ITAT 2013 Proceedings, CEUR Workshop Proceedings Vol. 1003, pp. 52–57
http://ceur-ws.org/Vol-1003, Series ISSN 1613-0073, © 2013 G. Andrejková, A. Almarimi, A. Mahmoud

Approximate Pattern Matching using Fuzzy Logic *

Gabriela Andrejková, Abdulwahed Almarimi and Asmaa Mahmoud

Institute of Computer Science, Faculty of Science
P. J. Šafárik University in Košice, Slovakia
gabriela.andrejkova@upjs.sk

Abstract: Pattern matching problem is still very interesting and important problem. Algorithms for the exact pattern matching search for exact patterns in some texts or figures. Algorithms for an approximate pattern matching search for exact and similar patterns with some errors. They use some measures to evaluate a similarity of found similar patterns. In the area of the pattern matching allowing errors is possible to use fuzzy logic theory. In the paper we present the algorithm for a fuzzification of a deterministic finite state automaton using a similarity function of characters. The fuzzified automaton will accept exact and similar words. The second presented algorithm is a fuzzy modification of Aho-Corasick pattern matching algorithm which still work in linear time with respect to the length of the searching text.
Key words: Pattern matching, fuzzy logic, fuzzy automaton.

1 Introduction

The motivation to the Fuzzy Pattern Matching Problem (FPMP) can be found in Exact Pattern Matching Problem (EPMP). Words they are very closed to patterns (maybe words with one error) will not be found in EPMP. A quite good example is the typing of some text on the keyboard [2]. The following errors can be done in typing some text:

1. Typing a different character, usually from the neighborhood of the current character on a keyboard.
2. Inserting one or more characters into the source text.
3. Omitting any single character from the text.
4. Transposition of neighbor elements in the source text.

The most frequent error is the following: instead of the required character is typed a character from the area on the keyboard adjacent to the required character. For example, the neighborhood of the character **f** is the set $\mathbf{f_n} = \{\mathbf{f},\mathbf{d},\mathbf{g},\mathbf{r},\mathbf{t},\mathbf{c},\mathbf{v}\}$. The set of characters $\mathbf{A} = \{\mathbf{f},\mathbf{r},\mathbf{o},\mathbf{l},\mathbf{i},\mathbf{c}\}$ belongs to the pattern **frolic**. In this case of typing errors, let us assign **similarity value** (f) to each element of the neighborhood in such way that the character itself has f equal to 1 and the characters from the f's neighborhood have f value < 1, because they really represent some error.

*Supported by the Slovak Scientific Grant Agency VEGA, Grant No. 1/0479/12.

The similarity values of characters could be prepared in many ways, for example the closest characters to the character on keyboard, the similar characters to the given character because of they have the same shape, and so on. And it is possible to use similarity values as some fuzzy values of characters [3]. Several fuzzifications of formal concept analysis have been proposed in [8]. For example, for the set $\mathbf{f_n}$ should be $f(\mathbf{f},\mathbf{f}) = 1, f(\mathbf{f},\mathbf{d}) = 0.4, f(\mathbf{f},\mathbf{r}) = 0, f(\mathbf{f},\mathbf{g}) = 0.1, f(\mathbf{f},\mathbf{t}) = 0.4, f(\mathbf{f},\mathbf{c}) = 0.3, f(\mathbf{f},\mathbf{v}) = 0.3$. We consider that in the text, it is necessary to find the words they are very closed to the pattern **frolic**. We could consider the sum of f's of a given string as a measure of its similarity of the found string to the pattern **frolic** [2]. But we can lose the information if the found word is exact or not if the pattern is a subsequence of the found word. For example, if the pattern **frolic** is found in the text then the measure of the similarity to the found word **frolic** is the length of the word **frolic** (equal to 6). The word **froolic** is very closed to the pattern and the measure of the similarity is 6 too, because the symbol **o** can be deleted. But the word **froolic** is not exact word. The used measure of the similarity words has some problems in this case. We will apply measures based on fuzzy logic.

