Journal of Theoretical and Applied Information Technology
31st December 2017. Vol.95. No 24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DECIDABILITY PROPERTIES OF THE CLASS OF FORMAL LANGUAGES RECOGNIZED BY K-EDGE FINITE STATE AUTOMATA

[1]ANUCHIT JITPATTANAKUL
[1]Intelligent Control and Nonlinear Research Center
Department of Mathematics, Faculty of Applied Science
King Mongkut's University of Technology North Bangkok, Bangkok, Thailand
E-mail: [1]anuchit.j@sci.kmutnb.ac.th

ABSTRACT

Finite state automata (FSA) are computational models which are used for studying theoretical complexity of computational problems. The computational model of *k*-edge finite state automata (*k*-FSA) have been introduced to compute naturally ordered data such as data in music processing and time series domain. The *k*-edge finite state automata contain theoretical interest that numbers of their edges are bounded with a *k* value. In order to use such model in practice, some theoretical properties need to be investigated by way of their classes of formal languages. This paper investigates the closure properties and decidability of the class of formal languages recognized by *k*-edge finite state automata called *k*-acceptable languages (*k*-ACC). The results include the following: (i) for $k \geq 1$, the class of *k*-acceptable languages is closed under complement. (ii) for $k \geq 2$, the class of *k*-acceptable languages is closed under complementation, union, intersection, concatenation and Kleene-star operation. (iii) The infiniteness problem, membership problem, equivalence problem and emptiness problem of *k*-acceptable languages are decidable.

Keywords: *Decidability, Closure Properties, K-Acceptable Languages, K-Edge Finite State Automata*

1. INTRODUCTION

Formal language theory is a research field originated from attempt to understand how human learns natural languages by Noam Chomsky in 1950s [1]. The paper defined the syntax of languages using precise mathematical rules called grammars. The research results have greatly benefited to many fields of computer science, including programming languages, compiler design, and artificial intelligence. After the advent of modern electronic computers, mathematical model of computer known as automata plays an important role of model of computation. Specifically, this computational model can be viewed a recognizer of the formal languages.

Nowadays, the formal languages and grammatical representations have many applications in other fields, including molecular biology and symbolic dynamic [2]. However theoretical properties have been studied in parallel to better understand the powerful of computational models. Some of these are closure properties and decidability of computational problems.

Closure properties of various formal languages under different operation are of considered theoretical interest [3]. The closure properties are often useful in constructing new languages from existing languages. That is every application of the operation on languages of the class yield a language of the class. Moreover, the established closure properties could be also use to prove some decision problems known as decidability. In the field of research, various classes of formal languages have been investigated this theoretical properties, including formal language classes in Chomsky hierarchy (regular languages, context-free languages, context-sensitive languages and recursively enumerable languages) shown in Fig. 1. Moreover, other classes such as quantitative

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

languages [4], quantum Turing languages [5], and ordered languages [6] have also investigated their closer properties and decidability.

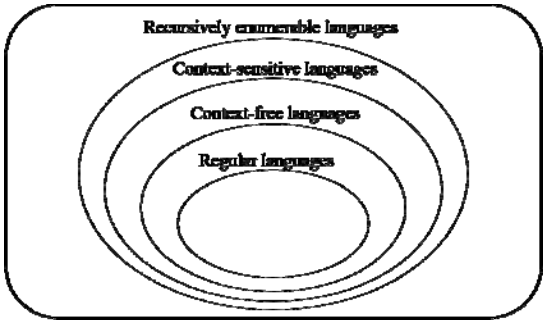


Fig 1. Chomsky hierarchy

In this paper we focus on the theoretical properties of class of formal languages called *k*-acceptable languages denoted by *k*-ACC. The studied languages is a subclass of regular languages recognized by *k*-edge finite state automaton, shortly called *k*-FSA, has been introduced by Higuera in [7] to be a computational model recognized strings defined on an ordered alphabet. These strings can be found in many real-world applications. For example the alphabet of melody is naturally ordered in the case of music [8]. Another example, in time series domain, the discretization of original data in numeric to discrete data needs to keep topological characteristics [9]. The interesting property of *k*-edge finite state automata is that its size depends on the *k* value instead of the size of alphabet. This is a reason why the study of *k*-edge finite state automata is useful and interesting.

The classes of formal languages that recognized by *k*-edge finite state automata have been studied in [10-11] to investigate their learnability. The results have shown that 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. However, there are no literatures concerned with theoretical properties of the language class. Research objectives for this work is to investigate the closure properties and decidability of the class of *k*-acceptable languages.

This paper is organized as follows. In section 2, we give preliminaries composing of related definitions and some examples. Section 3 studies algebraic properties of *k*-edge finite state automata. Section 4 investigates closure properties of *k*-acceptable languages. In section 5, we show that infiniteness problem, membership problem, equivalence problem and emptiness problem of the class of *k*-acceptable languages are decidable. The final section is conclusion.

2. PRELIMINARIES

The related definitions and notations used throughout this paper are provided in this section.

