *proper* substring, a *prefix* and a *suffix* of w if $w \neq \beta$, $\alpha = \varepsilon$ and $\gamma = \varepsilon$, respectively. The set of all prefixes of α is denoted by $pre(\alpha)$. Let $n \in \mathbb{N}$; then we set $\Omega^n = \{\alpha \in \Omega^*,\ |\alpha| = n\}$. Moreover, we set $\Omega^+ = \Omega^* \setminus \{\varepsilon\}$. Now, let $\Sigma \subset \Omega$ be a finite alphabet. By $||\Sigma||$, we denote the cardinality of Σ. Any set $L \subseteq \Omega^*$ is called a *language*. A *class* of languages is a collection of languages containing at least one nonempty language.

For a mapping ψ, we write ψ^{-1} to denote the inverse of ψ, provided it exists. For an alphabet Σ, a mapping π is said to be a *permutation* over Σ if π is a one-to-one and onto mapping over Σ.

Let $\sharp$ be a special symbol not belonging to Ω. An *assignment* is a finite string $x_1 x_2 \cdots x_n \in (\Omega \cup \{\sharp\})^n$ such that if $x_i = x_j$ and $i \neq j$, then $x_i = \sharp$ for $1 \leq i, j \leq n$.

DEFINITION 2.1. (Kaminski and Francez (1994)) A *finite-memory automaton* is a 6-tuple $A = \langle Q, q_0, \mathfrak{u}, \varrho, \mu, F \rangle$, where Q is a finite set of *states*, $q_0 \in Q$ is the *initial state*, $\mathfrak{u} \in (\Omega \cup \{\sharp\})^k$ is an assignment of length k called the *initial assignment*, ϱ is a mapping:

$Q \mapsto \{1, 2, \ldots, k\}$, $\mu \subseteq Q \times \{1, 2, \ldots, k\} \times Q$ is the *transition relation*, and $F \subseteq Q$ is the set of *final states*.

For a finite-memory automaton A defined as above, every pair of a state and an assignment is called a *configuration*. In particular, the pair $(q_0, \mathfrak{u})$ is referred to as the *initial* configuration. All configurations with final states are called *final* configurations.

We define a relation $\vdash$ as follows: Let $\mathfrak{u}_1 = x_1 x_2 \cdots x_k$ and $\mathfrak{u}_2 = y_1 y_2 \cdots y_k$ be assignments, and let $p, q \in Q$. Then, $(p, \mathfrak{u}_1) \vdash (q, \mathfrak{u}_2)$ iff there exists an $a \in \Omega$ such that:

(a) $a = x_i$ for some $1 \leq i \leq k$, $\mathfrak{u}_1 = \mathfrak{u}_2$, and $(p, i, q) \in \mu$, or

(b) $a \notin [\mathfrak{u}_1]$ and there exists a $j \in \{1, 2, \ldots, k\}$ such that $\varrho(p) = j$, $\mathfrak{u}_2[j] = a$ and for all $i \in \{1, 2, \ldots, k\} \backslash \{j\}$, $\mathfrak{u}_1[i] = \mathfrak{u}_2[i]$ and $(p, j, q) \in \mu$.

Intuitively, when reading a symbol a the finite-memory automaton A may change its actual configuration $(p, \mathfrak{u}_1)$ as follows. If a is equal to the i-th symbol of $\mathfrak{u}_1$ and $(p, i, q) \in \mu$, then A may change its configuration to $(q, \mathfrak{u}_2)$. If $a \notin [\mathfrak{u}_1]$, then $\mathfrak{u}_1[\varrho(p)]$ is replaced by a and all $\mathfrak{u}_1[i]$ for $i \neq \varrho(p)$ remain unchanged resulting into the new assignment $\mathfrak{u}_2$. Now, A may change its configuration to $(q, \mathfrak{u}_2)$ provided $(p, \varrho(p), q) \in \mu$.

When necessary, we write $(p, \mathfrak{u}_1) \vdash^a (q, \mathfrak{u}_2)$ by specifying the symbol a. The reflexive, transitive closure of $\vdash$ is denoted by $\vdash^*$. We say that A *accepts* $w = w_1 w_2 \cdots w_n \in \Omega^*$ ($w_i \in \Omega$, $1 \leq i \leq n$) if there exists a sequence of configurations $c_0, c_1, \ldots, c_n$ such that c_0 is the initial configuration, c_n is a final configuration and $c_{i-1} \vdash^{w_i} c_i$ for each $1 \leq i \leq n$. The language accepted by A is denoted by $L(A)$.

Let A be a finite-memory automaton and $w \in \Omega^*$. Then, we use $A(w)$ to denote a state that A has reached after processing w. Analogously, we write $\mathfrak{u}_w$ to denote an assignment of A after having read w. That is, $(A(w), \mathfrak{u}_w)$ denotes a configuration of A such that $(q_0, \mathfrak{u}) \vdash^w (A(w), \mathfrak{u}_w)$.

Let A and B be simple DFMAs. We say that A and B are *consistent* over an alphabet Σ if for every string $w \in \Sigma^*$, $w \in L(A)$ iff $w \in L(B)$. This is denoted by $A \simeq_\Sigma B$. In particular, when the alphabet is Ω, we say that A and B are *equivalent*, denoted by $A \simeq B$. We also use the same notation for DFAs, that is, DFAs M and M' are consistent over Σ, denoted by $M \simeq_\Sigma M'$, if for every string $w \in \Sigma^*$, $w \in L(M)$ iff $w \in L(M')$. The following proposition tells that the classes of finite-memory automata and finite automata are equivalent if a finite alphabet is fixed.

PROPOSITION 2.2. *(Kaminski and Francez (1994)) For every regular language L on Σ, there exists a finite-memory automaton A such that $L = L(A) \cap \Sigma^*$. Moreover, for every finite-memory automaton A and each finite alphabet Σ, the language $L(A) \cap \Sigma^*$ is regular.*

The definition above is nondeterministic in the sense that there may exist two or more computations for a string. The deterministic model is obtained by restricting ϱ and δ to total functions as follows.

DEFINITION 2.3. (Kaminski and Francez (1994)) A finite-memory automaton is said to be *deterministic* if for each $p \in Q$ and each $1 \leq i \leq k$, $\varrho(p)$ is defined and there exists exactly one $q \in Q$ such that $(p, i, q) \in \mu$.

In this paper, we introduce another class of finite-memory automata, called *simple*. Intuitively, a simple finite-memory automaton is considered that each window of the initial assignment is initialized by the blank symbol $\sharp$.

DEFINITION 2.4. A finite-memory automaton $A = \langle Q, q_0, \mathfrak{u}, \varrho, \mu, F \rangle$ is said to be *simple* if its initial assignment contains no symbol in Ω, that is, $\mathfrak{u} \in \{\sharp\}^*$.

We briefly call a deterministic finite-memory automata 'DFMA' below. The following propositions are useful for further discussions within this paper.

PROPOSITION 2.5. *(Kaminski and Francez (1994)) Let $A = \langle Q, q_0, \mathfrak{u}, \varrho, \mu, F \rangle$ be a finite-memory automaton and π a permutation on Ω which is identical for symbols in $[\mathfrak{u}]$. Then, $\pi(L(A)) = L(A)$, where $\pi L(A) = \cup\{\pi(w) | w \in L(A)\}$.*

Since an initial assignment of a simple DFMA does not contain any symbol from Ω, this proposition implies that the class of languages accepted by simple DFMAs is closed under permutations.

