KYBERNETIKA — VOLUME 36 (2000), NUMBER 6, PAGES 671 - 687

THE SIMILARITY OF TWO STRINGS OF FUZZY SETS

GABRIELA ANDREJKOVÁ

Let $\mathcal{A}$, $\mathcal{B}$ be the strings of fuzzy sets over χ, where χ is a finite universe of discourse. We present the algorithms for operations on fuzzy sets and the polynomial time algorithms to find the string $\mathcal{C}$ over χ which is a closest common subsequence of fuzzy sets of $\mathcal{A}$ and $\mathcal{B}$ using different operations to measure a similarity of fuzzy sets.

1. INTRODUCTION

The problems of automatic (or partially automatic) corrections of texts are still topical and very important. The mistakes in the text can have special properties and those properties can be used in the construction of correcting algorithms. For example, following mistakes can be made in typing some string on the keyboard: (1) Typing a different character, usually from the neighbour area of the given character, (2) inserting a single character into the source string, (3) omitting (skipping) any single source character, (4) transposition of neighbour elements.

In the most frequent mistakes, a character from the area on the keyboard adjacent to the required character was typed instead of the required character. For example, the neighborhood of the character f is the set $\mathbf{f} = \{f, d, g, r, t, c, v\}$. The sequence of sets $\mathbf{A} = \mathbf{f}, \mathbf{r}, \mathbf{e}, \mathbf{s}, \mathbf{c}, \mathbf{o}$ belongs to the word *fresco*. In this case (typing mistakes) let us assign *membership value (m.v.)* to each element of the neighborhood in such way that the character itself has m.v. 1 and the m.v.'s of "more erroneous" character are smaller than those of the "better one". For example, for set $\mathbf{f}$ we have $\mu(f) = 1$, $\mu(d) = 0.4$, $\mu(g) = 0.4$, $\mu(r) = 0.2$, $\mu(t) = 0.4$, $\mu(c) = 0.3$, $\mu(v) = 0.3$. We consider that in the text, it is necessary to find the words which are very close to the word *fresco*. For example, it is possible to consider the sum of m.v.'s of a given string as a measure of its similarity of the string to the given word *fresco*. The measure of the similarity of the found words can be different to the length of the given word *fresco*. For example, if the word *fresco* is found in the text then the measure of the similarity to the given word *fresco* is the length of the word *fresco* (6), if the word *tresc* is found then the measure of the similarity is 4.4 because the symbol t is very close to the symbol f and symbol o is omitted. From the theoretical point of view, we have one string of symbols with m.v.'s and one string of sets of symbols with m.v.'s and a measure of a closest common subsequence is founded.

If we consider a very high uncertainty of the words then we can analyze the strings

of sets of symbols with membership values, for example fuzzy sets. It is possible to consider the described problem as the closest common subsequence problem of the two similar strings and its repetition for text of strings.

The simpler problem – the common subsequence problem of two strings of symbols with the same m.v.'s 1, is to determine one of the subsequences that can be obtained by deleting zero or more symbols from each of the given strings. Usually, it is measured the length of the common subsequence, but we can consider some different measures for the common subsequence. The longest common subsequence problem (*LCS Problem*) of two strings is to determine the common subsequence with the maximal length. For example, the strings *AGI* is a common subsequence and the string *ALGI* is the longest common subsequence of the strings *ALGORITHM* and *ALLEGATION*.

D. S. Hirschberg and L. L. Larmore [6] have discussed a generalization of LCS Problem, which is called Set LCS Problem (*SLCS Problem*) of two strings where however the strings are not of the same type. The first string is a sequence of symbols and the second string is a sequence of subsets of elements with the same m.v's 1 over an alphabet Ω. The elements of each subset can be used as an arbitrary permutation of elements in the subset. The longest common subsequence in this case is a sequence of symbols with maximal length. D. S. Hirschberg and L. L. Larmore have presented $O(m \cdot n)$-time and $O(m + n)$-space algorithm, m, n are lengths of strings. The Set-Set LCS Problem (*SSLCS Problem*) is discussed by D. S. Hirschberg and L. L. Larmore [7]. In this case both strings are the strings of subsets of elements with the same m.v.'s 1 over an alphabet Ω. In the paper [7] is presented the $O(m \cdot n)$-time algorithm for the general SSLCS Problem.

In this paper we present algorithms for more general case of the Common Subsequence Problem, it means Closest Common Subsequence Problem of two strings of fuzzy sets – *SSCCS Problem*.

2. BASIC DEFINITION IN FUZZY LOGIC CONNECTIVES THEORY

A *fuzzy set* A over some universe of discourse χ (which itself is classical set) is characterized by its *membership function* $\mu_A : \chi \to [0,1] \subset \Re, \Re$ the set of real numbers and often called *fuzzy subset* of χ. Fuzzy sets are generalized characteristic functions. The membership values of elements in fuzzy sets can be considered as generalized truth values, i. e. as *truth degrees* of suitable many-valued logic **L**. We use definitions of connectives and quantifiers of the langauge **L** via truth functions and truth functional conditions according to [3, 4, 9].

A binary operation $\mathbf{t}$ in the real interval $[0,1]$ is a *t-norm* if and only if it is (i) associative and commutative, (ii) non-decreasing in each argument, (iii) has 1 as neutral element, i. e. $\mathbf{t}(x,1) = x$ for all $x \in [0,1]$. A binary operation $\mathbf{s}$ in the real interval $[0,1]$ is a *t-conorm* if and only if it is (i) associative and commutative, (ii) non-decreasing in each argument, (iii) has 0 as neutral element, i. e. $\mathbf{s}(x,0) = x$ for all $x \in [0,1]$. The t-norms are suitable candidates for conjunctions in many-valued logic and the t-conorms candidates for disjunctions. To given any t-norm $\mathbf{t}$, by $\mathbf{s_t}(x,y) =_{\text{def}} 1 - \mathbf{t}(1-x, 1-y)$ a t-conorm can be defined.

A binary operation φ in real unit interval $[0,1]$ is called Φ-operator (connected to a given t-norm $\mathbf{t}$) iff for all $x,y,z \in [0,1]$ the following hold true (Φ_1) $x \leq z \Rightarrow \varphi(y,x) \leq \varphi(y,z)$, (Φ_2) $\mathbf{t}(x,\varphi(x,y)) \leq y$, (Φ_3) $x \leq \varphi(y,\mathbf{t}(y,x))$.

If φ is Φ-operator which is connected to the t-norm $\mathbf{t}$, then for all $x,y \in [0,1]$ it holds that $\varphi(x,y) = \sup\{z|\mathbf{t}(x,z) \leq y\}$.