2.1 Formal languages and automata

Let Σ be an alphabet that is a finite and nonempty set of characters. The size of Σ is a number of letters, denoted by $|\Sigma|$. A finite sequence of letters from Σ 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 λ. The infinite set of all possible strings over Σ, denoted by Σ^*, is the set of all finite-length strings generated by concatenating zero or more letters of Σ. An alphabet Σ is called an ordered alphabet denoted by $\Sigma_{\leq}$. Given an order relation $\leq$ on Σ, we can define a lexicographic-length order over Σ^* 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 formal language over Σ denoted by L is any subset of Σ^*. The family of languages over Σ, denoted by Λ, is called a class of languages.

Definition 2.1

A *finite state automaton* (FSA) is a grammatical representation that is typically defined as a 5-tuple $M = (\Sigma, Q, q_0, F, \delta)$, where

- Σ 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
- $\delta : Q \times \Sigma \rightarrow Q$ is a state transition function.

The state transition function δ can be extended to a mapping $\delta^*: Q \times \Sigma^* \rightarrow Q$ in the following inductive way:

(*i*) $\delta^*(q, \lambda) = q$, for each state $q \in Q$, where λ is the null string, and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

(*ii*) $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$, for each state $q \subseteq Q$, each letter $a \in \Sigma$, and each string $w \subseteq \Sigma^*$.

An example of a finite state automata is shown in Fig. 2.

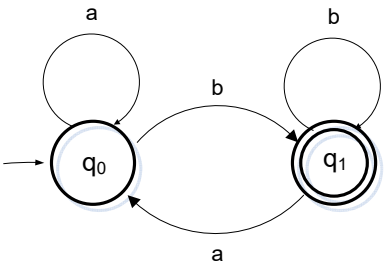


Fig 2. a finite state automata

Theoretically, an automaton plays an important role as a language recognizer. A string w is recognized by an finite state automaton $M = (\Sigma, Q, q_0, F, \delta)$ if $\delta^*(q, w) \in F$. The language recognized by M, denoted by $L(M)$, is the set of all strings that 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 $L = L(M)$.

Because of size of finite state automata, an automaton called k-edge finite state automaton has been introduced. They contains very nice properties with numbers of their edges are bounded with a k value. The definition are shown as follows.

Definition 2.2

A *k-edge finite state automaton* (k-FSA) is a 6-tuple denoted by $M_k = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ 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,
- $F_R \subseteq Q$ is a set of rejecting states, and
- $\delta_k : Q \times \Sigma_{\leq} \times \Sigma_{\leq} \rightarrow Q$ is the transition function.

The transition function δ_k 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 $x \leq a \leq y$ and $\delta_k(q, x, y) = q'$ such that $q, q' \in Q, a, x, y \in \Sigma_{\leq}, w \in \Sigma_{\leq}^*$.

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

Definition 2.3

Let $M_k = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ be a k-edge finite state automata. The *transition graph* of M_k, denoted by $\Delta(M_k)$, is a directed graph, where nodes are states from Q, and there is an edge from p to q labeled with an interval $[a, b]$ if and only if $(p, a, b, q) \in \delta_k$. A state $q \in Q$ is *reachable* if there is a walk from q_0 to q in $\Delta(M_k)$. A state $q \in Q$ is *terminating* if there is a walk from q to some $f \in F_A$.

Definition 2.4

Languages recognized by k-edge finite state automata $M_k = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ 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 $k \geq 0$.

Example 2.1

Let L be a formal language defined over $\Sigma_{\leq} = \{a_1, a_2, a_3, a_4, a_5\}$. The formal language L defined by regular expression as

$((a_1+a_2+a_3) + (a_4+a_5)\, a_1^*(a_2+a_3+a_4+a_5))^*$.

This automata accepting the language L is 2-edge finite state automaton because for any $q \in Q$, $|\{(x, y) : \delta_2(q, x, y) \neq \varnothing\}| = 2$ and for state $q_0 : \delta_2(q_0, a_1, a_3) \neq \delta_2(q_0, a_4, a_5) \neq \varnothing$ then $\{z : a_1 \leq z \leq a_3\} \cap \{z : a_4 \leq z \leq a_5\} = \varnothing$ for state $q_3 : \delta_2(q_3, a_1, a_1) \neq \delta_2(q_3, a_2, a_5) \neq \varnothing$ then $\{z : a_1 \leq z \leq a_1\} \cap \{z : a_2 \leq z \leq a_5\} = \varnothing$. The 2-edge finite automata in this example is depicted in Fig. 3

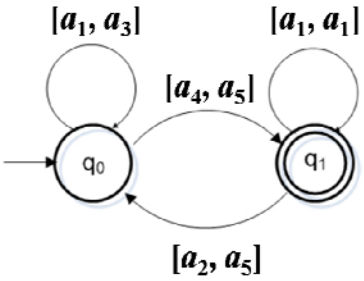


Fig 3. 2-edge finite state automata

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2.2 Closure properties

Theoretically, a set is closed under an operation if applying that operation to any members of the set yields a members of the set. In formal language theory, the closure is a property about classes of languages, not about individual strings.

Definition 2.5

Let be Λ a class of languages and L_1, L_2, L_3,…, L_n be languages in Λ. The language class Λ is said to be *closed* under an operation *op*, if for all languages L_1, L_2, L_3,…, L_n in Λ then we have $op(L_1, L_2, L_3,\ldots, L_n)$ in Λ.