Similarly to the case of finite automata, a finite-memory automaton can be described as a directed graph whose nodes denote states. There is an edge from nodes s_i to s_j if there exists a number k such that a relation (s_i, k, s_j) is defined in the transition relation. Such an edge is labeled k. Also, if for a node s, the value of ϱ is defined, then s is labeled $\varrho(s)$.

We exemplify the graph representation of a finite-memory automaton (cf. Fig. 1.). The three nodes denote states q_0, q_1 and q_2 from the left. Each label of the nodes denotes the mapping ϱ, that is, $\varrho(q_0) = 1$ and so on. Each label of edges denotes the transition relation μ, for example, the edge from q_0 to q_1 means $(q_0, 1, q_1), (q_0, 2, q_1) \in \mu$. The initial assignment is $\sharp\sharp$ and the final state is q_2. The string aba is accepted by the computation $(q_0, \sharp\sharp) \vdash^a (q_1, a\sharp) \vdash^b (q_1, ab) \vdash^a (q_2, ab)$. The finite-memory automaton accepts the language $\{\alpha\beta\alpha | \alpha \in \Omega, \beta \in \Omega^*\}$. A corresponding DFA on $\Sigma = \{a, b\}$ is also shown in Fig. 1.

In this paper, we assume two oracles, called *membership* and *equivalence queries*. A membership oracle answers the question, *given a simple DFMA A and a string w, whether or not $w \in L(A)$*. An equivalence oracle answers the question, *given simple DFMA A and B, whether or not $L(A) = L(B)$*. Additionally, an equivalence oracle returns a *counterexample* in $L(A) \triangle L(B)$ if not, where $L(A) \triangle L(B)$ is the symmetric difference of $L(A)$ and $L(B)$.

Since we can simulate a computation of a finite-memory automaton for any string by a standard algorithm, the membership problem is decidable. However, it is not trivial whether or not the equivalence problem for finite-memory automata is decidable. Then, we must confirm the soundness of our setting.

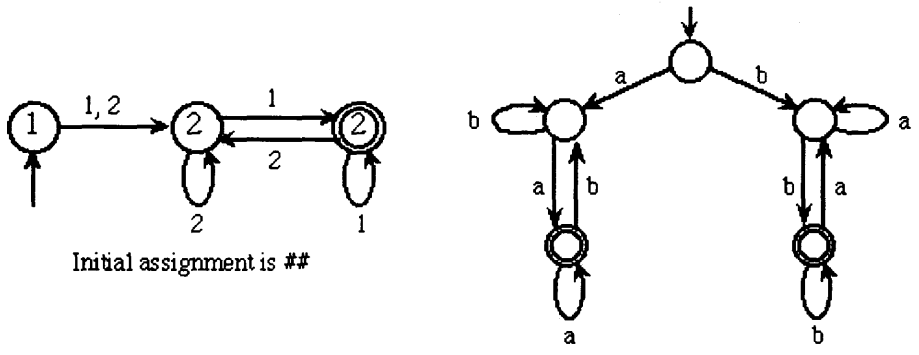

Figure 1: A finite-memory automaton and a corresponding DFA

Recently, Sakamoto and Ikeda (1998) investigated several decision problems for finite-memory automata. In particular, the problem whether two finite-memory automata are equivalent is shown to be PSPACE-complete. Thus, the equivalence problem for finite-memory automata is decidable. However, the completeness of deterministic case is open.

THEOREM 2.6. *(Sakamoto and Ikeda (1998)) The membership problem for finite-memory automata is NP-complete and it is P-complete if the automata are deterministic. The equivalence problem for deterministic finite-memory automata is decidable in PSPACE.*

3. Learnability of simple DFMAs

Our learning algorithm contains two procedures, denoted by $\mathcal{ANG}$ and $\mathcal{CONS}$. Their complete descriptions are given in Fig. 2 and 3, respectively. First, we analyze the procedures and show their correctness.

3.1. The procedure $\mathcal{ANG}$ and its correctness

Angluin (1987) introduced the notion of an *observation table*. An observation table for a DFA M on Σ is denoted by a triplet (S, E, T), where S (resp., E) is a nonempty, finite, and *prefix-closed* (resp., *suffix-closed*) set of strings from Σ (a set is prefix-closed iff each prefix of each element of the set is in the set, and suffix-closed is defined analogously) and T is a mapping from $(S \cup S \cdot \Sigma) \cdot E$ to $\{0,1\}$ such that $T(w) = 1$ if $w \in L(M)$ and $T(w) = 0$, otherwise. By $row(s)$, we denote $f : E \mapsto \{0,1\}$ defined by $f(e) = T(s \cdot e)$ for $s \in (S \cup S \cdot \Sigma)$. An observation table is said to be *closed* if for each $t \in S \cdot \Sigma$, there exists an $s \in S$ such that $row(t) = row(s)$, and is said to be *consistent* if for each $a \in \Sigma$ and $s_1, s_2 \in S$ such that $row(s_1) = row(s_2)$, it holds that $row(s_1 \cdot a) = row(s_2 \cdot a)$.

If (S, E, T) is a closed, consistent observation table, then we define a corresponding

Input: an observation table (S, E, T) on Σ
Begin
  While($M := M(S, E, T)$ is not permutation closed) **do**
    Find $w \in L(M) \triangle L(M_{a|b})$ for some $a, b \in \Sigma$.
    Let $S := S \cup \{pre(w)\}$ and extend T to $(S \cup S \cdot \Sigma) \cdot E$
    using membership queries.
    Compute a closed, consistent observation table (S, E, T).
  End While
  Output M and halt.
End

Figure 2: Procedure $\mathcal{ANG}$

DFA $M(S, E, T)$ on Σ such that $Q = \{row(s) | s \in S\}$, the set of states, $q_0 = row(\varepsilon)$, the initial state, $F = \{row(s) | s \in S, T(s) = 1\}$, the set of final states, and $\delta(row(s), a) = row(s \cdot a)$, the transition relation.

Next, we show that $M(S, E, T)$ is well-defined. Since S is a nonempty prefix-closed set, it must contain ε and hence q_0 is defined. Also, since E is a nonempty suffix-closed set, it must contain ε. Thus, if s_1 and s_2 are elements of S such that $row(s_1) = row(s_2)$, then $T(s_1 \cdot \varepsilon) = T(s_1) = T(s_2) = T(s_2 \cdot \varepsilon)$ and hence F is well-defined. To see that δ is well-defined, suppose that s_1 and s_2 are elements of S such that $row(s_1) = row(s_2)$. Then, since the observation table (S, E, T) is consistent, for each $a \in \Sigma$, $row(s_1 \cdot a) = row(s_2 \cdot a)$, and since it is closed, this common value is equal to $row(s)$ for some $s \in S$.

For any further details, the reader is referred to Angluin (1987). In particular, the following proposition is useful for our discussion.

