Mathematical Problems of Computer Science 56, 56–64, 2021.
doi:10.51408/1963-0079

UDC 519.688

Determining the Degree of Fuzzy Regularity of a String

Armen H. Kostanyan

IT Educational and Research Center
Yerevan State University
e-mail: armko@gmail.com

Abstract

The paper deals with the issue of determining the degree of fuzzy regularity of a crisp string. It is assumed that the concept of fuzzy regularity is formalized by a pattern given as a finite automaton with fuzzy properties of alphabet characters on transitions. Proceeding from this, we replace the problem of determining the degree of fuzzy regularity of a crisp string with the problem of determining the degree of belonging of such a string to the language of the corresponding automaton and propose an effective method for solving it using the dynamic programming approach.

The solution to the considered problem makes it possible to fuzzify the set of strings in a given alphabet based on a pattern defining fuzzy regularity. This work is a continuation of the author's previous works related to finding occurrences of a fuzzy pattern in the text. It may have applications in the field of pattern recognition, data clustering, bio-informatics, etc.
Keywords: Fuzzy string matching, Pattern recognition, Fuzzy automaton.
Article info: Received 4 October 2021; accepted 6 December 2021.
Acknowledgement: This work was supported by the Ministry of Education, Science, Culture and Sports of the Republic of Armenia, project 21T-1B326.

1. Introduction

This paper refers to the definition of a degree of fuzzy regularity of a crisp string in a given alphabet. To formalize the concept of fuzzy regularity, we use the highest degree of matching of such a string to the elements of a given set of periodical sequences of fuzzy properties of the alphabetic characters. We treat this set of sequences as a *matching pattern*.

As we noted, the sequences of fuzzy properties included in the pattern must have a certain periodicity. To achieve this goal, we propose to define the matching pattern as a finite automaton with fuzzy properties of alphabetic characters on transitions. As the pumping lemma states, any sufficiently large sequence of fuzzy properties accepted by such an automaton will be periodic with a period size not exceeding the number of states of the automaton. Thus, in the concept we propose, the problem of determining the fuzzy regularity of a crisp string is defined as a problem of determining the degree of belonging such a string to the language of a fuzzy automaton.

It should be noted that the concept of a fuzzy automaton, which we use as a pattern for determining the regularity, is somewhat different from the one generally accepted in the literature. For example, in the concept of a fuzzy automaton used in [1] and [2], the automaton states and the alphabetic characters are assumed to be crisp, while the start and final states as well as the transitions of the automaton are assumed to be fuzzy. A review of the results on fuzzy automata in their various interpretations, as well as the fuzzy languages they accept (including automaton analysis and synthesis, minimization, closure properties, etc.) is given in [3].

The problem of determining the regularity of a string is widely used in *pattern recognition*, where it is often necessary to detect regularities in strings encoding images [4]. Such regularity is not always specified precisely, so we have to deal with a fuzzy regularity. The concept of fuzzy regularity can also be used in the area of fuzzy *data clustering* [5], which deals with grouping elements into fuzzy clusters, a particular case of which is fuzzy *sequence labeling* [6].

The investigation on fuzzy regularity of a string that we propose in this paper continues our previous research in the field of fuzzy string matching [7], [8], [9]. The problem of splitting a string into adjacent segments to best match the pattern, which is a sequence of fuzzy properties of substrings, was considered in [10], [11]. In this paper, we concretize the concept of a fuzzy property of a substring, defining it as the degree of belonging the substring to the language of a fuzzy automaton.

The paper is organized as follows.

Section 2 presents the concepts of a fuzzy symbol and a finite automaton over a set of fuzzy symbols, called a *fuzzy automaton.* Section 3 considers the problem of matching a crisp string with a pattern given by a fuzzy automaton and provides an efficient algorithm for determining the degree of matching based on the dynamic programming approach. Finally, the conclusion summarizes the obtained results.

2. Preliminaries

2.1 Fuzzy Symbols

Suppose that $(L, \leq, \otimes, 0, 1)$ is a finite linearly ordered set of measures with the smallest element 0, the largest element 1, and the monotonic accumulation operation $\otimes$ such that L is a commutative monoid with unit element 1 and zero element 0. That is, for all $a, b, c \in L$

$$a \otimes 0 = 0, a \otimes 1 = a, a \leq b \Rightarrow a \otimes c \leq b \otimes c.$$

In our further considerations, we will assume that the maximum over the empty set of L - values is equal to 0.

