7-2013

Computing with Words: Towards a New Tuple-Based Formalization

Olga Kosheleva
The University of Texas at El Paso, olgak@utep.edu

Vladik Kreinovich
The University of Texas at El Paso, vladik@utep.edu

Ariel Garcia
The University of Texas at El Paso, adgarcia11@miners.utep.edu

Felipe Jovel
The University of Texas at El Paso, fjovel@miners.utep.edu

Luis A. Torres Escobedo
The University of Texas at El Paso, latorresesobedo@utep.edu

See next page for additional authors

Comments:

Technical Report: UTEP-CS-13-31a

To appear in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics IEEE SMC'2013*, Manchester, UK, October 13-16, 2013.

Computing with Words: Towards a New Tuple-Based Formalization

Olga Kosheleva, Vladik Kreinovich,
Ariel Garcia, Felipe Jovel, Luis A. Torres Escobedo
University of Texas at El Paso
500 W. University
El Paso, Texas 79968, USA
vladik@utep.edu, olgak@utep.edu,
adgarcia11@miners.utep.edu, fjovel@miners.utep.edu,
latorresescobedo@utep.edu

Thavatchai Ngamsantivong
Computer and Information Science
Faculty of Applied Sciences
King Mongkut's Univ. of Technology North Bangkok
1518 Piboonsongkhram Road, Bangsue
Bangkok 10800 Thailand
tvc@kmutnb.ac.th

***Abstract*—An expert opinion describes his or her opinion about a quantity by using imprecise ("fuzzy") words from a natural language, such as "small", "medium", "large", etc. Each of these words provides a rather crude description of the corresponding quantity. A natural way to refine this description is to assign degrees to which the observed quantity fits each of the selected words. For example, an expert can say that the value is reasonable small, but to some extent it is medium. In this refined description, we represent each quantity by a tuple of the corresponding degrees.**

Once we have such a tuple-based information about several quantities x_1, ..., x_m, and we know that another quantity y is related to x_i by a known relation $y = f(x_1, \ldots, x_m)$, it is desirable to come up with a resulting tuple-based description of y. In this paper, we describe why a seemingly natural idea for computing such a tuple does not work, and we show how to modify this idea so that it can be used.

***Index Terms*—computing with words, fuzzy logic**

I. Computing with Words: Formulation of the Problem

Need to use words. Often, to describe a quantity such as temperature, height, etc., we use words such as "small", "medium", "high", etc.

Need to go beyond simple words. If we only use the selected words, we get a rather crude description of the quantity.

To get a more accurate description, we can say, e.g., that someone is rather short, but closer to medium height. In this case, to a large extent, this person is short, but to some degree, this person is of medium height.

How to go beyond simple words: a natural idea. In general, an accurate description of the quantity may include not just one word, but several words, with degrees associated with different words. A similar description can be used to describe situations when we are uncertain: e.g., if we do not know whether a person is short, medium, or tall, we assign, e.g., the same degree 1 to the possible degree to which this person is short, medium, and tall.

Once we fix the words $w_1, \ldots, w_n$, each quantity is then represented by the corresponding tuple of degrees

$$d = (d_1, \ldots, d_n).$$

Need for data processing. In many practical situations, we are interested in the value of a real-valued quantity y which is difficult (or even impossible) to measure or estimate directly. For example, we are often interested in the future value y of some quantity (e.g., tomorrow's weather), and it is not possible to directly measure a future value of a quantity.

In such situations, a usual approach is:

- to estimate easier-to-estimate auxiliary real-valued quantities $x_1, \ldots, x_m$ which are related to y by a known dependence $y = f(x_1, \ldots, x_m)$, and
- to use the estimates of x_i to compute the estimate for y.

This computation of y based on $x_1, \ldots, x_m$ (or, to be more precise, computation of an estimate for y from the estimates for x_i) is known as *data processing*.

Need for the corresponding computing with words. When the estimates for x_i are given in the form of tuples d, we face the following problem:

- we know the tuples $d^{(j)} = \left(d_1^{(j)}, \ldots, d_n^{(j)}\right)$ which describes our knowledge about each input x_j, $1 \le j \le m$;
- we want to describe the resulting knowledge about y in a similar tuple form.