PROPOSITION 3.1. *(Angluin (1987)) If (S, E, T) is a closed, consistent observation table, then the DFA $M(S, E, T)$ is consistent with the finite function T. Any other DFA consistent with T but inequivalent to $M(S, E, T)$ must have more states.*

The procedure $\mathcal{ANG}$ takes as input an observation table. It computes a closed, consistent observation table using membership queries. When such a table is obtained, it makes a corresponding DFA M. Next, it decides whether or not M satisfies the following notion.

DEFINITION 3.2. A DFA M over Σ is said to be *permutation closed* if for every permutation $\pi : \Sigma \mapsto \Sigma$, $L(M) = \pi(L(M))$, where $\pi(L(M)) = \cup_{w \in L(M)} \{\pi(w)\}$.

Let M be any DFA over Σ. Then, there exists a finite-memory automaton $A = \langle Q, \mathfrak{u}, q_0, \varrho, \mu, F \rangle$ such that $M \simeq_\Sigma A$ (Kaminski and Francez (1994)). Let $\ell : \Omega \cup \{\sharp\} \mapsto \Omega \cup \{\sharp\}$ be an automorphism such that $\ell(\sharp) = \sharp$. Then, for each $w \in \Omega^*$, $w \in L(A)$ iff $\ell(w) \in L(A)$ (Kaminski and Francez (1994)). Since each initial assignment of each

simple DFMA has no symbol in Ω, we have that M is permutation closed on Σ iff there exists a simple DFMA A such that $M \simeq_\Sigma A$. We prove that it is decidable whether or not M is permutation closed over Σ.

DEFINITION 3.3. Let $M = \langle Q, \Sigma, q_0, \delta, F\rangle$ be a DFA. Define $M_{a|b} = \langle Q, \Sigma, q_0, \delta_{a|b}, F\rangle$ such that $\delta_{a|b}(p,c) = \delta(p,a)$ if $c = b$, or $\delta_{a|b}(p,c) = \delta(p,b)$ if $c = a$, or $\delta_{a|b}(p,c) = \delta(p,c)$ otherwise.

For M and $k \geq 1$, let us define a set of DFAs, denoted by M^k, recursively as: $M^1 = \{M_{a|b} | a, b \in \Sigma\}$ and $M^{k+1} = \{M'_{a|b} | M' \in M^k, a, b \in \Sigma\}$. Then, clearly, M is permutation closed iff $M \simeq M'$ for each $M' \in M^k$ and each $k \geq 1$. We say that M is *k-permutation closed* if $M \simeq M'$ for each $M' \in M^k$.

LEMMA 3.4. *M is 1-permutation closed iff M is permutation closed.*

PROOF. The left direction is trivial. Assume that for all $M' \in M^1$, $M \simeq M'$. This assertion is shown by induction on k for M^k. Let $M' \in M^2$. Then, there exists an $M_i \in M^1$ such that $M' = (M_i)_{a|b}$ for some $a, b \in \Sigma$. By the definition, $w \in L(M_i)$ iff $\pi(w) \in L(M')$, where π is the permutation $\{a \mapsto b, b \mapsto a\}$. On the other hand, there exists $M_j \in M^1$ such that $M_j = M_{a|b}$. Analogously, $w \in L(M)$ iff $\pi(w) \in L(M_j)$. By the assumption, $M_i \simeq M_j$. It follows that $\pi(w) \in L(M_i)$. Thus, $w \in L(M_i)$ iff $\pi(w) \in L(M_i)$ iff $\pi(\pi(w)) = w \in L(M')$. Thus, $M' \simeq M$ because $M_i \simeq M$. It follows that M is 2-permutation closed.

Assuming the induction hypothesis on some $k \geq 2$, we can similarly prove that for each $M' \in M^{k+1}$, $M \simeq M'$. Thus, we conclude that M is permutation closed. □

LEMMA 3.5. *The procedure $\mathcal{ANG}$ stops and outputs a permutation closed DFA.*

PROOF. Let (S, E, T) be an input for a finite alphabet Σ. Since no equivalence query is used in $\mathcal{ANG}$, Σ is fixed in the execution of $\mathcal{ANG}$. Let $||\Sigma|| = n$. There exists a minimum DFA M_* over Σ that is consistent with the target over Σ. Let m_* be the number of states of M_*.

After computing the DFA $M := M(S, E, T)$ for a closed, consistent observation table, assume that $M \not\simeq_\Sigma M'$ for some $M' \in M^1$. Let m be the number of states of M. Since M is a minimum DFA that is consistent with T, there are at least m counterexamples for $M \simeq M'$. Let $M' = M_{a|b}$ for some $a, b \in \Sigma$.

When we extend S, E and T, a next closed, consistent observation table (S, E, T) is computed. Let $M := M(S, E, T)$. Again, M is a minimum DFA consistent with T. By adding a counterexample, at least one new entry comes into (S, E, T), that is, the number of states of M is greater than m.

When $m < m_*$, after found at most $m_* n(n-1)/2$ counterexamples, we computes a DFA M such that $M \simeq_\Sigma M_{a|b}$ for all $a, b \in \Sigma$, because $||M^1|| = n(n-1)/2 + 1$ (M itself is in M^1). When $m \geq m_*$, clearly, $M \simeq_\Sigma M_*$, that is, M is permutation closed. Hence, $\mathcal{ANG}$ eventually terminates and outputs a permutation closed DFA for any input. □

Input: a permutation closed DFA $M = \langle Q, \Sigma, q_0, \delta, F \rangle$, where $||\Sigma|| = n$.
Begin
Set $\mathfrak{u} := \sharp^n$, $(p, i, q) \in \mu$ iff $\delta(p, a_i) = q$ and $\varrho = \emptyset$.
For $(j = 1; j \leq m; j++)$ **do**
If there exists $p \in Q$ such that $\delta(q_0, w) = p$ for some $w \in (\Sigma \setminus \{a_j\})^*$ and $\varrho(p)$ is not defined in ϱ, **then**
Let $\varrho := \varrho \cup \{\varrho(p) = j\}$.
End If
End For
Output $A := \langle Q, \mathfrak{u}, q_0, \varrho, \mu, F \rangle$.
End

Figure 3: Procedure *CONS*

3.2. The procedure *CONS* and its correctness

The procedure *CONS* takes as input a permutation closed DFA M on Σ and computes a simple DFMA A from M. A permutation closed DFA M is shown in Fig. 4. Using this DFA, let us demonstrate the computation of *CONS* . First, a DFA M' on the alphabet $\{1, 2\}$ is directly obtained from M by replacing labels a and b with 1 and 2, respectively. Taking account of the permutation $\ell : \{a \mapsto 1, b \mapsto 2\}$, we can identify M and M'. Starting with the initial state of M, if a state p is reachable with reading no 1, then we assign $\varrho(p) = 1$ for all such states. After that, all other states are assigned 2 because they are reachable with reading no 2. Including the initial assignment $\sharp\sharp$, a simple DFMA A in Fig. 4 is computed. Note that A is consistent with M.

The aim of this subsection is to prove that $M \simeq_\Sigma A$. For this purpose, we introduce the following notion concerned with a DFA $M = \langle Q, \Sigma, q_0, \delta, F \rangle$, where $p, q \in Q$.

DEFINITION 3.6. Let $W_p = \{w \in \Sigma^* | \delta(q_0, w) = p\}$ and $W_q = \{w \in \Sigma^* | \delta(q_0, w) = q\}$. Then, we write $p \equiv_\ell q$ if there exists a one-to-one and onto mapping $\ell : \Sigma \mapsto \Sigma$ such that $W_q = \ell(W_p) = \cup_{w \in W_p} \{\ell(w)\}$.