For each t-norm $\mathbf{t}$ we denote by $\wedge_{\mathbf{t}}$ the connective of many-valued logic that has this t-norm $\mathbf{t}$ as its truth function, similarly for each t-norm $\mathbf{t}$ a disjunction connective $\vee_{\mathbf{t}}$ with truth function $\mathbf{s_t}$. Let $LCS(\mathbf{t})$ be the denotation for $\mathbf{t}$ that is lower semicontinuous, it means that for each $x_0, y_0 \in [0,1]$ and each $\epsilon > 0$ there is a $\delta > 0$ such that $\mathbf{t}(x,y_0) > \mathbf{t}(x_0,y_0) - \epsilon$ for all $x \in (x_0 - \delta, x_0]$.

Using the Φ-operators we can introduce a special kind of a negation function with respect to each t-norm $\mathbf{t}$ with the property LSC($\mathbf{t}$) in the following way: $\mathbf{n_t}(u) =_{\text{def}} \varphi_{\mathbf{t}}(u,0)$ for all $u \in [0,1]$.

The basic binary predicate symbol "ε" will denote the membership relation of elements of the universe of discourse χ with respect to the fuzzy subset of χ. Let be $[\![x \,\varepsilon\, A]\!] =_{\text{def}} \mu_A(x)$.

For any t-norm $\mathbf{t}$ which has the property LSC($\mathbf{t}$) let $-_{\mathbf{t}}$ be that (unary) negation connective which has $\mathbf{n_t}$ as its truth function, that means always put $[\![-_{\mathbf{t}}H]\!] = \mathbf{n_t}([\![H]\!])$. For any t-norm $\mathbf{t}$ which fulfills LSC($\mathbf{t}$) by $\rightarrow_{\mathbf{t}}$ we denote that implication connective that has the Φ-operator $\varphi_{\mathbf{t}}$ as its truth function. Recall some classical logic connectives:

1. **The Gödel intuitionistic connectives**

   $\wedge_G(x,y) = \min(x,y)$, $\quad \vee_G(x,y) = \max(x,y)$, $\quad \rightarrow_G(x,y) = y$ if $x > y$ else 1.

2. **The Łukasiewicz connectives**

   $\wedge_L(x,y) = \max(0, x+y-1)$, $\quad \vee_L(x,y) = \min(1, x+y)$,
   $\rightarrow_L(x,y) = \min(1, 1-x+y)$.

3. **The product logic connectives**

   $\wedge_P(x,y) = x.y$, $\quad \vee_P(x,y) = x+y-xy$, $\quad \rightarrow_P(x,y) = \min(1, \frac{y}{x})$.

Whole above mentioned operators are called fuzzy logic connectives. Fuzzy logic connectives play important role in comparisons of symbols of fuzzy set strings since the different results are obtained by different kinds of fuzzy logic connectives. It is difficult to determine the optimum operator since there are so many kinds of fuzzy logic connectives.

For the quantifiers, we have considerations independent on the t-norms:

$$[\![\forall x H(x)]\!] = \inf_{a \in \chi} [\![H(x/a)]\!], \qquad [\![\exists x H(x)]\!] = \sup_{a \in \chi} [\![H(x/a)]\!], \tag{1}$$

where $H(x/a)$ is the obvious "substitution notation" and it means that the free variable x of H has to be given the value a from the universe discourse χ.

3. OPERATIONS ON FUZZY SETS

For each universe of discourse χ and each formula $H(x)$ of set theoretic language of many-valued logic we denote by $\{x \in \chi \| H(x)\}$ or simply by $\{x \| H(x)\}$ that fuzzy set A on χ whose membership function μ_A is characterized by: $\mu_A(a) = [\![H(a)]\!]$ for each $a \in \chi$; i.e. $\{x \in \chi \| H(x)\}$ has the characteristic property $[\![a \varepsilon \{x \in \chi \| H(x)\}]\!] = [\![H(x/a)]\!]$ for all $a \in \chi$.

For any fuzzy sets A, B and any t-norm $\mathbf{t}$, t-conorm $\mathbf{s_t}$ and negation function $\mathbf{n_t}$ we define

1. *intersection* "$\cap_{\mathbf{t}}$": $A \cap_{\mathbf{t}} B =_{\text{def}} \{x \| \wedge_{\mathbf{t}} (x \varepsilon A, x \varepsilon B)\}$,
2. *union* "$\cup_{\mathbf{t}}$": $A \cup_{\mathbf{t}} B =_{\text{def}} \{x \| \vee_{\mathbf{t}} (x \varepsilon A, x \varepsilon B)\}$,
3. *complement* "$\subset^c_{\mathbf{n}}$": $\subset^c_{\mathbf{n}} A =_{\text{def}} \{x \| -_{\mathbf{n}} (x \varepsilon A)\}$.

3.1. Comparison of two fuzzy sets

Let C, D be fuzzy sets defined over the same universe of discourse χ, i. e. $C, D : \chi \to [0, 1]$. We discuss the following approach to measure how distinct or how similar are the fuzzy sets C, D.

The approach for the comparison of fuzzy sets uses set theoretically oriented tools, especially the fuzzified identity $\equiv_{\mathbf{t}}$.

For any fuzzy sets C, D and any t-norm $\mathbf{t}$ with property $LCS(\mathbf{t})$ let

$$C \subseteq_{\mathbf{t}} D =_{\text{def}} \forall x \to_{\mathbf{t}} (x \varepsilon C, x \varepsilon D),$$

$$C \equiv_{\mathbf{t}} D =_{\text{def}} \wedge_{\mathbf{t}} (C \subseteq_{\mathbf{t}} D, D \subseteq_{\mathbf{t}} C).$$

The truth degree $[\![C \subseteq_{\mathbf{t}} D]\!]$ is a degree of containment of C in D and the truth degree $[\![C \equiv_{\mathbf{t}} D]\!]$ is a degree of equality for the fuzzy sets C, D.

For a readable formulation we use the following denotation for $C, D \in \mathcal{P}(\chi)$:

- $\text{supp}(C) = \{x \in \chi | \mu_C(x) > 0\} = \{x \in \chi | [\![x \varepsilon A]\!] \neq 0\}$
- $\{C > D\} =_{\text{def}} \{x \in \chi | [\![x \varepsilon C]\!] > [\![x \varepsilon D]\!]\}$,
- $\{C \neq D\} =_{\text{def}} \{x \in \chi | [\![x \varepsilon C]\!] \neq [\![x \varepsilon D]\!]\}$,
- $\Delta(C, D) =_{\text{def}} \sup_{x \in \{C > D\}} \{[\![x \varepsilon C]\!] - [\![x \varepsilon D]\!]\}$.

