ICIC Express Letters
Volume **12**, Number **10**, October **2018**

ICIC International ©2018 ISSN 1881-803X
pp. **981–989**

NFA TO DFA CONVERSION: A NEW APPROACH USING LANGUAGES OF BOUNDED WORDS

Ho Ngoc Vinh[1] and Nguyen Thi Thu Ha[2]

[1]Information Technology Faculty
Vinh University of Technology Education
No 117 Nguyen Viet Xuan, HungDung, Nghe An, Vietnam
hongocvinh@gmail.com

[2]Department of E-Commerce
Electric Power University
235 Hoang Quoc Viet, TuLiem, Hanoi, Vietnam
hantt@epu.edu.vn

Received March 2018; accepted June 2018

Abstract. *In this paper, concepts of bounded words on an alphabet A, ♢-languages and monoid ♢-morphisms are introduced. Some basic results for recognizable languages and regular languages of bounded words on A are obtained. These allow defining an extension of the automaton on the set of bounded words. Hence, a new perspective of mathematical model of sets of states, edges, languages recognized by the finite automaton according to the length of languages is given and a new checking algorithm is proposed, greatly reducing the complexity of the checking algorithm.*
Keywords: Bounded word, Monoid morphism, ♢-automata, ♢-recognizable, Algorithm

1. **Introduction.** The theory of formal languages, finite automata and complexity are modern branches in computer theory and their mathematical models play very important roles. There are a lot of works considering the relationship between these mathematical models. For example, a popular issue in studying the theory of formal languages and automata is checking whether the strings are recognized by a finite automaton. Many different checking algorithms have been proposed.

In this paper, we introduce the notions of bounded words on an alphabet A, ♢-languages and monoid ♢-morphisms (see also [3]). In addition, we give the new definitions of the regular ♢-expressions, regular ♢-languages and finite ♢-automata. Hence, some basic results for recognizability of the ♢-automata, (Proposition 3.1, Proposition 3.3), the relationship between regular language and regular ♢-languages (Proposition 3.2), in special, the relationship between ♢-automata, ♢-recognizable and regular ♢-languages are obtained. In Section 3, the algorithms of checking if a language L can be recognized by an automaton and their complexity are considered. Then, a new algorithm (Algorithm 4), with the ♢-automaton approach, reducing the complexity compared with previous algorithms is proposed.

2. **Languages of Bounded Words.** At first, we recall some notions and notations; for more details, we refer to [4,6]. Let A be a finite alphabet and the set $B = \{0, 1\}$. The sets of *bounded words* (*♢-words*) on A are $A_\diamondsuit = \{(i, a, j) | a \in A \textit{ or } a = \varepsilon, i, j \in B\}$ and $A^*_\diamondsuit = \{(i, w, j) | w \in A^*, i, j \in B\} \cup \{\theta, e\}$. Then, each element (i, w, j), $w \in A^*$, is called a *♢-word* (*or a bounded word with borders* i, j) extended from w in which e, θ are two new elements as the unit, the zero of the monoid $A^*_\diamondsuit$ of all *♢-words* respectively. It is easily seen that, $A^*_\diamondsuit$ is a monoid by a product defined as follows: for any $x_1 = (i_1, w_1, j_1)$,

DOI: 10.24507/icicel.12.10.981

$x_2 = (i_2, w_2, j_2)$ in $A^*_\diamondsuit$, if $j_1 = i_2$ then $x_1.x_2 = (i_1, w_1w_2, j_2)$, else $x_1.x_2 = \theta$ and $\forall x \in A^*_\diamondsuit$, $x.\theta = \theta.x = \theta$, $x.e = e.x = x$.

We call $A^*_\diamondsuit$ the $\diamondsuit$-*monoid* defined by A. A set $L \subseteq A^*_\diamondsuit$ is called *an extended language* ($\diamondsuit$-*language*) on A. Whenever none of mistakes are made, we also use notation $|x|$ as the length of x. In particular we make a convention: $|\theta| = -\infty$, $|e| = 0$ and $|x| = 0$ if $x \in \{(i, \varepsilon, j) | i,\ j \in B\}$. For X, $Y \subseteq A^*_\diamondsuit$, left and right quotients are defined as $Y^{-1}X = \{u \in A^*_\diamondsuit | \exists y \in Y\colon\ y.u \in X\}$ and $XY^{-1} = \{u \in A^*_\diamondsuit | \exists\ y \in Y\colon\ u.y \in X\}$. The function *Proj*: $A^*_\diamondsuit \to A^* \cup \{0\}$ is defined by *Proj*$(e) = \varepsilon$, *Proj*$(\theta) = 0$ and *Proj*$(i, w, j) = w$ (where $0 \notin A^*$ as the new zero of the monoid $A^* \cup \{0\}$).