LEMMA 3.7. *If* $p \not\equiv_\ell q$, *then* $\ell(W_p) \cap W_q = \emptyset$.

PROOF. By Proposition 3.1, we can assume that M is a minimum DFA. Suppose that there exist $w_1, w_2 \in W_p$ such that $\ell(w_1) \in W_q$ and $\ell(w_2) \in W_{q'}$ for some $\ell : \Sigma \mapsto \Sigma$, where $q \neq q'$. For each $w \in \Sigma^*$, $\delta(q_0, w_1 w) \in F$ iff $\delta(q_0, w_2 w) \in F$. Since $\ell(L(M)) = L(M)$, we have that, for each $w \in \Sigma^*$, $\delta(q_0, \ell(w_1 w)) \in F$ iff $\delta(q_0, \ell(w_2 w)) \in F$, that is, $\delta(q, \ell(w)) \in F$ iff $\delta(q', \ell(w)) \in F$. Since ℓ is an automorphism, $\cup_{w \in \Sigma^*} \ell(w) = \Sigma^*$. Thus, for each $w \in \Sigma^*$, $\delta(q, w) \in F$ iff $\delta(q', w) \in F$. Note that M is minimum. Contradiction. □

For a string $w \in \Sigma^*$, let $\delta(q_0, w) = p$ and $A(w) = q$. We can regard q as a state of M. Since $p \equiv_\ell q$ implies $p, q \in F$ or $p, q \in Q \setminus F$, in order to prove that $M \simeq_\Sigma A$, it is sufficient to show that for each $w \in \Sigma^*$, $\delta(q_0, w) \equiv_\ell A(w)$.

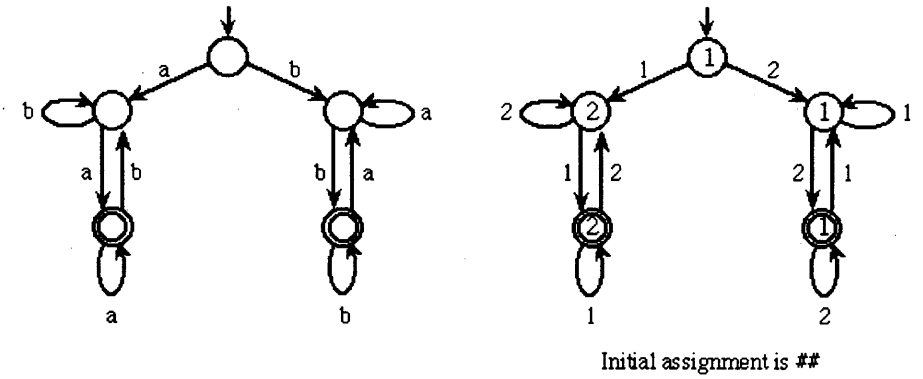

Figure 4: A permutation closed DFA and a corresponding simple DFMA

LEMMA 3.8. *Let M be an output of $\mathcal{ANG}$ and A an output of $\mathcal{CONS}$ for the input M. Then, $A \simeq_\Sigma M$, where M is defined on Σ.*

PROOF. This proof is done by induction on the length of $w \in \Sigma^*$. For a string w of length m such that $w = w_1 w_2 \cdots w_m$, let $(p_{i-1}, \mathfrak{u}_{i-1}) \vdash^{w_i} (p_i, \mathfrak{u}_i)$ for $(p_{i-1}, j_i, p_i) \in \mu$ $(1 \leq i \leq m)$, where $(p_0, \mathfrak{u}_0) = (q_0, \mathfrak{u})$. Define a string $w_\lambda = \lambda^{-1}(j_1 j_2 \cdots j_m) \in \Sigma^m$, where $\lambda = \{a_i \mapsto i | a_i \in \Sigma\}$.

We first show that $A(w) = A(w_\lambda)$ for $w \in \Sigma^+$. This is true for each $a \in \Sigma$. Then, assume $A(w) = A(w_\lambda)$ on each $w \in \Sigma^n$, that is, $(p_{i-1}, \mathfrak{u}'_{i-1}) \vdash^{\lambda^{-1}(j_i)} (p_i, \mathfrak{u}'_i)$, where $\mathfrak{u}'_0 = \mathfrak{u}$.

For $a \in \Sigma$, if $a = \mathfrak{u}_w[k]$, then there exists $1 \leq i \leq m$ such that $(p_{i-1}, \mathfrak{u}_{i_1}) \vdash^a (p_i, \mathfrak{u}_i)$. By the induction hypothesis, $(p_{i-1}, \mathfrak{u}'_{i_1}) \vdash^{\lambda^{-1}(k)} (p_i, \mathfrak{u}'_i)$. Thus, if $(p_m, \mathfrak{u}_m) \vdash^a (p_{m+1}, \mathfrak{u}_{m+1})$ for some $(p_m, k, p_{m+1}) \in \mu$, then $(p_m, \mathfrak{u}'_m) \vdash^{\lambda^{-1}(k)} (p_{m+1}, \mathfrak{u}'_{m+1})$. Otherwise, $\varrho(p_m) = k$ and $(p_m, \mathfrak{u}_m) \vdash^a (p_{m+1}, \mathfrak{u}_{m+1})$, where $(p_m, k, p_{m+1}) \in \mu$. By the induction hypothesis, $\lambda^{-1}(k) \notin [\mathfrak{u}'_m]$. Thus, $(p_m, \mathfrak{u}'_m) \vdash^{\lambda^{-1}(k)} (p_{m+1}, \mathfrak{u}'_{m+1})$. Hence, $A(wa) = A((wa)_\lambda)$.

Next, we show that for each $w \in \Sigma^*$, $\delta(q_0, w) \equiv_\ell A(w)$. The base step is clear. Then, assume that $\delta(q_0, w) = p$, $A(w) = p'$ and $p \equiv_\ell p'$. Since $A(w) = A(w_\lambda) = \delta(q_0, w_\lambda) = p'$, there exists $\ell^{-1}(w_\lambda) \in \Sigma^n$ such that $\delta(q_0, \ell^{-1}(w_\lambda)) = p$. We denote $(wa)_\lambda = w_\lambda a_\lambda$.

By Lemma 3.7, we note that if $p \equiv_\ell q$, then for each $a \in \Sigma$, $\delta(p, a) \equiv_\ell \delta(q, a)$. If $a \in [\ell^{-1}(w_\lambda)]$, then $\ell(\ell^{-1}(w_\lambda)a) = w_\lambda a_\lambda = (wa)_\lambda$. Thus, by lemma 3.7, $\delta(q_0, wa) = \delta(q_0, \ell^{-1}(w_\lambda)a) \equiv_\ell A((wa)_\lambda) = A(wa)$.