Straightforward calculations for above defined logic connectives give the following results:

1. **The Gödel intuitionistic connectives, $\mathbf{t} = \mathbf{t}_G$**

$$[\![C \subseteq_{\mathbf{t}_G} D]\!] = \inf_{x \in \{C > D\}} [\![x \varepsilon D]\!]$$

$$[\![C \equiv_{\mathbf{t}_G} D]\!] = \inf_{x \in \{C \neq D\}} [\![x \varepsilon C \cap_{\mathbf{t}_G} D]\!].$$

2. **The Łukasiewicz connectives, $\mathbf{t} = \mathbf{t}_L$**

$$[\![C \subseteq_{\mathbf{t}_L} D]\!] = 1 - \Delta(C, D),$$
$$[\![C \equiv_{\mathbf{t}_L} D]\!] = \max\{0, 1 - (\Delta(C, D) + \Delta(D, C))\}.$$

3. **The product logic connectives, $\mathbf{t} = \mathbf{t}_P$**

$$[\![C \subseteq_{\mathbf{t}_P} D]\!] = \text{ if } \{C \cap_{\mathbf{t}} D\} = \emptyset \text{ then 1 else } \min_{x \in \{C > D\}} \left\{ \frac{[\![x \varepsilon D]\!]}{[\![x \varepsilon C]\!]} \right\},$$
$$[\![C \equiv_{\mathbf{t}_P} D]\!] = [\![C \subseteq_{\mathbf{t}_P} D]\!] \wedge_P [\![D \subseteq_{\mathbf{t}_P} C]\!].$$

Modelling the involved connectives for implication and conjunction in our fuzzy set theoretic setting via any residuation operator, i.e. any Φ-operator φ for implication and any (left) continuous t-norm $\mathbf{t}$ for conjunction, in addition to equality degree $[\![C \equiv_{\mathbf{t}} D]\!]$ also some *local* degree to which two fuzzy sets C and D are equal each other *at a point* $a \in \chi$ is given as a number $\|C = D\|(a)$ defined e.g. in analogy with definition of $\equiv_{\mathbf{t}}$, but deleting the universal quantifier $\forall$ there, i.e. the *inf*-operator. That means we take this local degree of equality as the value

$$\|C = D\|(a) =_{\text{def}} \mathbf{t}(\varphi_{\mathbf{t}}(\mu_C(a), \mu_D(a)), \varphi_{\mathbf{t}}(\mu_D(a), \mu_C(a)). \tag{2}$$

The simple fact that always $\mu_C(a) \leq \mu_D(a)$ or $\mu_D(a) \leq \mu_C(a)$ is the case, i.e. that always $\varphi_{\mathbf{t}}(\mu_C(a), \mu_D(a)) = 1$ or $\varphi_{\mathbf{t}}(\mu_D(a), \mu_C(a)) = 1$, allows to simplify (2) to

$$\|C = D\|(a) = \wedge_{\mathbf{t}}(\varphi_{\mathbf{t}}(\mu_C(a), \mu_D(a)), \varphi_{\mathbf{t}}(\mu_D(a), \mu_C(a))). \tag{3}$$

Thus one has, together with the Φ-operator $\varphi_{\mathbf{t}}$, only to take the *min*-operator instead of the t-norm $\mathbf{t}$ to finally find $\|C = D)\|(a)$.

Nevertheless one usually also likes to have a number expressing a (unique) degree to which C and D are equal to each other in a global sense. For this one has to aggregate the partial evaluations 2 and 3 of equality over the whole space χ.

On the base of (2) we have the following results for the above defined logic connectives if $a \varepsilon C \cap_{\mathbf{t}} D$:

1. **The Gödel intuitionistic connectives**

$$\|C = D\|_G(a) = \min\{\mu_C(a), \mu_D(a)\}. \tag{4}$$

2. **The Łukasiewicz connectives**

$$\|C = D\|_L(a) = \begin{cases} 1 - \mu_D(a) + \mu_C(a), & \text{if } \mu_C(a) > \mu_D(a), \\ 1 - \mu_C(a) + \mu_D(a), & \text{if } \mu_D(a) > \mu_C(a), \\ 1 & \text{otherwise.} \end{cases} \tag{5}$$

3. **The product logic connectives**

$$\|C = D\|_P(a) = \min\left\{ \frac{\mu_D(a)}{\mu_C(a)}, \frac{\mu_C(a)}{\mu_D(a)} \right\}. \tag{6}$$

There is, however, no unique way to perform these tasks. Perhaps the most preferred ways in the engineering community are to take either a so-called optimistic or a pessimistic form of aggregation.

In the *optimistic* case one prefers to modelize the degree of the statement "C and D are equal to each other" by the maximal value of $\|C = D\|(x)$ over χ, viz.

$$\|C = D\| =_{\text{def}} \sup_{x \in \chi} \|C = D\|(x). \tag{7}$$

On the opposite, i.e. *pessimistic* pole, one often uses the formula

$$\|C = D\| =_{\text{def}} \inf_{x \in \chi} \|C = D\|(x). \tag{8}$$

This degree of equality (8) has a "pessimistic character" because obviously it indicates the worst case of all the local degrees (2).

The possibility to get an intermediate value of global equality not as likely to cause overestimation or underestimation as (7) or (8) is to take for example the average value of all the local equality degrees, i.e. to consider

$$\|C = D\| =_{\text{def}} \sum_{x \in \chi} \|C = D\|(x)/card(\chi) \tag{9}$$

in the case of a finite universe of discourse χ or of at least a finite support of the fuzzy set $C \cup D$. It means, in the average case the sum of membership values is very important and we will use the sum as the measure of the similarity of two fuzzy sets.

Straightforward computations give the following results:

1. **The Gödel intuitionistic connectives**

   - *optimistic case (pessimistic case)*
     $$\|C = D\|_G =_{\text{def}} \sup_{x \in \chi} (\inf_{x \in \chi}) \{\min\{\mu_C(x), \mu_D(x)\}\},$$
   - *average case*
     $$\|C = D\|_G =_{\text{def}} \sum_{x \in \chi} \{\min\{\mu_C(x), \mu_D(x)\}\}/card(\chi).$$

2. **The Łukasiewicz connectives**

   - *optimistic case (pessimistic case)*
     $$\|C = D\|_L =_{\text{def}} \sup_{x \in \chi} (\inf_{x \in \chi}) \{1 - \mu_D(x) + \mu_C(x), 1 - \mu_C(x) + \mu_D(x)\}.$$

3. **The product logic connectives**

   - *optimistic case (pessimistic case)*
     $$\|C = D\|_P =_{\text{def}} \sup_{x \in \chi} (\inf_{x \in \chi}) \left\{\min\left\{\frac{\mu_D(x)}{\mu_C(x)}, \frac{\mu_C(x)}{\mu_D(x)}\right\}\right\}.$$

3.2. Algorithms for operations on fuzzy sets

The most important operations on sets and fuzzy sets are: (1) intersection, (2) union and (3) comparison. We will study algorithms for (4) difference of sets. The algorithms for computation of new sets by using of above operations on the sets depend on representations and data structures applied to sets. Let χ be the finite set, $|\chi| = s$. We will consider two representations only: (a) ordered elements of the set, (b) a set without an ordering of elements.

The algorithms for above operations on sets:

(a) The elements of the universe of discourse are ordered according some linear order. Then any fuzzy set can be represented by 1-dimensional array of the length $|\chi|$. It is necessary to have m.v.'s of all elements in the fuzzy set (0, if element is not in the set). Let A, B be two fuzzy sets represented by two arrays A and B. In the case both sets have the elements ordered in the same order.