Definition 2.1. *Let M be a monoid with the unit 1, the zero 0. Let φ: $A^*_\diamondsuit \to M$ be a function. Then, φ is called a monoid $\diamondsuit$-morphism (or $\diamondsuit$-morphism for short) if it satisfies the following conditions:*

(1) $x, y \in A^*_\diamondsuit$ and $x.y \neq \theta$ then $\varphi(x.y) = \varphi(x).\varphi(y)$

(2) $\varphi(e) = 1$

(3) $\varphi(\theta) = 0$

Definition 2.2. *Let $L \subseteq A^*_\diamondsuit$ and M be a monoid. We say that M saturates L if there exists a $\diamondsuit$-morphism φ: $A^*_\diamondsuit \to M$ such that $L = \varphi^{-1}(N)$ for some $N \subseteq M$. In this case, we also say that L is saturated by φ.*

From Definition 2.2, if N_1, $N_2 \subseteq M$, imply that $\varphi^{-1}(N_1 \cap N_2) = \varphi^{-1}(N_1) \cap \varphi^{-1}(N_2)$, $\varphi^{-1}(N_1 \cup N_2) = \varphi^{-1}(N_1) \cup \varphi^{-1}(N_2)$, $\varphi^{-1}(N_1 \backslash N_2) = \varphi^{-1}(N_1) \backslash \varphi^{-1}(N_2)$. Moreover, if φ is subjective, we have $\varphi^{-1}\left(N_1^{-1}N_2\right) = \varphi^{-1}(N_1)^{-1}\varphi^{-1}(N_2)$, $\varphi^{-1}\left(N_1N_2^{-1}\right) = \varphi^{-1}(N_1)\varphi^{-1}$ $(N_2)^{-1}$.

For each $L \subseteq A^*_\diamondsuit$, due to S. Eilenberg [1], we can apply a similar way to constructing a monoid M saturating L. We denote by $\mathcal{R}(A, M)$, the set of all $\diamondsuit$-languages saturated by M on $A^*_\diamondsuit$. According to [4], $\mathcal{R}(A, M)$ is closed under the boolean operations. Further, if φ is an epimorphism then $\mathcal{R}(A, M)$ is closed under the left quotients and right quotients.

3. **Extended Recognizable Languages.** In this section, we propose the notions of regular $\diamondsuit$-expression and regular $\diamondsuit$-language by following definitions.

Definition 3.1. *Let A be a finite alphabet. A regular $\diamondsuit$-expression on $A^*_\diamondsuit$ is defined recursively as follows.*

(i) $\varnothing$, e, θ are regular $\diamondsuit$-expressions.

(ii) $\forall a \in A$ or $a = \varepsilon$, $\forall i, j \in B, (i, a, j)$ is a regular $\diamondsuit$-expression.

(iii) If E_1 and E_2 are the regular $\diamondsuit$-expressions, then $(E_1 + E_2)$, $E_1.E_2$ and E_1^* are the regular $\diamondsuit$-expressions.

(iv) There is not any regular $\diamondsuit$-expressions except the regular $\diamondsuit$-expressions defined by (i), (ii) and (iii).

Then, we define *regular $\diamondsuit$-languages.*

Definition 3.2. *Let A be a finite alphabet. A regular $\diamondsuit$-languages determined by the regular $\diamondsuit$-expression E on $A^*_\diamondsuit$, denoted by $L(E)$, is defined recursively as follows.*

(i) $E = \varnothing$ then $L(E) = \varnothing$.

(ii) $E = e$ then $L(E) = \{e\}$.

(iii) $E = \theta$ then $L(E) = \{\theta\}$.

(iv) $E = (i, \varepsilon, j)$ then $L(E) = \{(i, \varepsilon, j)\}$, $\forall i, j \in B$.

(v) $\forall (i, a, j) \in A_\diamondsuit$, $E = (i, a, j)$ then $L(E) = \{(i, a, j)\}$.

(vi) If E_1 and E_2 are the regular $\diamondsuit$-expressions and $L(E_1)$ and $L(E_2)$ have been defined, $E = (E_1 + E_2)$ then $L(E) = L(E_1) \cup L(E_2)$, $E = E_1.E_2$ then $L(E) = L(E_1).L(E_2)$ and $E = E_1^*$ then $L(E) = L(E_1)^*$.

(vii) Only the $\diamondsuit$-languages which are defined by (i), (ii), (iii), (iv), (v) and (vi) are regular $\diamondsuit$-languages.

Combining with a finite automaton $\mathcal{A} = (A, Q, \delta, I, T)$, we define a special form of finite automata that accepts a set of $\diamondsuit$-words on $A^*_\diamondsuit$ as follows.

Definition 3.3. *Let $\mathcal{A} = (A, Q, \delta, I, T)$ be a nondeterministic finite automaton, we define a finite extended automaton (for brevity, $\diamondsuit$-automaton) $\mathcal{A}_\diamondsuit$ by a 5-tuple $\mathcal{A}_\diamondsuit = (A_\diamondsuit, Q_\diamondsuit, \delta_\diamondsuit, I_\diamondsuit, T_\diamondsuit)$ satisfying:*

– $A_\diamondsuit = \{(i, a, j) | a \in A, i, j \in B\} \cup \{e, \theta\}$ *is considered as the alphabet of* $\mathcal{A}_\diamondsuit$.

– $Q_\diamondsuit = \{(i, q, j) | q \in Q,\ i, j \in B\} \cup \{q_\theta\}$ *is the finite nonempty set of the states, where* q_θ *is a new sink state.*

– $I_\diamondsuit = \{(i, q, j) | q \in I, i, j \in B\}$ *is the set of initial states.*

– $T_\diamondsuit = \{(i, q, j) | q \in T, i, j \in B\}$ *is the set of final states.*

– *Denote by* $\mathcal{P}(Q_\diamondsuit)$ *the set of all subsets of* $Q_\diamondsuit$*, then the transition function* $\delta_\diamondsuit$*:* $Q_\diamondsuit \times A^*_\diamondsuit \to \mathcal{P}(Q_\diamondsuit)$ *is defined as follows: for any* (i, q, j) *in* $Q_\diamondsuit$*,*

$\delta_\diamondsuit((i, q, j), e) = (i, q, j)$ *where* $e \in A^*_\diamondsuit$.

