Chiang Mai J. Sci. 2013; 40(2) : 248-260
http://it.science.cmu.ac.th/ejournal/
Contributed Paper

# The Study of Learnability of the Class of k-acceptable Languages on Gold's Learning Model

**Anuchit Jitpattanakul and Athasit Surarerks***
Engineering Laboratory in Theoretical Enumerable System (ELITE), Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand.
*Author for correspondence; e-mail: athasit.s@chula.ac.th.

Received: 9 December 2011
Accepted: 5 April 2012

**ABSTRACT**

The study of learnability of formal languages is an attractive topic in grammatical inference. Many classes of the formal languages have been theoretically investigated their learnability from some presentations. These obtained results are applicable for some real-world problems such as speech recognition, protien-motif classification. However, the study is limited to formal languages defined over unordered alphabets. In this article, we focus our attention on formal languages that are defined over ordered alphabets called *k*-acceptable languages. We theoretically show that the class of this language is not learnable by using only positive examples. By using both positive and negative examples, we show that the class of *k*-acceptable languages is learnable in the limit from polynomial time and data.

**Keywords:** k-acceptable languages, identification in the limit, learnability, grammatical inference

## 1. INTRODUCTION

In research field of grammatical inference (shortly called GI), language learning refers to a process of identifying a formal language in terms of its grammatical representation by a learner who is given language information. One of most attractive topics in GI is of theoretically studying on learnability of some classes of languages. The study is mostly based on three learning models, e.g. Gold's explanatory learning model [1], Angluin's active learning model [2], and Valiant's probably approximately correct (PAC) learning model [3]. In this paper, we focus on the learnability on Gold's model.

The Gold's learning model is viewed as a framework of explanatory learning. In the process of learning an unknown language, a number of examples will be provided at each time to a learner who is to hypothesize a grammatical representation of the language on the basis of the examples received so far. The process continues forever. The success of learning process is considered by using a criterion called *identification in the limit*. However, this criterion only tells us that the correct grammatical representation of the target language would be eventually identified, but the issue of efficiency of learning is regardless. Several learning criteria have been introduced to deal with this complexity issue. Some constraints are added in learning process within the

framework of identification in the limit [4]. One of those criteria is introduced in a learning model called *identification in the limit from polynomial time and data* introduced by Higuera [5].

The problem of learning regular languages has been a central problem in GI. This is because it is almost only one class of formal languages that is both efficiently learnable and general enough to represent many nontrivial real-life phenomena. The learnability of many subclasses of regular languages has been studied through identification of specifically defined automata [6] such as $k$-testable languages [7], local languages [8], and strictly deterministic regular languages [9]. Learning algorithms from these works have been experimentally tested to various applications such as DNA sequences analysis [10], music style recognition [11], and speech recognition [12]. The obtained results show that the algorithms are an effective and efficient alternative to solve the problems. Unfortunately, one disadvantage of using those algorithms is about a size of returned automata depending on size of alphabet of inputs. This leads to inconvenience in practical ways.

In order to solve these problems, $k$-edge deterministic finite automata ($k$-DFA) were first introduced by Higuera [13] to recognize languages defined over an ordered alphabet. The languages are recognized by $k$-DFA called $k$-acceptable languages. The interesting property of $k$-DFA is that its size depends on the $k$ value instead of the size of alphabet. This is a reason why the inference of $k$-acceptable languages is useful.

In this paper, we study the learnbility of $k$-acceptable languages on Gold's learning model. Two types of presentation, *i.e.* only positive examples and both positive examples and negative examples will be theoretically investigated. We first show that $k$-acceptable languages are not learnable by using only positive examples. Moreover, we study the identification in the limit from polynomial time and data of $k$-acceptable languages by using positive and negative examples. We propose a polynomial-time algorithm to infer $k$-acceptable languages from positive and negative examples and show that there exist characteristic sets with size polynomial for each $k$-acceptable language.

The remains are organized as follows. Section 2 presents basic definitions and notations that will be used throughout this paper. In section 3, we introduced $k$-acceptable languages and prove some properties of the language class. In section 4 we investigate learnability of the class of $k$-acceptable languages. The last section provides the conclusion of this work.

## 2. PRELIMINARIES

The basic definitions and notations used throughout this paper are provided as follows.

### 2.1 Formal Languages and Automata

Let $\Sigma$ be an *alphabet* that is a finite and nonempty set of symbols called *letters*. The size of $\Sigma$ is a number of letters, denoted by $|\Sigma|$. A finite sequence of letters from $\Sigma$ is called a *string*. Given a string $w$, the length of strings is the total number of letters appearing in $w$ and it is denoted by $|w|$. The string with length zero is called the *null string* denoted by $\lambda$. The infinite set of all possible strings over $\Sigma$, denoted by $\Sigma^*$, is the set of all finite-length strings generated by concatenating zero or more letters of $\Sigma$. Given a string $w = uv$, a string $u$ is a *prefix* of $w$ if and only if there exists a string $v$ in $\Sigma^*$. An alphabet $\Sigma$ with a partial order is called an *ordered alphabet* denoted by $\Sigma_<$. Given an order relation $<$ on $\Sigma$, we

can define a *lexicographic-length order* over $\Sigma^*$ for two strings $u, v \in \Sigma^*$, by setting $u <_{lex} v$ if and only if $|u| < |v|$ or there exist strings $w, u', v' \in \Sigma^*$ and two letters $x < y \in \Sigma$ such that $|u'| < |v'|$ and $u = wxu', v = wyv'$. A *language* over $\Sigma$ denoted by $L$ is any subset of $\Sigma^*$. The *complement of a language* $L$ is defined by $L' = \Sigma^* - L$. The family of languages over $\Sigma$, denoted by $\mathcal{L}$, is called *a class of languages*. A *super-finite class* is a class that contains all finite languages and at least one infinite language. A class $\mathcal{L}$ of languages has a *limit point* if there exists an infinite sequence $L_n \in \mathcal{L}$ such that $L_0 \subset L_1 \subset ... \subset L_i \subset ...$, where $i \in \mathbb{N}$ and there exists another language $L_p \in \mathcal{L}$ such that $L_p = \cup_{n \in \mathbb{N}} L_n$. The language $L_p$ is called a limit point for $\mathcal{L}$.

A *finite automaton* is a grammatical representation that is typically defined as a 5-tuple $M = (\Sigma, Q, q_0, F, \delta)$, where $\Sigma$ is a finite alphabet, $Q$ is a finite non-empty set of states, $q_0 \in Q$ is an initial state, $F \subseteq Q$ is a set of final states, and $\delta : Q \times \Sigma \to Q$ is a state transition function. The state transition function $\delta$ can be extended to a mapping $\delta^*: Q \times \Sigma^* \to Q$ in the following inductive way: (*i*) $\delta^*(q, \lambda) = q$, for each state $q \in Q$, where $\lambda$ is the null string, and (*ii*) $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$, for each state $q \in Q$, each letter $a \in \Sigma$, and each string $w \in \Sigma^*$. The *size of a finite automaton* $M$ is defined in this paper by a number of states of $Q$, denoted by $|M|$.