Investigation of the closure properties of a class of formal languages is one of the most interesting and fundamental research tasks in formal languages theory. Normally any progress leads to insights and techniques that yield a better understanding of the class. In case of languages in Chomsky's hierarchy, the results are shown in Table 1 as below.

Table 1 Closure properties of classes in Chomsky's hierarchy

Operation	Class of languages			
	REG	CFL	CSL	RE
Union	Yes	Yes	Yes	Yes
Intersection	Yes	No	Yes	Yes
Complementation	Yes	No	Yes	No
Difference	Yes	No	Yes	No
Reverse	Yes	Yes	Yes	Yes

2.3 Decidability

Decidability is an important property concerning to ask decision problems about formal languages. Informally, we use the word problem to refer to a question such as "Is a given *k*-acceptable language finite?". By restricting our attention to problems with yes-no answers and encoding instances of the problem by strings over some finite alphabet, we can transform the question to whether or not there exists an algorithm for solving a problem. A problem whose language there exists an algorithm for solving is said to be decidable. Otherwise, the problem is undecidable. That is, a problem is undecidable if there is no algorithm that takes as input and instance of the problem and determines whether the answer to that instance is "yes" or "no".

3. *k*-EDGE FINITE STATE AUTOMATA AND ALGEBRAIC PROPERTIES

To investigate closure properties and decidability of the class of *k*-acceptable languages, some algebraic properties are needed. In this section we give some propositions that will be referred in next section.

Proposition 3.1 Let $M_k = (\Sigma_\leq, Q, q_0, F_A, F_R, \delta_k)$ be *k*-edge finite state automata, we can always construct a finite state automata represents by $M_d = (\Sigma, Q_d, q_{0d}, F_d, \delta_d)$ such that $L(M_k) = L(M_d)$.

Proof: Let $M_k = (\Sigma_\leq, Q, q_0, F_A, F_R, \delta_k)$ defined on the followings.

- $\Sigma_\leq = \{a_1, a_2, \ldots, a_n\}$,
- $\delta_k = \{(p, a, b, q) : p, q \in Q$ and $a, b \in \Sigma_\leq\}$ such that $\#(\delta_k) = k$.

From the definition 2.2, the M_k is a *k*-edge finite state automata recognizing a language *L*.

We can construct a finite state automata recognized the language *L* as follows. We give a finite state automata defined by

$$M_d = (\Sigma, Q_d, q_{0d}, F_d, \delta_d)$$

such that

- $\Sigma = \{a : a \in \Sigma_\leq\}$,
- $Q_d = Q$,
- $q_{0d} = q_0$,
- $F_d = F_A$ and
- $\delta_d = \{(p, z, q) : a \leq z \leq b$ for all $(p, a, b, q) \in \delta_k\}$.

From this construction, it follows that M_d recognizes *L*. Therefore, $L(M_k) = L(M_d)$. +

Proposition 3.2 For any integer $k \geq 0$, a *k*-FSA is a $(k+1)$-FSA.

Proof: Let $M_k = (\Sigma_\leq, Q, q_0, F_A, F_R, \delta_k)$ be a *k*-edge finite state automata.

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 $\{z : a_1 \leq z \leq b_1\} \cap \{z : a_2 \leq z \leq b_2\} = \varnothing$.

For all integer *k*, it is algebraically obvious that

$$k < k + 1.$$

It follows that $|\{[x, y] : \delta_k(q, x, y) \neq \varnothing\}| \leq k+1$. Therefore, M_k is a $(k+1)$-edge finite state automata by definition. +

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Proposition 3.3 For any integer $k \geq 0$, k-ACC $\subseteq$ (k+1)-ACC.

Proof: Let $M_k = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ be a k-edge finite state automata 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)-finite state automata to recognize the language L_k.

Suppose $M_m = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_m)$ is a m-edge finite state automata. We define the transition function δ_{k+1} of M_m as

$\delta_m = \delta_k - \{(q_0, a_0, a_i, p) : a_i \in \Sigma_{\leq}, p \in Q\} \cup \{(q_0, a_0, a_0, p), (q_0, a_1, a_i, p)\}$.

It follows that

for all $q \in Q$, $|\{[x, y] : \delta_m(q, x, y) \neq \varnothing\}| = k$+1.

It follows that M_m is a (k+1)-edge finite state automata recognizing the language L_k. That is the language $L_k \in$ (k+1)-ACC.

Thus we can conclude that k-ACC $\subseteq$ (k+1)-ACC for $k \geq 0$. +

The relation between classes of k-acceptable languages and regular languages shown as the Venn diagram in Fig.4.