$\delta_\diamondsuit((i, q, j), \theta) = q_\theta$ *where* $\theta \in A^*_\diamondsuit$.

$\delta_\diamondsuit((i, q, j),\ (j', a, k)) \ni (i, q', k) \Leftrightarrow \forall a \in A$*:* $\delta(q, a) \ni q'$ *and* $j = j'$*, otherwise if* $j \neq j'$ *then* $\delta_\diamondsuit((i, q, j), (l, a, k)) = q_\theta$.

For simplicity, with s, $s' \in Q$, we write $s.x$ instead of $\delta_\diamondsuit(s, x)$, $x = e$, θ or $x = (l, a, k)$, $a \in A$, and it can be extended inductively on length to any $\diamondsuit$-word $x \in A^*_\diamondsuit$: $s.x = \Upsilon_{s' \in s.u, x = u.y} s'.y$. A sequence x_1, x_2, …, x_n of $\diamondsuit$-words in $A^*_\diamondsuit$ is said to be *accepted* by $\mathcal{A}_\diamondsuit$ if and only if there exists $q_\diamondsuit \in T_\diamondsuit$, such that $q_\diamondsuit \in (((q_0.x_1).x_2)\ldots).x_n$ and in that case $\diamondsuit$-word $x = x = x_1.x_2 \ldots x_n$ is said to be *accepted* by $\mathcal{A}_\diamondsuit$. Denote by $\boldsymbol{\mathcal{L}}(\mathcal{A}_\diamondsuit)$ the set of all $\diamondsuit$-words recognized by $\mathcal{A}_\diamondsuit$, that is $\boldsymbol{\mathcal{L}}(\mathcal{A}_\diamondsuit) = \{x \in A^*_\diamondsuit | \exists q_0 \in I_\diamondsuit \text{ such that } q_0.x \cap T_\diamondsuit \neq \varnothing\}$. We call a $\diamondsuit$-language L to be *accepted* by $\mathcal{A}_\diamondsuit$ if $L = \boldsymbol{\mathcal{L}}(\mathcal{A}_\diamondsuit)$.

For brevity, from now on, we write finite $\diamondsuit$-automaton (automaton) instead of nondeterministic finite $\diamondsuit$-automaton (automaton).

Definition 3.4. *A set $L \subseteq A^*_\diamondsuit$ is called an extended recognizable language (or $\diamondsuit$-recognizable) if $L = \boldsymbol{\mathcal{L}}(\mathcal{A}_\diamondsuit)$ for some finite $\diamondsuit$-automaton $\mathcal{A}_\diamondsuit$.*

We call a language $L \subseteq A^*_\diamondsuit$ a $\diamondsuit$-recursive language if the membership problem for L is solvable. The following results are fundamental for the case of finite $\diamondsuit$-automata which can be verified directly by definition.

Fact 1. *If $L \subseteq A^*_\diamondsuit$ is $\diamondsuit$-recognizable then L is $\diamondsuit$-recursive.*

Let $L \subseteq A^*$, we build an extension operator of language $\diamondsuit$: $L' = \{(i, w, j) \in A^*_\diamondsuit | w \in L, i, j \in B\}$ and $L_\diamondsuit = L'$ if $\varepsilon \notin L$; $L_\diamondsuit = L' \cup \{e\}$ if $\varepsilon \in L$. A non-trivial relationship between $\diamondsuit$-recognizable languages and recognizable languages is showed by Fact 2 below.

Fact 2. *Let $L \subseteq A^*$. If L is recognizable by a finite automaton $\mathcal{A}$ then $L_\diamondsuit$ is recognizable by a finite $\diamondsuit$-automaton $\mathcal{A}_\diamondsuit$, where $\mathcal{A}_\diamondsuit$ is a $\diamondsuit$-automaton expanded form $\mathcal{A}$.*

The properties above and the classical results confirm the equivalent of deterministic and nondeterministic finite automata. So, we have following corollary.

Corollary 3.1. *Let $L \subseteq A^*_\diamondsuit$. If L is recognizable by a finite $\diamondsuit$-automaton then it is also recognizable by a deterministic finite $\diamondsuit$-automaton.*

Next, we will show some fundamental results for $\diamondsuit$-automata, which can be proved classically similar to the methods used in [2,5,7]. Some results are well-known so their proofs are ignored in this paper.

Proposition 3.1. *Let $L \subseteq A^*_\diamondsuit$. Then L is $\diamondsuit$-recognizable if and only if L is regular $\diamondsuit$-language.*

A non-trivial relationship between regular $\diamondsuit$-languages and regular languages is shown by the following result.

Proposition 3.2. *Given a regular language $L \subseteq A^*$. Then $Proj^{-1}(L)$ is a regular $\diamondsuit$-language on $A^*_\diamondsuit$. There exists L which is not a regular $\diamondsuit$-language on $A^*_\diamondsuit$ but $Proj(L)$ is a regular language on A^*.*

Proof: By definition and assumption, $L = L(E)$ for some regular $\diamondsuit$-expression E. We will prove by induction on the construction of E.

+ If $E = \varnothing$ then $L(E) = \varnothing$. Then $Proj^{-1}(L) = \varnothing$ is a regular $\diamondsuit$-language on $A^*_\diamondsuit$.

+ If $E = \varepsilon$ then $L(E) = \{\varepsilon\}$. Therefore, $Proj^{-1}(L) = \{e, (i, \varepsilon, j) : i, j \in B\}$ is a finite set; this implies that L is a regular $\diamondsuit$-language on $A^*_\diamondsuit$.