According to [12], the fuzzy subset X of the universal set U is defined by the membership function $\mu_X : U \to L$ that associates with each element $u \in U$ the value $\mu_X(u) \in L$, representing the degree of belonging u to X. A fuzzy subset X of U can be represented by the additive form

$$X = \sum_{u \in U} u/\mu_X(u).$$

We say that an element $u \in U$ certainly belongs to X if $\mu_X(u) = 1$, and it certainly does not belong to X if $\mu_X(u) = 0$. Conversely, if $0 < \mu_X(u) < 1$, then we say that u belongs to X with degree $\mu_X(u)$.

Given an alphabet Σ of characters, we define a *fuzzy symbol* α over Σ as a fuzzy subset of Σ. Given a character $x \in \Sigma$ and a fuzzy symbol α over Σ, we say that x matches α with degree $\mu_\alpha(x)$. This definition can be extended in the usual way to equal length sequences of characters and fuzzy symbols, respectively. That is, for a set of fuzzy symbols Ξ, $x = x_1...x_n \in \Sigma^*$ and $\omega = \omega_1...\omega_n \in \Xi^*$, we define the matching degree of x to ω as the L-value

$$\mu_\omega(x) = \begin{cases} \mu_{\omega_1}(x_1) \otimes ... \otimes \mu_{\omega_n}(x_n), & \text{if } x \neq \epsilon, \\ 1, & \text{if } x = \epsilon. \end{cases}$$

Example 1 . Suppose $\Sigma = \{1, 2, 3, 4, 5, 6, 7\}$ and the measures are rational numbers that belong to the segment $[0, 1]$ with the accumulation operation defined as multiplication. Define the fuzzy symbols S (*small*), M (*middle*) and L(*large*) to be the following fuzzy subsets of Σ:

- $S = 1/1 + 2/0.9 + 3/0.6 + 4/0.3 + 5/0.1 + 6/0 + 7/0$,
- $M = 1/0 + 2/0.25 + 3/0.75 + 4/1 + 5/0.75 + 6/0.25 + 7/0$,
- $L = 1/0 + 2/0 + 3/0.1 + 4/0.3 + 5/0.6 + 6/0.9 + 7/1$.

Let $x = 35634$, $\omega_1 = SMLSM$, $\omega_2 = MLMSL$. Then

- $\mu_{\omega_1}(x) = \mu_S(3) \otimes \mu_M(5) \otimes \mu_L(6) \otimes \mu_S(3) \otimes \mu_M(4) = \frac{3}{5} \cdot \frac{3}{4} \cdot \frac{9}{10} \cdot \frac{3}{5} \cdot 1 = \frac{243}{1000}$,
- $\mu_{\omega_2}(x) = \mu_M(3) \otimes \mu_L(5) \otimes \mu_M(6) \otimes \mu_S(3) \otimes \mu_L(4) = \frac{3}{4} \cdot \frac{3}{5} \cdot \frac{1}{4} \cdot \frac{3}{5} \cdot \frac{3}{10} = \frac{81}{4000}$.

2.2 Fuzzy Automaton

Given a finite set Ξ of fuzzy symbols over the alphabet Σ, we define a *fuzzy automaton* as a deterministic finite automaton using symbols from Ξ on transitions. That is, a fuzzy automaton is a 5-tuple $A = (Q, \Xi, \delta, q_{in}, F)$, where

- Q is the finite non-empty set of states,
- $\delta \subseteq Q \times \Xi \to Q$ is the transition function,
- $q_{in} \in Q$ is the initial state,
- $F \subseteq Q$ is the set of final states.

Let

$$h = p_0 \xrightarrow{\alpha_1} p_1 \xrightarrow{\alpha_2} ... \xrightarrow{\alpha_n} p_n, \; n \geq 0,$$

be a path leading from the state $p_0 \in Q$ to the state $p_n \in Q$ in the diagram of the fuzzy automaton A. We say that the path h generates a sequence $\alpha = \alpha_1...\alpha_n \in \Xi^*$ of fuzzy symbols. We define the language $L(A) \subseteq \Xi^*$ of the automaton A as the set of all sequences of fuzzy symbols generated by all paths leading from the initial state to a final state. For $k \geq 0$ we denote by $L(A)/k$ the set of all k-length strings in $L(A)$.