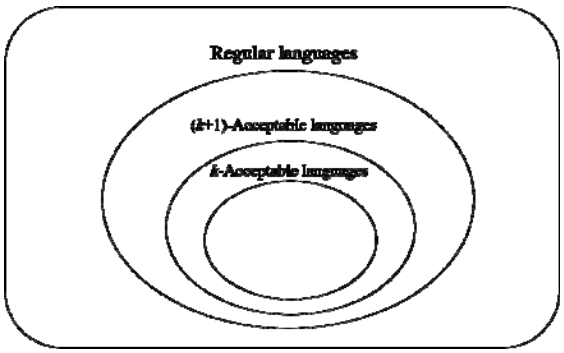


Fig. 4 The relation between classes of k-acceptable languages and regular languages

Definition 3.1

A *k-edge finite state automaton with null transition* (k-FSA$^{\lambda}$) is a 6-tuple $M_k = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k^{\lambda})$ where

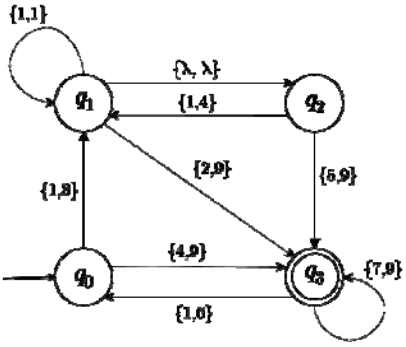
- $\Sigma_{\leq}$ is an ordered alphabet,
- Q is a finite set of states,
- q_0 is the initial state,
- $F_A \subseteq Q$ is a set of accepting states and
- $F_R \subseteq Q$ is a set of rejecting states,
- $\delta_k^{\lambda} : Q \times \Sigma_{\leq} \cup \{\lambda\} \times \Sigma_{\leq} \cup \{\lambda\} \rightarrow Q$ is the transition function with null defined as for any $q \in Q$, $\#(\delta_k^{\lambda}) = |\{[x, y]: \delta_k^{\lambda}(q, x, y) \neq \varnothing\}| = k$, and if $\delta_k^{\lambda}(q, a_1, b_1) \neq \delta_k^{\lambda}(q, a_2, b_2)$ then $\{z : a_1 \leq z \leq b_1\} \cap \{z : a_2 \leq z \leq b_2\} = \varnothing$.

The extended transition function $\delta_k^{\lambda *}: Q \times \Sigma^* \rightarrow Q$ is defined as $\delta_k^{\lambda *}(q, \lambda) = q$ and $\delta_k^{\lambda *}(q, aw) = \delta_k^{\lambda *}(q', w)$ where $x \leq a \leq y$ and $\delta_k^{\lambda}(q, x, y) = q'$ such that $q, q' \in Q$, $a, x, y \in \Sigma_{\leq}$, $w \in \Sigma_{\leq}^*$.

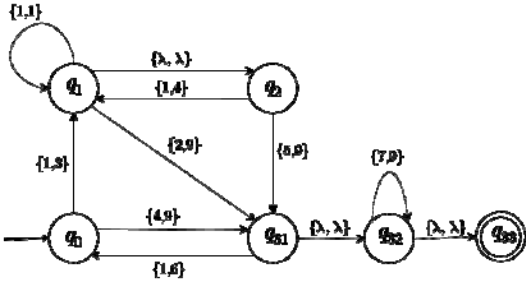
From above definition, k-FSA$^{\lambda}$ can do transition a state to another state by null string, but not for k-FSA.

Definition 3.2

Let $M_k = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k^{\lambda})$ be a k-edge finite state automaton with null transition. We call M_k that a *k-edge finite state automaton with null transition and zero successor* denoted by k-FSA$_0^{\lambda}$ if for all $q \in F_A$ has no successor states.



(a) 3-FSA$^{\lambda}$



(b) 3-FSA$_0^{\lambda}$

Fig 5. Example of 3-FSA$^{\lambda}$ and 3-FSA$_0^{\lambda}$

Fig. 5 shows 3-FSA$^{\lambda}$ and 3-FSA$_0^{\lambda}$. To prove equivalence, we show the equivalence in the below proposition.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Proposition 3.4 Let $M_k = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ be a k-edge finite state automata recognizing the language $L(M_k)$. There exists a k-edge finite state automaton with null transition that recognizes $L(M_k)$.

Proof: To show the existence of a k-edge finite state automaton with null transition (k-FSA$^{\lambda}$), we construct a k-FSA$^{\lambda}$ denoted by

$$M_k^{\lambda} = (\Sigma_{\leq}, Q', q_0, F'_A, F_R, \delta_k^{\lambda}).$$

We defined each tuples as follows.

- $F'_A = \{q_{i\lambda} : q_i \in F_A\}$,
- $Q' = Q \cup F'_A$,
- $\delta_k^{\lambda} = (\delta_k \cup \{(q_{i\lambda}, a, b, p) : q_i \in F_A\} \cup \{(Pred(q_{i\lambda}), \lambda, \lambda, q_{i\lambda})\}) - \{(q, a, b, p) : \forall q \in F_A\}$

From this construction, we see that M_k^{λ} can recognize $L(M_k)$. +

4. CLOSURE PROPERTIES OF THE CLASS OF *k*-ACCEPTABLE LANGUAGES

All closure properties that we present in this paper are constructive: when the class of languages Λ is closed under an operator, we always constructs the automaton M such that $L(M) \in \Lambda$ give L_1, $L_2 \in \Lambda$. In this section, we give constructively proofs that the class ok k-acceptable languages (k-ACC) is closed under complementation, union, Kleene-star operation, intersection and difference as follows.

Theorem 4.1 The class of k-acceptable languages is closed under complement.

Proof: To show closure under complementation, we let $M_{k1} = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ be a k-edge finite state automata that accept $L_1 \in k$-ACC. Then we construct a k-edge finite state automata defined by $M_{k12} = (\Sigma_{\leq}, Q, q_0, Q - F_A, Q - F_R, \delta_k)$ to recognize the complementation of L_1.