In the string matching problem allowing errors, an input text, maybe containing errors, and a pattern string are compared in order to find an imperfect pattern in input text. Very interesting fuzzy measure between strings using fuzzy automata with ε-moves was given in [4]. Fuzzified Aho-Corasick (FAC) search automata were described in [10]. Some examples of an application of a similarity function to nondeterministic finite automata is described in [9]. We propose a theoretical description of fuzzy operations in a fuzzy automaton using relations on fuzzy sets. We developed a modified fuzzy automaton, named a FAC automaton working in a linear time in the length of searched texts. In the applied FAC algorithm we did some restrictions coming from practice: (1) restrictions on sets of similar symbols, (2) restrictions on the number of mistakes in words. The applied FAC automaton was tested on some small texts with acceptable results.

The paper is organized as follows: Section 2 presents used notions and concepts needed to understand the problem, mainly fuzzy logic connectives and fuzzy automata. In Section 3 we built fuzzy automaton from a deterministic finite state automaton and a similarity function. In the following sections we present the algorithm for fuzzy pattern matching and some results of its using.

2 Preliminaries

A **fuzzy set** A in a referential universal set U is characterized by a membership function which associates to each element $x \in U$ a real number $A(x) \in [0,1]$, $[0,1]$ is the interval of real numbers, $\{0,1\}$ is the set of two numbers. The value of the membership function of $x \in U$ represents the **membership degree** of x in A. We use $\mathscr{F}(U)$ to denote the set of all fuzzy sets on U. As the basic book for notions in the theory of fuzzy sets and fuzzy logic, we used Gottwald's book [5].

In the theory of fuzzy sets, triangular norms have been used for defining the intersection of fuzzy sets and for modeling the logical conjunction in fuzzy logic.

Definition 1. *A mapping $T : [0,1] \times [0,1] \to [0,1]$ is called a* **triangular norm**, *or* **t-norm**, *if it satisfies at least axioms of*

- *identity element, 1, i. e. $T(x,1) = T(1,x) = 1Tx = xT\,1 = x$ for each $x \in [0,1]$.*
- *monotonicity, non-decreasing in each argument,*
- *commutativity and associativity.*

□

t-norms are considered as truth functions of generalized conjunction operators. s-conorms are considered as truth functions of generalized disjunction operators.

Remark 1. *Well known t-norms:*

1. *Gödel conjunction: $xT_G y = T_G(x,y) = min(x,y)$,*
2. *Lukasiewicz conjunction: $xT_L y = T_L(x,y) = max\{x+y-1,0\}$,*
3. *Product conjunction: $xT_P y = T_P(x,y) = x*y$.*

Let A be a fuzzy set of set U. A binary fuzzy relation on A is any fuzzy set of $A \times A$.

Definition 2. *For a fuzzy subset V of A and a binary fuzzy relation R on A, the compositions $V \bullet R$ and $R \bullet V$ are fuzzy subsets of A defined, for any $x \in A$, by*

$$(V \bullet R)(x) = \max_{y \in A}\{T(V(y), R(y,x))\}, \quad (1)$$

$$(R \bullet V)(x) = \max_{y \in A}\{T(R(x,y), V(y))\} \quad (2)$$

□

Example 1. *Let $V = \{(a,0.4),(b,0.7),(c,0.5)\}$ be the fuzzy subset of A. Let $A \times A$ be the binary fuzzy relation defined by the Table 1. The composition of V by $A \times A$ can be done from the left and from the right side. If $A \times A$ is not symmetric relation then we can get different fuzzy sets.*

Definition 3. *A* **fuzzy finite state automaton** *FA is a quintuple (Σ, Q, μ, S, F), where:*

	$A \times A$			$(V \bullet R)_G$	$(R \bullet V)_G$
	a	b	c		
a	0.3	0.4	0.2	0.3	0.4
b	0.2	0.6	0.3	0.6	0.6
c	0.1	0.9	0.3	0.7	0.3

Table 1: The composition of a fuzzy set and a binary fuzzy relation ($\bullet$), index G means using of Gödel conjunction.