Theoretically, an automaton plays an important role as a language recognizer. A string $w$ is *recognized* by an automaton $M = (\Sigma, Q, q_0, F, \delta)$ if $\delta^*(q, w) \in F$. The language recognized by $M$, denoted by $\mathbb{L}(M)$, is the set of all strings which are recognized by the automaton $M$ and this set is called a *regular language*. A language $L$ is *recognizable* if there exists an automaton $M$ such that $L = \mathbb{L}(M)$. Any two automata $M$ and $M'$ are said to be *equivalent* if and only if they recognize the same language, *i.e.* $\mathbb{L}(M) = \mathbb{L}(M')$.

The finite automaton $M = (\Sigma, Q, q_0, F, \delta)$ is *deterministic* if $|\delta(q, a)| \leq 1$ for each $q \in Q$ and for each $a \in$ S. Then $M$ is called *deterministic finite automata* (shortly denoted by *DFA*s). Let $M_1 = (\Sigma, Q_1, q_0, F_1, \delta_1)$ and $M_2 = (\Sigma, Q_2, q_0, F_2, \delta_2)$ be two deterministic finite automata. Then $M_1$ is called a *subautomaton* of $M_2$ if $Q_1 \subseteq Q_2$, $F_1 \subseteq F_2$, and $\delta_1 \subseteq \delta_2$. It is easily seen that if $M_1$ is a subautomaton of $M_2$ then $\mathbb{L}(M_1) \subseteq \mathbb{L}(M_2)$.

## 2.2 Language Learning and Convergence Criterions

Let $L$ be a target language in a language class $\mathcal{L}$. With context of Gold-style learning, any strings in $\Sigma^*$ are sampled as examples for learning. A set of strings $S$ is called a *positive sample* if $S$ is a subset of $L$, defined as $S+ = \{w \in \Sigma^* : w \in L\}$ and elements of $S+$ are called *positive examples*. But if $S$ is a subset of the complement of $L$ then $S$ is called a negative sample of $S- = \{w \in \Sigma^* : w \in \Sigma^* - L\}$ and elements of $S-$ are called *negative examples*. In the process of learning, the main purpose is to construct an equivalent grammatical representation by using an *input sample* $S = (S+, S-)$ for $L$, where $S+ \subseteq L$ is a set of positive examples and $S- \subseteq \Sigma^* - L$ is a set of negative examples. Given a learning sample $S$, a *learning sample at time n* is denoted by $S_n = \{S(j) \in \Sigma^* : j \leq n\}$ where $n \in \mathbb{N}$. The size of learning sample $S$ is total numbers of size of all example $w_i$ contained in $S$, defined by $||S||$.

In Gold's learning framework is concerned two types of presentation. For the first one, only the *positive sample* denoted by $S = (S+, \varnothing)$ is considered. The second one contains *positive and negative sample* denoted as $S = (S+, S-)$ which is called an *informant* in Gold's work. In this

paper, we modify the inclusion operators of set for input samples such that they operate on the set of positive and the set of negative examples separately, *i.e.* if $S = (S+, S-)$ and $S' = (S'+, S'-)$ then $S \subseteq S'$ means $S+ \subseteq S'+$ and $S- \subseteq S'-$.

A *learning algorithm* $\mathcal{A}$ is a mapping function defined as $\mathcal{A} : S \rightarrow \mathcal{G}$, where $S$ is a set of all samples and this set is used for learning any language $L$ in language class $\mathcal{L}$ by identifying a grammatical representation $G$ in a class $\mathcal{G}$ of corresponding grammatical representations. This function can be viewed as a learning algorithm. For learning an unknown language $L$, we say that the algorithm $\mathcal{A}$ *converges* to $G \in \mathcal{G}$ from $S \in \mathcal{S}$ if and only if $\mathbb{L}(\mathcal{A}(\mathcal{S})) = L$.

The learning criterion that will be used in this paper can be found in [1] and if is well known as *identification in the limit*. A formal definition of this criterion is restated as follows.

**Definition 2.1**

Let $G$ be a grammatical representation of a language $L$ and $S$ be a learning sample of $L$. A learning algorithm $\mathcal{A}$ is said to *identify in the limit* $L$ in terms of $\mathcal{G}$ by using $S$ if and only if, the algorithm $\mathcal{A}$ converge to a grammatical representation $G'$ on $S_i \subseteq S$ where $i \in \mathbb{N}$ such that $\mathbb{L}(G') = \mathbb{L}(G)$.

This definition can be extended to the class of grammatical representations. Thus, a class of grammatical representation $\mathcal{G}$ can be said to be *identifiable in the limit* if and only if there exists a learning algorithm that identifies in the limit all grammatical representation $G$ in the class $\mathcal{G}$.

**Definition 2.2**

Given $\mathcal{G}$ is a grammatical representation corresponding to a language class $\mathcal{L}$ and $S$ is a learning sample in $S$. $A$ learning algorithm $\mathcal{A}$ is said to *learn in the limit* $\mathcal{L}$ in terms of $\mathcal{G}$ by using $S$ if and only if, for any $L \in \mathcal{L}$, there exists a learning algorithm $\mathcal{A}$ that converges to $G$ on $S_i$ where $i \in \mathbb{N}$ such that $\mathbb{L}(G) = L$. A class $\mathcal{L}$ of languages is said to be *learnable in the limit* if there exists a learning algorithm $\mathcal{A}$ that identify in the limit any $L$ in $\mathcal{L}$ and *not learnable in the limit* if there is no such the learning algorithm.

Note that learnability is therefore a property of a class of languages, not of any one individual language. In Gold's seminal work, two theoretical results concerning the learnability of non-trivial classes in Chomsky's hierarchy were given. We recall them as below.

**Theorem 2.1**

*No super-finite class of languages is learnable in the limit from only a positive sample.*

**Proof :** See [1]. ■

**Theorem 2.2**

*Any class of recursive languages is learnable in the limit in term of Turing machine that always halts by using an informant.*

**Proof :** See [1]. ■

The language identification in the limit is proposed for sequential presentation of data. This learning framework tells us only that we will eventually find a grammatical representation to be able to describe the target language. Gold [14] also proposed a related identification for fixed learning data and it is called *identification from given data*. In this framework, an interesting concept about good learning sample was proposed. This kind of the such examples is called *characteristic sample* [5]. We restate the related definitions below.