From the definition 2.2 of a k-edge finite state automata, we assume δ_k^* be a total function. So that $\delta_k^*(q_0, w)$ is defined for all $w \in \Sigma_{\leq}^*$. Consequently either $\delta_k^*(q', w)$ is a final state, in which case $w \in L_1$, or which $\delta_k^*(q_0, w) \in Q - F_A$ and $w \in \overline{L_1}$.

$$\begin{aligned} L(M_{k12}) &= \{w \in \Sigma_{\leq}^* : \delta_k^*(q_0, w) \in Q - F_A\} \\ &= \{w \in \Sigma_{\leq}^* : \delta_k^*(q_0, w) \notin F_A\} \\ &= \Sigma_{\leq}^* - \{w \in \Sigma_{\leq}^* : \delta_k^*(q_0, w) \in F_A\} \\ &= \Sigma_{\leq}^* - L(M_{k1}) \\ &= \overline{L(M_{k1})} \end{aligned}$$

It follows that $\overline{L_1} \in k$-ACC., Thus, the class of k-acceptable languages is closed under complement. +

Theorem 4.2 The class of k-acceptable languages is closed under union.

Proof: Let $L_1 = L(M_{k1})$ and $L_2 = L(M_{k2})$ be languages in k-ACC, where $M_{k1} = (\Sigma_{\leq}, Q_1, q_{01}, F_{A1}, F_{R1}, \delta_{k1})$ and $M_{k2} = (\Sigma_{\leq}, Q_2, q_{02}, F_{A2}, F_{R2}, \delta_{k2})$ are k-edge finite state automata.

To show closure under union, we construct a combined k-edge finite state automata $M_{k12} = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ recognizing $L_1 \cup L_2$ by following steps.

Step 1 : From proposition 3.4, we can construct the k-edge finite state automata with null transition and zero successor $M_{k1\_0}^{\lambda}$ from M_{k1} and $M_{k2\_0}^{\lambda}$ from M_{k2}.

Step 2 : Construct Q of M_{k12} as follows. $Q = Q_1 \cup Q_2$.

Step 3 : Construct $F_A = \{q_{\text{final}}\}$.

Step 4 : Construct $q_0 = q_{\text{ini}}$.

Step 5 : We add q_0 and the set of accepting state F_A into Q. Now, we have $Q = Q \cup F_A \cup \{q_{\text{ini}}\}$.

Step 6 : Construct $F_R = Q - F_A$.

Step 7 : Construct the k-transition function δ_k as following.

7.1 : Construct two null transition by linking q_0 to initial states of $M_{k1\_0}^{\lambda}$ and $M_{k2\_0}^{\lambda}$. That is $\delta_k = \{(q_{\text{ini}}, \lambda, \lambda, q_{01}), (q_{\text{ini}}, \lambda, \lambda, q_{02})\}$.

7.2 : Add all transitions of $M_{k1\_0}^{\lambda}$ into δ_k. We have $\delta_k = \delta_k \cup \delta_{k1}$.

7.3 : Add all transitions of $M_{k2\_0}^{\lambda}$ into δ_k. We have $\delta_k = \delta_k \cup \delta_{k2}$.

7.4 : Construct two null transition by linking for each $q \in F_{A1}$ of $M_{k1\_0}^{\lambda}$ into δ_k of M_k. Then we get $\delta_k = \delta_k \cup \{(q, \lambda, \lambda, q_{\text{final}}) : q \in F_{A1}\}$.

7.5 : Construct two null transition by linking for each $q \in F_{A2}$ of $M_{k2\_0}^{\lambda}$ into δ_k of M_k.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Then we get $\delta_k = \delta_k \cup \{(q, \lambda, \lambda, q_{\text{final}}) : q \in F_{A2}\}$

Then it is simple matter to show that w $\in L_1 \cup L_2$ if and only if it is accepted by M_{k12}.

Consequently, $L_1 \cup L_2 \in k$-ACC. Thus, the class of k-acceptable languages is closed under union. +

Theorem 4.3 The class of k-acceptable languages is closed under intersection.

Proof: For proving closure property under intersection, we start with DeMorgan's law, taking the complement of both sides. Then

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

For any language $L_1, L_2 \in k$-ACC. Now, if L_1 and L_2 are k-acceptable languages, then by closure under complementation, so are $\overline{L_1}$ and $\overline{L_2}$. Using the closure property under union (Theorem 4.2), we next get that $\overline{L_1} \cup \overline{L_2}$ is k-acceptable. Using the closure property under complementation (Theorem 4.1) once more, we see that $\overline{\overline{L_1} \cup \overline{L_2}}$ is k-acceptable. Thus, the class of k-acceptable languages is closed under intersection. +

Theorem 4.4 The class of k-acceptable languages is closed under concatenation.

Proof: Let $L_1 = L(M_{k1})$ and $L_2 = L(M_{k2})$ be languages in k-ACC, where $M_{k1} = (\Sigma_{\leq}, Q_1, q_{01}, F_{A1}, F_{R1}, \delta_{k1})$ and $M_{k2} = (\Sigma_{\leq}, Q_2, q_{02}, F_{A2}, F_{R2}, \delta_{k2})$ are k-edge finite state automata.