1. **intersection** – time complexity $O(s)$:
   for $i := 1$ to s do $C[i] := \{A[i] \,\mathbf{t}\, B[i]\}$;
2. **union** – time complexity $O(s)$:
   for $i := 1$ to s do $C[i] := \{A[i] \,\mathbf{s_t}\, B[i]\}$;
3. **comparison** – time complexity $O(s)$:
   $c := 0$; for $i := 1$ to s do $c := c + \|A[i] = B[i]\|$; $c := c/s$;
4. **difference A-B** – time complexity $O(s)$:
   for $i := 1$ to s do
   if $(A[i] > 0)$ and $(B[i] > 0)$ then $C[i] := 0$ else
   if $(A[i] > 0)$ and $(B[i] = 0)$ then $C[i] := A[i]$ else $C[i] := 0$;

(b) It is possible to represent any element of the universe of discourse by record (if it is necessary to remember the names of elements): $(e, \mu(e))$. The fuzzy sets can be represented by the arrays of records. The representation is better than the first one if the universe of discourse has many elements and the numbers of elements in fuzzy sets are small. Let A, B be two fuzzy sets represented by two arrays A and B of records, $|A| = a, |B| = b$.

1. **intersection** – time complexity $O(a \cdot b)$:
   for $i := 1$ to a do for $j := 1$ to b do if $A[i] \cdot e = B[j] \cdot e$ then
   begin $C[i] \cdot e := A[i] \cdot e$; $C[i] := A[i] \cdot \mu_A(e) \,\mathbf{t}\, B[j] \cdot \mu_B(e)$ end;
2. **union** – time complexity $O(a \cdot b)$:
   $k := 0$; for $i := 1$ to a do
   for $j := 1$ to b do begin $C[i] \cdot e := A[i] \cdot e$; $C[i] := A[i] \cdot \mu_A(e) \,\mathbf{s_t}\, B[j] \cdot \mu_B(e)$
   end;
3. **comparison** – time complexity $O(a \cdot b)$:
   $c := 0$; for $i := 1$ to a do for $j := 1$ to b do $c := c + \|A[i] = B[j]\|$; $c := c/s$;

4. **difference A-B** – time complexity $O(a \cdot b)$:
   $k := 0$; for $i := 1$ to a do begin $inc(k); C[k] := A[i]$;
   for $j := 1$ to b do if $B[j] \cdot e = A[i] \cdot e$ then $dec(k)$; end;

4. BASIC DEFINITIONS IN STRINGOLOGY

In this section, some basic definitions and results concerning to CCS Problem, SCCS and SSCCS Problem are presented.

Let Ω be a finite alphabet, $|\Omega| = s, 1 \leq s$. In the connection to the fuzzy sets $\Omega = \chi$.

Let $A = a_1 a_2 \dots a_m, a_i \in \Omega, 1 \leq i \leq m$ be a string over an alphabet Ω, where $|A| = m$ is the length of the string A.

The string $C = c_1 \dots c_p$ is a *subsequence* of the string $A = a_1 \dots a_m$, if a monotonous increasing sequence of natural numbers $i_1 < \dots < i_p$ exists such that $c_j = a_{i_j}, 1 \leq j \leq p$. The string C is a *common subsequence* of two strings A, B if C is a subsequence of A and C is a subsequence of B. $|C|$ is the length of the common subsequence. The classical problem to find the longest common subsequence is defined and solved in Hirschberg [5]. In the classical problem, each element in the string is in his position as full member. But sometimes we are not sure about it in texts. The element should be in his position with 70%, it means, the element is in his position with some membership value. It means, we can suppose that in some position should be one element of some set of elements with membership values.

Let $\mu_A(a_i) \in (0,1\rangle, 1 \leq i \leq m$, be some membership values of elements in the string A. The pair (A, μ_A) is *the string A with the membership function* μ_A, *m-string* μA for short. $Val(\mu A)$ is a measure of μA defined by the (10).

$$Val(\mu A) = \Sigma_{i=1}^{m} \mu_A(a_i). \tag{10}$$

The string $\mu C = (C, \mu_C)$ is a *subsequence with the membership function* μ_C, *shortly m-subsequence* of the m-string μA if C is a subsequence of the string A and $0 < \mu_C(c_t) \leq \mu_A(a_{i_t})$, for $1 \leq t \leq p$. The m-subsequence μC is a *closest m-subsequence* if $Val(\mu C) = \Sigma_{j=1}^{p} \mu_C(c_j) = \Sigma_{j=1}^{p} \mu_A(a_{i_j})$.

The string μC is a *common m-subsequence* of two m-strings μA and μB if μC is a m-subsequence of μA and μC is a m-subsequence of μB.

The string μC is a *closest common m-subsequence* of the m-strings μA and μB if μC is a common m-subsequence with the maximal value $Val(\mu C)$. It means, if μD is a common m-subsequence of the strings μA and μB then $Val(\mu D) \leq Val(\mu C)$.

If μC is a closest common m-subsequence of the m-strings, μA and μB then $\mu_C(c_t) = \min\{\mu_A(a_{k_t}), \mu_B(b_{l_t})\}$, for $1 \leq t \leq p$.

The CCS Problem: Let μA and μB be m-strings. To find a closest common m-subsequence of the m-strings μA and μB, $CCS(\mu A, \mu B)$ for short.

The MCCS Problem is to find the measure of CCS $(\mu A, \mu B)$, MCCS for short. It means, $\text{MCCS}(\mu A, \mu B) = Val(CCS(\mu A, \mu B))$.

Algorithms for CCS and MCCS Problem Andrejková are described in [2].

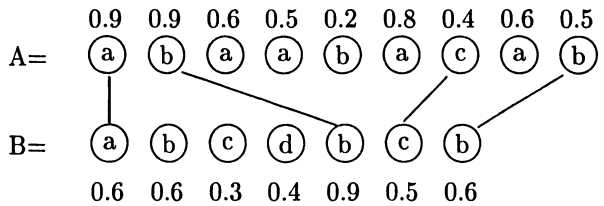

Fig. 1. The closest common m-subsequence of two m-strings A and B.

Example 1. $\Omega = \{a, b, c\}, A = abaabacab, m = 9, \mu_A = (.9, .9, .6, .5, .2, .8, .4, .6, .5)$, $B = abcdbcb, n = 7, \mu_B = (.6, .6, .3, .4, .9, .5, .6)$. The string $C = abcb$ is a subsequence, $D = abbcb$ is the longest common subsequence of the strings A and B, and μE, $E = abcb, \mu_E = (.6, .9, .4, .5)$ is the closest common m-subsequence of the m-strings μA and μB, $Val(\mu E) = \text{MCCS}(\mu A, \mu B) = 2.4$ as it is shown in Figure 1.