In particular, in the simplest case when we have only two inputs x_1 and x_2, and data processing consists of applying a simple arithmetic operation, e.g., $f(x_1, x_2) = x_1 + x_2$, $f(x_1, x_2) = x_1 - x_2$, or $f(x_1, x_2) = x_1 \cdot x_2$, we face the following problem:

- if we have one quantity x_1 which is characterized by the tuple $d^{(1)}$ and
- we have another quantity x_2 which is characterized by the tuple $d^{(2)}$,

978-1-4799-0652-9/13/$31.00

- then we should be able to produce a tuple characterizing the sum $x_1 + x_2$ of these quantities, a tuple characterizing their difference $x_1 - x_2$, their product $x_1 \cdot x_2$, etc.

In general, instead of computing with numbers, we should be able to compute with words. The need for such computing with words was first emphasized by L. Zadeh; see, e.g., [9].

II. Analysis of the Problem

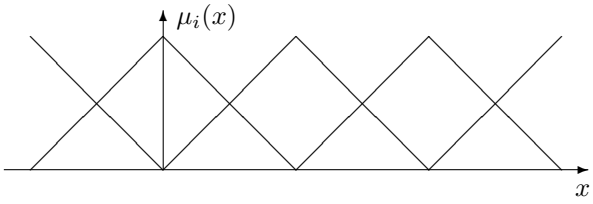
How to represent the original words: general idea. A natural way to represent the original words in the computer-understandable form is to use fuzzy logic [2], [3], [8], i.e., to assign, to each word w_i, a function $\mu_i(x)$ that describes, for each value x, the degree to which this value satisfies the corresponding property. The corresponding function $\mu_i(x)$ is known as the *membership function* corresponding to the word w_i.

Comment.

- The degree $\mu_i(x)$ can be obtained, e.g., by polling experts; in this case, if m out of n experts think that x satisfies the property w_i, then we take $\mu_i(x) = m/n$.
- Alternatively, we can ask a single expert to mark the desired degree by a value on a scale from 0 to n; if the expert marks the value m, we take $\mu_i(x) = m/n$.

How to represent the original words: details. Usually, the corresponding membership functions are triangular, i.e., first linearly increase from 0 to 1, then linearly decrease from 1 to 0 at the same rate. Different functions $\mu_i(x)$ usually differ by a shift, so for some some starting point s and step h, we have

$$\mu_i(x) = \max\left(0, 1 - \frac{|x - (s + i \cdot h)|}{h}\right). \qquad (1)$$



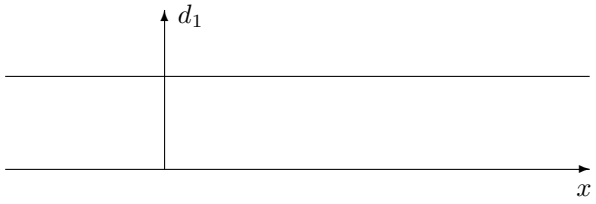
Comment. For example, these membership functions can describe the usual terms "negligible", "small", "medium", etc.

From a words-related tuple representation to a membership function. A tuple $d = (d_1, \ldots, d_n)$ represents a value x if one of the following conditions hold:

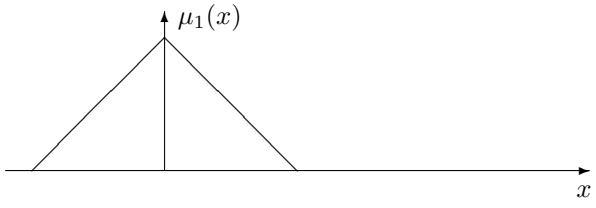
- the corresponding quantity q is characterized by the first word w_1, and x satisfies the property described by this word;
- the corresponding quantity q is characterized by the second word w_2, and x satisfies the property described by this word;
- $\ldots$
- the corresponding quantity q is characterized by the n-th word w_n, and x satisfies the property described by this word.

Here:

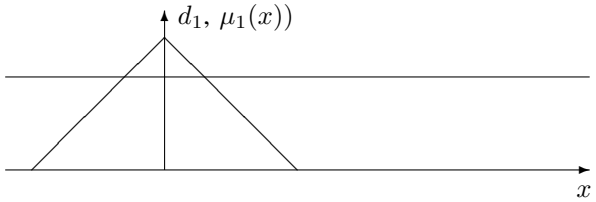
- the degree to which q is characterized by the first word is d_1, and

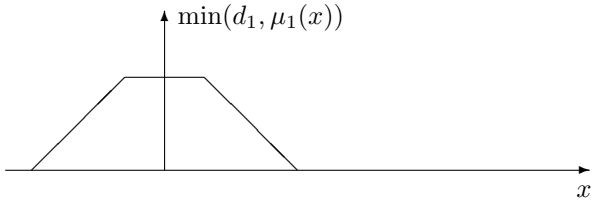


- the degree to which x satisfies the property described by this word is $\mu_1(x)$.



