7-2012

Membership Functions or alpha-Cuts? Algorithmic (Constructivist) Analysis Justifies an Interval Approach

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

Comments:
Technical Report: UTEP-CS-12-22
Short version published in *Proceedings of the International Conference "Mathematical Logics and Applications"*, Yerevan, Armenia, November 1-3, 2012, pp. 70-71; full paper published in *Applied Mathematical Science*, 2013, Vol. 7, No. 5, pp. 217-228.

Membership Functions or α-Cuts? Algorithmic (Constructivist) Analysis Justifies an Interval Approach

Vladik Kreinovich
Department of Computer Science
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
vladik@utep.edu

Abstract

In his pioneering papers, Igor Zaslavsky started an algorithmic (constructivist) analysis of fuzzy logic. In this paper, we extend this analysis to fuzzy mathematics and fuzzy data processing. Specifically, we show that the two mathematically equivalent representations of a fuzzy number – by a membership function and by α-cuts – are *not* algorithmically equivalent, and only the α-cut representation enables us to efficiently process fuzzy data.

1 Need for Fuzzy Logic and Fuzzy Mathematics: Brief Reminder

Need for automating expert knowledge. In many application areas, we actively use expert knowledge: skilled medical doctors know how to diagnose and cure diseases, skilled pilots know how to deal with extreme situations, etc. The big problem with expert knowledge is that there are only a few top experts, and it is not realistically possible to use them every time. For example, there are only a few top experts in heart diseases, and there are millions of patients; similarly, there are a few top pilots, and there are thousands of daily flights. Since we cannot use the top experts in all situations, it is desirable to design automated systems that would incorporate the knowledge of these experts.

Need for fuzzy knowledge. One of the main challenges in incorporating expert knowledge into an automated system is that experts often cannot describe their knowledge in precise terms. Instead, they express a significant part of their knowledge by using imprecise ("fuzzy") words from natural languages such as "small", "slightly", "a little bit", etc. So, to design automated expert systems, we need to translate these words into a language that a computer can understand – i.e., translate this knowledge in precise terms.

Fuzzy logic: truth values. Techniques for translating imprecise (fuzzy) knowledge into precise formulas were pioneered by Lotfi Zadeh who called them *fuzzy logic*; see, e.g., [4, 11]. The main idea is that in contrast to well-defined properties (like $x < 1.0$) which are either true or not for every real value x, a property like "x is small" is fuzzy: for very small values x, everyone would agree that x is small; for large values x, everyone agrees that these values are not small; however, for intermediate values x, some experts may consider them small, and some not. A reasonable way

to describe, for each x, its "degree of smallness" is, e.g., to ask N experts. If n of these experts think that x is small, we can take the ratio n/N as the degree that x is small.

Even a single expert may not be sure whether a given number x is small. In such situations, we can ask an expert to describe his or her degree of certainty on a scale from, e.g., 0 to 10. If an expert marks his/her certainty by a mark n on a scale from 0 to N, it makes sense to take the ratio n/N as the degree to which x is small.

In all these cases, for each statement instead of two possible values "true" and "false" – which are, in a computer, usually represented by 1 and 0 – we have a whole range of possible values from the interval $[0, 1]$. These degrees are called *degree of confidence*, or *truth values* $d(S)$ of the corresponding statements S.

Fuzzy logic: operations with truth values. Expert statements often use propositional connectives like "and", "or": e.g., a medical doctor may say that if a tumor is small *and* grows slowly, then conservative methods should be applied. In the classical two-valued logic, once we know the truth values of two statements S and S', we can *uniquely* determine the truth values of propositional combinations like $S \,\&\, S'$ and $S \vee S'$. In the fuzzy case, we no longer have this uniqueness: e.g., if half of the experts believe in the statement S, i.e., if $d(S) = 0.5$, then half of the experts believe in its negation $\neg S$: $d(\neg S) = d(S) = 0.5$. Here, $S \,\&\, S$ means the same as S, so $d(S \,\&\, S) = d(S) = 0.5$. On the other hand, the conjunction $S \,\&\, \neg S$ is always false, so $d(S \,\&\, \neg S) = 0$. Here, $d(S) = d(\neg S)) = 0.5$, but $d(S \,\&\, S) \neq d(S \,\&\, \neg S)$.