- *Σ is a non-empty finite set of input characters (input alphabet),*
- *Q is a non-empty finite set of states,*
- *$S, S \subseteq Q$ is a fuzzy set of a starting states on Q,*
- *$F, F \subseteq Q$ is a fuzzy set of final (accepting) states on Q,*
- *$\mu : Q \times Q \times \Sigma \to [0,1]$ is the state transition function,*
- *μ can be decomposed to $|\Sigma|$ binary relations, for each character $x \in \Sigma$, in the following way:*

  $\mu_x : Q \times Q \to [0,1]$ is a binary fuzzy relation on Q - fuzzy transition matrix of order $|Q|$.
- *μ can be decomposed to $|Q|$ binary relations, for each state $q \in Q$, in the following way:*

  ${}_q\mu : Q \times \Sigma \to [0,1]$ is a binary fuzzy relation on $Q \times \Sigma$ - fuzzy transition matrix of order $|Q| \times |\Sigma|$.

□

For each $x \in \Sigma$, μ_x is **binary fuzzy relation** on a set of states Q. It is a fuzzy set of the Cartesian product $Q \times Q$. $\mu_x(p,q) \in [0,1]$ for each pair $(p,q) \in Q \times Q$ and it can be explained as the compliance degree of interaction between states p and $q \in Q$ in symbol $x \in \Sigma$.

μ	μ_a			μ_b			μ_c		
Sts	q_0	q_1	q_2	q_0	q_1	q_2	q_0	q_1	q_2
q_0	.4	0	0	0	.1	0	0	0	0
q_1	0	0	.5	0	0	0	0	1	0
q_2	0	0	0	0	0	.7	0	0	0

Table 2: Decomposition of the transition function for all characters of fuzzy automaton FA_1 in the Example 2.

Example 2. *The fuzzy finite automaton $FA_1 = (\Sigma, Q, \mu, S, F), \Sigma = \{a,b,c\}, Q = \{q_0,q_1,q_2\}, S = \{q_0\}, S(q_0) = 1, F = \{q_2\}, F(q_2) = 1$, and transition functions for symbols in Σ are in the Table 2, the automaton is drawn in the Figure 1 . $\mu_b(q_0,q_1) = 0.1$ and it can be explained as the compliance degree of interaction between states q_0 and q_1 for the symbol b. $\mu_a(q_0,q_0) = 0.4$ is the compliance degree of interaction between states q_0 and q_0 for the symbol a.*

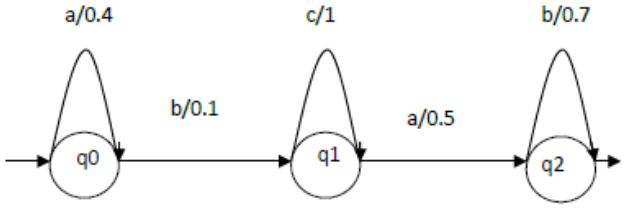

Figure 1: The graph of transition functions of the fuzzy finite automaton FA_1 in the example 1. The symbols with membership values 0 are not drawn.

Remark 2. *In the following text*

1. Σ^+ *is the set of all non-empty strings over* Σ *and* $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$.
2. *The term* **fuzzy state of an automaton** *(shortly fuzzy state) is used to refer to a fuzzy set of states over* Q. *Fuzzy state* $V \in \mathscr{F}(Q)$ *is some set of states with membership values. For example,* $V = \{(q_0,1),(q_1,0),(q_2,0)\}$ *is a fuzzy state.*
3. *The composition of a fuzzy state* $V \in \mathscr{F}(Q)$ *and a binary fuzzy relation* μ_x, $x \in \Sigma$, *on* $Q \times Q$ *is defined, for each* $p \in Q$, *by*

$$(V \bullet_T \mu_x)(p) = \max_{q \in Q}\{T(V(q), \mu_x(q,p))\} \quad (3)$$

Example 3. *The composition of a state* $V_1 = \{(q_0,1),(q_1,0),(q_2,0)\}$ *and the binary fuzzy relation* μ_b *from the example 1 using Gödel norm* T_G *gives the new fuzzy state* V_2*:*