**Definition 2.3**

Given a sample $S$ for a language $L$,

a learning algorithm $\mathcal{A}$ *identifies a language* $L$ *from given data* if there exists a sample $CS \subseteq S$ such that for any learning sample $S$, the grammatical representation $M$ proposed by the algorithm $\mathcal{A}$ is correct, that is $\mathbb{L}(\mathcal{A}(S)) = L$. The sample $CS$ is called a *characteristic sample* for the language $L$ and the learning algorithm $\mathcal{A}$.

The following theorem states the equivalence of the two learning frameworks.

**Theorem 2.3**

*Let $\mathcal{L}$ be a class of languages, a learning algorithm $\mathcal{A}$ identifies $\mathcal{L}$ from given data if and only if it identifies $\mathcal{L}$ in the limit.*

**Proof:** See [15]. ■

**Corollary 2.1**

*A language class $\mathcal{L}$ is learnable in the limit from given data if and only if $\mathcal{L}$ is learnable in the limit from a sequential presentation.*

The two learning frameworks proposed by Gold do not consider a complexity aspect in learning process. A learning model including such the aspect was introduced by Higuera [5].

**Definition 2.4**

A learning sample composed of positive and negative examples, denoted by $CS = (CS+, CS-)$, is said to be a characteristic sample of a target language $L$ for a learning algorithm $\mathcal{A}$ if $CS$ satisfies following conditions:

1. Given a learning sample $CS$, the learning algorithm $\mathcal{A}$ returns a grammatical representation $G \in \mathcal{G}$ such that $\mathbb{L}(G) = L$.
2. Given any learning sample $S = (S+, S-)$ such that $S \supseteq CS$, the algorithm $\mathcal{A}$ returns a grammatical representation $G \in \mathcal{G}$ such that $\mathbb{L}(G) = L$.

**Definition 2.5**

Let $\mathcal{L}$ be a class of languages and $\mathcal{G}$ be a class of grammatical representations corresponding with $\mathcal{L}$.The class $\mathcal{L}$ is *identifiable in the limit from polynomial time and data* in terms of $G$ if and only if there exists two polynomial functions $p(\,)$ and $q(\,)$ and a learning algorithm $\mathcal{A}$ such that:

1. Given any learning sample $S = (S+, S-)$, the algorithm $\mathcal{A}(S)$ returns a grammatical representation $G \in \mathcal{G}$ such that $S+ \subseteq \mathbb{L}(G)$ and $S- \cap \mathbb{L}(G) = \varnothing$ in $\mathcal{O}(p(\|S\|))$.
2. $\forall L \in \mathcal{L}$ and $\forall G \in \mathcal{G}$, $\mathbb{L}(G) = L$, there exists a characteristic sample $CS = (CS+, CS-)$ such that $\|CS\| < q(\|G\|)$, for which, if $CS \subseteq S$ then the algorithm $\mathcal{A}(S)$ converges to a grammatical representation $G'$ such that $\mathbb{L}(G') = \mathbb{L}(G)$.

## 3. THE CLASS OF *k*-ACCEPTABLE LANGUAGES AND ITS PROPERTIES

We begin this section with giving a formal definition of an automaton called a $k$-edge deterministic finite automaton. This kind of automata introduced in [13] to recognize languages are called $k$-acceptable languages. In this section, some properties of the $k$-acceptable languages will be investigated for studying their learnability.

**Definition 3.1**

A *$k$-edge deterministic finite automaton* ($k$-DFA) is a 6-tuple $M_k = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ where $\Sigma_{\leq}$ is a finite ordered alphabet, $Q$ is a finite set of states, $q_0$ is the initial state, $F_A \subseteq Q$ is a set of accepting states and $F_R \subseteq Q$ is a set of rejecting states, $\delta: Q \times \Sigma_{\leq} \times \Sigma_{\leq} \rightarrow Q$ is the transition function defined as $q \in Q$, $|\{[x, y]: \delta_k(q, x, y) \neq \varnothing\}| \leq k$, and if $\delta_k(q, a_1, b_1) \neq \delta_k(q, a_2, b_2)$ then $\{z : a_1 \leq z \leq b_1\} \cap \{z : a_2 \leq z \leq b_2\} = \varnothing$. The extended transition function $\delta_k^*: Q \times \Sigma^* \rightarrow Q$ is defined as

$\delta_k^*(q, \lambda) = q$ and $\delta_k^*(q, aw) = \delta_k^*(q', w)$ where $x \le a \le y$ and $\delta_k(q, x, y) = q'$ such that $q, q' \in Q, a, x, y \in \Sigma_{\le}, w \in \Sigma_{\le}^*$.

**Remark :** Notice that the maximum number of edges of a state is used to determines the value of $k$ in definition of $k$-DFA. Thus a $k$-DFA can be also viewed as a $m$-DFA such that $m > k$ for $m, k \in \mathbb{N}$.

**Example 3.1**

The finite automaton in Figure 1. recognizes language $((1+2+3)+(4+5)1^*(2+3+4+5))^*$. This automaton is 2-DFA because for any $q \in Q$, $|\{(x, y) : \delta_2(q, x, y) \neq \varnothing\}| \le 2$ and for state $q_0 : \delta_2(q_0, 1, 3) \neq \delta_2(q_0, 4, 5) \neq \varnothing$ then $\{z : 1 \le z \le 3\} \cap \{z : 4 \le z \le 5\} = \varnothing$ for state $q_3 : \delta_2(q_3, 1, 1) \neq \delta_2(q_3, 2, 5) \neq \varnothing$ then $\{z : 1 \le z \le 1\} \cap \{z : 2 \le z \le 5\} = \varnothing$. Notice that the same language can also be recognized by a 3-DFA, but not by a 1-DFA.