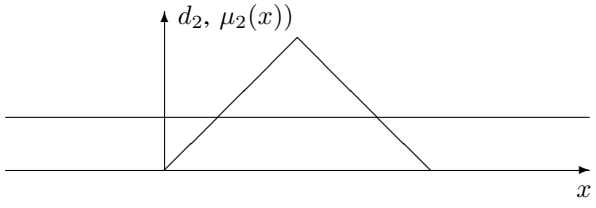
From the computational viewpoint, the simplest way to describe "and" in fuzzy logic is to use minimum: out of all t-norms, only $a \cdot b$ and $\min$ can be implemented by a single arithmetic operation, and computing $\min$ requires fewer bit operations and is, thus, faster. So, the degree to which q is characterized by the first word w_1 *and* x satisfies the property described by this word can be described as $\min(d_1, \mu_1(x))$.

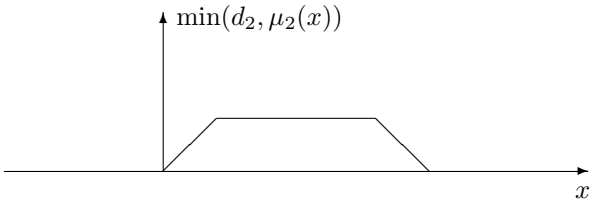




Similarly:

- the degree to which q is characterized by the second word w_2 *and* x satisfies the property described by this word can be described as $\min(d_2, \mu_2(x))$;

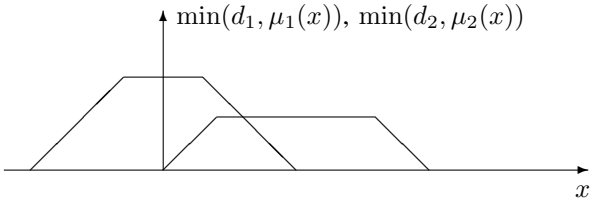


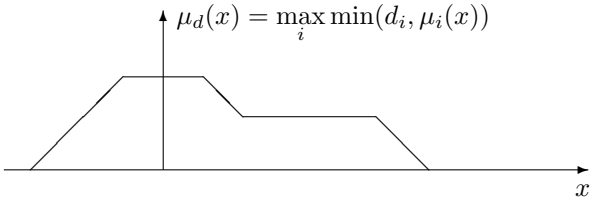


- ...
- the degree to which q is characterized by the n-th word w_n *and* x satisfies the property described by this word can be described as $\min(d_n, \mu_n(x))$.

The simplest way to describe "or" in fuzzy logic is to use maximum, so the degree to which one of these conditions is satisfied is equal to

$$\mu_d(x) = \max(\min(d_1, \mu_1(x)), \ldots, \min(d_n, \mu_n(x))). \quad (2)$$





Resulting fuzzy-based formalization of computing with words: a natural idea. We know the tuples $d^{(1)}$ and $d^{(2)}$ describing the two quantities x_1 and x_2, and we want to find a tuple d corresponding to the value $y = f(x_1, x_2)$ for some known function $f(x_1, x_2)$. A natural idea to do it is as follows:

- first, we use the formula (1) to generate membership functions $\mu^{(1)}(x_1)$ and $\mu^{(2)}(x_2)$ corresponding to the tuples $d^{(1)}$ and $d^{(2)}$;
- then, by applying Zadeh's extension principle to these membership functions, we compute the membership function $\mu(x)$ corresponding to $y = f(x_1, x_2)$;
- finally, we need to generate the tuple d corresponding to the resulting membership function $\mu(x)$.

What is necessary to implement the above idea. To implement the above idea, we need to be able to generate a tuple corresponding to a given membership function.