If $a \notin [\ell^{-1}(w_\lambda)]$, then $a_\lambda \notin [w_\lambda]$. Thus, there exists $\ell' = (\ell \setminus \{a \mapsto b\}) \cup \{a \mapsto a_\lambda\}$ for some $b \in \Sigma$ such that $\ell'(\ell^{-1}(w_\lambda a)) = w_\lambda a_\lambda = (wa)_\lambda$. Thus, by Lemma 3.7, $\delta(q_0, wa) = \delta(q_0, \ell^{-1}(w_\lambda)a) \equiv_{\ell'} A(w_\lambda a_\lambda) = A(wa)$. Hence, this proof is completed. □

3.3. The algorithm $\mathcal{DFMA}$ and its correctness

The aim of this subsection is to show that for any simple DFMA A and B, there exists a number k depending on A and B such that $A \simeq_\Sigma B$ for a finite alphabet Σ of cardinality k implies $A \simeq B$. First, we consider reflecting a string $w \in \Omega^*$ to a string $w' \in \Sigma^*$ such that the transition of states of A on w is preserved. That is, we propose a method computing for every $w \in \Omega^*$ a string $w' \in \Sigma^*$ such that $A(w) = A(w')$ for a sufficiently large but finite Σ. In order to formalize this idea, we need the following definition.

DEFINITION 3.9. Let $A = \langle Q, \mathbf{u}, q_0, \varrho, \mu, F\rangle$ be a simple DFMA and Σ a finite alphabet of cardinality $k = |\mathbf{u}|$. Then, for $w \in \Omega^+$, we define sets w^A_Σ recursively as follows: For each $a \in \Omega$, $a^A_\Sigma = \Sigma$, and for each $w \in \Omega^+$ and $a \in \Omega$, $(wa)^A_\Sigma = \bigcup_{\substack{w' \in w^A_\Sigma \\ a' \in \Sigma}} \{w'a'\}$, where

$$\begin{cases} a' = \mathbf{u}_{w'}[i], & \text{if } a = \mathbf{u}_w[i] \text{ or } a \notin [\mathbf{u}_w],\ a = \mathbf{u}_{wa}[i] \text{ and } \sharp \notin [\mathbf{u}_w] \\ a' = \mathbf{u}_{w'}[i] \text{ or } a' \in \Sigma \setminus [\mathbf{u}_{w'}], & \text{if } a \notin [\mathbf{u}_w],\ a = \mathbf{u}_{wa}[i],\ \mathbf{u}_w[i] \neq \sharp \text{ and } \sharp \in [\mathbf{u}_w] \\ a' \in \Sigma \setminus [\mathbf{u}_{w'}], & \text{if } a \notin [\mathbf{u}_w],\ a = \mathbf{u}_{wa}[i],\ \mathbf{u}_w[i] = \sharp. \end{cases}$$

For the reader's understanding, we show an application of this notion taking the example in Fig. 1. Note that this is a simple DFMA (denoted by A). Since $|\mathbf{u}| = 2$, set $\Sigma = \{a_1, a_2\}$. Now, $a_5a_6a_{11}a_6 \in \Omega^+$. By definition, $a_1 \in (a_5)^A_\Sigma = \Sigma$. Since $\mathbf{u}_{a_5} = a_5\sharp$, $\mathbf{u}_{a_5a_6} = a_5a_6$ and $\mathbf{u}_{a_1} = a_1\sharp$, this case corresponds to the third condition of the definition. Then, $a_2 \in \Sigma \setminus [\mathbf{u}_{a_1}]$ follows $a_1a_2 \in (a_5a_6)^A_\Sigma$. Since $\mathbf{u}_{a_5a_6} = a_5a_6$, $\mathbf{u}_{a_5a_6a_{11}} = a_5a_{11}$ and $\mathbf{u}_{a_1a_2} = a_1a_2$, this case corresponds to the first condition of the definition. Then, $a_2 = \mathbf{u}_{a_1a_2}[2]$ follows $a_1a_2a_2 \in (a_5a_6a_{11})^A_\Sigma$. Similarly, $a_1a_2a_2a_2 \in (a_5a_6a_{11}a_6)^A_\Sigma$.

On this simple DFMA A, the first symbol of every input is substituted in the first window of the assignment. Once A reads other symbols, they are substituted in the second window. Thus, there is no computation such as $(p, a\sharp) \vdash^b (q, b\sharp)$. Hence, in this example, the second condition of the definition does not happen.

On the other hand, the string $a_2a_1a_1a_1$ is also in $(a_5a_6a_{11}a_6)^A_\Sigma$ and the two strings $a_1a_2a_2a_2$ and $a_2a_1a_1a_1$ can simulate the computation $(q_0, \sharp\sharp) \vdash^{a_5a_6a_{11}a_6} (q_1, a_5a_6)$. That is, $a_5a_6a_{11}a_6 \in L(A)$ iff $a_1a_2a_2a_2, a_2a_1a_1a_1 \in L(A)$. Next, we formalize the property of strings in w^A_Σ.

LEMMA 3.10. *Let $w \in \Omega^n$ and $w' \in w^A_\Sigma$, where the length of the initial assignment of A is k. Then, for each $1 \leq i \leq k$, $\mathbf{u}_w[i] = \sharp$ iff $\mathbf{u}_{w'}[i] = \sharp$.*

PROOF. The proof is done by induction on the length of $w \in \Omega^+$. For the induction basis, let $a \in \Omega$. Then, $a^A_\Sigma = \Sigma$ and since $\mathbf{u} \in \{\sharp\}^*$, it is also true that for each $a' \in a^A_\Sigma$, $\mathbf{u}_a[i] = \sharp$ iff $\mathbf{u}_{a'}[i] = \sharp$. Assume the induction hypothesis for each $w \in \Omega^n$. Let $w' \in w^A_\Sigma$. Taking a symbol $a \in \Omega$, consider $(wa)^A_\Sigma$.

In case $a = \mathbf{u}_w[i]$, or $a \notin [\mathbf{u}_w]$, $a = \mathbf{u}_{wa}[i]$ and $\sharp \notin [\mathbf{u}_w]$ (the first case of Definition 3.9), there exists $\alpha, \alpha b \in pre(w)$ such that $\mathbf{u}_\alpha[i] = \sharp$ and $\mathbf{u}_{\alpha b}[i] = b$. Let $\alpha'b' \in (\alpha b)^A_\Sigma$. By the induction hypothesis, $\mathbf{u}_\alpha[i] = \sharp$ implies $\mathbf{u}_{\alpha'}[i] = \sharp$ as well as $\mathbf{u}_{\alpha b}[i] = b$ implies

$\mathbf{u}_{\alpha'b'}[i] = b'$. Thus, $\mathbf{u}_{w'}[i] \neq \sharp$. By the definition, $w'a' \in (wa)^A_\Sigma$ for some symbol $a' \in \Sigma$ such that $a' = \mathbf{u}_{w'}[i]$. Hence, for each $1 \leq j \leq |\mathbf{u}|$, $\mathbf{u}_{wa}[j] = \sharp$ iff $\mathbf{u}_{w'a'}[j] = \sharp$.