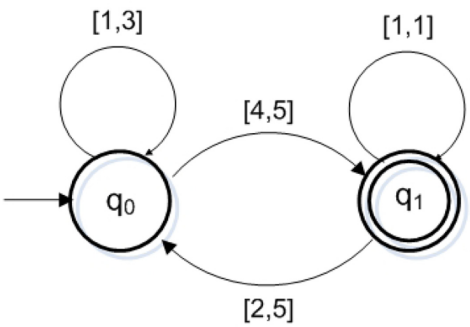


**Figure 1.** 2-DFA.

**Proposition 3.1** *For any integer $k \ge 0$, a $k$-DFA is a $(k+1)$-DFA.*

**Proof:** Let $M_k = (\Sigma_{\le}, Q, q_0, F_A, F_R, \delta_k)$ be a $k$-DFA. Suppose that $q$ is a state in $Q$ such that $q \in Q$, $|\{[x, y]: \delta_k(q, x, y) \neq \varnothing\}| = k$ and if $\delta_k(q, a_1, b_1) \neq \delta_k(q, a_2, b_2)$ then $\{z : a_1 \le z \le b_1\} \cap \{z: a_2 \le z \le b_2\} = \varnothing$. For all integer $k$, it is algebraically obvious that $k < k + 1$. It follows that $|\{[x, y]: \delta_k(q, x, y) \neq \varnothing\}| \le k+1$. Therefore, $M_k$ is a $(k+1)$-DFA by definition. ■

**Definition 3.2**

Languages recognized by $k$-DFA $M_k = (\Sigma_{\le}, Q, q_0, F_A, F_R, \delta_k)$ are called *$k$-acceptable languages* defined as $L = \{w : \delta^*_k(q_0, w) \in F_A\}$. A set of all $k$-acceptable languages is called that a *class of $k$-acceptable language* denoted by $k$-ACC for any integer $k \ge 0$.

**Proposition 3.2** *For any integer $k \ge 0$, $k$-ACC $\subseteq (k+1)$-ACC.*

**Proof:** Let $M_k = (\Sigma_{\le}, Q, q_0, F_A, F_R, \delta_k)$ be a $k$-DFA and $L_k \in k$-ACC be a $k$-acceptable language recognized by $M_k$. In order to prove this proposition, we will show that $L_k$ is a language in the class $(k+1)$-ACC. In other word, we have to show there exists a $(k+1)$-DFA to recognize the language $L_k$. Suppose $M_m = (\Sigma_{\le}, Q, q_0, F_A, F_R, \delta_m)$ is a $m$-DFA. We define the transition function $\delta_{k+1}$ of $M_m$ as $\delta_m = \delta_k - \{(q_0, a_0, a_i, p) : a_i \in \Sigma_{\le}, p \in Q\} \cup \{(q_0, a_0, a_0, p), (q_0, a_1, a_i, p)\}$. It follows that $q \in Q$, $|\{[x, y] : \delta_m(q, x, y) \neq \varnothing\}| = k+1$. Thus, $M_m$ is a $(k+1)$-DFA recognizing the language $L_k$. That is $L_k \in (k+1)$-ACC. Thus we can conclude that $k$-$ACC \subseteq (k+1)$-$ACC$ for $k \ge 0$. ■

We then show that $k$-ACC has a limit point for any integer $k \ge 1$.

**Proposition 3.3** *For an integer $n > 0$, $\cup_{i = 0 \text{ to } n} \Sigma^i_{\le} \in 1$-ACC .*

**Proof:** This statement can be proved by using mathematical induction.

*Basis step*: Let $L_0 = \cup_{i = 0 \text{ to } 0} \Sigma^i_{\le} = \{\lambda\}$. It is easily to see that 1-DFA for $L_0$ can be defined as $M_{(0)} = (\Sigma_{\le}, Q_{(0)}, q_0, F_{A(0)}, F_{R(0)}, \delta_{1(0)})$ such that $\Sigma_{\le} = \{a_1, a_2, ..., a_n\}$, $Q_{(0)} = \{q_0, q_1\}$, $F_{A(0)} = \{q_0\}$, $F_{R(0)} = \{q_1\}$ and $\delta_{(0)} = \{(q_0, a_1, a_n, q_1), (q_1, a_1, a_n, q_1)\}$. It follows

that $\#(\delta_{(0)}) = 1$. By definition 3.1, $M_{(0)}$ is a 1-DFA. Thus $L_0 \in$ 1-ACC.

*Induction step* : Suppose that $L_k = \cup_{i = 0 \text{ to } k}\Sigma^i_{\leq} \in$ 1-$ACC$ and then we will show that $L_{k+1} = \cup_{i = 0 \text{ to } k+1}\Sigma^i_{\leq} \in$ 1-$ACC$.

We let $M_{(k)} = (\Sigma_{\leq}, Q_{(k)}, q_0, F_{A(k)}, F_{R(k)}, \delta_{(k)})$ be a 1-DFA of $L_k$ that is defined as follows. $\Sigma_{\leq} = \{a_1, a_2, ..., a_n\}$, $Q_{(k)} = \{q_0, q_1, ... q_{k+1}\}$, $F_{A(k)} = \{q_0, q_1, ... q_k\}$, $F_{R(k)} = \{q_{k+1}\}$ and $\delta_{(k)} = \{(q_0, a_1, a_n, q_1), (q_1, a_1, a_n, q_2), ..., (q_k, a_1, a_n, q_{k+1}), (q_{k+1}, a_1, a_n, q_{k+1})\}$ such that $\#(\delta_{(k)}) = 1$ for $L_k$.

Consider $L_{k+1} = \cup_{i=0 \text{ to } k+1}\Sigma^i_{\leq} = \cup_{i=0 \text{ to } k}\Sigma^i_{\leq} \cup \{w \in \Sigma^*_{\leq} : |w| = k+1\}$. We can build a 1-DFA $M_{(k+1)} = (\Sigma_{\leq}, Q_{(k+1)}, q_0, F_{A(k+1)}, F_{R(k+1)}, \delta_{(k+1)})$ recognizing $L_{k+1}$ from $M_{(k)}$ as follows : $Q_{(k+1)} = Q_{(k)} \cup \{q_{k+2}\}$, $F_{A(k+1)} = F_{A(k)} \cup \{q_{k+1}\}$, $F_{R(k+1)} = \{q_{k+2}\}$ and $\delta_{(k+1)} = (\delta_{(k)} - \{(q_{k+1}, a_1, a_n, q_{k+1})\}) \cup \{(q_{k+1}, a_1, a_n, q_{k+2}), \{(q_{k+2}, a_1, a_n, q_{k+2})\}\}$. It is clear to see that $\#(\delta_{(k+1)}) = 1$. It follows that $M_{(k+1)}$ is a 1-DFA. Therefore, $L_{k+1} \in$ 1-$ACC$ by definition. ■

**Proposition 3.4** *The class 1-ACC has a limit point.*

**Proof:** To prove this proposition, we have to show that there exist a limit point in 1-ACC.

As $\Sigma^*_{\leq}$ is an infinite language, we will show that the language $\Sigma^*_{\leq} \in$ 1-ACC. Firstly, we suppose $M_1 = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_1)$ is a 1-DFA defined as follows: $\Sigma_{\leq} = \{a_0, a_1, ... , a_n\}$, $Q = \{q_0\}$, $F_A = \{q_0\}$, $F_R = \varnothing$, $\delta_1 = \{(q_0, a_0, a_n, q_0)\}$. It is obvious to see that the language $\Sigma^*_{\leq}$ can be recognized by $M_1$. It follows that the language $\Sigma^*_{\leq}$ is a 1-acceptable language.