+ Now, we suppose E_1 and E_2 are the regular expressions on A^* and $Proj^{-1}(L(E_1))$ and $Proj^{-1}(L(E_2))$ are the regular $\diamondsuit$-languages on $A^*_\diamondsuit$ already. Then, from the definition of the projection function $Proj^{-1}$, we easily get:

$$Proj^{-1}(L(E_1) \cup L(E_2)) = Proj^{-1}(L(E_1)) \cup Proj^{-1}(L(E_2))$$
$$Proj^{-1}(L(E_1).L(E_2)) = Proj^{-1}(L(E_1)).Proj^{-1}(L(E_2))$$
$$Proj^{-1}(L(E_1)^*) = Proj^{-1}(L(E_1))^*$$

Hence, the language $Proj^{-1}(L)$ is a regular $\diamondsuit$-language on $A^*_\diamondsuit$.

For the second statement, we consider the following example: let $A = \{a\}$ be a singleton alphabet and let $L_1 \subsetneq A^*$ be a non-recursive language on A^* (according to classical results, there exists such an L_1). Therefore, it does not have any decision algorithm for the membership problem of L_1. We define

$$L = \{(1, w, 1) | w \in L_1\} \cup \{(0, w, 0) | w \notin L_1\}$$

It is easily seen that L is also non-recursive, that means the membership problem for L is also not decidable, but the image $Proj(L)$ is exactly A^*, the regular one. □

Lemma 3.1. *Let $\mathcal{A}_\diamondsuit$ be a finite $\diamondsuit$-automaton and x be a $\diamondsuit$-word of $A^+_\diamondsuit$ admitting two different factorizations $x = x_1.x_2 \ldots x_n = x'_1.x'_2 \ldots x'_m$ where n, $m \geq 1$, $x_i, x'_j \in A^*_\diamondsuit$, $i = 1, \ldots, n$, $j = 1, \ldots, m$. If the sequence $x_1, x_2, \ldots, x_n$ is recognized by $\mathcal{A}_\diamondsuit$, then the sequence x'_1, $x'_2, \ldots, x'_m$ is also recognized by $\mathcal{A}_\diamondsuit$.*

Lemma 3.2. *Let $\mathcal{A}_\diamondsuit$ be a finite $\diamondsuit$-automaton and x, y, $z \in A^*_\diamondsuit$, $z = x.y \neq \theta$ with some factorizations $x = x_1.x_2 \ldots x_n$, $y = y_1.y_2 \ldots y_m$, $z = z_1.z_2 \ldots z_k \in A^+_\diamondsuit$ where $n, m, k \geq 1$, $x_i, y_j, z_l \in A^*_\diamondsuit$, $i = 1, \ldots, n$, $j = 1, \ldots, m$, $l = 1, \ldots, k$. If the sequence $x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m$ are recognized by $\mathcal{A}_\diamondsuit$ then the sequence $z_1, z_2, \ldots, z_k$ is also recognized by $\mathcal{A}_\diamondsuit$.*

Form Lemma 3.1 and Lemma 3.2, we have following result.

Proposition 3.3. *Let $L \subseteq A^*_\diamondsuit$. Then, L is $\diamondsuit$-recognizable if and only if there exists a $\diamondsuit$-morphism φ: $A^*_\diamondsuit \to M$, M is finite, such that L is saturated by φ.*

From the propositions above, we have

Corollary 3.2. *Let $L \subseteq A^*_\diamondsuit$. The following conditions are equivalent.*
(i) L is $\diamondsuit$-recognizable.
(ii) L is saturated by a finite monoid.
(iii) L is regular $\diamondsuit$-language.

4. **Conversion of Nondeterministic Finite Automata (NFA) to Deterministic Finite Automata (DFA).** A popular issue when studying on the theory of formal language and automata is to check whether a string $S \in A^*$ is recognized by a finite automaton or not. Many different algorithms of checking have been presented. Next, we are presenting those algorithms again and propose a new algorithm in the approach of $\diamondsuit$-language to considerably reduce complexity of checking algorithm.

Problem: Let language $L = \{S_1, S_2, \ldots, S_N\}$, $S_i \in A^*$ is a string with the size $\leq l$ characters and finite automaton $\mathcal{A} = (A, Q, \delta, I, T)$ on the alphabet A with m elements, the set of states Q consists of k states. Find out strings $S_i \in L$ so that $S_i \in \boldsymbol{\mathcal{L}}(\mathcal{A})$.

Algorithm 1. With finite automaton $\mathcal{A}$, check whether all strings $S_i \in L$ are recognized by automaton $\mathcal{A}$? Then, this algorithm has complexity of $O(l.m.k^k.N)$.

Algorithm 2. (Deterministicitization).

Step 1. Change the nondeterministic finite automaton $\mathcal{A}$ into the deterministic finite automaton $\mathcal{A}'$. This step has complexity of $\geq O(2^k)$.

Step 2. Use the deterministic finite automaton $\mathcal{A}'$ to solve the problem: Find out strings $S_i \in L$ so that $S_i \in \boldsymbol{\mathcal{L}}(\mathcal{A}')$. This step has complexity of $O(l.N)$.

Therefore, this algorithm has approximate complexity of $\geq O(2^k + l.N)$.