Because of this non-uniqueness, ideally, to fully capture the expert knowledge, we should elicit, from the experts, not only the degree of confidence of the basic statement $S_1, \ldots, S_n$, but also the degree of confidence in all possible propositional combinations like $S_1 \,\&\, \neg S_2 \,\&\, \ldots$

The problem with this approach is that there are exponentially many (2^n) such combinations. A knowledge base may contain hundreds and thousands of statement. For $n \approx 10^2$, for $n \approx 10^3$, it is not possible to elicit 2^n degrees from experts. Thus, instead of eliciting degree of composite statements from experts, we need to estimate these degrees based on the truth values of the original statements.

For example, we must be able, knowing the degrees of confidence $a = d(A)$ and $b = d(B)$ in statements A and B, to estimate our degree of confidence in a statement $A \,\&\, B$. This estimate depends only on a and b, and it is usually denoted by $f_{\&}(a, b)$. This function $f_{\&}$ is sometimes called an *and-operation*. It is easy to describe reasonable properties of and-operations. For example, since the statements $A \,\&\, B$ and $B \,\&\, A$ are equivalent, it makes sense to require that the corresponding estimates $f_{\&}(a, b)$ and $f_{\&}(b, a)$ coincide, i.e., that the and-operation is *commutative*: $f_{\&}(a, b) = f_{\&}(b, a)$. Similarly, since the statements $A \,\&\, (B \,\&\, C)$ and $(A \,\&\, B) \,\&\, C$ are equivalent, it makes sense to require that the and-operation be associative: $f_{\&}(a, f_{\&}(b, c)) = f_{\&}(f_{\&}(a, b), c)$. If our degree of belief in A or B increases, then our degree of belief in $A \,\&\, B$ should also increase – or at least remain the same; thus, we can require that the and-operation be $\leq$-monotonic. Since "A and true" is the same as A, we should have $f_{\&}(a, 1) = a$; since "A and false" is always false, we should have $f_{\&}(a, 0) = 0$.

It also makes sense to require that since $A \,\&\, A$ is the same as A, we should get $f_{\&}(a, a) = a$. These requirements uniquely determine the approximating function $f_{\&}(a, b)$: namely, if $a \leq b$, then, due to monotonicity, we should get $f_{\&}(a, a) \leq f_{\&}(a, b) \leq f_{\&}(a, 1)$. Due to our requirements, we have $f_{\&}(a, a) = a$ and $f_{\&}(a, 1) = a$, thus $a \leq f\&(a, b) \leq a$ and $f_{\&}(a, b) = a$. Similarly, if $a \geq b$, we get $f_{\&}(a, b) = b$. These two cases can be combined into a single formula $f_{\&}(a, b) = \min(a, b)$.

For a similar or-operation $f_{\vee}(a, b)$, similar requirements $f_{\vee}(a, b) = f_{\vee}(b, a)$, $f_{\vee}(a, 0) = a$, monotonicity, and $f_{\vee}(a, a) = a$ lead to $f_{\vee}(a, b) = \max(a, b)$: indeed, if $a \leq b$, then $b = f_{\vee}(0, b) \leq$

$f_\vee(a,b) \le f_\vee(b,b) = b$ hence $f_\vee(a,b) = b$. Similarly, if $a \ge b$, then $f_\vee(a,b) = a$, and in both cases, $f_\vee(a,b) = \max(a,b)$.

Comment. In some applications, it makes sense not to require that $f_\&(a,a) = f_\vee(a,a) = a$. In this case, we get more general and- and or-operations (also known as t-norms and t-conorms). However, in this paper, we will mostly consider the simplest operations min and max; a special section of this paper explains why we use these simplest operations.

From fuzzy logic to fuzzy mathematics and data processing. For each property, the function that maps a real value x into a degree to which the value x satisfies this property is called a *membership function*. Most properties like "small", "medium", etc., are "monotonic" in the following sense: for each of these properties, as the value x increases, the degree increases from 0 to 1 and then decreases back from 1 to 0. Membership functions with this monotonicity property are known as *fuzzy numbers*.

It is known that this "monotonicity" can be equivalently described as a "convexity" condition: if $a \le b \le c$, then $\mu(b) \ge \min(\mu(a), \mu(c))$. Also, usually, we know the lower bounds $\underline{\Delta}$ and $\overline{\Delta}$ that contain all possible values of the quantity. In this case, a fuzzy number can be defined as a function $\mu : \left[\underline{\Delta}, \overline{\Delta}\right] \to [0,1]$ for which $\mu\left(\underline{\Delta}\right) = \mu\left(\overline{\Delta}\right) = 0$, $\max\limits_x \mu(x) = 1$, and if $a \le b \le c$, then $\mu(b) \ge \min(\mu(a), \mu(c))$. We will call such membership functions *c-membership functions* (c for convexity).