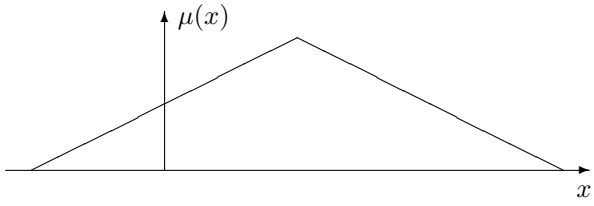
III. A Seemingly Natural Idea and Its Limitations

How to transform a membership function into a tuple: a seemingly natural idea. At first glance, we have a natural way of computing the degree d_i to which it is possible that a quantity described by the new membership function $\mu(x)$ satisfies the property described by the word w_i. There are several possible value x, so the quantity corresponds to w_i if one of the following statements hold:

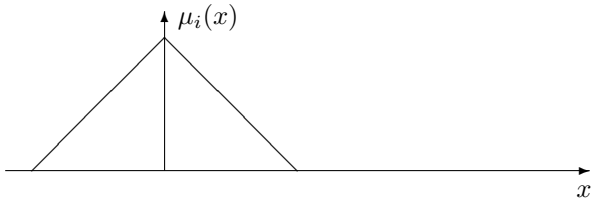
- for one possible value x, this value is in agreement with the new membership function and with the word w_i;
- for another possible value x, this value is in agreement with the new membership function and with the word w_i;
- ...

For each number x:

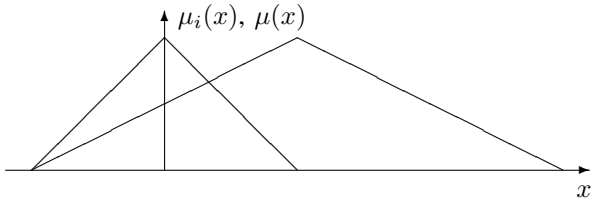
- the degree to which this number is in agreement with the new membership function is equal to $\mu(x)$,

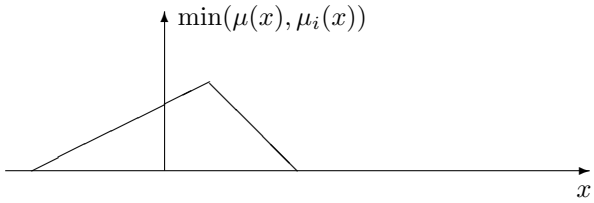


- the degree to which this number is consistent with the word w_i is equal to $\mu_i(x)$.



Thus, the degree to which this number x is in agreement with the new membership function *and* in agreement with the word w_i is equal to $\min(\mu(x), \mu_i(x))$:





The degree to which one of the conditions corresponding to different value x hold is equal to the maximum of all such values, i.e., to

$$A_i \stackrel{\text{def}}{=} \max_x(\min(\mu(x), \mu_i(x))). \quad (3)$$

Comment. In deriving the formula (3), we used $\max$ for "or" and $\min$ for "and". If we use sum for "or" and product for

"and", we get F-transform instead of the formula (2); see, e.g., [1], [4], [5], [6], [7].

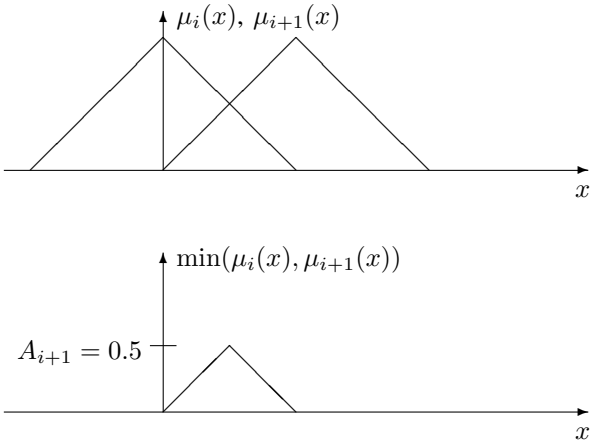
How to transform a membership function into a tuple: a reasonable requirement. It is reasonable to require that:

- if we start with the word w_i itself, i.e., with the tuple $d = (0, \ldots, 0, 1, 0, \ldots, 0)$ in which $d_i = 1$ and $d_j = 0$ for all $j \neq i$, and
- we perform no operations at all, i.e., use the function $f(x) = x$,
- then we would like to get back the same tuple

$$d = (0, \ldots, 0, 1, 0, \ldots, 0).$$

How to transform a membership function into a tuple: limitations of a seemingly natural idea. Unfortunately, the above seemingly reasonable approach does not satisfy this requirement. Specifically, when we apply this approach to the word w_i, then:

- for this index i, we do get $A_i = 1 = d_i$;
- however, instead of the desired $d_{i-1} = d_{i+1} = 0$, we get $A_{i-1} = A_{i+1} = 0.5$.



Thus, we do not get the original tuple back.