To show closure under concatenation, we construct a combined k-edge finite state automata $M_{k12} = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ recognizing L_1L_2 by following steps.

Step 1 : From proposition 3.4, we can construct a k-edge finite state automata with null transition and zero successor $M_{k1\_0}{}^{\lambda}$ from M_{k1} and a k-edge finite state automata with null transition and zero successor $M_{k2\_0}{}^{\lambda}$ from M_{k2}.

Step 2 : Construct Q of M_{k12} as follows.
$Q = Q_1 \cup Q_2$.

Step 3 : Construct $F_A = \{q_{\text{final}}\}$.

Step 4 : Construct $q_0 = q_{\text{ini}}$.

Step 5 : We add q_0 and the set of accepting state F_A into Q. Now, we have
$Q = Q \cup F_A \cup \{q_{\text{ini}}\}$.

Step 6 : Construct $F_R = Q - F_A$.

Step 7 : Construct the k-transition function δ_k as following.

7.1 : Construct a null transition by linking q_0 to initial states of $M_{k1\_0}{}^{\lambda}$. That is $\delta_k = \{(q_{\text{ini}}, \lambda, \lambda, q_{01})\}$.

7.2 : Add all transitions of $M_{k1\_0}{}^{\lambda}$ into δ_k. We have $\delta_k = \delta_k \cup \delta_{k1}$.

7.3 : Construct null transition by linking every final states of $M_{k1\_0}{}^{\lambda}$ to the initial state of $M_{k2\_0}{}^{\lambda}$. That is $\delta_k = \delta_k \cup \{(q, \lambda, \lambda, q_{02}) : q \in F_{A1}\}$.

7.4 : Add all transitions of $M_{k2\_0}{}^{\lambda}$ into δ_k. We have $\delta_k = \delta_k \cup \delta_{k2}$.

7.5 : Construct null transitions by linking every final states of $M_{k2\_0}{}^{\lambda}$ to the final state of M_k. That is $\delta_k = \delta_k \cup \{(q, \lambda, \lambda, q_{\text{final}}) : q \in F_{A2}\}$.

Then it is simple matter to show that $w \in L_1L_2$ if and only if it is accepted by M_{k12}. Consequently, the language L_1L_2 is a k-acceptable language. Thus, the class of k-acceptable languages is closed under concatenation. +

Theorem 4.5 The class of k-acceptable languages is closed under Kleene-star operation.

Proof: Let $L_1 = L(M_{k1})$ be a language in k-ACC, where $M_{k1} = (\Sigma_{\leq}, Q_1, q_{01}, F_{A1}, F_{R1}, \delta_{k1})$.

To show closure under Kleene-star operation, we construct a combined k-edge finite state automata $M_{k12} = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ recognizing L_1* by following steps.

Step 1 : From proposition 3.4, we can construct k-edge finite state automata with null transition and zero successor $M_{k1\_0}{}^{\lambda}$ from M_{k1}.

Step 2 : Construct Q of M_{k12} as follows.
$Q = Q_1$

Step 3 : Construct $F_A = \{q_{\text{final}}\}$.

Step 4 : Construct $q_0 = q_{\text{ini}}$.

Step 5 : We add q_0 and the set of accepting state F_A into Q. Now, we have $Q = Q \cup F_A \cup \{q_{\text{ini}}\}$.

Step 6 : Construct $F_R = Q - F_A$.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Step 7 : Construct the k-transition function δ_k as following.

7.1 : Construct a null transition by linking q_0 to initial states of $M_{k1\_0}{}^{\lambda}$. That is $\delta_k = \{(q_{ini}, \lambda, \lambda, q_{01})\}$.

7.2 : Construct a null transition by linking the initial state of M_k to the final state of M_k that is

$$\delta_k = \delta_k \cup \{(q_{ini}, \lambda, \lambda, q_{final})\}.$$

7.3 : Add all transitions of $M_{k1\_0}{}^{\lambda}$ into δ_k. We have $\delta_k = \delta_k \cup \delta_{k1}$.

7.4 : Construct a null transition by linking the final state of M_k to the initial state of M_k that is

$$\delta_k = \delta_k \cup \{(q_{final}, \lambda, \lambda, q_{ini})\}$$

Then it is simple matter to show that $w \in L_1{}^*$ if and only if it is accepted by M_{k12}. Consequently, $L_1{}^*$ is k-acceptable.

Thus, the class of k-acceptable languages is closed under concatenation +

Theorem 4.6 The class of k-acceptable languages is closed under difference.

Proof: To show closure property under difference, we need to show that if L_1 and L_2 are k-acceptable languages, then L_1 - L_2 is a k-acceptable language also.

From the definition of a set difference,

$$L_1 - L_2 = L_1 \cap \overline{L_2}\ .$$

The fact that L_2 is k-acceptable implies $\overline{L_2}$ is also k-acceptable. Then, because of the closure of k-ACC under intersection, we know that $L_1 \cap \overline{L_2}$ is k-acceptable. Therefore, L_1 - L_2 is k-acceptable.

Thus, the class of k-acceptable languages is closed under difference +

5. DECIDABILITY OF k-ACC

In this section, we consider the decidability status of some decision problems for k-acceptable languages.