$(V_1 \bullet_{T_G} \mu_b)(q_0) = \max_{q\in Q}\{T_G(V_1(q), \mu_b(q,q_0))\} = V_2(q_0) = 0,$
$(V_1 \bullet_{T_G} \mu_b)(q_1) = \max_{q\in Q}\{T_G(V_1(q), \mu_b(q,q_1))\} = V_2(q_1) = 0.1,$
$(V_1 \bullet_{T_G} \mu_b)(q_2) = \max_{q\in Q}\{T_G(V_1(q), \mu_b(q,q_2))\} = V_2(q_2) = 0,$
$V_2 = \{(q_0,0),(q_1,0.1),(q_2,0)\}.$

Definition 4. *Let* R_1 *and* R_2 *be two fuzzy binary relations on* $Q \times Q$, T *be some t-norm. The* **max-T** *composition between* R_1 *and* R_2, *denoted* $R_1 \circ_T R_2$, *is the fuzzy set on* $Q \times Q$ *such that for all* $(p,q) \in Q \times Q$

$$R_1 \circ_T R_2(p,q) = max_{r\in Q}\{T(R_1(p,r), R_2(r,q))\}. \quad (4)$$

□

Example 4. *Let* $R_1 = \mu_a$ *and* $R_2 = \mu_b$ *be two fuzzy relations on* $Q \times Q$ *in the Example 2. Let* T_G *be Gödel t-norm. The* $max - T_G$ *composition between* μ_a *and* μ_b, *is given in the Table 3. For example,* $\mu_a \circ_{T_G} \mu_b(q_0,q_1)$ *can be computed as*

$$\mu_a \circ_{T_G} \mu_b(q_0,q_1) = max_{r\in Q}\{T_G\{\mu_a(q_0,r), \mu_b(r,q_1)\}\} = .1,$$

where
$min\{\mu_a(q_0,q_0), \mu_b(q_0,q_1)\} = min\{0.4, 0.1\} = 0.1,$
$min\{\mu_a(q_0,q_1), \mu_b(q_1,q_1)\} = min\{0,0\} = 0,$
$min\{\mu_a(q_0,q_2), \mu_b(q_2,q_1)\} = min\{0,0\} = 0,$
$max\{0.1,0,0\} = 0.1$

μ	μ_a			μ_b			max-T		
Sts	q_0	q_1	q_2	q_0	q_1	q_2	q_0	q_1	q_2
q_0	.4	0	0	0	.1	0	0	0	0
q_1	0	0	.5	0	0	0	0	0	.5
q_2	0	0	0	0	0	.7	0	0	0

Table 3: $Max - T_G$ composition of relations μ_a and μ_b from Example 2.

The computation of a fuzzy finite state automaton *FA* is formally described in terms of strings of input symbols that are accepted by it.

Definition 5. *Let* $FA = (\Sigma, Q, \mu, S, F)$ *be a fuzzy finite state automaton.*

(i) $\hat{\mu} : \mathscr{F}(Q) \times \Sigma \to \mathscr{F}(Q)$ *is the* **fuzzy state transition function**. *Given a fuzzy state* $V \in \mathscr{F}(Q)$ *and a symbol* $a \in \Sigma$ *it is* $\hat{\mu}(V,a) = V \bullet_T \mu_a$, *the result is a fuzzy state.*

(ii) $\mu^* : \mathscr{F}(Q) \times \Sigma^* \to \mathscr{F}(Q)$ *is the* **extended transition function** *defined as*

(a) $\mu^*(V,\varepsilon) = V$, *for all* $V \in \mathscr{F}(Q)$ *(the result is a fuzzy state).*

(b) $\mu^*(V,\alpha x) = \hat{\mu}(\mu^*(V,\alpha),x) = \mu^*(V,\alpha) \bullet_T \mu_x$, *for all* $V \in \mathscr{F}(Q), \alpha \in \Sigma^*$ *and* $x \in \Sigma$.

- *The* **language accepted by** *FA, denoted* $\mathscr{L}(FA)$, *is the fuzzy set on* Σ^* *such that* $\mathscr{L}(FA)(\alpha) = \max_{q\in Q}\{T(\mu^*(S,\alpha)(q), F(q))\}$ *for all* $\alpha \in \Sigma^*$.