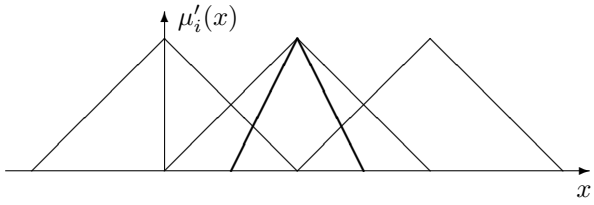
What is needed. We therefore need to modify the above seemingly natural procedure, to make sure that it returns the original tuple if no operation is performed.

How can we do this? The main problem with the above idea is that the neighboring membership functions overlap. This leads us to the natural idea.

IV. Main Idea, Resulting Definition, and Preliminary Results

Main idea. Since intersection is a problem, let us remove the intersecting parts from the membership function before applying a formula of type (3). We still want to compute the i-th coefficient d_i corresponding to the membership function $\mu(x)$ by comparing this membership function with the membership function $\mu_i(x)$ representing the i-th word w_i. However, instead of directly comparing these functions, let us first cut off, from both of them, parts intersecting with the neighboring membership functions. In other words, we first compute "reduced" functions

$$\mu_i'(x) = \max(0, \mu_i(x) - \max(\mu_{i-1}(x), \mu_{i+1}(x))) \qquad (4)$$



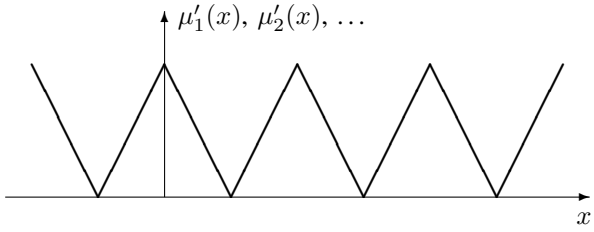
and

$$\mu'(x) = \max(0, \mu(x) - \max(\mu_{i-1}(x), \mu_{i+1}(x))), \qquad (5)$$

and then compute the degrees based on these reduced functions, as

$$A_i = \max_x(\min(\mu'(x), \mu_i'(x))). \qquad (6)$$

Here, the reduced functions $\mu_i'(x)$ no longer overlap:



Resulting definition of a membership function-to-tuple transformation. Thus, we arrive at the following definitions.

Definition 1.

- *Let $\mu_i(x)$ be a sequence of membership functions described by the formula* (1)*, and*
- *let $\mu(x)$ be a membership function.*

By a tuple $d = (d_1, \ldots, d_n)$ *corresponding to the membership function $\mu(x)$, means a sequence of values obtained by using formulas* (4)–(6).

The following easy-to-prove result shows that this modification of the original formula (3) indeed enables us to reconstruct the original degrees d_i:

Proposition 1.

- *Let $\mu_i(x)$ be a sequence of triangular functions* (1)*,*
- *let $d = (d_1, \ldots, d_n)$ be a tuple of a numbers $d_i \in [0, 1]$,*
- *and let $\mu_d(x)$ be described by the formula* (2).

For this function $\mu_d(x)$, formulas (4)–(6) *lead to $A_i = d_i$ for all i.*

Proof.

1°. We are interested in the maximum of the function $\min(\mu_d'(x), \mu_i'(x))$. Let us first show that this maximum is always attained on the interval $[t_{i-1}, t_{i+1}]$, where we denoted

$t_i \stackrel{\text{def}}{=} s + i \cdot h$, because outside this interval, the above function is equal to 0.

Indeed, the minimum function is always non-negative. The function $\mu_i(x)$ (as defined by the formula (1)) is only different from 0 on the interval $[t_{i-1}, t_{i+1}]$. By definition of a reduced function(formula (4)), $\mu_i'(x) \le \mu_i(x)$ and thus, the reduced function $\mu_i'(x)$ can only be different from 0 on the interval $[t_{i-1}, t_{i+1}]$. By definition (4) of the reduced function, we have $\mu_i'(x) \le \mu_i(x)$, hence $\min(\mu_d'(x), \mu_i'(x)) \le \mu_i'(x) \le \mu_i(x)$.

Since outside the interval $[t_{i-1}, t_{i+1}]$, we have $\mu_i(x) = 0$, the minimum $\min(\mu_d'(x), \mu_i'(x))$ is also equal to 0 for

$$x \notin [t_{i-1}, t_{i+1}].$$

2°. We want to prove that the largest value of $\min(\mu_d'(x), \mu_i'(x))$ is equal to d_i. To prove this, we will prove two auxiliary statements:

- that $\min(\mu_d'(t_i), \mu_i'(t_i)) = d_i$, and
- that $\min(\mu_d'(x), \mu_i'(x)) \le d_i$ for all other values

$$x \in [t_{i-1}, t_{i+1}].$$

2.1°. Let us first prove that $\min(\mu_d'(t_i), \mu_i'(t_i)) = d_i$. To prove this equality, we will:

- first compute $\mu_d'(t_i)$,
- then compute $\mu_i'(t_i)$,
- and finally compute the minimum of these two values.