Given a string $x \in \Sigma^*$ and a fuzzy automaton A, we say that x matches A with degree $\mu_A(x)$, if

$$\mu_A(x) = max\{\mu_\omega(x)| \text{ for all } \omega \in L(A)/|x|\}$$

Example 2 . Let $x = 56364$, $|x| = 5$, $\Xi = \{S, M, L\}$, A is the fuzzy automaton in Fig. 1.

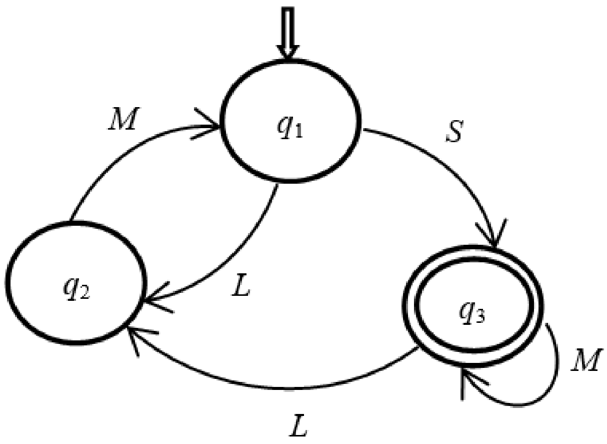

Fig. 1. A fuzzy automaton

Listed below are all the strings in $L(A)/5$:

$\omega_1 = SMMMM,\ \omega_2 = SMLMS,\ \omega_3 = SLMSM,\ \omega_4 = LMSMM,\ \omega_5 = LMLMS.$

Note that

$$\mu_A(x) = max\{\mu_{\omega_1}(x), \mu_{\omega_2}(x)\mu_{\omega_3}(x), \mu_{\omega_4}(x), \mu_{\omega_5}(x)\} = max\Big\{\mu_{SMMMM}(56364), \mu_{SMLMS}(56364), \mu_{SLMSM}(56364), \mu_{LMSMM}(56364), \mu_{LMLMS}(56364)\Big\} = max\left\{\left(\frac{1}{10}\cdot\frac{1}{4}\cdot\frac{3}{4}\cdot\frac{1}{4}\cdot 1\right), \left(\frac{1}{10}\cdot\frac{1}{4}\cdot\frac{1}{10}\cdot\frac{1}{4}\cdot\frac{3}{10}\right), \left(\frac{1}{10}\cdot\frac{9}{10}\cdot\frac{3}{4}\cdot 0\cdot 1\right), \left(\frac{3}{5}\cdot\frac{1}{4}\cdot\frac{3}{5}\cdot\frac{1}{4}\cdot 1\right), \left(\frac{3}{5}\cdot\frac{1}{4}\cdot\frac{1}{10}\cdot\frac{1}{4}\cdot\frac{3}{10}\right)\right\} = $$

$$= max\left\{\frac{3}{640}, \frac{3}{16000}, 0, \frac{9}{400}, \frac{9}{8000}\right\} = \frac{9}{400}.$$

3. Calculation of Fuzzy Regularity

3.1 The Fuzzy Regularity Determination Problem

Let Σ be a finite alphabet of characters, Ξ be a finite set of fuzzy symbols over Σ, P be a fuzzy automaton over Ξ called a *fuzzy regularity pattern* (or, in short, a *regularity pattern*).

For a given $x \in \Sigma^*$, we define the *(x, P) - matching problem* as the problem of determining the value $\mu_P(x)$. We assume that this value represents the degree of fuzzy regularity of the crisp string x according to the regularity pattern P.

3.2 Recursive Solution

Let $P = (Q, \Xi, \delta, q_{in}, F)$ be a regularity pattern, $q \in Q$.

Let us denote $P_q = (Q, \Xi, \delta, q, F)$ the regularity pattern obtained from P by replacing the initial state q_{in} with the state q. The value $\mu_{P_q}(x)$, representing the solution to the (x, P_q) - matching problem for the string x and the regularity pattern P_q, let us denote $\mu_P(q, x)$. In particular, we have that $\mu_P(q_{in}, x) = \mu_P(x)$.