Let $L_\infty = \Sigma^*_{\leq}$ and $L_n = \cup_{i=0 \text{ to } n}\Sigma^i_{\leq}$ for $n \in \mathbb{N}$. These languages are 1-acceptable languages. Since $L_0 \subset L_1 \subset ... \subset L_n \subset ...$ and $L_\infty = \cup_{i=0 \text{ to } n} L_n$ and, thus it follows that $L_\infty$ is a limit point of 1-$ACC$. ■

**Proposition 3.5** *For any integer $k \geq 1$, the language class k-ACC has a limit point.*

**Proof:** It is obvious from proposition 3.3 and 3.4. ■

## 4. LEARNABILITY OF THE CLASS OF *k*-ACCEPTABLE LANGUAGES

Language learnability is a property of classes of languages, not for a language. Any language class containing this property means there exists at least one algorithm that is learnable every language in the class on some types of presentation. In this section, we study on learnability of the class of *k*-acceptable languages by using two different types of presentation. For the first one, only positive examples are available in learning context. And for the second one, negative examples are additionally given.

### 4.1 Learning *k*-acceptable Languages by Using only Positive Examples

We show that the class of *k*-acceptable languages is not learnable from positive examples.

**Theorem 4.1** *For any integer $k \geq 1$, language class k-ACC is not learnable in terms of k-DFAs by using only positive examples.*

**Proof:** By proposition 3.5, we now know that the language class *k*-ACC has a limit point for $k \geq 1$. By definition, it implies that *k*-ACC language class is not learnable in terms of *k*-DFAs by using only positive examples. ■

### 4.2 Learning *k*-acceptable languages by using both positive and negative examples

This section we show that the class of *k*-acceptable languages is learnable in terms of *k*-edge deterministic finite automata by using both positive and negative examples.

We firstly give a learning algorithm called *KACLI* that is learnable *k*-acceptable languages. In order to prove learnability, time complexity and existence of characteristic samples will also be shown in this section.

### 4.2.1 The learning algorithm *KACLI*

In this section, we describe a learning algorithm from positive and negative examples (shown in Figure 2) for *k*-acceptable languages. This algorithm is called *KACLI* for learning *k*-acceptable languages in terms of *k*-DFAs from positive and negative examples. The idea is to greedily merge states in order to find a solution that is always consistent with a given set of examples. The incompatibility constraint is used to prevent merging incompatible states that may lead to build an inconsistent *k*-DFA. An augmented prefix tree acceptor (APTA) consistent with a given sample is initially built to hold all information from the given examples. Pairs of states for merging are chosen in lexicographic-length order. The merging process may return temporary solution which may be nondeterministic. To reduce the non-determinism of temporary solution in learning process, a tree invariant property is maintained by considering two states to be merged; at least one of them is the root of a tree. The property of tree invariant is a sufficient condition for the determinization process to be finite and also helps implementing simple and fast. In the best of cases, if a characteristic set of the language includes in the random sample then the proposed algorithm returns the correct target *k*-DFA.

*KACLI* algorithm starts from building a *k*-edge augmented prefix tree acceptor consistent with a given set of examples and return a *k*-DFA as an output. The set $M$ in this algorithm is a set of states, which are selected for merging. The main state-merging loop begins with choosing a state $q_\alpha$ in set *Q-K* by using function *CHOOSE*. The function *CHOOSE* returns a smallest state $q_\alpha$ in the set *Q-K* in lexicographic-length order. We then make the state $q_\alpha$ secure by using function *MAKE_SECURE*. This function is used to merge any consecutive children of $q_\alpha$ which they are not incompatible. The merge process is recursively performed to merge children states $q_\beta$ into their previous states $q_\omega$ in the set $M$ by using the function *REC_MERGE*. The incompatibility constraint is considered to prevent leading to obtain inconsistent *k*-DFA. In *KACLI* algorithm, the function *COMPATIBLE* is used to check whether the *k*-DFA is consistent to the given set of

**Algorithm :** *KACLI*
**Input** : positive and negative samples
$S = (S+, S-)$
**Output** : a *k*-DFA $M = (\sum_{\leq}, Q, q_0, F_A, F_R, \delta)$
$M \leftarrow BUILD\_APTA(S)$
$K \leftarrow \varnothing$
**While** $Q\text{-}K \neq \varnothing$ **do**
$q_\alpha \leftarrow CHOOSE(Q\text{-}K)$
$K \leftarrow K \cup \{q_\alpha\}$
$M \leftarrow MAKE\_SECURE(M, q_\alpha)$
$B \leftarrow \{ q_\beta : q_\beta \in \delta(q_\alpha, a, b) : a, b \in \sum_{\leq} \}$
**For** $q_\beta \in B$ **do**
**For** $q_\omega \in K$ **do**
$M_{new} \leftarrow REC\_MERGE\ (M, q_\omega, q_\beta)$
**If** $COMPATIBLE(M_{new}, S)$
**Then**
$M \leftarrow M_{new}$
**Endif**
**Endfor**
$K \leftarrow K \cup \{q_\beta\}$
**Endfor**
**Endwhile**
**Return** $M$

**Figure 2.** *KACLI* algorithm.

examples. If it returns false then the merging $q_\beta$ into $q_\omega$ is allowed. The algorithm terminate when all state in $Q$ is considered.

### 4.2.2 Time analysis of the *KACLI* algorithm

We analyze the time complexity of *KACLI*.

**Lemma 4.1** *The algorithm KACLI runs in* $\mathcal{O}(\|S\|^4)$.

**Proof** : We prove this statement by considering computing time of each functions in *KACLI*. Firstly note that *KACLI* begin with building an APTA by the function *BUILD_APTA*. The computing time of this function is bounded with the size of APTA. That is $\|S\|$. It follows that function *BUILD_APTA* computes in $\mathcal{O}(\|S\|)$.

Then, we consider functions in the while-loop. Function *CHOOSE* is used for choosing a state, so this function runs at most |APTA|. It follows that computing time of this function is $\mathcal{O}(\|S\|)$. Function *MAKE_SECURE* is used for reducing numbers of successor of a given state. We will see that its computing time is bounded with $\mathcal{O}(|\Sigma_{\leq}|)$. Function *COMPATIBLE* is used for checking compatibility of $k$-DFA and a given sample. It computes at most $\mathcal{O}(\|S\|^2)$. For function *REC_MERGE*, a state in $B$ will be recursively merged into a state in $K$. The time of this step is bounded by $\mathcal{O}(\|S\|^2)$. Thus, we will see that computing time for each iteration is $\mathcal{O}(\|S\|^2)$.