Lemma 5.1 Let $M_k = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ be a k-edge finite state automata. Then, the language $L(M_k)$ is infinite if and only if $\delta_k{}^*(p, u) = p$ in M_k for some $u \in \Sigma^+$ and $p \in Q$ such that p is both reachable and terminating in M_k.

Proof :

(If part) Let $M_k = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ be a k-edge finite state automata such that if $\delta_k{}^*(p, w) = p$ is in M_k for some $w \in \Sigma^+$ and $p \in Q$ such that p is both reachable and terminating in M_k.

Then,

$$\delta_k{}^*(q_0, w) = \delta_k{}^*(p, u) = \delta_k{}^*(p, v) = f$$

where $w \in L(M_k)$, $u, v \in \Sigma^*{}_{\leq}$, $p \in Q$, $f \in F_A$. Consequently,

$$\delta_k{}^*(q_0, w) = \delta_k{}^*(p, uy') = \delta_k{}^*(p, v) = f$$

where $y' = y^n$ for all $n \geq 0$.

Therefore, $L(M_k)$ is infinite, so the if part holds.

(Only if part) : Let $M_k = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ be a k-edge finite state automata such that $L(M_k)$ is infinite.

Then,

$$\delta_k{}^*(q_0, w) = \delta_k{}^*(p, u) = \delta_k{}^*(p, v) = f$$

for some $w \in L(M_k)$, $u, v \in \Sigma^*{}_{\leq}$, $p \in Q$, $f \in F_A$.

This implies that p is both reachable and terminating in M_k. Let $y \in \Sigma^+{}_{\leq}$ be a string read by M_k during $\delta_k{}^*(p, u) = \delta_k{}^*(p, v)$.Then, $\delta_k{}^*(p, w) = p$ in M_k, so the only-if part holds. +

Theorem 5.2 The infiniteness problem is decidable for the class of k-acceptable languages.

Proof : Let $M_k = (\Sigma_{\leq}, Q, q_0, F_A, F_R, \delta_k)$ be a k-edge finite state automata. By Lemma 5.1, $L(M_k)$ is infinite if and only if $\delta_k{}^*(p, u) = p$ is in M_k for some $y \in \Sigma^+{}_{\leq}$ and $p \in Q$ such that p is both reachable and terminating in M_k. This condition can be checked by any graph searching algorithm, such as breadth-first search.

Therefore, the infiniteness problem is decidable for the class of k-acceptable languages. +

The membership problem for $L \in k$-ACC is to that test whether a given k-edge finite state automata M accepts a given input string w. We consider the language as follows.

Theorem 5.3 The membership problem is decidable for the class of k-acceptable languages.

Proof : The language contains the encodings of all k-edge finite state automata together with strings that the k-edge finite state automata accept.

Let $A_{k\text{-}FSA} = \{\langle M, w \rangle : M$ is a k-edge finite state automata that accepts input string $w\}$.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

We present an algorithm that decides the language $A_{k\text{-FSA}}$ as below.

Input : a k-FSA M_k and a string $w \in \Sigma^*$
Output : *yes* or *no*
Step 1 : Simulate M_k on input w.
Step 2 : If the simulation ends in an accept state, then return *yes*.
If it ends in a nonaccepting state, then return *no*.

Therefore, the membership problem is decidable for the class of k-acceptable languages. +

The emptiness problem for the class of k-acceptable languages is to test whether the language recognized by a given k-edge finite state automata M is empty or not.

Theorem 5.4 The emptiness problem is decidable for the class of k-acceptable languages.
Proof : We consider the language of the emptiness problem defined as

$E_{k\text{-FSA}} = \{ \langle M \rangle : M$ is a k-FSA and $L(M) = \varnothing \}$.

We note that $L(M) = \varnothing$ if and only if there is no path in the state diagram of M from the initial state q_0 to a accepting state. If $F_A = \varnothing$, then clearly $L(M) = \varnothing$.

To show the emptiness problem is decidable, we present an algorithm that decides the language $E_{k\text{-FSA}}$ as follows.

Input : a k-FSA M_k
Output : *yes* or *no*
Step 1 : Mark the initial state of M_k.
Step 2 : Repeat until no new states get marked:
Mark and state that has a transition coming into it from any that is already marked.
Step 3 : If no accepting state is marked, then return *yes*; otherwise return *no*.

Therefore, the emptiness problem is decidable for the class of k-acceptable languages.. +

The language equivalence problem for the class of k-acceptable languages is to test whether two given k-edge finite state automata recognize the same language.

Theorem 5.5 The equivalence problem is decidable for the class of k-acceptable languages.
Proof : We consider the language of an equivalence problem $EQ_{k\text{-FSA}}$ defined as

$EQ_{k\text{-FSA}} = \{ \langle M_{k1}, M_{k2} \rangle : M_1$ and M_2 are k-edge finite state automata and $L(M_{k1}) = L(M_{k2}) \}$.