□

Example 5. *The value of the membership function for some word in* $\mathscr{L}(FA)$ *depends on used t-norm. For example, the word* $\alpha =' bca'$ *is accepted by* FA_1 *in the Example 1. The starting fuzzy state is* $S = \{(q_0,1),(q_1,0),(q_2,0)\}$ *and the finite fuzzy state is* $S = \{(q_0,0),(q_1,0),(q_2,1)\}$. *The membership values are:*

- T_G: $\mathscr{L}(FA_1)(\alpha) = \max_{q\in Q}\{T_G(\mu^*(S,\alpha)(q), F(q))\}$,
$\mu^*(S,'bca') = \hat{\mu}(\mu^*(S,'bc'),a),$
$\mu^*(S,'bc') = \hat{\mu}(\mu^*(S,'b'),c),$
$\mu^*(S,'b') = \hat{\mu}(\mu^*(S,\varepsilon),b) = S \bullet_{T_G} \mu_b = \{(q_0,0),(q_1,0.1),(q_2,0)\} = S_1,$
$\mu^*(S,'bc') = \hat{\mu}(\mu^*(S,'b'),c) = \hat{\mu}(S_1,c) = \{(q_0,0),(q_1,0.1),(q_2,0)\} = S_2,$
$\mu^*(S,'bca') = \hat{\mu}(\mu^*(S,'bc'),a) = \hat{\mu}(S_2,a) = \{(q_0,0),(q_1,0),(q_2,0.1)\} = S_3.$

Using the Gödel conjunction the membership value of the word is 0.1.

- T_L: $\mathscr{L}(FA_1)(\alpha) = \max_{q\in Q}\{T_L(\mu^*(S,\alpha)(q),F(q))\}$, *Using the Lukasiewics conjunction the membership value of the word is* 0. *It means the word is not accepted by the automaton.*

- T_P: $\mathscr{L}(FA_1)(\alpha) = \max_{q\in Q}\{T_P(\mu^*(S,\alpha)(q),F(q))\}$, *Using the Product conjunction the membership value of the word is* 0.02.

3 Fuzzification of DFA using a similarity function

We will work with some deterministic finite state automaton (DFA) M defined by [6], $M = (\Sigma, Q, \delta, q_0, F)$, where Σ is non-empty finite set of characters, Q is a non-empty finite set of states, q_0 is the starting state, F is the set of finite states, and $\delta : Q \times \Sigma \to Q$ is the state transition function. δ can be decomposed to binary relations according to states in Q, $\delta_q : Q \times \Sigma \to \{0,1\}, q \in Q$.

Example 6. *Let $\mathscr{A}$ be a deterministic finite state automaton $\mathscr{A} = (\Sigma, Q, \delta, q_0, F)$, $\Sigma = \{A,B,C\}, Q = \{q_0,\ldots,q_7\}$, $F = \{q_4, q_6, q_7\}$, δ is in the Table 6. Some results of the decomposition of δ according to states is in the Table 5.*

Sts	A	B	C
$\to q_0$	q_1	q_5	–
q_1	–	q_2	–
q_2	q_3	q_5	–
q_3	–	q_4	–
$q_4 \to$	–	q_5	–
q_5	–	q_6	q_7
$q_6 \to$	–	–	–
$q_7 \to$	–	–	–

Table 4: Transition function δ of the automaton $\mathscr{A}$.

Definition 6. *Let R be a binary fuzzy relation on Q, T be a t-norm.*

- *R is* **t-transitive** *if for all $p,q,r \in Q$, $T(R(p,r),R(r,q)) \le R(p,q)$.*
- *R is* **reflexive** *if $R(p,p)$ is defined for each $p \in Q$.*
- *R is* **symmetric** *if $R(p,q) = R(q,p)$ for all $p,q \in Q$.*

If R is reflexive, and symmetric then R is called **proximity relation**. *If R is also t-transitive, then R is called* **t-similarity** *relation.*