For the number of iteration, we will see that it is depended on two sets $B$ and $K$ that is a subset of $Q$ in a $k$-DFA. So, the number of iteration is $\mathcal{O}(\|S\|^2)$. It follows that *KACLI* runs in $\mathcal{O}(\|S\|^4)$ for while-loop. Therefore we can conclude that the algorithm *KACLI* runs in $\mathcal{O}(\|S\|^4)$. ■

### 4.2.3 Characteristic sets of the *KACLI* algorithm

In this section, we show the existence of a characteristic set $CS = (CS+, CS-)$ of $k$-acceptable languages $L$ for *KACLI*. The $CS$ ensures the *KACLI* algorithm will return a $k$-DFA $M_k = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ such that $\mathbb{L}(M_k) = L$.

To construct a characteristic set, we need to define the *short prefix* of a state $q \in Q$ denoted by $Short(q)$, the set of short prefixes of $L$ denoted by $SP(\mathbb{L}(M_k))$, and the *kernel set* of $L$ denoted by $N(\mathbb{L}(M_k))$ as follows:

- $Short(q) = \min\{u \in \Sigma^* : \delta_k^*(q_\lambda, u) = q\}$,
- $SP(\mathbb{L}(M_k)) = \{u \in \Sigma^* : \forall q \in Q,\ u = Short(q)\}$,
- $N(\mathbb{L}(M_k)) = \{\lambda\} \cup \{uz : \forall q \in Q,\ u = Short(q),\ z = Lb(\delta_k(q, a, b))\} \cup \{uz : \forall q \in Q,\ u = Short(q),\ z = Ub(\delta_k(q, a, b))\}$.

The set $CS = (CS+, CS-)$ is a characteristic set of $\mathbb{L}(M_k)$ for the algorithm *KACLI* if it satisfies the following conditions:

- $\forall u \in N(\mathbb{L}(M_k))$, if $\delta_k^*(q_0, u) \in F_A$ then $u \in CS+$ and if $\delta_k^*(q_0, u) \in F_R$ then $u \in CS-$,
- $\forall u \in SP(\mathbb{L}(M_k)),\ \forall v \in N(\mathbb{L}(M_k))$, if $\delta_k^*(q_0, u) \neq \delta_k^*(q_0, v)$ then $uw \in CS+$ and $vw \in CS-$ or $vw \in CS+$ and $uw \in CS-$, where $w$ is a distinguishing string formally defined as $w = \min\{w \in \Sigma^* : (\delta_k^*(q_u, w) \in F_A \wedge \delta^*(q_v, w) \in F_R) \wedge (\delta_k^*(q_u, w) \in F_R \wedge \delta_k^*(q_v, w) \in F_A)\}$.

**Example 4.1** *Construct a characteristic set* $CS = (CS+, CS-)$ *for k-DFA* $M_k = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ *in Figure 1.*

**Solution** For each $q \in Q$, the short prefixes of state $q_0$ and state $q_3$ are "λ" and "4", respectively.

We construct the set of short prefixes of $L$ recognized by $M_k$ as

$SP(\mathbb{L}(M_k)) = \{\lambda, 4\}$.

Then, we construct the kernel set of $L$ from $SP(\mathbb{L}(M_k))$. So we have

$N(\mathbb{L}(M_k)) = \{\lambda, 1, 3, 4, 5, 41, 42, 45\}$.

Finally, we have $CS(M_k) = (CS+, CS-)$ such that $CS+ = \{\lambda, 1, 3, 42, 45\}$ and $CS- = \{4, 5, 41\}$. ■

Let $\mathcal{A}$ denote the *KACLI* algorithm that returns $k$-DFA from a given input set $S = (S+, S-)$ and $CS = (CS+, CS-)$ be a characteristic set of a target $k$-acceptable language $L$ recognized by $k$-DFA $M_k$. From the definition of the characteristic set, we now must show the following lemmas:

**Lemma 4.2** $\mathbb{L}(\mathcal{A}(CS)) \subseteq \mathbb{L}(M)$.

**Proof:** (By induction) Let $L$ be a target $k$-acceptable language on an ordered alphabet $\Sigma_{\leq}$, and we denote two $k$-DFAs in step of proving as follows:

- $M = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta)$ is a canonical $k$-DFA for a target $L$.
- $M_{(i)} = (\Sigma_{\leq}, Q_{(i)}, q_0, F_{A(i)}, F_{R(i)}, \delta_{(i)})$ is a $k$-DFA built from *KACLI* algorithm at $i$th iteration.

In order to show that $\mathbb{L}(A(CS)) \subseteq \mathbb{L}(M)$, we have to show that the automaton $M_t$, that we build at any moment, is a subautomaton of $M$ for a target language $L$.

We now define a homomorphism $f: Q_{(i)} \rightarrow Q$ as $f(q) = \delta^*(q_0, u_q)$ for each $q \in Q_{(i)}$, where $u_q$ is a prefix of strings in the given $CS$ such that the number of transitions in $\delta^*_{(i)}(q_0, u_q) = q$ is minimum. So, we will show that $M_{(i)}$ is homomorphic to a subautomaton of $M$ by showing the following conditions hold:

(*i*) if $q$ is in $Q_{(i)}$, then $f(q)$ is in $Q$,
(*ii*) if $q$ is in $F_{A(i)}$, then $f(q)$ is in $F_A$,
(*iii*) if $q$ is in $F_{R(i)}$, then $f(q)$ is in $F_R$,
(*iv*) for each $q \in Q_{(i)}$, and for each $a \in \Sigma_{\leq}$, if $\delta_{(i)}(q, a, b) = p$, then $f(\delta_{(i)}(q, a, b)) = \delta(f(q), a, b)$.

*Basis step*: [show $\mathbb{L}(M_0) \subseteq \mathbb{L}(M)$]

As *KACLI* initiates with building a prefix tree acceptor for $CS$, we have that $M_{(0)} = \text{APTA}(CS) = (\Sigma_{\leq}, Q_{(0)}, q_0, F_{A(0)}, F_{R(0)}, \delta_{(0)})$, where $q_0 = q_\lambda$, $Q_{(0)} = \{q_u : u \in Pref(CS)\}$, $F_{A(0)} = \{q_u \in Q_{(0)}: u \in CS+\}$, $F_{R(0)} = \{q_u \in Q_{(0)}: u \in CS-\}$, $\delta_{(0)}(q_u, a, a) = q_{ua}$ such that $u, ua \in Pref(CS)$.

To show (*i*), we suppose $q_u \in Q_{(0)}$. Since $u \in Pref(CS) \subseteq Pref(\mathbb{L}(M))$ and $\delta^*(q_0, u) \in Q$, so we have that $f(q_u) \in Q$ by the defined homomorphism.