Let $P(\Omega)$ be the set of all subsets of Ω. *A string of sets* $\mathcal{B}$ over an alphabet Ω, *set-string* for short, is any finite sequence of subsets from $P(\Omega)$. Formally, $\mathcal{B} = B_1 B_2 \dots B_n, B_i \in P(\Omega), 1 \leq i \leq n$, n is the number of sets in $\mathcal{B}$. The length of the symbol string described by $\mathcal{B}$ is $N = \Sigma_{i=1}^{n} |B_i|$. A string of symbols $C = c_1 c_2 \dots c_p, c_i \in \Omega, 1 \leq i \leq p$, is a *subsequence of symbols* (subsequence, for short) of the set-string $\mathcal{B}$ if there is a nondecreasing mapping $F : \{1, 2, \dots, p\} \to \{1, 2, \dots, n\}$, such that

1. if $F(i) = k$ then $c_i \in B_k$, for $i = 1, 2, \dots, p$
2. if $F(i) = k$ and $F(j) = k$ and $i \neq j$ then $c_i \neq c_j$.

Let $\mathcal{A} = A_1 \dots A_m, \mathcal{B} = B_1 \dots B_n, 1 \leq m \leq n$, be two set-strings of sets over an alphabet Ω. The string of symbols C is a *common subsequence of symbols* of $\mathcal{A}$ and $\mathcal{B}$ is C a subsequence of symbols of $\mathcal{A}$ and C is a subsequence of symbols of the set-string $\mathcal{B}$.

As similar as for strings, let define *f-set* as a set with membership function defined on its elements. More exactly, if Ω is universe of discourse, *f-set* A over Ω is fuzzy subset of Ω which is characterised by its *membership function* $\mu_A : \Omega \to [0, 1]$.

Let $\mu_{B_i}, i = 1, 2, \dots, n$ be the membership functions of the sets $B_i, i = 1, 2, \dots, n$ in the string $\mathcal{B}$. It means, $\mu\mathcal{B} = \mu B_1 \mu B_2 \dots \mu B_n$. $\mu\mathcal{B}$ is the f-set-string $\mathcal{B}$ of f-sets $B_i, i = 1, 2, \dots, n$ with the membership functions μ_{B_i}, f-set-string $\mu\mathcal{B}$ for short. The *weight of the f-set* B with membership function μ_B is

$$W(B) = \sum_{x \in B} \mu_B(x). \tag{11}$$

A string μC is an *m-subsequence* of the f-set-string $\mu\mathcal{B}$ if (1) μC is the subsequence of symbols of the set-string $\mathcal{B}$ and (2) if $c = c_i, c_i \in B_k$ then $\mu_C(c) \leq \mu_{B_k}(c_i)$.

The m-string μC is *a common m-subsequence* of the f-set-strings $\mu\mathcal{A}$ and $\mu\mathcal{B}$ if μC is m-subsequence of $\mu\mathcal{A}$ and μC is m-subsequence of $\mu\mathcal{B}$.

The string μC is *a closest common m-subsequence* of the f-set-strings $\mu\mathcal{A}$ and $\mu\mathcal{B}$ if μC is a common subsequence with maximal value $Val(\mu C)$. Note that μC is not in general unique.

The SSCCS Problem: Let $\mu\mathcal{A}, \mu\mathcal{B}$ be two f-set-strings. The Set-Set Closest Common Subsequence problem of the f-set-strings $\mu\mathcal{A}$ and $\mu\mathcal{B}$, SSCCS($\mu\mathcal{A}, \mu\mathcal{B}$) for short, consists of finding a closest common m-subsequence μC with the maximal value $Val(\mu C)$.

The MSSCCS Problem consists of finding the measure of SSCCS f-set-string, MSSCCS($\mu\mathcal{A}, \mu\mathcal{B}$) for short.

It means, MSSCCS($\mu\mathcal{A}, \mu\mathcal{B}$) = Val(SSCCS($\mu\mathcal{A}, \mu\mathcal{B}$)),

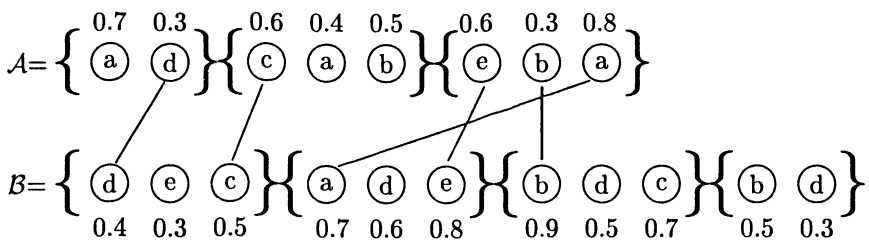

Fig. 2. The closest common subsequence of two f-set-strings $\mathcal{A}$ and $\mathcal{B}$.

Example 2. Let $\mathcal{A} = \{a,d\}\{c,a,d\}\{e,b,a\}, m = 3$, $\mu_{A_1} = (.7,.3), \mu_{A_2} = (.6,.4,.5), \mu_{A_3} = (.6,.3,.8), \mathcal{B} = \{d,e,c\}\{a,d,e\}\{b,d,c\}\{b,d\}, n = 4$. $\mu_{B_1} = (.4,.3,.5), \mu_{B_2} = (.7,.6,.8), \mu_{B_3} = (.9,.5,.7), \mu_{B_4} = (.5,.3)$. The membership values are described according to the named order in the set. For example, $\mu_{A_1}(a) = 0.7, \mu_{A_1}(d) = 0.3$. Then MSSCCS($\mu\mathcal{A}), \mu\mathcal{B}) = 2.4$ as it is shown in Figure 2.

5. ALGORITHM FOR MSSCCS PROBLEM WITH GÖDEL CONNECTIVES

The basic idea of the algorithm starts from the definition of MSSCCS Problem.

$$\text{MSSCCS}(\mu\mathcal{A}, \mu\mathcal{B}) = \max_{\mu C}\{Val(\mu C) : \mu C \text{ is the common } m\text{-subsequence of } \mu\mathcal{A} \text{ and } \mu\mathcal{B}\}. \tag{12}$$

In the following part of the paper we will use the f-sets only and for simpler description we will omit the symbol μ in the names of sets.

A *flattening* of a sequence of sets is defined as a concatenation, in order of the sequence, of strings formed by some permutation of individual elements of the sets in the sequence. For example, the flattening of the f-set-string $\mathcal{A}$ in Example 2 is $\mathcal{A}_{fl1} = dabaceba, \mu_{\mathcal{A}_{fl1}} = (.3,.7,.5,.4,.6,.6,.3,.8)$ and so is $\mathcal{A}_{fl2} = daacbbae, \mu_{\mathcal{A}_{fl2}} = (.3,.7,.4,.6,.5,.3,.8,.6)$.

If we have some flattenings of both set-strings then it is possible to apply the MCCS algorithm, Andrejková [2]. It is necessary to compute MCCS values of all pairs of all flattenings both set-strings but it is too much time consuming.

If we have the flattening of one set-string and the second is as set-string then it is possible to use the MSCCS algorithms. But it is necessary to compute MSCCS value for all flattenings of one string. It is to much time consuming too. Both algorithms have the exponential time complexity.