In case $a \notin [\mathbf{u}_w]$, $a = \mathbf{u}_{wa}[i]$, $\mathbf{u}_w[i] \neq \sharp$ and $\sharp \in [\mathbf{u}_w]$ (the second case of Definition 3.9), since $\mathbf{u}_w[i] \neq \sharp$, by the induction hypothesis, $\mathbf{u}_{w'}[i] = a' \neq \sharp$. Thus, $w'a' \in (wa)^A_\Sigma$. Similarly to the above case, $\mathbf{u}_{wa}[i] = \sharp$ iff $\mathbf{u}_{w'a'}[i] = \sharp$. In addition, $\sharp \in [\mathbf{u}_w]$. By the induction hypothesis, $\mathbf{u}_w[j] = \mathbf{u}_{w'}[j] = \sharp$ for some $j \neq i$. Since $||\Sigma|| = |\mathbf{u}|$, there exists an $b' \in \Sigma \setminus [\mathbf{u}_{w'}]$, namely, $w'b' \in (wa)^A_\Sigma$. Moreover, $\mathbf{u}_{w'}[i] = a'$ and $\mathbf{u}_{w'b'}[i] = b'$. Hence, $\mathbf{u}_{wa}[i] = \sharp$ iff $\mathbf{u}_{w'b'}[i] = \sharp$.

The last case is proved analogously. Therefore, we have proved that $(wa)^A_\Sigma \subseteq \Sigma^{n+1}$ and that for each $w'a' \in (wa)^A_\Sigma$ and each $1 \leq i \leq k$, $\mathbf{u}_{wa}[i] = \sharp$ iff $\mathbf{u}_{w'a'}[i] = \sharp$. The proof is completed. □

By Lemma 3.10, for each $wa \in \Omega^+$ and each $w'a' \in (wa)^A_\Sigma$, if $(p, \mathbf{u}_w) \vdash^a (q, \mathbf{u}_{wa})$, then $(p, \mathbf{u}_{w'}) \vdash^{a'} (q, \mathbf{u}_{w'a'})$. Hence, we have the following lemma.

LEMMA 3.11. *For each $w \in \Omega^+$ and $w' \in w^A_\Sigma$, $A(w) = A(w')$.*

For each simple DFMA A and B and for each finite alphabet Σ, we can decide whether or not $A \simeq_\Sigma B$. Hence, it is sufficient to prove that $A \simeq_\Sigma B$ implies $A \simeq B$ for the equivalence problem, where $||\Sigma|| = max\{|\mathbf{u}^A|, |\mathbf{u}^B|\}$. Moreover, by Lemma 3.11, this condition is simplified to that for each $w \in \Omega^+$, there exists $w' \in \Sigma^+$ such that $w' \in w^A_\Sigma \cap w^B_\Sigma$.

LEMMA 3.12. *Let A and B be simple DFMAs with initial assignments $\mathbf{u}^A$ and $\mathbf{u}^B$, respectively. Let Σ be an alphabet of cardinality $k = max\{|\mathbf{u}^A|, |\mathbf{u}^B|\}$. Then, for each $w \in \Omega^+$, $w^A_\Sigma \cap w^B_\Sigma \neq \emptyset$.*

PROOF. Suppose that $|\mathbf{u}^A| \geq |\mathbf{u}^B|$. We show the desired result by induction on the length of w. For the induction base, let $a \in \hat{\Sigma}$. Clearly, for each $a \in \Omega$, $a^A_\Sigma = a^B_\Sigma = \Sigma$. Now, assume the induction hypothesis on Ω^n, $n \geq 1$. Let $wa \in \Omega^{n+1}$.

Let $a \in [\mathbf{u}^A_w] \cap [\mathbf{u}^B_w]$. Then, there exists $w' \in w^A_\Sigma \cap w^B_\Sigma$ such that $w'\mathbf{u}^A_{w'}[i] \in (wa)^A_\Sigma$ for $a = \mathbf{u}^A_w[i]$ and $w'\mathbf{u}^B_{w'}[j] \in (wa)^B_\Sigma$ for $a = \mathbf{u}^B_w[j]$, where $1 \leq i \leq |\mathbf{u}^A|$ and $1 \leq j \leq |\mathbf{u}^B|$. Since $a \in [\mathbf{u}^A_w]$, we can assume that $w = \alpha a \beta$ for some $\alpha \in \Omega^*$ and $\beta \in (\Omega \setminus \{a\})^*$. By the induction hypothesis, there exists $\alpha'a'\beta' \in (\alpha a\beta)^A_\Sigma \cap (\alpha a\beta)^B_\Sigma$. Since $a \notin [\beta]$ and $a \in [\mathbf{u}^A_w] \cap [\mathbf{u}^B_w]$, $\mathbf{u}^A_{\alpha a}[i] = u^A_w[i] = \mathbf{u}^B_{\alpha a}[j] = \mathbf{u}^B_w[j] = a$. Thus, by Lemma 3.10, $\mathbf{u}^A_{\alpha'a'}[i] = \mathbf{u}^A_{w'}[i] = \mathbf{u}^B_{\alpha'a'}[j] = \mathbf{u}^B_{w'}[j] = a'$. Thus, $w'a' = w'\mathbf{u}^A_{w'}[i] \in (wa)^A_\Sigma$ and $w'a' = w'\mathbf{u}^B_{w'}[j] \in (wa)^B_\Sigma$. That is, $(wa)^A_\Sigma \cap (wa)^B_\Sigma \neq \emptyset$.

Let $a \in [\mathbf{u}^B_w] \setminus [\mathbf{u}^A_w]$. Since $a \in [w]$, let $w = \alpha a \beta$ for some $\alpha \in \Omega^*$ and $\beta \in (\Omega \setminus \{a\})^*$. By the induction hypothesis, let $\alpha'a' \in (\alpha a)^B_\Sigma$. By Lemma 3.10, there exists j such that $\mathbf{u}^B_{\alpha a}[j] = a$ iff $\mathbf{u}^B_{\alpha'a'}[j] = a'$. Let $w' = \alpha'a'\beta' \in w^B_\Sigma$. Since $a \notin [\beta]$, by Lemma 3.10, $a' \notin [\beta']$. When we take $w'a' \in (wa)^B_\Sigma$, since $a \in [\mathbf{u}^A_{\alpha a}]$, we can assume that $a = \mathbf{u}^A_{\alpha a}[i]$ and $a' = \mathbf{u}^A_{\alpha'a'}[i]$. Since $a \notin [\mathbf{u}^A_w]$, by Lemma 3.10, $\mathbf{u}^A_{\alpha a}[i] = a \neq \mathbf{u}^A_w[i]$ implies $\mathbf{u}^A_{\alpha'a'}[i] = a' \neq \mathbf{u}^A_{w'}[i]$. Since $a' \notin [\beta']$, we have that $a' \notin [\mathbf{u}^A_{w'}]$. Thus, $w'a' \in (wa)^A_\Sigma$ because $w' \in w^A_\Sigma$, $a \notin [\mathbf{u}^A_w]$ and $a' \notin [\mathbf{u}^A_{w'}]$. Consequently, $w'a' \in (wa)^A_\Sigma \cap (wa)^B_\Sigma$.