Theorem 1: (Optimal substructure of the (x, P_q) - matching problem)

1. $x = \epsilon \Rightarrow$ [If $q \in F$ then $\mu_P(q,x) = 1$ else $\mu_P(q,x) = 0$].
2. $x = ax' \Rightarrow [\mu_P(q,x) = max\{\mu_\alpha(a) \otimes \mu_P(q',x')|$ for all $\delta(q,\alpha) = q'\}]$.

Proof: The first statement is obvious.

The second statement follows from

$$[x = ax'] \Rightarrow$$

$$[\mu_P(q,x) = max\{\mu_\alpha(a) \otimes \mu_\omega(x')|\ \delta(q,\alpha) = q', \alpha \in \Xi, \omega \in L(P_{q'})] \Rightarrow$$

$$[\mu_P(q,x) = max\{\mu_\alpha(a) \otimes \mu_P(q',x')| \text{ for all } \delta(q,\alpha) = q'\}].$$

Theorem 1 implies the following recurrent equation for calculation $\mu_P(q,x)$:

$$\mu_P(q,x) = \begin{cases} (q \in F)?1:0, & \text{if } \quad x = \epsilon, \\ max\{\mu_\alpha(a) \otimes \mu_P(q',x')| \text{ for all } \delta(q,\alpha) = q'\}, & \text{if } x = ax'. \end{cases}$$

□

Direct calculation of the value of $\mu_P(q,x)$ using this formula will be inefficient due to overlapping subproblems. In order to get a more efficient solution, let us apply the dynamic programming approach.

3.3 Dynamic Programming Solution

Suppose Q is represented as an m-tuple $< q_1, ..., q_m >$, so that $q_{in} = q_1$. For $1 \le i \le m, 1 \le j \le n+1$, let us denote $s[i,j] = \mu_P(q_i, x[j..n])$ (we assume that $x[n+1..n] = \epsilon$).

The optimal substructure of the (x, P_q) - matching problem dictates the following recurrent equation for calculating $s[i,j]$:

$$s[i,j] = \begin{cases} q_i \in F)?1:0, & \text{if } j = n+1, \\ max\{\mu_\alpha(x[j]) \otimes s[k, j+1]| \text{ for all } \delta(q_i,\alpha) = q_k\}, & \text{if } 1 \le j \le n. \end{cases} \quad (1)$$

Note that the (x, P) - matching problem can be represented as the problem of determining the value $s[1,1] = \mu_P(x)$.

For $q \in Q$, let us denote $out(q) = \{(\alpha, p)|\delta(q,\alpha) = p\ \}$, which is the set of pairs consisting of states and fuzzy symbols corresponding to transitions outgoing from q.

The algorithm in Fig. 2 presents the process of calculating the matrix $\{s[i,j]|, 1 \le i \le m, 1 \le j \le n+1\}$ of matching degrees according to formula (1):

Algorithm 1: *Calculate − Matching − Degrees*

Input:
An n-length string x and an m-state regularity pattern P
Output:
The L-value matrix $s[1..m, 1..n+1]$ of matching degrees

```
for i = 1 to m do
    s[i, n+1] = ((q_i ∈ F)?1 : 0)
end
for j = n downto 1 do
    for i = 1 to m do
        max = 0
        for all (α, q_k) ∈ out(q_i)
        if μ_α(x[j]) ⊗ s[k, j+1] > max then
            max = μ_α(x[j]) ⊗ s[k, j+1]
        end
        s[i, j] = max
    end
end
return s
```

Fig. 2. Building the matrix of matching degrees

The solution to the (x, P) - matching problem, presented in Fig. 3, is simply reduced to extracting the value $s[1, 1]$ from the matrix s.