To show (*ii* and *iii*), we suppose $q_u \in F_{A(0)}$ and then we have that $u \in CS+$ such that $CS+ \subseteq \mathbb{L}(M)$. Thus, $\delta^*(q_0, u) \in F_A$ implies that $f(q_u) \in F_A$ by definition. Similarly we suppose $q_u \in F_{R(0)}$ and then we have that $u \in CS-$ such that $CS- \subseteq \Sigma^*_{\leq} - \mathbb{L}(M)$. Thus, $\delta^*(q_0, u) \in F_R$ implies that $f(q_u) \in F_R$ by definition.

To complete proving in the basis step, we will show (*iv*) hold. For $q, p \in Q_0$ such that $\delta_{(0)}(q, a, b) = p$, we let $u_p = \{u_q z : a \leq z \leq b\}$. From the manner of choosing $u_p$ in APTA($CS$), when $u_q$ is a prefix of string in $CS$ such that the number of transition in $\delta^*_{(0)}(q_0, u_q) = q$ is minimum, then $\delta^*_{(0)}(q_0, u_p) = p$ is minimum as well. Since $CS \subseteq \mathbb{L}(M)$, so we have that $f(p) = \delta^*(q_0, u_p)$.

R.H.S. $= \delta(f(q), a, b)$
$= \delta(\delta^*(q_0, u_q), a, b)$; by definition
$= \delta^*(q_0, u_q z)$ ; $a \leq z \leq b$
$= \delta^*(q_0, u_p)$ ; $u_q z = u_p$
$= f(p)$ ; by definition
$= f(\delta_0(q, a, b))$ ; $\delta_0(q, a, b) = p$
$=$ L.H.S

Therefore, we have $f(\delta_{(0)}(q, a, b)) = \delta(f(q), a, b)$.

The above proving (*i*), (*ii*), (*iii*) and (*iv*) imply that $M_{(0)}$ is homomorphic to a subautomaton of the canonical $k$-DFA $M$. Thus, the statement $\mathbb{L}(M_{(0)}) \subseteq \mathbb{L}(M)$ is true.

*Inductive step*: [show if $\mathbb{L}(M_{(t)} \subseteq \mathbb{L}(M)$ then $\mathbb{L}(M_{t+1}) \subseteq \mathbb{L}(M)$] ■

In this step of proving, we suppose this

lemma holds for $M_{(t)}$ and we show that it also holds for $M_{(t+1)}$. From *KACLI* algorithm, $M_{(t+1)}$ is derived from merging states in $M_{(t)}$. Thus, we distinguish possible state merging into three cases.

*Case 1:* [$M_{(t+1)} = Merge(M_{(t)}, q_\omega, q_\beta)$ such that $\delta_{(t)}(q_u, a_1, b_1) = q_\omega$ and $\delta_{(t)}(q_u, a_2, b_2) = q_\beta$]. From merging with this case, we have $Q_{(t+1)} = Q_{(t)} - \{q_\beta\} \subseteq Q_{(t)}$, $F_{A(t+1)} = F_{A(t)} - \{q_\beta\} \subseteq F_{A(t)}$, $F_{R(t+1)} = F_{R(t)} - \{q_\beta\} \subseteq F_{R(t)}$, and $\delta_{(t+1)} = \delta_{(t)} - (\{(q_u, a, b, q_\beta)\} \cup \{(q_\beta, a, b, q_v) : q_v \in Q_{(t)}\}) \subseteq \delta_{(t)}$. By supposition, it is obvious that the conditions (*i*), (*ii*), (*iii*) and (*iv*) holds for $M_{(t+1)}$.

*Case 2*: [$M_{(t+1)} = Merge(M_{(t)}, q_\omega, q_\beta)$ such that $\delta_{(t)}(q_u, a_1, b_1) = q_\omega$ and $\delta_{(t)}(q_v, a_2, b_2) = q_\beta, q_u \neq q_v$]. From merging with this case, we have $Q_{(t+1)} = Q_{(t)} - \{q_\beta\} \subseteq Q_{(t)}$, $F_{A(t+1)} = F_{A(t)} - \{q_\beta\} \subseteq F_{A(t)}$, $F_{R(t+1)} = F_{R(t)} - \{q_\beta\} \subseteq F_{R(t)}$, and $\delta_{(t+1)} = \delta_{(t)} - (\{(q_v, a, b, q_\beta)\} \cup \{(q_\beta, a, b, q_v) : q_v \in Q_{(t)}\}) \cup \{(q_v, a, b, q_\omega)\} \subseteq \delta_t$. Thus, it is obvious that the conditions (*i*), (*ii*), (*iii*) and (*iv*) hold for $M_{(t+1)}$ by supposition for $M_{(t)}$.

*Case 3*: [$M_{(t+1)} = M_{(t)}$ because $Merge(M_{(t)}, q_\omega, q_\beta)$ fails] From merging with this case, we have $Q_{(t+1)} = Q_{(t)}$, $F_{A(t+1)} = F_{A(t)}$, $F_{R(t+1)} = F_{R(t)}$, and $\delta_{(t+1)} = \delta_{(t)}$. Thus, it is obvious that the conditions (*i*), (*ii*), (*iii*) and (*iv*) holds for $M_{(t+1)}$ by supposition for $M_{(t)}$. With three cases of the merging, we have that the statement is true. Thus, it follows that this lemma is true. ■

**Lemma 4.3.** $\mathbb{L}(M) \subseteq \mathbb{L}(\mathcal{A}(CS))$.

**Proof:** To prove this lemma, let $M_{(n)} = (\Sigma_{\leq}, Q_n, q_0, F_{A(n)}, F_{R(n)}, \delta_{(n)})$ is *k*-DFA returned from *KACLI*, i.e. $A(CS) = M_{(n)}$. That means we must show that $M$ is a subautomaton of $M_{(n)}$.

*Prove (i)*: [If $q$ is in $Q$, then $f(q)$ is in $Q_{(n)}$] We suppose that $q \in Q$. If $q \in F_A$ then clearly $u_q \in CS+$ by definition of the characteristic sample. It follows that $u_q \in \mathbb{L}(M_{(n)})$. Hence, $\delta^*(q_0, u_q) \in Q_n$. It follows that $f(q) \in Q_{(n)}$ because of $f(q) = \delta^*(q_0, u)$ by definition. If $q \in F_R$ then clearly $u_q \in CS-$ by definition of the characteristic sample. It follows that $u_q \in \Sigma^* - \mathbb{L}(M_{(n)})$. Hence, $\delta^*(q_0, u_q) \in Q_{(n)}$. If follows that $f(q) \in Q_{(n)}$ because of $f(q) = \delta^*(q_0, u)$ by definition. Therefore, if $q$ is in $Q$, then $f(q)$ is in $Q_{(n)}$.