Definition 7. *Let $\Sigma = \{a_1, a_2, \ldots a_n\}$ be some finite alphabet of characters used in some text. Each function $f : \Sigma \times \Sigma \to [0,1]$ defined by (5) is called* **similarity function**.

$$f(a_i,a_j) = f(a_j,a_i) = \begin{cases} 1, & \text{if } a_i = a_j, \\ v, & v \in [0,1), \text{ if } a_i \neq a_j. \end{cases} \quad (5)$$

□

δ_*	δ_{q_0}			δ_{q_2}			δ_{q_5}		
Sts	*A*	*B*	*C*	*A*	*B*	*C*	*A*	*B*	*C*
q_0	0	0	0	0	0	0	0	0	0
q_1	1	0	0	0	0	0	0	0	0
q_2	0	0	0	0	0	0	0	0	0
q_3	0	0	0	1	0	0	0	0	0
q_4	0	0	0	0	0	0	0	0	0
q_5	0	1	0	0	1	0	0	0	0
q_6	0	0	0	0	0	0	0	1	0
q_7	0	0	0	0	0	0	0	0	1

Table 5: Examples of the decomposition δ_* functions.

The similarity function defines similarity level between each pair of characters in Σ. A value $v \in [0,1]$ is depending on the similarity of characters a_i and a_j. The similarity function can be used as a proximity relation, it is a binary symmetric and reflexive fuzzy relation on $\Sigma \times \Sigma$.

To the composition of the state transition function δ and the similarity function it is necessary to choose some adequate fuzzy logic connectives. One of them is a **max-T** composition of two binary relations defined by (6).

Definition 8. *Let R_1 be a fuzzy binary relation on $Q \times \Sigma$ and R_2 be a fuzzy binary relation on $\Sigma \times \Sigma$. The* **max − T** *composition between R_1 and R_2, denoted $R_1 \circ_T R_2$, is the fuzzy set on $Q \times \Sigma$ such that for all $p \in Q$ and $a \in \Sigma$*

$$R_1 \circ_T R_2(p,a) = max_{x\in\Sigma}\{T(R_1(p,x),R_2(x,a))\}. \quad (6)$$

□

Let δ_q, for some $q \in Q$, be a binary (fuzzy) relation on $Q \times \Sigma$ and R_2 be a proximity relation representing a similarity function f. Using (6) we will get the decomposed (according to states) fuzzy transition functions of the corresponding fuzzy automaton ${}_q\mu : Q \times \Sigma \to [0,1]$ for each $q \in Q$. The fuzzy transition function is $\mu : Q \times Q \times \Sigma \to [0,1]$.

$$\begin{aligned} \mu(q,p,a) &= {}_q\mu(p,a) = (\delta_q \circ_T f)(p,a) \\ &= max_{x\in\Sigma}\{T(\delta_q(p,x), f(x,a))\} \end{aligned} \quad (7)$$

Example 7. *The application of* **max − T$_G$** *(Gödel conjunction) composition to δ_q and the similarity function f.*

Similarity function:
$f(A,A) = f(B,B) = f(C,C) = 1, f(A,B) = f(B,A) = 0$, $f(B,C) = f(C,B) = 0, f(C,A) = f(A,C) = 0.3$.

μ	μ_{q_0}			μ_{q_2}			μ_{q_5}		
Sts	A	B	C	A	B	C	A	B	C
q_0	0	0	1	0	0	0	0	0	0
q_1	1	0	0	0	0	0	0	0	0
q_2	0	0	0	0	0	0	0	0	0
q_3	0	0	0	1	0	0.3	0	0	0
q_4	0	0	0	0	0	0	0	0	0
q_5	0	1	0	0	1	0	0	0	0
q_6	0	0	0	0	0	0	0	1	0
q_7	0	0	0	0	0	0	0	0	1

Table 6: Examples of a fuzzification of a transition function.