Algorithm 2: *Determine − Fuzzy − Regularity*

Input:
An n-length string x
An m-state regularity pattern P, where $Q =< q_1, ..., q_m >$ and $q_{in} = q_1$
Output:
The degree of matching x to P

```
s = Calculate − Matching − Degrees(x, P)
return s[1, 1]
```

Fig. 3. Determination of fuzzy regularity

Example 3 . The matrix s of matching degrees, constructed by the $Calculate{-}Matching{-}Degrees$ algorithm on the input $x = 56364$ and the regularity pattern in Fig. 1, is presented in Fig. 4.

$$
\begin{array}{c|cccccc|}
 & \mathbf{5} & \mathbf{6} & \mathbf{3} & \mathbf{6} & \mathbf{4} & \epsilon \\
\hline
q_1 & \mu_L(5)\cdot\frac{3}{80}=\frac{9}{400} & \mu_S(6)\cdot\frac{3}{16}=0 & \mu_S(3)\cdot\frac{1}{4}=\frac{3}{20} & \mu_S(6)\cdot 1=0 & \mu_S(4)\cdot 1=\frac{3}{10} & 0 \\
q_2 & \textit{doesn't matter} & \mu_M(6)\cdot\frac{3}{20}=\frac{3}{80} & 0 & \mu_M(6)\cdot\frac{3}{10}=\frac{3}{40} & 0 & 0 \\
q_3 & \textit{doesn't matter} & \mu_M(6)\cdot\frac{3}{16}=\frac{3}{64} & \mu_M(3)\cdot\frac{1}{4}=\frac{3}{16} & \mu_M(6)\cdot 1=\frac{1}{4} & \mu_M(4)\cdot 1=1 & 1
\end{array}
$$

Fig. 4. Memoization results

In line with $Determine - Fuzzy - Regularity$ algorithm, the value $s[1, 1] = \frac{9}{400}$ is the solution to the (x, P) - matching problem for the string $x = 56364$ and regularity pattern in Fig. 1, which is consistent with the result obtained in Example 2. According to our approach, this value determines the degree of fuzzy regularity of the crisp string x based on the pattern P.

3.4 Analysis

To estimate the complexity of the proposed solution to the (x, P)-matching problem, let us assume that there are k transitions in the graph of the m-state fuzzy automaton P, and that this graph is represented as an array $A[1..m]$ such that $A[i] = out(q_i), 1 \leq i \leq m$.

In this case, the construction of the previous column of the matrix s in lines 5-12 of the $Calculate - Matching - Degrees$ algorithm takes time $O(1 + k)$ and, consequently, the construction of the entire matrix s for a string of length n takes time $O(n(1 + k))$. As a result, the instruction in line 1 of the $Determine - Fuzzy - Regularity$ algorithm runs in $O(n(1 + k))$ time. The instruction in line 2 of this algorithm obviously runs in O(1) time, which makes the time complexity of the $Determine - Fuzzy - Regularity$ algorithm equal to $O(n(1 + k))$.

For an n-length string x and an m-state regularity pattern P, the algorithm uses $O(mn)$ extra memory to represent the matrix $s[1..m, 1..n + 1]$ of matching degrees.

4. Conclusion

The problem of determining the fuzzy regularity of a crisp string has been considered in this paper, where the concept of fuzzy regularity is formalized by means of a finite automaton with fuzzy properties of alphabet characters on transitions. Using the dynamic programming approach, we propose a solution to this problem with

$O(n(k + 1))$ time complexity, and

$O(mn)$ space complexity,

where n is the length of the input string; m and k are the number of states and the number of transitions of the automaton, respectively.

The proposed algorithm can be used in the field of pattern recognition, data clustering, DNA analysis, etc.

References

[1] Y. Cao and Y. Ezawa, "Non-determinstic fuzzy automata", Information Sciences, vol. 191, pp. 86-97, 2012.

[2] J. N. Mordeson and D.S. Malik, *Fuzzy Automata and Languages: Theory and Applications*, Chapman & Hall, CRC, Boca Raton, London 2002.

[3] R. K. Singh, A. Rani and M.K. Sachan, "Fuzzy automata: A quantitative review", *International Journal on Future Revolution in Computer Science & Communication Engineering*, vol 3, no. 7, pp. 11-17, 2017.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[5] Wang Zhen Zhou, "Image segmentation by combining the global and local properties", *Expert Systems with Applications*, vol. 87, pp. 30-40, 2017.

[6] Bezdek, James C, *Pattern Recognition with Fuzzy Objective Function Algorithms*, 1981.