How do such fuzzy numbers propagate through data processing? If we have a general data processing algorithm $y = f(x_1, \ldots, x_n)$ that transforms n inputs $x_1, \ldots, x_n$ into an output y, and we only known fuzzy numbers $\mu_1(x_1), \ldots, \mu_n(x_n)$ that describe the inputs, then what can we conclude about the output? For example, if we use Ohm's law $V = I \cdot R$ and we know that the current I is small and the resistance R is medium, what can we conclude about the voltage $V = I \cdot R$?

To answer this question, let us reformulate it in logical terms and use fuzzy logic. For each value y, we are interested in the degree $\mu(y)$ that this value y is possible, i.e., that there exists values $x_1, \ldots, x_n$ such that x_1 is a possible value of the first input, $\ldots$, x_n is a possible value of the n-th input, and $y = f(x_1, \ldots, x_n)$. The degree to which x_1 is a possible value of the first input is $\mu_1(x_1)$, $\ldots$, the degree to which x_n is a possible value of the n-th input is $\mu_n(x_n)$, the degree to which $y = f(x_1, \ldots, x_n)$ is 1 or 0, depending on whether this equality holds or not, and the degree to which x_1 is a possible value, *and* x_2 is a possible value, etc., can be determined by applying the and-operation (min) to these degrees:

$$\min(\mu_1(x_1), \ldots, \mu_n(x_n), y = f(x_1, \ldots, x_n)).$$

The value y is possible if this condition is satisfied for one of the tuples $(x_1, \ldots, x_n)$, i.e., if it is satisfied either for one tuple, *or* for another tuple, etc. Using the or-operation max, we conclude that the degree $\mu(y)$ to which y is possible can be described as

$$\mu(y) = \max_{x_1,\ldots,x_n} \min(\mu_1(x_1), \ldots, \mu_n(x_n), y = f(x_1, \ldots, x_n)).$$

This formula can be simplified if we take into account that when $y \ne f(x_1, \ldots, x_n)$, then the degree to which the equality $y = f(x_1, \ldots, x_n)$ is satisfied is 0, hence

$$\min(\mu_1(x_1), \ldots, \mu_n(x_n), y = f(x_1, \ldots, x_n)) = 0.$$

Thus, when we compute the maximum, it is sufficient to only consider the tuples $(x_1, \ldots, x_n)$ for which $y = f(x_1, \ldots, x_n)$. When we restrict ourselves to only such tuples, we get the following formula:

$$\mu(y) = \max_{x_1,\ldots,x_n : y=f(x_1,\ldots,x_n)} \min(\mu_1(x_1), \ldots, \mu_n(x_n)). \tag{1}$$

This formula was first proposed by L. Zadeh; it is known as *Zadeh's extension principle*.

Fuzzy data processing: computational aspects. Sometimes, the membership function $\mu(y)$ is explicitly computed by using the formula (1). In other cases, this computation is performed by using *α-cuts*, i.e., sets $\{x : \mu(x) \geq \alpha\}$. For a fuzzy number, one can check that each α-cut is an interval.

It is known that if the inputs are fuzzy numbers with continuous membership functions and the data processing algorithm is also continuous, then, for every α, the α-cut $\mathbf{y}(\alpha)$ is equal to the range of the function $f(x_1, \ldots, x_n)$ on the α-cuts of the inputs:

$$\mathbf{y}(\alpha) = f(\mathbf{x}_1(\alpha), \ldots, \mathbf{x}_n(\alpha)), \tag{2}$$

where

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=} \{f(x_1, \ldots, x_n) : x_1 \in \mathbf{x}_1, \ldots, \mathbf{x}_n\}.$$

Computing the range of a given function $f(x_1, \ldots, x_n)$ on given intervals is one of the main problems of *interval computations* (see, e.g., [3, 10]); thus, we can use numerous efficient algorithms developed in interval computations for fuzzy data processing.

2 Membership Functions and α-Cuts: Two Alternative Representations of a Fuzzy Number

Two representations: reminder. As we have mentioned, each imprecise ("fuzzy") property can be described by a *membership function*, i.e., by a mapping μ from real numbers into the interval $[0, 1]$. Alternatively, this same property can be described by α-cuts, i.e., by a function that maps each number α from the interval $[0, 1]$ into an interval $\mathbf{x}(\alpha) = \{x : \mu(x) \geq \alpha\}$.