It is possible to use the following algorithm of polynomial time complexity. The algorithm works in two steps:

1. to create the string of symbols for each of set-string; each set can be encoded as the string of all permutations of its elements (the length of such string is $k^2 - 2 \cdot k + 4$, k is the number of elements in set [10]); for example, the shortest m -string of elements in the f-set-string $\mathcal{A}$ in example 2 is *dadcabcabcbeabeab* and so is *adacabcabcebaebae*.

2. to apply the MCCRS algorithm, Andrejková [1] for the two in the previous step constructed m-strings (each element of the f-set can be used once at most).

The algorithm works in polynomial time: $O(M^2 \cdot N^2 \cdot K)$, where $M = \Sigma_{i=1}^{m} |A^i|$, $N = \Sigma_{j=1}^{n} |B^j|$, and K is the number of elements in closest common restricted subsequence.

We formulate the following algorithm with the better time complexity according to Hirschberg idea [7]. The algorithm uses the intersection, union, equivalence and difference of fuzzy sets and checks subsets of used and free elements.

5.1. Description of the simple algorithm

The algorithm extends the strings of fuzzy sets one by one set and finds the closest common subsequence for the partial substrings. For each substring it memorizes free elements and used elements with their membership values of the current fuzzy sets in the strings. If we analyze the f-sets A_i and B_j from the common elements point of view then we can consider three parts as shown in Figure 3. Part I contains free elements of the f-set A_i and from the part I we can choose elements for prolongation of the closest common subsequence. Analogously for part III of the f-set B_j. Part II contains common elements but the f-subset actually used has a new membership function and it is f-subset of the f-set represented by part II. Since Gödel logic connectives will be used throughout the subscripts indicating this will be omitted.

We will now present a rigorous formulation of the above description.

For convenience, we define $A_0 = B_0 = \emptyset$.

We define $Ent(i, j)$ to be the set of quintuples (k, F_f, F_u, G_f, G_u) such that:

(1) k is the measure of γ, a common m-subsequence of some flattening of $A_1 \ldots A_i$ and some flattening of $B_1 \ldots B_j$, defined by (10).

(2) *free* f*-set* $F_f \subseteq A_i$ is the f-set of elements of A_i which are not used by γ,

(3) *free* f*-set* $G_f \subseteq B_j$, is the f-set of elements of B_j not used by γ,

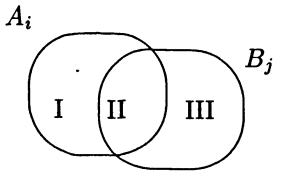

Fig. 3. Common and free elements of f-sets A_i and B_j.

(4) *f-set of used elements* $F_u \subseteq A_i$ is the f-set of elements of A_i used by γ, and

(5) *f-set of used elements* $G_u \subseteq B_j$ is the f-set of elements of B_j used by γ.

Example 3. $(0.8, \{(a, 0.4), (b, 0.5)\}, \{(c, 0.5)\}, \{(e, 0.3)\}\{(d, 0.3), (c, 0.5)\})$ is in $Ent(1,2)$, $(1, \emptyset, \{(a, 0.7), (d, 0.3)\}, \{(d, 0.6), (e, 0.8)\}, \{(a, 0.7)\})$ is in $Ent(2,1)$ and $(1.2, \{(b, 0.5)\}, \{(c, 0.5), (a, 0.4)\}, \{(d, 0.6), (e, 0.8)\}, \{(a, 0.4)\})$ is in $Ent(2,2)$ for f-set-strings in Example 2.

We refer to such quintuple as an *entry*. The measure of the CCS of some flattening of $A_1 \dots A_m$ and some flattening of $B_1 \dots B_n$ is then, by definition, the largest k such that $(k, F_f, F_u, G_f, G_u) \in Ent(m,n)$ for some f-sets F_f, F_u, G_f and G_u. $Ent(0,0)$ contains just one entry, namely $(0, \emptyset, \emptyset, \emptyset, \emptyset)$, while $Ent(i,j)$ can be computed dynamically from $Ent(i-1,j)$ and $Ent(i,j-1)$. The problem is that the cardinality of $Ent(i,j)$ could become very large, making such an algorithm exponential in the worst case.

Let $e = (k, F_f, F_u, G_f, G_u) \in Ent(i-1,j)$ and $F^s, F_u \subseteq F^s$ be the f-set with the following property: $x \varepsilon F^s \Leftrightarrow x \varepsilon F_u$ and $\mu_{F_u}(x) \leq \mu_{F^s}(x) = \mu_{A_i}(x)$. It means, the f-set F^s is the maximal f-set that has the same elements as the f-set F_u, but membership values of elements in F^s are the same as in the f-set A_i.

Let S be any subset of $A_i \cap G_f$. We say, e *vertically generates* $e' \in Ent(i,j)$ iff

1. $e' = (k + W(S) - W(A_i \cap G_u) + W(S'), A_i - S, F_u, G_f - S, G_u)$ for any subset S' of $A_i \cap G^s, W(S') > W(A_i \cap G_u)$, or
2. $e' = (k + W(S), A_i - S, F_u, G_f - S, G_u)$ and for each subset S' of $A_i \cap G^s$ is $W(S') \leq W(A_i \cap G_u)$.

The element $e' \in Ent(i,j)$ and it is shown by the following: If α is common m-subsequence with a measure $k = Val(\alpha)$ of the flattening of $A_1 \dots A_{i-1}$ and some flattening of $B_1 \dots B_j$, where $F_f \subseteq A_{i-1}$ and $G_f \subseteq_t B_j$ are free f-sets, and β is a m-sequence consisting of the elements of $S \subseteq A_i \cap G_f$ written in any order, then $\alpha\beta$ (having measure $k + Val(S)$) is common subsequence of a flattening of $A_1 \dots A_i$ and a flattening of $B_1 \dots B_j$, with free f-sets $A_i - S$ and $G_f - S$. The used elements from f-set G_u can be used with some better membership values and it is evaluated by the comparison of the weights of the sets $A_i \cap G^s$ and $A_i \cap G_u$. If $W(A_i \cap G_u) < W(A_i \cap G^s)$ then there exists some better using of elements in A_i.

Similarly, if $(k, F_f, F_u, G_f, G_u) \in Ent(i, j-1)$ and $S \subseteq F_f \cap B_j$ and S' is any subset of $B_j \cap F^s, W(S') > W(B_j \cap F_u)$, we say (k, F_f, F_u, G_f, G_u) *horizontally generates* $(k + W(S) - W(B_j \cap F_u) + W(S'), F_f - S, F_u, B_j - S, G_u) \in Ent(i, j)$ or $(k+W(S), F_f - S, F_u, B_j - S, G_u) \in Ent(i, j)$ according to the relation $W(B_j \cap F_u) < / \geq W(S')$. It means, the following lemma is fulfilled:

Lemma 1. If $e \in Ent(i, j)$ for $i + j \geq 0$ then e is generated by some element of either $Ent(i-1, j)$ or $Ent(i, j-1)$.