Algorithm 3. *(Breadth First Search)*

Check the string $S_i = a_1 a_2 \ldots a_l$, $a_j \in A$, on the finite automation.

```
1. N_0 = I
2. For j = 1 to l do
   {  //Known N_{j-1}, calculate N_j
3.     N_j = ∅
4.     For each q in N_{j-1} do
5.          Find a neighbor q' of q with the label a_j such that (q, a_j, q') ∈ E(A) then
6.               Add q' in N_j.
   }
```

It is realized that N_{j-1} has k states, q has m neighbors and each neighbor has k accessible states. Therefore, checking a string S_i has the approximate complexity of $O(l.m.k^2)$. Therefore, this algorithm has complexity of $O(l.m.k^2.N)$.

In this part, we are going to present an extended form of the finite automaton $\mathcal{A}$ and if without confusion, we call it finite $\diamondsuit$-automaton $\mathcal{A}_\diamondsuit$, the set of bounded B is not only $\{0, 1\}$, but also extended to $B = \{0, 1, 2, \ldots, l\} \subseteq \mathbb{N}$. Next, we use finite $\diamondsuit$-automaton $\mathcal{A}_\diamondsuit$ to solve the problem of checking whether the strings are recognized by the finite automaton $\mathcal{A}$ or not.

Let $\mathcal{A}$ be a finite automaton and l be the length of the longest string recognized by the finite automaton $\mathcal{A}$. Then, finite $\diamondsuit$-automaton $\mathcal{A}_\diamondsuit$ extended from the finite automaton $\mathcal{A}$ by a 5-tuple $\mathcal{A}_\diamondsuit = (A_\diamondsuit, Q_\diamondsuit, \delta_\diamondsuit, I_\diamondsuit, T_\diamondsuit)$

– $A_\diamondsuit = \{(i, a, i+1) | a \in A, i = 0, \ldots, l-1\}$ is the alphabet of $\diamondsuit$-automaton $\mathcal{A}_\diamondsuit$.

– $Q_\diamondsuit = \{(0, q, i) | q \in Q, i = 0, \ldots, l\}$ is the finite nonempty set of the states.

– $I_\diamondsuit = \{(0, q, 0) | q \in I\}$ is the set of initial states.

– $T_\diamondsuit = \{(0, q, i) | q \in T, i = 0, \ldots, l\}$ is the set of final states.

Let a word $w = a_1 \ldots a_l \in A^*$. Then, the word w is extended into $w_\diamondsuit = (0, a_1, 1) \ldots (l-1, a_l, l)$ and we denote $\boldsymbol{\mathcal{L}}(\mathcal{A}_\diamondsuit)$ as a set of $\diamondsuit$-word recognized by $\diamondsuit$-automaton $\mathcal{A}_\diamondsuit$, we have: $\boldsymbol{\mathcal{L}}(\mathcal{A}_\diamondsuit) = \{w \in A^* | \exists (0, q, 0) \in I_\diamondsuit \text{ such as } \delta_\diamondsuit((0, q_0, 0), w_\diamondsuit) \cap T_\diamondsuit \neq \varnothing\}$.

With the above definition of $\diamondsuit$-automata, we have a new mathematic overview of sets of states $V_i = \{(0, q, i) | q \in Q, i \in B\}$, sets of edges $E_i = \{((0, q, i-1), (i-1, a, i), (0, q', i))\}$ and set of languages $S = \{(0, a_1, 1) \ldots (l-1, a_l, l) | l \in B, a_i \in A, i = 1, \ldots, l\}$ recognized by the finite automaton $\mathcal{A}$ according to the length of language. In the data structure view, for example: the set of edges $E_i = \{((0, q, i-1), (i-1, a, i), (0, q', i))\}$ is represented as $E[i][q, a] = q'$ where i is index of the array or the register (It will be described in detail in the next section).

The following procedure is to build layers of the set of states $V_0, \ldots, V_l$ and edges $E_1, \ldots, E_l$ corresponding to the length of language recognized by the finite automaton $\mathcal{A}$.

Procedure 1. *To build layers V_i and E_i*

1. $V_0 = I_\diamondsuit$, $E_i = \varnothing$
2. <u>Repeat:</u> *Known V_{i-1} and calculate V_i, E_i (for $i = 1, \dots, l$)*
3. $V_i = \varnothing$, $E_i = \varnothing$
4. *For each* $(0, q, i-1) \in V_{i-1}$
5. *For each* $a \in A$ and $q' \in Q$
6. *If* $(q, a, q') \in E(A)$ *then Add* $(0, q', i)$ *to* V_i. *That is* $V_i = \{(0, q', i)$ $\in Q_\diamondsuit | \exists\}$ *path has length i from* $q_0 \in I \rightsquigarrow q' \in \text{Proj}(V_i)$ $\}$
7. $E_i = E_i \cup \{((0, q, i-1), (i-1, a, i), (0, q', i))\}$
8. <u>Stop:</u> $(V_i = V_{k<i})$ *or* $(V_i = \varnothing)$

Procedure 1 stops if $(V_i = V_{k<i})$ or $(V_i = \varnothing)$. To mark these two cases, we use the variable $LAP = (i, k)$ in the case $V_i = V_{k<i}$; otherwise, $LAP = (i, i)$.

To reduce complexity of Procedure 1, in the initial setup of the finite automaton $\mathcal{A}$, corresponding to each edge (q, a, q'), we add an array variable *Trans*$(q, a, q') \in \{$*TRUE*, *FALSE*$\}$ to mark whether edges change successfully.