[7] A. Kostanyan, Fuzzy string matching with finite automata, *Proc. on 2017 IEEE Conference CSIT-2017*, Yerevan, Armenia, pp. 25-29. IEEE Press, USA 2018.

[8] A. Kostanyan, Fuzzy string matching using prefix table, *Transactions of IIAP NAS RA, Mathematical Problems of Computer Science*, vol. 54, 2020, pp.116 -121.

[9] A. Kostanyan and A. Karapetyan, *String Matching in Case of Periodicity in the Pattern*, In: Dolinina O., Brovko A., Pechenkin V., Lvov A., Zhmud V., Kreinovich V. (eds) Recent Research in Control Engineering and Decision Making. ICIT 2019. Studies in Systems, Decision and Control, vol 199. Springer, 2019.

[10] A. Kostanyan and A. Harmandayan, Segmentation of string to match a fuzzy pattern, *In Proc. of Computer Science & Information Technologies (CSIT) Conference*, Yerevan, Armenia, September, pp. 17-19, 2019.

[11] A. Kostanyan and A. Harmandayan, Mapping a fuzzy pattern onto a string, *in Proc. on 2019 IEEE Conference "2019 Computer Science and Information Technologies (CSIT)*, Yerevan, Armenia, 23-27 Sep., 2019, IEEE Press, USA, pp. 5-8, 2019.

[12] L. A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning-I, *Information Sciences*, vol. 8, pp. 199-249, 1975.

Տողի ոչ հստակ կանոնավորության աստիճանի որոշում

Արմեն Հ. Կոստանյան

ՏՏ կրթական և հետազոտական կենտրոն
Երևանի պետական համալսարան
e-mail: armhko@gmail.com

Ամփոփում

Հոդվածում դիտարկվում է տողի ոչ հստակ կանոնավորության չափի որոշման խնդիրը: Ենթադրվում է, որ ոչ հստակ կանոնավորության հասկացությունը ֆորմալիզացվում է վերջավոր ավտոմատի (շաբլոնի) միջոցով, որի անցումներին վերագրված են այբուբենի նշանների ոչ հստակ հատկություններ: Արդյունքում, հստակ տողի ոչ հստակ կանոնավորության չափի որոշման խնդիրը փոխարինվում է համապատասխան ավտոմատի լեզվին նշված տողի պատկանելիության չափի որոշման խնդրով, որի լուծման համար առաջարկվում է օգտագործել դինամիկ ծրագրավորման մեթոդը:

Նշված խնդրի լուծումը հնարավորություն է տալիս կառուցել տրված այբուբենի բառերի ոչ հստակ բազմություն՝ ոչ հստակ կանոնավորության որոշման շաբլոնի հիման վրա: Տվյալ աշխատանքը շարունակությունն է հեղինակի նախորդ աշխատանքների՝ տողում ոչ հստակ ենթատողի (շաբլոնի) որոնման ուղղությամբ: Այն կարող է կիրառվել շաբլոնների ճանաչման, տվյալների խմբավորման, բիոինֆորմատիկայի և այլ բնագավառներում:

Բանալի բառեր՝ նմուշի ոչ հստակ համադրում, շաբլոնների ճանաչում, ոչ հստակ ավտոմատ:

Определение степени нечеткой регулярности строки

Армен Г. Костанян

Образовательный и научный центр информационных технологий
Ереванский государственный университет
e-mail: armhko@gmail.com

Аннотация

В статье рассматривается задача определения степени нечеткой регулярности данной строки. Предполагается, что нечеткая регулярность формализуется посредством патерна, представленного в виде конечного автомата с нечеткими свойствами символов алфавита на переходах. В результате, задача определения степени нечеткой регулярности строки заменяется задачей определения степени ее принадлежности языку нечеткого автомата, для решения которой предлагается использовать метод динамического программирования.

Решение рассматриваемой задачи позволяет фаззифицировать множество слов в данном алфавите на основе паттерна определения нечеткой регулярности. Данная работа является продолжением ряда предыдущих работ автора по поиску нечеткого паттерна в строке. Она может иметь применения в таких областях, как распознавание паттернов, кластеризация данных, боинформатика, и т. д.

Ключевые слова: нечеткое сопоставление с образцом, распознавание паттернов, нечеткий автомат.