Proof. $e = (k, F_f, F_u, G_f, G_u) \in Ent(i, j)$, it means $e = \alpha\beta$, β is the part of elements in $A_i \cap B_j$. According to above construction, the part β is the prolongation of some element $e' \in Ent(i-1, j)$ or $Ent(i, j-1)$. In the part α should be elements with higher membership values. □

The element $e \in Ent(i, j)$ is generated from elements in $E(i-1, j)$ or $Ent(i, j-1)$ using of two sets: the free subset and the used subset of B_j, respectively A_i. The following algorithm is a dynamic programming algorithm in which the boundary conditions are set and then the interval entries are determined:

Algorithm A.

for all i *do* $Ent(i, 0) := \{(0, A_i, \emptyset, \emptyset, \emptyset)\}$
for all j *do* $Ent(0, j) := \{(0, \emptyset, \emptyset, B_j, \emptyset)\}$
for $i := 1$ *to* m *do*
  for $j := 1$ *to* n *do*
    $Ent(i, j) :=$ {*all entries vertically generated from* $Ent(i-1, j)$}
      $\bigcup$ {*all entries horizontally generated from* $Ent(i, j-1)$}
  max_k := *the largest* k *such that* $(k, F_f, F_u, G_f, G_u) \in Ent(m, n)$ *for some* F_f, F_u, G_f, G_u.

5.2. Description of the better algorithm

The above algorithm may be very time-consuming because of too many quintuples is necessary to analyze. We will speed the algorithm by eliminating consideration of many quintuples.

If $(k, F_f, F_u, G_f, G_u), (k', F'_f, F'_u, G'_f, G'_u) \in Ent(i, j)$, we say that (k, F_f, F_u, G_f, G_u) *dominates* $(k', F'_f, F'_u, G'_f, G'_u)$ $((k', F'_f, F'_u, G'_f, G'_u) \preceq (k, F_f, F_u, G_f, G_u))$ if the following conditions hold:

1. $d = k - k' \geq 0$,
2. $\left(W(F'_f - F_f) \leq d \text{ and } F'_u \subseteq F_u\right)$ or $\left(W(F'_u - F_u) \leq d \text{ and } F'_f \subseteq F_f\right)$,
3. $\left(W(G'_f - G_f) \leq d \text{ and } G'_u \subseteq G_u\right)$ or $\left(W(G'_u - G_u) \leq d \text{ and } G'_f \subseteq G_f\right)$.

The relation "$\preceq$" is a transitive, antisymmetric and reflexive relation. The elements of $Ent(i,j)$ can be ordered according to relation "$\preceq$", it means they are ordered in *chains*. The last element of the chain has maximal measure and that is very important as can be seen according to the following lemma:

Lemma 2. Any element of $Ent(i,j)$ which is not maximal with respect to the relation "$\preceq$" can be discarded during execution of the algorithm without affecting the final value of $\max\_k$.

Proof. It will be proved by downward induction on both indices i and j. The value of $\max\_k$ is obtained from $Ent(m,n)$ in the last step and all other elements may be discarded with no effect.

Suppose $i+j < m+n$ and $e' \in Ent(i,j), e'$ is not maximal. Let $e \in Ent(i,j)$ be maximal. It means, $e' \preceq e$. It is necessary to prove that maximal element of $Ent(i+1,j)$ or $Ent(i,j+1)$ which is generated by e' can be generated by e too. And the element e' can be discarded.

Let $e = (k, F_f, F_u, G_f, G_u), e' = (k', F_f', F_u', G_f', G_u')$ and e' vertically generates f'. f' should have two forms for some m-set $P \subseteq A_{i+1} \cap G_f'$

(a) $f' = (k' + W(P) - W(A_{i+1} \cap G_u') + W(A_{i+1} \cap G'^s), A_{i+1} -_m P, P, G_f' -_m P, G_u')$, if $W(A_{i+1} \cap G_u') < W(A_{i+1} \cap G'^s)$, or

(b) $f' = (k' + W(P), A_{i+1} -_m P, P, G_f' -_m P, G_u')$, if $W(A_{i+1} \cap G_u') \geq W(A_{i+1} \cap G'^s)$.

Let $S = P \cap G_f$, and f is vertically generated by e. f should have two forms: (1) $f = (k + W(S) - W(A_{i+1} \cap G_u) + W(A_{i+1} \cap G^s), A_{i+1} -_m S, S, G_f -_m S, G_u)$ or (2)$f = (k + W(S), A_{i+1} -_m S, S, G_f -_m S, G_u)$. It is necessary to analyze four cases to prove the Lemma (a)-(1), (a)-(2), (b)-(1), (b)-(2). We start with the first one, it means (a)-(1), and $W(A_{i+1} \cap G_u') < W(A_{i+1} \cap G'^s)$ and $W(A_{i+1} \cap G_u) < W(A_{i+1} \cap G^s)$.

Since $e' \preceq e, d = k - k'$,
$d \geq 0$, $\Big(\Big(W(F_f' -_m F_f) \leq d$ and $F_u' \subseteq F_u\Big)$ or $\Big(W(F_u' -_m F_u) \leq d$ and $F_f' \subseteq F_f\Big)\Big)$, and $\Big(\Big(W(G_f' -_m G_f) \leq d$ and $G_u' \subseteq G_u\Big)$ or $\Big(W(G_u' -_m G_u) \leq d$ and $G_f' \subseteq G_f\Big)\Big)$. Then $W(P -_m S) = W(P -_m P \cap G_f) = W(P -_m G_f) \leq W(F_f' -_m F_f) \leq d$ and $W(P -_m S) \leq W(P) - W(S)$. Let $d' = (W(P) - W(S)) - (W(A_{i+1} \cap G^s) - W(A_{i+1} \cap G'^s)) - (W(A_{i+1} \cap G_u) - W(A_{i+1} \cap G_u') \leq d$ We prove that $f' \preceq f$, it means f' is not maximal or $f = f'$. According to definition of "$\preceq$" it is necessary to check three conditions 1–3.

1. $z = k + W(S) - W(A_{i+1} \cap G_u) + W(A_{i+1} \cap G^s) - (k' + W(P) - W(A_{i+1} \cap G_u') + W(A_{i+1} \cap G'^s)) = k - k' - (W(P) - W(S)) + W(A_{i+1} \cap G^s) - W(A_{i+1} \cap G'^s) + W(A_{i+1} \cap G_u) - W(A_{i+1} \cap G_u' \geq d - d' \geq 0$
2. $W(P -_m S) \leq d$ and $A_{i+1} -_m P \subseteq_m A_{i+1} -_m S$
3. $W(G_f' -_m P -_m (F_f -_m S)) = W(G_f' -_m G_f) \leq d$ and $G_u' \subseteq_m G_u$.