To prove $EQ_{k\text{-FSA}}$ is a decidable language, we use Theorem 5.4. We construct a new k-edge finite state automata M_{k3} from k-edge finite state automata M_{k1} and M_{k2}, where accept only those string that are accepted by either M_{k1} or M_{k2} but not by both. Thus, if the k-edge finite state automata M_{k1} and M_{k2} recognize the same language, the k-edge finite state automata will accept nothing. The language of M_{k3} is

$$L(M_{k3}) = \left(L(M_{k1}) \cap \overline{L(M_{k2})} \right) \cup \left(\overline{L(M_{k1})} \cap L(M_{k2}) \right).$$

We can construct M_{k3} from M_{k1} and M_{k2} with the construction for proving the class of k-acceptable closed under complementation, union and intersection proved in the section 4. These construction are algorithms that can be carried out by Turing machines. Once we have constructed M_{k3} we can use Theorem 5.4 to test whether $L(M_{k3})$ is empty. If it is empty, $L(M_{k1})$ and $L(M_{k2})$ must be equal. We present an algorithm that decides the language $EQ_{k\text{-FSA}}$ as below.

Input : k-FSA M_{k1} and M_{k2}
Output : *yes* or *no*
Step 1 : Construct a k-FSA M_{k3} as described above.
Step 2 : Run the algorithm from Theorem 5.4 on the input $\langle M_{k3} \rangle$.
Step 3 : If the algorithm return yes, return *yes*; otherwise return *no*.

Therefore, the equivalence problem is decidable for the class of k-acceptable languages.. +

Theorem 5.6 The disjointness problem is decidable for the class of k-acceptable languages.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Proof : We consider the language of an equivalence problem $D_{k\text{-}FSA}$ defined as

$D_{k\text{-}FSA} = \{ \langle M_{k1}, M_{k2} \rangle : M_1$ and M_2 are k-edge finite state automata and $L(M_{k1}) \cap L(M_{k2}) = \varnothing \}$.

To prove $D_{k\text{-FSA}}$ is a decidable language, we use Theorem 4.3 that it follows that $L(M_{k1}) \cap L(M_{k2})$ is a k-acceptable language. Clearly, we can construct a k-edge finite state automata accepting $L(M_{k1}) \cap L(M_{k2})$. Then the algorithm in Theorem 5.5 can be used to test the emptiness of the language $L(M_{k1}) \cap L(M_{k2})$. We present an algorithm that decides the language $D_{k\text{-}FSA}$ as below.

Input : k-FSA M_{k1} and M_{k2}
Output : *yes* or *no*
Step 1 : Construct a k-FSA M_{k3} that accepts $L(M_{k1}) \cap L(M_{k2})$ as described above.
Step 2 : Run the algorithm from Theorem 5.5 on the input $\langle M_{k3} \rangle$.
Step 3 : If the algorithm return yes, return *yes*; otherwise return *no*.

Therefore, the disjointness problem is decidable for the class of k-acceptable languages.. +

6. CONCLUSION

We were motivated by the aim to come to a better understanding of formal languages recognized by k-edge finite state automata called k-acceptable languages. In this paper, we have achieved for investigation of the closure properties and proved that the class of k-acceptable languages is closed under complementation, union, intersection, concatenation, difference and Kleene-star operation. Moreover, we have shown that infiniteness problem, membership problem, equivalence problem, emptiness problem and disjointness problem of k-acceptable languages are decidable. The theoretical properties obtained in this work benefit to better understand the powerful of k-edge finite state automata.

ACKNOWLEDGEMENT

This research was funded by King Mongkut's University of Technology North Bangkok Contract no. KMUTNB-GOV-58-30.

REFRENCES:

[1] N. Chomsky, "Three models for the description of languages", *Proceedings of the Symposium on Information Theory*, 1956.

[2] C.D. Higuera, Grammatical inference: Learning Automata and Grammars, *Cambridge University Press*, Cambridge, 2010.

[3] L. Peter, An Introduction to Formal Languages and Automata, Third Editions, *Jones & Bartlett*, 2001.

[4] K. Chatterjee, L. Doyen, T.A. Henzinger, "Expressiveness and Closure Properties for Quantitative Languages", *Proceedings of the 24th Annual Symposium on Logic in Computer Science (LICS-09)*, 2009.

[5] T. Shang, X. Lu, R. Lu, "Closure Properties of Quantum Turing Languages", *Proceedings of the Computability in Europe 2012 (CiE2012)*, 2012.

[6] H. Fernau, "Closure Properties of Ordered Languages", *EATCS Bullentin*, Vol. 58, 1996, pp. 162-165.

[7] C. D. Higuera, "Ten open problems in grammatical inference", *Proceedings of 8th International Colloquium on Grammatical Inference*, 2006.

[8] P. Cruz-Alc´azar, E. Vidal, "Two grammatical inference applications in music processing", *Applied Artificial Intelligence*, Vol. 22(1–2), 2008, pp. 53–76.

[9] N. Abe, H . Mamitsuka, "Predicting protein secondary structure using stochastic tree grammars", *Machine Learning Journal*, 1997 pp. 275-30.

[10] A. Jitpattanakul, A. Surarerks, "The study of learnability of the class of k-acceptable languages on Gold's learning model", *Chiang Mai Journal of Science*, Vol. 40, No.2, 2013, pp. 248-260.

[11] A. Jitpattanakul, "Learnability of strictly k-acceptable languages", *Far East Journal of Mathematical Science*, Vol. 71, No.1, 2012, pp. 169-184.