Let A_* be a target and A a simple DFMA such that $L(A) = \emptyset$.
Initialize: $S = E = \Sigma = \Sigma' = \{\varepsilon\}$ and $T : (S \cup S \cdot \Sigma) \cdot E \mapsto \{0, 1\}$.
Begin
  While (an equivalence query returns a counterexample w) **do**
    Let $\Sigma' := \Sigma \cup [w]$, $S := S \cup \{pre(w)\}$, and
    extend T to $(S \cup S \cdot \Sigma') \cdot E$ using membership queries.
    Compute $M' :=$ANG(S, E, T).
    If ($\Sigma = \emptyset$ or $M \simeq_\Sigma M'$), **then**
      Let $M := M'$, $\Sigma := \Sigma'$, and $A := \mathcal{CONS}\ (M)$.
    End If
    If ($\Sigma \neq \emptyset$ and $M \not\simeq_\Sigma M'$), **then**
      Let $M := M'_\Sigma$ for $L(M'_\Sigma) = \Sigma^* \cap L(M')$ and $A := \mathcal{CONS}\ (M)$.
    End If
  End While
  Output A and halt.
End

Figure 5: Algorithm $\mathcal{DFMA}$ for a target A_*

Let $a \notin [\mathfrak{u}^A_w] \cup [\mathfrak{u}^B_w]$. If $[\mathfrak{u}^B_{w'}]$ is a proper subset of Σ, that is at least one window of $\mathfrak{u}^B_{w'}$ contains no symbol in Σ, then a symbol $a' \in \Sigma \setminus [\mathfrak{u}^B_{w'}]$ is not in $[\mathfrak{u}^A_{w'}]$. Thus, $w'a' \in (wa)^A_\Sigma \cap (wa)^B_\Sigma$. If $[\mathfrak{u}^B_{w'}] = \Sigma$, then, by Lemma 3.10, $\mathfrak{u}^B_w[j] = b$ and $\mathfrak{u}^B_{wa}[j] = a$ implies $\mathfrak{u}^B_{w'}[j] = b'$ for some $b' \in \Sigma$ and $1 \leq j \leq |\mathfrak{u}^B|$. Then, let $w' = \alpha' b' \beta'$, where $b' \notin [\beta]$. Since $b' \in [\mathfrak{u}^A_{\alpha' b'}] \cap [\mathfrak{u}^B_{\alpha' b'}]$ and $b' \notin [\beta']$, $w' \in w^A_\Sigma \cap w^B_\Sigma$ implies $w'b' = \alpha' b' \beta' b' \in (wb)^A_\Sigma \cap (wb)^B_\Sigma$, where $b = \mathfrak{u}^B_w[j]$. Since $a \notin [\mathfrak{u}^B_w]$, $\sharp \notin [\mathfrak{u}^B_w]$ and $b \in [\mathfrak{u}^B_w]$, by the definition, $(wb)^B_\Sigma = (wa)^B_\Sigma$. Similarly, $(wb)^A_\Sigma = (wa)^A_\Sigma$ if $b \notin [\mathfrak{u}_w]$, and $(wb)^A_\Sigma \subseteq (wa)^A_\Sigma$ otherwise. Thus, $w'b' =\in (wa)^A_\Sigma \cap (wa)^B_\Sigma$, namely, $(wa)^A_\Sigma \cap (wa)^B_\Sigma \neq \emptyset$. Therefore, we have that for each $w \in \Omega^{n+1}$, $w^A_\Sigma \cap w^B_\Sigma \neq \emptyset$. □

THEOREM 3.13. *Let A and B be simple DFMAs with initial assignments $\mathfrak{w}^A$ and $\mathfrak{w}^B$, respectively. Let Σ be an alphabet of cardinality $k = max\{|\mathfrak{w}^A|, |\mathfrak{w}^B|\}$. Then, $A \simeq_\Sigma B$ iff $A \simeq B$.*

PROOF. The if-part is trivial, thus we have to prove the only-if-part. Let $w \in L(A)$. By Lemma 3.11, $w^A_\Sigma \subseteq L(A)$. Since $A \simeq_\Sigma B$, $w^A_\Sigma \subseteq L(B)$. By Lemma 3.12, there exists $w' \in w^A_\Sigma \cap w^B_\Sigma$. By Lemma 3.11, $B(w') = B(w)$. Thus, $w \in L(B)$. The converse direction is proved analogously. □

Let A be a hypothesis computed from a DFA $M_A = \langle Q^A, \Sigma_A, q^A_0, \delta^A, F^A \rangle$ in one stage. After a counterexample for $A \simeq A_*$ is returned, assume that a DFA $M_B = \langle Q^B, \Sigma_B, q^B_0, \delta^B, F^B \rangle$ is computed.

LEMMA 3.14. *If $M_A \simeq_{\Sigma_A} A_*$, then $M_A \simeq_{\Sigma_A} M_B$.*

PROOF. Let $M_A := M(S^A, E^A, T^A)$ and $M_B := M(S^B, E^B, T^B)$. Assume that $M_A \simeq_{\Sigma_A} A_*$. Let a counterexample w be returned. Then, there exists $a \in [w]$ such that $a \in \Sigma_B \setminus \Sigma_A$. If each nonempty prefix of w contains a, then $S^A = \{s \in S^B | a \notin [s]\}$, $E^A = \{e \in E^B | a \notin [e]\}$ and $T^A = \{(\alpha, b) \in T^B | a \notin [\alpha]\}$. It follows that $M_A \simeq_{\Sigma_A} M_B$. Suppose that there exists $s \in S^B \cap \Sigma_A^*$ such that $s \notin S^A$. If there exists $s' \in S^A$ such that $T^A(s' \cdot e) = T^B(s \cdot e)$ for all $e \in E^A$, then $M_A \simeq_{\Sigma_A} M_B$, because $M_A \simeq_{\Sigma_A} A_*$. Then, let $T^A(s' \cdot e) = T^B(s \cdot e)$ for all $s' \in S^A$ and all $e \in E^A$. Since $M_A \simeq_{\Sigma_A} A_*$, there exists $s'' \in S^A$ such that $row(s) = row(s'')$ on (S^A, E^A, T^A). Since $S^A \subset S^B$, $s'' = s$ implies $T^A(s'' \cdot e) = T^B(s \cdot e)$ for all $e \in E^A$. This contradiction follows $M_A \simeq_{\Sigma_A} M_B$. □

LEMMA 3.15. *Let* $||\Sigma_A|| \geq |\mathfrak{w}^{A_*}|$. *If* $M_A \not\simeq_{\Sigma_A} A_*$, *then* $M_A \not\simeq_{\Sigma_A} M_B$.

PROOF. Let $A = \langle Q^A, \mathfrak{u}^A, q_0^A, \varrho^A, \mu^A, F^A \rangle$ be the hypothesis from $\mathcal{CONS}(M_A)$. Assume that $M_A \not\simeq_{\Sigma_A} A_*$. Then, a counterexample $w \in \Sigma^*$ for some $\Sigma \supset \Sigma^A$ is returned. Let $B = \langle Q^B, \mathfrak{u}^B, q_0^B, \varrho^B, \mu^B, F^B \rangle$ be a next hypothesis computed by $\mathcal{CONS}(M_B)$. Without loss of generality, we can assume that $w \in L(A_*) \setminus L(A)$. Note that $|\mathfrak{u}^A| = |\mathfrak{u}^B|$, because B is computed from the DFA M such that $L(M) = L(M_B) \cap \Sigma_A^*$. Here, we recall Definition 3.9. Since $||\Sigma_A|| \geq |\mathfrak{u}^A| = |\mathfrak{u}^B|$, $(w)_{\Sigma_A}^A$ and $(w)_{\Sigma_A}^B$ are well-defined. By Lemma 3.12, there exists $w' \in (w)_{\Sigma_A}^A \cap (w)_{\Sigma_A}^B$. Since $w \in L(A_*)$ and $w \notin L(A)$, by Lemma 3.11, $w' \notin L(A)$ and $w' \in L(B)$. Thus, $M_A \not\simeq_A M_B$. The proof is completed. □