*Prove (ii)*: [If $q$ is in $F_A$, then $f(q)$ is in $F_{A(n)}$] We suppose $q$ is in $F_A$. It follows that $u_q \in CS+$ by definition of the characteristic sample. Then, We have $\delta^*(q_0, u_q) \in F_{A(n)}$ by manner of *KACLI*. It follows that $f(q) \in F_{A(n)}$. Therefore, if $q$ is in $F_A$, then $f(q)$ is in $F_{A(n)}$.

*Prove (iii)*: [If $q$ is in $F_R$, then $f(q)$ is in $F_{R(n)}$] We suppose $q$ is in $F_R$. It follows that $u_q \in CS-$ by definition of the characteristic sample. Then, We have $\delta^*(q_0, u_q) \in F_{R(n)}$ by manner of *KACLI*. It follows that $f(q) \in F_{R(n)}$. Therefore, if $q$ is in $F_R$, then $f(q)$ is in $F_{R(n)}$.

*Prove (iv)*: [For every $q, p \in Q$, and every $a \in \Sigma_{\leq}$ such that $\delta(q,a,b) = p, f(\delta(q,a,b)) = \delta_{(n)}(f(q),a,b))$]

R.H.S. $= \delta_{(n)}(f(q), a, b))$
$= \delta_{(n)}(\delta^*_n(q_0, u_q), a, b))$ ; by definition
$= \delta^*_{(n)}(q_0, u_q z)$ ; $a \leq z \leq b$
$= \delta^*_{(n)}(q_0, u_p)$ ; $u_q z = u_p$
$= f(p)$ ; by definition
$= f(\delta(q, a, b))$ ; $\delta(q, a, b) = p$
$=$ L.H.S.

As (*i*), (*ii*), (*iii*), and (*iv*) hold, so this lemma is true. ■

**Lemma 4.4** *If $CS(M) \subseteq S \subseteq L$ then $\mathbb{L}(\mathcal{A}(S)) = \mathbb{L}(M)$.*

**Proof:** It is obvious that the proof is obtained from Lemma 4.2 and Lemma 4.3. ■

**Theorem 4.2** *There exist a characteristic set of k-acceptable language for KACLI algorithm.*

**Proof:** By Lemma 4.2, Lemma 4.3 and Lemma 4.4, we conclude that there exist a

characteristic set of *k*-acceptable language for *KACLI* algorithm. ■

**Theorem 4.3** *The size of characteristic sets for k-acceptable languages is $O(n^3k)$ such that n is size of k-DFA recognizing the language.*
**Proof:** By constructing of the characteristic set, the size of short prefix set $|SP(\mathbb{L}(M))|$ is as many as the number of states of the canonical *k*-DFA $M = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta)$. Suppose $|Q| = n$, so we have that $|SP(\mathbb{L}(M))| = n$. It clear that the number of strings in $N(\mathbb{L}(M))$ is $n \times 2k + 1$. Therefore the number of strings in $CS = (CS+, CS-)$ is $|CS+| < 2n^2 \times k$ and $|CS-| < 2n^2 \times k$. In the worst case, the maximum length of strings in $SP(\mathbb{L}(M))$ is equal to $n$ and $N(\mathbb{L}(M))$ is equal to $n+1$. For the maximum length of a distinguishing string $w$ is equal $n$. By considering the size of the strings in *CS*, we have that the possible length of strings in *CS* is less than $2n+1$. Thus, it show that $||CS|| \in O(n^3k)$. ■

**Theorem 4.4** *The class of k-acceptable languages is learnable in the limit from polynomial time and data in term of k-edge deterministic finite automata by using both positive and negative examples.*
**Proof:** By theorem 4.2, 4.3 and lemma 4.4, we conclude that the class of *k*-acceptable languages is learnable in the limit from polynomial time and data. ■

## 5. CONCLUSION

In this work we study learnability of the *k*-acceptable languages on Gold's learning model with two different types of presentation. First, we have proved that the language class is not learnable in the limit from only positive examples. For the second result, the class of *k*-acceptable languages is shown that it is learnable in the limit from polynomial time and data from positive and negative examples. To do so, we propose *KACLI* algorithm that polynomially identifies *k*-edge deterministic finite automata in the limit. We have also studied existence of characteristic sets for the class and have shown that its size is polynomial.

## REFERENCES


[1] Gold E.M., Language identification in the limit, *Inform. Control*, 1967; **10(5):** 447-474.

[2] Angluin D., Queries and concept learning, *Mach. Learn. J.*, 1987; **2:** 319-342.

[3] Valiant L.G., A theory of the learnable, *Comm. ACM*, 1984; **27(11):** 1134-1142

[4] Pitt L., Inductive inference, DFA's, and computational complexity, Proc. of the Int. Workshop AII'89, 1989; 18-44.

[5] de la Higuera C., Characteristic sets for polynomial grammatical inference, *Mach. Learn. J.*, 1997; **27:** 125-138.

[6] de la Higuera C., A bibliographical study of grammatical inference, *Pattern Recogn.*, 2005; **38:** 1332-1348.

[7] Garcia P. and Vidal E., Inference of k-testable languages in the strict sense and applications to syntactic pattern recognition, *IEEE Trans. Pattern Anal. Mach. Intell.*, 1990; **12:** 920-925.

[8] Yokomori T. and Kobayashi S., Learning local languages and their application to DNA sequence analysis, *IEEE Trans. Pattern Anal. Mach. Intell.*, 1998; **20(10):** 1067-1079.

[9] Yokomori T., On polynomial-time learnability in the limit of strictly deterministic automata, *Mach. Learn. J.*, 1995; **19:** 153-179.

[10] Sakakibara Y., Grammatical Inference in Bioinformatics, *IEEE Trans. Pattern Anal. Mach. Intell*., 2005; **27(7):** 1051-106.

[11] Cruz P. and Vidal E., Learning regular grammars to model musical style: Comparing different coding scheme, Proc. of ICGI'98, 1998; 211-222.

[12] García P. *et al*., On the use of the morphic generator grammatical inference (mggi) methodology in automatic speech recognition, *Int. J. Pattern Recogn. Artif. Intell*., 1994; **4:** 667-685.

[13] de la Higuera C., Ten open problems in grammatical inference, Proc. of ICGI'06, 2006; 32-44.

[14] Gold E. M., Complexity of automaton identification from given data, *Inform. Control*, 1978; **37:** 302-320.

[15] Dupont P., Utilisation et apprentissage de modeles de langage pour la reconnaissance de la parole continue. Doctoral dissertation, l' ENST Paris, 199.