$2.1.1^\circ$. Let us first compute $\mu_i'(t_i)$.

From the formula (1), we can conclude that for $x = t_i$, we have $\mu_i(t_i) = 1$ and $\mu_j(t_i) = 0$ for all $j \ne i$. Thus, by definition (4) of the reduced function $\mu_i'(x)$, we have

$$\mu_i'(t_i) = \max(0, \mu_i(t_i) - \max(\mu_{i-1}(t_i), \mu_{i+1}(t_i))) =$$
$$\max(0, 1 - \max(0, 0)) = \max(0, 1) = 1.$$

$2.1.2^\circ$. Let us now compute $\mu_d'(t_i)$.

From the fact that $\mu_i(t_i) = 1$ and $\mu_j(t_i) = 0$ for all $j \ne i$, we conclude that $\min(d_i, \mu_i(t_i)) = \min(d_i, 1) = d_i$ and $\min(d_j, \mu_j(t_i)) = \min(d_j, 0) = 0$ for all $j \ne i$. Thus, the value $\mu_d(t_i)$ (as defined by the formula (2)) is equal to $\mu_d(t_i) = \max(d_i, 0, \ldots, 0) = d_i$. Hence,

$$\mu_d'(t_i) = \max(0, \mu_d(t_i) - \max(\mu_{i-1}(t_i), \mu_{i+1}(t_i))) =$$
$$\max(0, d_i - \max(0, 0)) = \max(0, d_i) = d_i.$$

$2.1.3^\circ$. We have computed $\mu_i'(t_i) = 1$ and $\mu_d'(t_i) = d_i$; thus, $\min(\mu_d'(t_i), \mu_i'(t_i)) = \min(d_i, 1) = d_i$.

The first auxiliary statement is proven.

2.2°. Let us now prove that $\min(\mu_d'(x), \mu_i'(x)) \le d_i$ for all $x \in [t_{i-1}, t_{i+1}]$.

On this interval, only the function $\mu_i(x)$ and two neighboring membership functions $\mu_{i-1}(x)$ and $\mu_{i+1}(x)$ are different from 0, all the other are equal to 0. Thus, for these x, the value $\mu_d(x)$ (as defined by the formula (2)) is equal to the largest of the three values:

- the value $\min(\mu_{i-1}(x), d_{i-1})$,
- the value $\min(\mu_i(x), d_i)$, and
- the value $\min(\mu_{i+1}(x), d_{i+1})$.

The maximum of the three numbers is equal to one of them. Let us consider these three cases one by one and show that in all three cases, we have $\min(\mu_d'(x), \mu_i'(x)) \le d_i$.

$2.2.1^\circ$. When $\mu_d(x) = \min(\mu_{i-1}(x), d_{i-1})$, then $\mu_d(x) \le \mu_{i-1}(x)$ and thus, $\mu_d(x) \le \max(\mu_{i-1}(x), \mu_{i+1}(x))$. Hence,

$$\mu_d'(x) = \max(0, \mu_d(x) - \max \mu_{i-1}(x), \mu_{i+1}(x))) = 0,$$

and so, $\min(\mu_d'(x), \mu_i'(x)) = 0 \le d_i$.

$2.2.2^\circ$. When $\mu_d(x) = \min(\mu_i(x), d_i)$, then $\mu_d(x) \le d_i$ and thus, $\mu_d'(x) = \max(0, \mu_d(x) - \max \mu_{i-1}(x), \mu_{i+1}(x))) \le \mu_d(x) \le d_i$. Hence, $\min(\mu_d'(x), \mu_i'(x)) \le \mu_d'(x) \le d_i$.

$2.2.3^\circ$. When $\mu_d(x) = \min(\mu_{i+1}(x), d_{i+1})$, then $\mu_d(x) \le \mu_{i+1}(x)$ and thus, $\mu_d(x) \le \max(\mu_{i-1}(x), \mu_{i+1}(x))$. Hence,

$$\mu_d'(x) = \max(0, \mu_d(x) - \max \mu_{i-1}(x), \mu_{i+1}(x))) = 0,$$

and so, $\min(\mu_d'(x), \mu_i'(x)) = 0 \le d_i$.