THEOREM 3.16. *Every simple deterministic finite-memory automaton is exactly learnable using membership and equivalence queries.*

PROOF. Let A_* be a target. By Lemmas 3.14 and 3.15, every counterexample contributes to learning a simple DFMA A such that $A \simeq_{\Sigma_A} A_*$, where $||\Sigma_A|| \geq |\mathfrak{u}^{A_*}|$. Thus, by Theorem 3.13, we conclude this theorem. □

THEOREM 3.17. *The running time of the algorithm $\mathcal{DFMA}$ to compute a target A_* is bounded by $\mathcal{O}(m^4 n^3 t_{m,n})$, where m is the length of a longest counterexample returned, n is the number of states of the minimum DFA M such that $L(M) \simeq_\Sigma L(A_*)$ for $||\Sigma|| = m$, and $t_{m,n}$ is the time to compute M by Angluin's (1987) algorithm.*

PROOF. When a counterexample is returned, the procedure $\mathcal{ANG}$ first computes a closed, consistent observation table using membership queries. Once such a table is obtained, the procedure $\mathcal{ANG}$ computes a corresponding DFA M from the table and check whether or not the DFA is permutation closed. The time is $t_{m,n}$ (Angluin (1987)). If M is not permutation closed, we must find at most $nm(m-1)/2$ strings of length at most n. Thus, the time used by $\mathcal{ANG}$ is bounded by $\mathcal{O}(m^2 n^2 t_{m,n})$.

It is easy to see that the time used by $\mathcal{CONS}$ is bounded by $\mathcal{O}(mn)$. Thus, the update-time of the algorithm $\mathcal{DFMA}$ is $\mathcal{O}(m^2 n^2 t_{m,n} + mn)$. At most mn counterexamples of length at most m are returned by the time a DFA M such that $M \simeq_\Sigma A_*$ is computed, where $||\Sigma|| = m$. Thus, the total running time of $\mathcal{DFMA}$ is bounded by $\mathcal{O}(m^2 n(m^2 n^2 t_{m,n} + mn)) = \mathcal{O}(m^4 n^3 t_{m,n})$. □

4. Concluding remarks

Simple DFMAs, a subclass of finite-memory automata have been defined and investigated. Kaminski and Francez (1994) discussed various potential applications and benefits of their computational model; thus we refer the reader to their article for further information in this regard. The main new feature of finite-memory automata is their ability to perform computations over an infinite alphabet. The attractiveness of this computational model naturally suggested the problem to study the learnability of such computing devices. We have chosen, in this paper, the query learning model introduced by Angluin (1988) for providing the first answer to this problem. In particular, we have established the learnability of simple DFMAs via membership and equivalence queries. Moreover, we could prove the equivalence problem for simple DFMAs to be decidable. While answering membership queries has to be considered as an unobjectionable ability to be required of a teacher, the latter result considerably supports our viewpoint that the equivalence queries for simple DFMAs are reasonable. Additionally, the insight obtained in solving the equivalence problem turned out to be helpful for designing the desired learning algorithm.

Nevertheless, several problems remain open. Obviously, the most interesting question is whether or not our results are generalizable to the whole class of deterministic finite-memory automata. Inspecting our proofs we see that almost all results obtained heavily depend on the following closure property: for every simple DFMA A and every automorphism $\ell : \Omega \mapsto \Omega$, it holds that $\ell(L(A)) = L(A)$. This closure property is concerned with initial assignments of simple DFMAs including only the distinguished symbol $\sharp$. However, an initial assignment of a finite memory-automata may contain some alphabet symbols. When a deterministic finite-memory automaton reads an input symbol contained in its initial assignment, it may pay special attention to this symbol compared with other symbols not contained initially in its assignment. In this sense, every symbol is fairly judged by simple DFMAs but not by general DFMAs. On the other hand, there is no other difference between definitions of DFMAs and simple DFMAs except their initial assignments. Thus, we conjecture that the problem of learning the whole class of DFMAs is reducible to task of identifying initial assignments. We already made some progress along this line, but the final solution could not be obtained yet.

Acknowledgment. The author is very grateful to Prof. Thomas Zeugmann who read all part of an earlier version of this paper with great patience, and spent so much time and energy for a careful proof-reading. Without his helpful comments, this paper would never have been completed. The author thanks the referees of BIC for their helpful comments.

References.

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

Angluin, D. (1988), Queries and concept learning, Machine Learning, 2:319 – 342.

Burago, A. (1994), Learning structurally reversible context-free grammars from queries

and counterexamples in polynomial time, Proceedings of the 7th Workshop on the Computational Learning Theory, New Brunswick, USA, pp.140 – 146, ACM Press.

Bergadano, F. and Varricchio, S. (1995), Learning behaviors of automata from shortest counterexamples, Proceedings of the 2nd European Conference on Computational Learning Theory, In Vitanyi, P. (Ed.), Barcelona, Spain, pp.380 – 391, LNAI 904, Springer-Verlog.

Gasarch, W. I. and Smith, C. H. (1992), Learning via queries, Journal of the ACM, 39 (3):649 – 674.

Gavaldà, R. (1994), On the power of equivalence queries, Proceedings of the 1st European Conference on Computational Learning Theory, In Taylor, J. S. and Anthony, M. (Eds.), Clarendon Press, Oxford, Royal Holloway University, London.

Hopcroft, J. E. and Ullman, J. D. (1979), Introduction to automata theory, languages, and computation, Addison-Wesley.

Ishizaka, H. (1990), Polynomial time learnability of simple deterministic languages, Machine Learning, 5(2)151 – 164.

Kaminski, M. and Francez, N. (1994), Finite-Memory automata, Theoretical Computer Science, 134:329 – 363.

Kinber, E. B. and Zeugmann, T. (1989), Refined query inference, Proceedings of the 2-nd International Workshop on Analogical and Inductive Inference, In Jantke, K. P. (Ed.), LNAI 397, pp.148 – 160, Springer-Verlog.

Sakakibara, Y. (1990), Learning context-free grammars from structural data in polynomial time, Theoretical Computer Science, 76:223 – 242.

Sakamoto, H. and Ikeda, D. (1998), Intractability of decision problems for finite-memory automata, Proceedings of the International Colloquium of Universal Machines and Computations, to appear.

Sammut, C. and Banerji, R. (1986), Learning concepts by asking questions, In Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (Eds.), Machine Learning: An artificial intelligence approach, Vol. 2, Morgan Kaufmann, San Mateo, Ca.

Shapiro, E. (1981), A general incremental algorithm that infers theories from facts, Proceedings of the 7th International Joint Conference on Artificial Intelligence, pp.446 – 451, Morgan Kaufmann, San Mateo, Ca.

Shapiro, E. (1982), Algorithmic program diagnosis, Proceedings of the 9th ACM Symposium on Principles of Programming Languages, pp. 299 – 308, ACM Press.

Shapiro, E. (1983), Algorithmic program debugging, Cambridge, MA: MIT Press.

Received December 25, 1997