Procedure 2. *Put in and mark successfully transition edges.*

For $q = 1$ *to* k *do* // k *is the number of states.*
  For $a = 1$ *to* m *do* // m *is the number of characters in the alphabet* A.
    For $q = 1$ *to* k *do*
      Trans(q, a, q') = *FALSE*
For $i = 1$ *to CountArc do* // *CountArc is the number of edges.*
{ *Put in values* q, a, q'. // *Put in Edge* (q, a, q').
  $E[i][q, a] = q'$, *Trans*(q, a, q') = *TRUE* }
For each q *in* T *do* *Fin*(q) = *TRUE* // *Mark final states.*

Procedure 2 has the approximate complexity size of $O(m.k^2)$. Then, the line 6 of Procedure 1 can be replaced by:

If *Trans*(q, a, q') = *TRUE* *then* *Add* $(0, q', i)$ *to* V_i.

Therefore, each set V_{i-1} has the size $k_1 \leq k$. Then, if we have l sets $\{V_1, V_2, \dots, V_l\}$ then Procedure 1 has approximate complexity of $O(m.k^2 + l.m.k_1.k) \leq O(l.m.k^2)$.

Example 4.1. *Let $\mathcal{A} = (A, Q, \delta, I, T)$ be a finite automaton where with $A = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3\}$, $I = \{q_0\}$, $T = \{q_3\}$ and the edges (q_0, a, q_1), (q_0, b, q_2), (q_1, b, q_3), (q_2, a, q_1), (q_2, b, q_3).*

Easy to see that, $\mathcal{A}$ recognizes the language $\{(ab + cb + a)c^*ac, (a + c)d\}$ (cf. Figure 1). Then, the layers V_i and E_i of $\diamondsuit$-automaton are defined as follows: (cf. Figure 2)

$V_0 = \{(0, q_0, 0)\}$, $V_1 = \{(0, q_1, 1), (0, q_2, 1)\}$

$E_1 = \{((0, q_0, 0), (0, a, 1), (0, q_1, 1)), ((0, q_0, 0), (0, b, 1), (0, q_2, 1))\}$

$V_2 = \{(0, q_3, 2), (0, q_1, 2)\}$

$E_2 = \{((0, q_1, 1), (1, b, 2), (0, q_3, 2)), ((0, q_2, 1), (1, a, 2), (0, q_1, 2)),$
$((0, q_2, 1), (1, b, 2), (0, q_3, 2))\}$

$V_3 = \{(0, q_3, 3)\}$, $E_3 = \{((0, q_1, 2), (2, b, 3), (0, q_3, 3))\}$

$V_4 = \varnothing$, $LAP = (4, 4)$

Example 4.2. *Let $\mathcal{A} = (A, Q, \delta, I, T)$ be a finite automaton where $A = \{a, b, c, d\}$, $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $I = \{q_0, q_2\}$, $F = \{q_4\}$ and the edges (q_0, a, q_1), (q_0, a, q_2), (q_0, c, q_1), (q_1, b, q_2), (q_1, d, q_4), (q_2, a, q_3), (q_2, c, q_2), (q_3, c, q_4).*

Easy to see that, $\mathcal{A}$ recognizes the language $\{ab, bab, bb\}$ (cf. Figure 3). The layers V_i and transition states $E(\mathcal{A}_\diamondsuit)$ of $\diamondsuit$-automaton $\mathcal{A}_\diamondsuit$ are defined as follows: (cf. Figure 4)

$$V_0 = \{(0, q_0, 0), (0, q_2, 0)\},\ V_1 = \{(0, q_1, 1), (0, q_2, 1), (0, q_3, 1)\}$$

$$
\begin{aligned}
E_1 &= \{((0,q_0,0),(0,a,1),(0,q_1,1)),((0,q_0,0),(0,c,1),(0,q_1,1)),\\
&\quad ((0,q_0,0),(0,a,1),(0,q_2,1)),((0,q_2,0),(0,c,1),(0,q_2,1)),\\
&\quad ((0,q_2,0),(0,a,1),(0,q_3,1))\}\\
V_2 &= \{(0,q_2,2),(0,q_4,2),(0,q_3,2)\}\\
E_2 &= \{((0,q_1,1),(1,b,2),(0,q_2,2)),((0,q_1,1),(1,d,2),(0,q_4,2)),\\
&\quad ((0,q_2,1),(1,c,2),(0,q_2,2)),((0,q_2,1),(1,a,2),(0,q_3,2)),\\
&\quad ((0,q_3,1),(1,c,2),(0,q_4,2))\}\\
V_3 &= V_2,\ LAP = (3,2)
\end{aligned}
$$

Similar to Procedure 1, we can design an array to mark the edges $((0,q,j-1),(j-1,a_j,j),(0,q',j)) \in E_t$. Then, the set U_{j-1} has s_1 states $(s_1 \le k)$ and V_t has s_2 states