The rest three cases can be proved by a very similar method. And the vertical case is very similar. □

If $e = (k, F_f, F_u, G_f, G_u) \in Ent(i,j)$, we define the *horizontal child* of e to be $hor(e) = k+W(F_f \cap B_{j+1}) - W(A_i \cap G_u) + W(A_i \cap G^s), F_f - B_{j+1}, F_u, B_{j+1} - F_f, G_u)$ or $hor(e) = k+W(F_f \cap B_{j+1}), F_f - B_{j+1}, F_u, B_{j+1} - F_f, G_u)$ and define the *vertical child* of e to be $ver(e) = k+W(A_{i+1} \cap G_f) - W(B_{j+1} \cap G_u) + W(B_{j+1} \cap G^s), B_{j+1} - G_f, F_u, G_f - B_{j+1}, G_u)$or $ver(e) = k+W(A_{i+1} \cap G_f), B_{j+1} - G_f, F_u, G_f - B_{j+1}, G_u)$. We define $MaxEnt(i,j)$ to be the set of maximal elements of $Ent(i,j)$ under the dominance relation "$\preceq$".

Lemma 3. Any entry horizontally generates at most one maximal entry and vertically generates at most one maximal entry.

Proof. Let $e = (k, F_f, F_u, G_f, G_u) \in Ent(i,j)$. The only elements vertically generated by e which can be maximal are in the $ver(e)$, since they dominates any others vertically generated by e. Similarly, $hor(e)$ dominates any entries horizontally generated by e. □

We say that (k, F_f, F_u, G_f, G_u) *strongly dominates* (k', F_f, F_u, G_f, G_u) if $k > k'$. If $S \subseteq Ent(i,j)$, defines $Dom(S) \subseteq S$ to be the set obtained by deleting every element of S which is strongly dominated by another element of S. We now inductively define sets $Chain(i,j) \subseteq Ent(i,j)$ by:

1. $Chain(i,0) = \{(0, A_i, \emptyset, \emptyset, \emptyset)\}$,
2. $Chain(0,j) = \{(0, \emptyset, \emptyset, B_j, \emptyset)\}$,
3. $Chain(i,j) = Dom(\{ver(e) | e \in Chain(i-1,j)\} \cup \{hor(e) | e \in \in Chain(i,j-1)\})$.

We refer to entries $Chain(i,j)$ as *weekly maximal.* We observe the following theorem.

Theorem 1. $MaxEnt(i,j) \subseteq Chain(i,j)$.

Proof. By induction. For $i = 0$ or $j = 0$ the two sets $MaxEnt(i,j)$ and $Chain(i,j)$ are identical. For $i,j > 0$, and $e \in MaxEnt(i,j)$ must be vertical or horizontal child of some maximal element, which is weakly maximal by induction. It means, e must be weakly maximal, since it is maximal and thus cannot be deleted by operator Dom. □

Using the results of the Lemmas 2 and 3 and Theorem 1 we have the following algorithm:

Algorithm B.

{Using weakly maximal entries.}

for all i *do* $Chain(i,0) := \{(0, A_i, \emptyset, \emptyset, \emptyset)\}$;

for all j *do* $Chain(0,j) := \{(0, \emptyset, \emptyset, B_j, \emptyset)\}$;

for i:=1 *to* m *do*

for j:=1 *to* n *do*

begin

$Chain(i,j) := \emptyset$;

for all $(k, F_f, F_u, G_f, G_u) \in Chain(i, j-1)$ *do begin*

$F^s := F_u; \mu_{F^s} := \mu_{A_i}$;

$help := W(B_j \cap F^s) - W(B_j \cap F_u)$;

if $help \leq 0$ *then*

insert $(k + W(F \cap B_j), F_f - B_j, F_u, B_j - F_f, G_u)$ *into* $Chain(i,j)$ *else*

insert $(k + W(F_f \cap B_j) + help, F_f - B_j, F_u, B_j - F_f, G_u)$ *into* $Chain(i,j)$

end;

for all $(k, F_f, F_u, G_f, G_u) \in Chain(i-1, j)$ *do begin*

$G^s := G_u; \mu_{G^s} := \mu_{B_j}$;

$help := W(A_i \cap G^s) - W(A_i \cap G_u)$;

if $help \leq 0$ *then*

insert $(k + W(A_i \cap G), A_i - G_f, F_u, G_f - A_i, G_u)$ *into* $Chain(i,j)$ *else*

insert $(k + W(G_f \cap A_i) + help, A_i - G_f, F_u, G_f - A_i, G_u)$ *into* $Chain(i,j)$

end;

delete all nonweakly maximal elements from $Chain(i,j)$

end

max_k := *the maximum value of* k *such that* $(k, F_f, F_u, G_f, G_u) \in Chain(m,n)$ *for some* F_f, F_u, G_f *and* G_u.

The algorithm works in $O(m \cdot n \cdot K \cdot t)$-time, where K is the maximal number of elements in $Chain(i,j)$ and t is the maximal time spent for computing of the intersection of two sets. The algorithm works in $O(m \cdot n \cdot k)$-space, where k is the maximal number of elements in the f-sets A_i, B_j.

6. CONCLUDING REMARKS

The polynomial algorithms for the operations on fuzzy sets can be used to find the closest common subsequence (according to Gödel connectives) of two fuzzy sets strings (MSSCCS Problem) in polynomial time. The algorithms should be modified for using of Łukasiewicz and product logic connectives.

ACKNOWLEDGEMENT

This research was supported by the Slovak Grant Agency through Grant 1/7557/20.

(Received June 13, 2000.)

REFERENCES

[1] G. Andrejková: The longest restricted common subsequence problem. In: Proc. Prague Stringology Club Workshop'98, Prague 1998, pp. 14–25.
[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] S. Gottwald: Fuzzy Sets and Fuzzy Logic. Vieweg, Wiesbaden 1993, p. 216.
[4] P. Hájek: Mathematics of Fuzzy Logic. Kluwer, Dordrecht 1998.
[5] D. S. Hirschberg: Algorithms for longest common subsequence problem. J. Assoc. Comput. Mach. *24* (1977), 664–675.
[6] D. S. Hirschberg and L. L. Larmore: The set LCS problem. Algorithmica *2* (1987), 91–95.
[7] D. S. Hirschberg and L. L. Larmore: The set-set LCS problem. Algorithmica *4* (1989), 503–510.
[8] A. Kaufmann: Introduction to Theory of Fuzzy Subsets. Vol. 1: Fundamental Theoretical Elements. Academic Press, New York 1975.
[9] E. P. Klement, R. Mesiar, and E. Pap: Triangular Norms. Kluwer, Dordrecht 2000.
[10] S. P. Mohanty: Shortest string containing all permutations. Discrete Math. *31* (1980), 91–95.
[11] N. Nakatsu, Y. Kombayashi, and S. Yajima: A longest common subsequence algorithm suitable for similar text strings. Acta Inform. *18* (1982), 171–179.
[12] L. Zadeh: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems *1* (1978), 3–28.

RNDr. Gabriela Andrejková, CSc., Department of Computer Science, Faculty of Science, P. J. Šafárik University, 041 54 Košice. Slovak Republic.
e-mail: andrejk@kosice.upjs.sk