If we analyze the formula (7), we should see that the value $\delta_q(p,x)$ represent the transition value in DFA, it means $\delta_q(p,x) \in \{0,1\}$. From the property follows that in the formula (7) should be used arbitrary t-norm T. The algorithm needs information about DFA and the similarity function. Using nested four cycles to formula (7) we will get the fuzzy transition function of the new automaton $\mathscr{FA} = (\Sigma, Q, \mu, \{q_0^f\}, F^f)$, where $\{q_0^f\}, F^f)$ are sets of fuzzy states. The time complexity is $O(|Q|^2.|\Sigma|^2)$ in the worst case. In the common case, $\mathscr{FA}$ can be a nondeterministic finite automaton. Using some special conditions in DFA or in similarity functions $\mathscr{FA}$ can be deterministic.

4 Pattern matching

Aho-Corasick algorithm (ACA) [1] is used to search the small number of exact key words in some long texts. AC algorithm constructs the special automaton to accept key words, the automaton is DFA. The construction is modified using a similarity function using composition operation of binary relations described by formula (7). The similarity function will be prepared according to errors done by a typing of people on keyboard. We suppose that each set of similar characters has less or equal two elements and the similarity function is symmetric one.

The three basic functions of Aho-Corasick algorithm are:

- a **Goto** function based on an automaton DFA, which maps (state, character) pairs to states and occasionally emits an output,
- a **Failure** function, which tells the Goto function which state to jump into when the character it just read doesn't match anything,
- an **Output** function, which maps states to outputs - sometimes more outputs than one per state.

4.1 Failure function

The construction of failure function $fail$ depends on some transitions with fuzzy values greater than 0. The construction is done in a recursive way using a queue of states they are waiting for processing.

Method:

1. All states in the automaton for them exists some transition from state q_0 will be put into the queue. The queue in our Example 7: $(:1,5:)$.

   The value of the failure function $fail$ in the state q_0 and in all states with possible transition from the state q_0 will be q_0, $fail[q_0] = q_0.fail[s] = q_0$ if there exists some transition from state q_0 to s.

2. While the queue is not empty the first element is taken from the queue to r and transitions from r for all symbols of alphabet are analyzed. If the transition for symbol i is possible then to t is assigned $fail(r)$ and to s the result state of the transition from the state r through symbol i. While a transition from state t through symbol i is not possible then $t := fail(t)$. After finding the possible transition from state t through symbol i to state v, the value of the failure function in the state s is put to state v. The state s is put in the end of the queue.

4.2 Fuzzy Aho-Corasick algorithm:

1. To prepare the searched patterns, let P be the length of all patterns. To build the alphabet of used characters $\Sigma, |\Sigma| \leq P$. Σ will be the alphabet of DFA. To prepare the function f of similarity for characters. The similarity function describes the measure of character similarities.

2. To process all searched patterns and built the Aho-Corasick Automaton (ACA) with the transition function δ. $|Q|, |Q| \leq P$ is the number of states. To remove all transitions from q_0 to q_0. Let Σ_{q_0} be the alphabet of characters of possible transitions from state q_0 to the some other state (not to q_0). The time complexity is $O(|Q|^2.|\Sigma|)$.

3. To prepare the fuzzification of ACA and to process the output of each state together with membership values. To return all removed transitions from q_0 to q_0 using the alphabet Σ_{q_0}. We get FACA. The time complexity is $O(|Q|^2.|\Sigma|^2) = O(P^4)$ in the worst case.

4. To build failure function $fail$ and modified output of all states in FACA. The time complexity is $O(|Q|^2)$.

5. To use FACA for a searching of patterns in some text given in the file and to print all founded positions of the found patterns. The FACA finds the patterns and their membership values. the text is read is analyzed in one step, it means, the linear time in the length of the text. Many texts should be searched by the prepared automaton and they will be read once only.

5 Results in the application

In the application we used the following restrictions: (a) one error at most in the word, (b) the set of similar characters has two characters at most.

Example 8. *We construct the fuzzy automaton to the following patterns.*
Patterns: *"ABAB", "BB", "BC".*
Alphabet: $\Sigma = \{A, B, C\}$.
The deterministic finite state automaton *is* $A = (\Sigma, Q, \delta, q_0, F)$, $Q = \{q_0, \ldots, q_7\}$, $F = \{q_4, q_6, q_7\}$. δ *is in the Table 6.*

We will use the similarity function from Example 7.