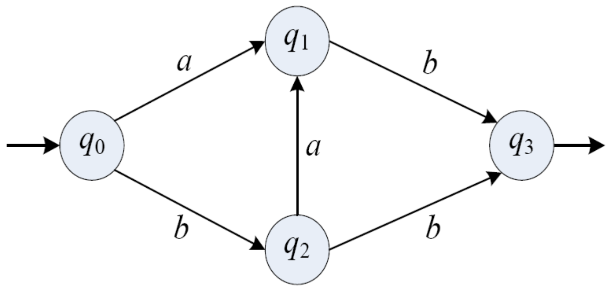

FIGURE 1. The finite automaton $\mathcal{A}$ recognizes $\{ab, bab, bb\}$

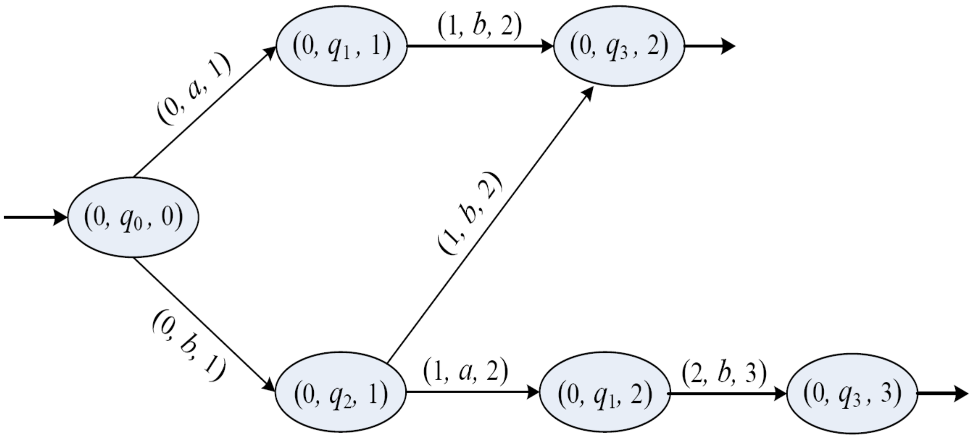

FIGURE 2. The finite $\Diamond$-automaton $\mathcal{A}_\Diamond$ extended from the finite automaton $\mathcal{A}$ in Figure 1

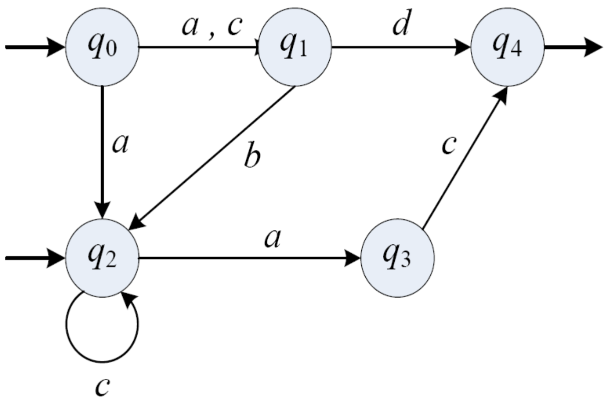

FIGURE 3. The automaton $\mathcal{A}$ recognizes $\{(ab \cup cb \cup c)c^*ac, (a \cup c)d\}$

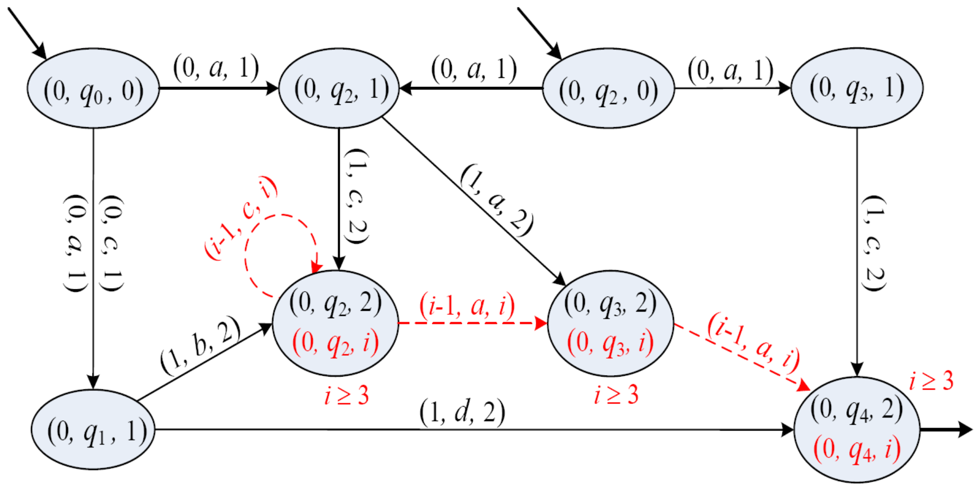

FIGURE 4. The ♢-automaton $\mathcal{A}_{\diamondsuit}$ extended from the automaton $\mathcal{A}$ in Figure 3

Algorithm 4. *Approach according to ♢-automata*

Step 1. Build sets of states V_i and the edges E_i recognizing a language with the length i. This step has the complexity size of $O(l.m.k^2)$.

Step 2. Check the string $S_i = a_1a_2 \dots a_l, a_j \in A$.

To check the string S_i, we check the string $S'_i = (0, a_1, 1).(1, a_2, 2) \dots (l-1, a_l, l)$ on finite ♢-automaton $A_{\diamondsuit}$.

```
(n, s) = LAP
If (n < l) and (n = s) then {KQ = False and Exit}
U_0 = I_◇, t = 0
Repeat: Consider the labels a_j in the string S_i
        U_j = ∅, t = t + 1
        For each pair of states (0, q, j − 1) ∈ U_{j−1} and (0, q′, j) ∈ V_t
            If ((0, q, j − 1), (j − 1, a_j, j), (0, q′, j)) ∈ E_t then Add (0, q′, j) to U_j.
        If (t = n) then t = s               // Repeat if V_t = V_s
Stop (1) If (j = l) and (U_j ∩ T_◇ ≠ ∅) then      {KQ = True and Exit}.
     (2) If (j = l) or (U_j = ∅) then             {KQ = False and Exit}.
```

$(s_2 \leq k)$. Therefore, Step 2 has the complexity size of $O(l.s_1.s_2)$. Checking the strings $\{S_1, S_2, \dots, S_N\}$ on the finite ♢-automaton $\mathcal{A}_{\diamondsuit}$ has the complexity size of $O(l.s_1.s_2.N)$. Therefore, Algorithm 4 has the approximate complexity size of $O(l.m.k^2 + l.s_1.s_2.N) \leq O(l.k^2.(m + N))$.