$2.2.4^\circ$. In all three cases, we have the desired inequality. Thus, the inequality always holds, and the proposition is proven.

Comment. As we can see from the proof, Proposition is valid not only for the triangular functions, but also for any set of membership functions $\mu_i(x)$ for which, for some sequence of values t_i:

- $\mu_i(t_i) = 1$, and
- $\mu_i(x)$ is only different from 0 for $x \in [t_{i-1}, t_{i+1}]$.

Resulting definition of an operation with tuples. Now that we know that our idea enables us to recover the original degrees, we can formalize what it means to perform computations with words.

Definition 2.

- *Let $\mu_i(x)$ be a family of membership functions described by the formula (1),*
- *let $y = f(x_1, \ldots, x_m)$ be a function of n real numbers,*
- *and let $d^{(j)} = (d_1^{(j)}, \ldots, d_n^{(j)})$, $j = 1, \ldots, m$, be tuples.*

By the result $f\left(d^{(1)}, \ldots, d^{(m)}\right)$ of applying the function $f(x_1, \ldots, x_m)$ to the tuples $d^{(1)}, \ldots, d^{(m)}$, *we mean the tuple obtained by using the following three-stage procedure:*

- *first, we use the formula (2) to compute the membership functions $\mu^{(j)}(x_j)$ corresponding to the given tuples*

$$d^{(1)}, \ldots, d^{(m)};$$

- *then, we apply Zadeh's extension principle to the membership functions $\mu^{(j)}(x_j)$, producing a new membership function*

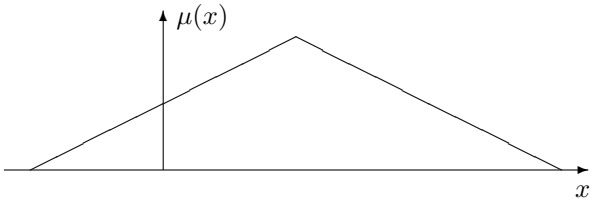
$$\mu(y) = \sup_{x_i : f(x_1, \ldots, x_m) = y} \min\left(\mu^{(1)}(x_1), \ldots, \mu^{(m)}(x_m)\right);$$

- *finally, we use the formulas* (4)–(6) *to transform the resulting membership function* $\mu(y)$ *into a tuple* d.

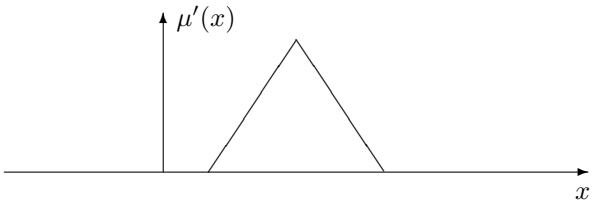
Examples. Let us consider triangular membership functions starting with $s = 0$. Each original word w_i is described by a tuple $d = (d_1, \ldots, d_n)$ in which $d_i = 1$ and $d_j = 0$ for all $j \neq i$.

Example 1. Let us first consider the case when we add two words $w_{i'}$ and $w_{i''}$ corresponding to tuples $(0, \ldots, 0, 1, 0, \ldots, 0)$ (with 1 on the i-th place) and $(0, \ldots, 0, 1, 0, \ldots, 0)$ (with 1 on the i'-th place). In this case, we will show that we get a tuple with $d_{i'+i''} = 1$, $d_{(i'+i'')-1} = d_{(i'+i'')+1} = 0.5$, and $d_j = 0$ for all other j, i.e., a tuple $(0, \ldots, 0, 0.5, 1, 0.5, 0, \ldots, 0)$.

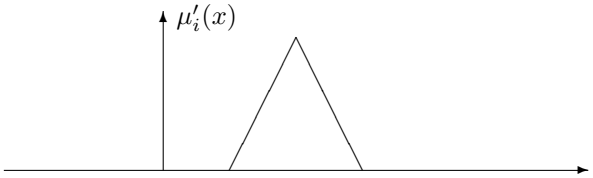
Indeed, here, Zadeh's extension principle leads to the following membership function:

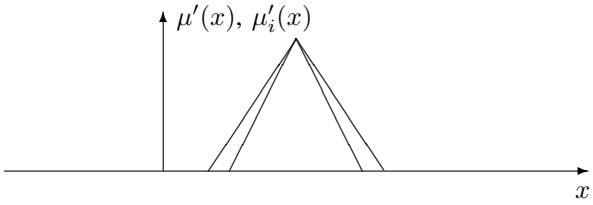


For $i = i' + i''$, the reduced function has the form

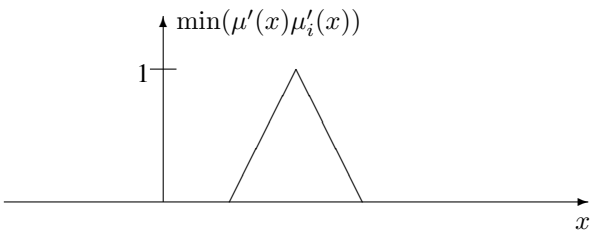


The reduced function $\mu'_i(x)$ has the form:





Here, $\min(\mu'(x), \mu'_i(x)) = \mu'_i(x)$:



Thus, the maximum $d_i = \max\limits_x \min(\mu'(x), \mu'_i(x))$ is equal to 1.

Similarly, for $i = i' + i'' + 1$ and for for $i = i' + i'' - 1$, we get $d_i = \max\limits_x \min(\mu'(x), \mu'_i(x)) = 0.5$, and we get $d_i = 0$ for all other i.

Example 2. For subtracting two words $w_{i'}$ (corresponding to the tuple $(0, \ldots, 0, 1, 0, \ldots, 0)$) and $w_{i''}$ (corresponding to the tuple $(0, \ldots, 0, 1, 0, \ldots, 0)$), we similarly get a tuple with $d_{i'-i''} = 1$ and $d_{(i'-i'')-1} = d_{(i'-i'')+1} = 0.5$, i.e., a tuple $(0, \ldots, 0, 0.5, 1, 0.5, 0, \ldots, 0)$.

Example 3. A shift of a word w_i as described by the tuple $(0, \ldots, 0, 1, \ldots, 0)$, i.e., the result of applying a function $f(x) = x + a \cdot h$ with $0 < a < 1$ to the word w_i, leads to $d_i = 1 - a$ and $d_{i+1} = a$, i.e., to the tuple $(0, \ldots, 0, 1 - a, a, 0, \ldots, 0)$.

Need for extending these results. In all three examples, we get reasonable results. The fact that we get reasonable results for simple examples shows that this approach is worth pursuing. To make this approach useful, we need to come up with similar explicit formulas for the result of applying other functions $f(x_1, \ldots, x_m)$ to tuples.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation grants HRD-0734825 and HRD-1242122 (Cyber-ShARE Center of Excellence) and DUE-0926721, by Grants 1 T36 GM078000-01 and 1R43TR000173-01 from the National Institutes of Health, and by a grant N62909-12-1-7039 from the Office of Naval Research.

The authors are thankful to the anonymous referees for valuable suggestions.

REFERENCES

[1] M. Holčapek and T. Tichý, "A smoothing filter based on fuzzy transform", *Fuzzy Sets and Systems*, 2011, Vol. 180, pp. 69–97.

[2] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic*, Prentice Hall, Upper Saddle River, New Jersey, 1995.

[3] H. T. Nguyen and E. A. Walker, *First Course In Fuzzy Logic*, CRC Press, Boca Raton, Florida, 2006.

[4] V. Novák, M. Štěpnička, A. Dvořák, I. Perfilieva, V. Pavliska, and L. Vavřícková, "Analysis of seasonal time series using fuzzy approach", *International Journal of General Systems*, 2010, Vol. 39, No. 3, pp. 305–328.

[5] V. Novák, M. Štěpnička, I. Perfilieva, and V. Pavliska, "Analysis of periodical time series using soft computing techniques", *Proceedings of the 8th International FLINS Conference on Computational Intelligence in Decision and Control FLINS'2008*, Madrid, Spain, September 21–24, 2008.

[6] I. Perfilieva, "Fuzzy transforms: theory and applications", *Fuzzy Sets and Systems*, 2006, Vol. 157, pp. 993–1023.

[7] I. Perfilieva, V. Novák, V. Pavliska, A. Dvořák, and M. Štěpnička, "Analysis and prediction of time series using fuzzy transform", *Proceedings of IEEE World Congress on Computational Intelligence WCCI'2008*, 2008, pp. 3875–3879.

[8] L. A. Zadeh, "Fuzzy sets", *Information and Control*, 1965, Vol. 8, pp. 338–353.

[9] L. A. Zadeh, "From computing with numbers to computing with words – from manipulation of measurements to manipulation of perceptions", *Int. J. Appl. Math. Comput. Sci.*, 2002, Vol. 12, No. 3, pp. 307–324.