The fuzzy transition function from a state using a character to some state. The value -1 *means any transition.*

Real numbers present membership values. In the output, there are exact and similar words together with their membership values to the accepted language.

Failure function $fail$**:**

From	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
To state	q_0	q_0	q_5	q_7	q_2	q_0	q_5	q_1

Transition function of the FACA

Sts	A	B	C	Output
q_0	q_1 : 1.0	q_5 : 1.0	q_1 : 0.3	-
q_1	-1 : 0.0	q_2 : 1.0	-1 : 0.0	-
q_2	q_3 : 1.0	-1 : 0.0	q_3 : 0.3	-
q_3	-1 : 0.0	q_4 : 1.0	-1 : 0.0	:BC, 1.00: :BA, 0.30:
q_4	-1 : 0.0	-1 : 0.0	-1 : 0.0	:ABAB, 1.00: :CBAB, 0.30: :ABCB, 0.30:
q_5	q_7 : 0.3	q_6 : 1.0	q_7 : 1.0	-
q_6	-1 : 0.0	-1 : 0.0	-1 : 0.0	:BB, 1.00:
q_7	-1 : 0.0	-1 : 0.0	-1 : 0.0	:BC, 1.00: :BA, 0.30:

Accepting states: $3, 4, 6, 7$.

Output:

State	# words	words exact and similar
3	2	BC, BA
4	3	ABAB, CBAB, ABCB
6	1	BB
7	2	BC, BA

The application of the constructed automaton to the following text string: '**ABABBCCCBAB**' gives the following results:
The number of found words: 5

pattern	position in text	fuzzy value
ABAB	1	1.00
BC	2	0.30
BB	4	1.00
BC	5	1.00
ABAB	7	0.30

6 Conclusion

In the paper we give some information about possibility to use fuzzy logic theory in the area of the pattern matching. The exact pattern matching will find the exact words only and do not give any information about words with one error only. The information that the word is very closed to the pattern should be very important. It is enough for people to analyze this word in next time.

We developed the fuzzy version of Aho-Corrasick algorithm which can find the similar fuzzy words. In the following work we will test some fuzzy aggregate functions to evaluate a similarity of the words.

References

[1] A. V. Aho, M. J. Corasick: Efficient string matching: an aid to bibliographic search. Communications of the ACM, Vol. 18, 1975, p. 333 - 340.

[2] G. Andrejková: The set closest common subsequence problem. In: Proceedings of 4th International Conference on Applied Informatics'99, Eger - Noszvaj 1999, p. 8.

[3] G. Andrejková: The similarity of two strings of fuzzy sets. Kybernetika, vol. 36 (2000), issue 6, pp. 671 - 687

[4] J. J. Astrain, J. R. Gonzalez de Mendivil, J. R. Garitagoitia: Fuzzy automata with ε–moves compute fuzzy measures between strings. Fuzzy Sets and Systems 157 (2006) p. 1550 -1559.

[5] S. Gottwald: Fuzzy sets and fuzzy logic. Verlag Vieweg, Braunschweig, 1993.

[6] J. E. Hopcroft, J. D. Ullman: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading, MA, USA, 1979.

[7] Z. Horák, V. Snášel, A. Abraham, A. E. Hassanien: Fuzzified Aho-Corrasick Search Automata. In Proceedings of IAS, 2010, p. 338-342.

[8] J. Medina, M. Ojeda-Aciego, J. Ruiz-Calvino: Formal concept analysis via multi-adjoint concept lattices. Fuzzy Sets and Systems, Volume 160 Issue 2, January, 2009, p. 130-144.

[9] V. Ramaswamy, H. A. Girijamma: Fuzzy Automata for String Comparison. International Journal of Computer Application, Vol. 37, No. 8, 2012, p. 1 - 4.

[10] V. Snášel, A. Keprt, A. Abraham, and A. E. Hassanien: Approximate pattern matching using fuzzy automata. In: K. A. Cyran et (Eds.): Man-Machine Interactions, AISC 59, Springer-Verlag Berlin Heidelberg 2009, p. 281 - 290.