Example 4.3. *Using the finite automaton $\mathcal{A}$ in Example 4.2, check whether following strings $S \in A^*$ are recognized by the automat $\mathcal{A}$ or not.*

a) With $S = abcac \in A^*$. Implement steps of Algorithm 4:

$l = |S| = 5$ *and* $S_{\diamondsuit} = (0, a, 1).(1, b, 2).(2, c, 3).(3, a, 4).(4, c, 5)$

$U_0 = I_{\diamondsuit} = \{(0, q_0, 0), (0, q_2, 0)\}$, $U_1 = \{(0, q_1, 1), (0, q_2, 1), (0, q_3, 1)\}$

$U_2 = \{(0, q_2, 2)\}$; $U_3 = \{(0, q_2, 3)\}$; $U_4 = \{(0, q_3, 4)\}$; $U_5 = \{(0, q_4, 5)\}$

It is realized that, $(0, q_4, 5) \in T_{\diamondsuit}$. Therefore, the string $S = abcac$ is recognized by the finite automaton $\mathcal{A}$.

b) With $S = acbc \in A^*$. Implement steps of Algorithm 4:

$l = |S| = 5$ *and* $S_{\diamondsuit} = (0, a, 1).(1, b, 2).(2, d, 3).(3, c, 4)$

$U_0 = I_{\diamondsuit} = \{(0, q_0, 0), (0, q_2, 0)\}$, $U_1 = \{(0, q_1, 1), (0, q_2, 1), (0, q_3, 1)\}$

$U_2 = \{(0, q_2, 2)\}$, $U_3 = \varnothing$.

Therefore, the string $S = acbc$ is not recognized by the finite automaton $\mathcal{A}$.

With the above finite ◇-automaton model approach, we have a new view of the layers based on the length of the recognized word. If we use function *Proj*() defined in Section 1, we can present Algorithm 4 in array structure form: sets of states V_i, U_j are corresponding to the 2-dimensional arrays $V[i][p]$ and $U[j][q]$, the edge E_i is a 3-dimensional array $E[i][q, a]$, we have

Algorithm 5. *Set up in the array structure.*

Step 1. Build sets $V[i][q]$ *and the edges* $E[i][q, a]$.

Step 2. Check the string $S_i = a_1 a_2 \ldots a_l, a_j \in A$.

$(n, s) = LAP$
If $(n < l)$ *and* $(n = s)$ *then* $\{KQ = False \text{ and } Exit\}$
For $k = 1$ *to* $|I|$ *do* $U[0][k] = I[k]$,
$Count\_U[0] = |I|,\ j = 0,\ t = 0$
Do While $(j < l)$
{ // *Known* $U[j-1]$ *and the label* $a[j]$. *Calculate* $U[j]$
  $Count\_U[j] = 0,\ j = j + 1,\ t = t + 1$
  For $p = 1$ *to* $Count\_U[j-1]$ *do*
  For $q = 1$ *to* $Count\_V[t]$ *do*
    If $Trans(U[j-1][p], a[j], V[t][q]) = TRUE$ *then*
    $\{Count\_U[j] = Count\_U[j] + 1,\ U[j][Count\_U[j]] = q\}$
  If $(t = n)$ *then* $t = s$ // *Repeat if* $V[t] = V[s]$.
  If $(j = l)$ *and* $(U[j] \cap T \neq \varnothing)$ *then* $\{KQ = True \text{ and } Exit\}$.
  If $(j = l)$ *or* $(Count\_U[j] = 0)$ *then* $\{KQ = False \text{ and } Exit\}$. }

5. **Conclusion.** In this paper, new types of automata are introduced. Our result shows that these automata can be considered as extension forms of traditional automata. Hence, a new perspective on mathematical model of automaton is given and a new checking algorithm is proposed, greatly reducing the complexity of the checking algorithm. Results obtained to ◇-automaton enrich theory of languages and can provide us some applications such as establishing new trapdoors in the area of cryptography.

REFERENCES

[1] S. Eilenberg, *Automata, Languages and Machines*, Volume B, Academic Press, New York, 1976.

[2] D. J. E. Hopcroft and J. D. Ullman, *Formal Languages and Their Relation to Automata*, Addison-Wesley Publishing Company, 1969.

[3] H. N. Vinh and P. T. Huy, Codes of bounded words, *Proc. of the 3rd International Conference on Computer and Electrical Engineering (ICCEE 2010)*, Chengdu, China, 2010.

[4] H. N. Vinh and N. D. Han, A novel method based on the post correspondance problem for designing cryptosystems, *Journal of Science & Technology – Technical Universities*, no.121, pp.58-63, 2017.

[5] T. Jastrzab, On parallel induction of nondeterministic finite automata, *Procedia Computer Science*, vol.80, pp.257-268, 2016.

[6] N. D. Han, H. N. Vinh, D. Q. Thang and P. T. Huy, Quadratic algorithms for testing of codes and ◇-codes, *Fundamenta Informaticae*, vol.130, pp.1-15, 2014.

[7] T. Marschall, Construction of minimal deterministic finite automata from biological motifs, *Theoretical Computer Science*, vol.412, pp.922-930, 2011.