From the traditional mathematical viewpoint, these two representations are equivalent. Indeed, once we know the membership function, we can determine the α-cuts. Vice versa, if we know all the α-cuts $\mathbf{x}(\alpha)$, then we can reconstruct each value $\mu(x)$ as the largest value α for which $x \in \mathbf{x}(\alpha)$.

Need for an algorithmic approach. Since our main objective is applications, it is desirable to check the algorithmics:

- are the two approaches algorithmically equivalent?
- and if not, which one of them makes fuzzy data processing algorithmic?

These are the questions that we will answer in this paper.

Historical comment. Algorithmic analysis of fuzzy logic was pioneered by Igor D. Zaslavsky and his students; see, e.g., [6, 7, 8, 9, 14] (see also [2]). In this paper, we extend his ideas from fuzzy logic to fuzzy mathematics and fuzzy data processing.

3 Computable Numbers and Functions: Brief Reminder

In this paper, we will use the main ideas and results about computable numbers and functions; see, e.g., [1, 5, 12, 13].

It is reasonable to call a real number computable if we can compute it with any given accuracy. In precise terms, a real number x is called *computable* if there is an algorithm that, given a natural number k, returns a rational number r_k for which $|x - r_k| \le 2^{-k}$.

A function from real numbers to real numbers is computable if for each desired accuracy we know with what accuracy to compute the input, and we also know how, based on this input, we can compute the result. In precise terms, a function $f(x_1, \ldots, x_n)$ defined on a box $\left[\underline{\Delta}_1, \overline{\Delta}_1\right] \times \ldots \times \left[\underline{\Delta}_n, \overline{\Delta}_n\right]$ with computable endpoints $\underline{\Delta}_i$ and $\overline{\Delta}_i$ is called *computable* if there exist two algorithms:

- an algorithm that, given a natural number k, computes a natural number ℓ such that if $|x_i - x_i'| \le 2^{-\ell}$ for all i, then $|f(x_1, \ldots, x_n) - f(x_1', \ldots, x_n')| \le 2^{-k}$; and
- an algorithm that, given n rational numbers $r_1, \ldots, r_k$ and an integer k, returns a a rational number r for which $|r - f(r_1, \ldots, r_n)| \le 2^{-k}$.

Now, we are ready to formulate our first result – that from the algorithmic viewpoint, the membership function and α-cut representations are not equivalent.

4 First Result: Two Representations Are Not Equivalent

Definition 1. *By a* c-membership function*, we mean a tuple consisting of two real numbers* $\underline{\Delta}$ *and* $\overline{\Delta}$ *and a function* $\mu : \left[\underline{\Delta}, \overline{\Delta}\right] \to [0,1]$ *for which* $\mu(\underline{\Delta}) = \mu\left(\overline{\Delta}\right) = 0$, $\max\limits_x \mu(x) = 1$*, and* $a \le b \le c$ *implies that* $\mu(b) \ge \min(\mu(a), \mu(c))$.

Definition 2. *We say that a c-membership function* $\langle \underline{\Delta}, \overline{\Delta}, \mu \rangle$ *is* computable *if both real numbers* $\underline{\Delta}$ *and* $\overline{\Delta}$ *are computable and the function* μ *is computable.*

Definition 3. *By a* family of α-cuts (or simply *α-cuts*, for short) corresponding to a c-membership function μ, we mean a pair of mappings $\underline{x} : [0,1] \to \mathbb{R}$ and $\overline{x} : [0,1] \to \mathbb{R}$ for which, for every $\alpha \in [0,1]$, we have $\{x : \mu(x) \ge \alpha\} = [\underline{x}(\alpha), \overline{x}(\alpha)]$.

Definition 4. *We say that α-cuts are* computable *if both mapping* $\underline{x}$ *and* $\overline{x}$ *are computable.*

Proposition 1. *There exists a computable c-membership function for which the corresponding α-cuts are not computable.*

Proposition 2. *There exist computable α-cuts for which the corresponding c-membership function is not computable.*

Proof of Proposition 1. Let us define the following membership function $\mu(x)$ on the interval $[-2, 2]$: $\mu(-2) = 0$, $\mu(-1) = 1$, $\mu(0) = \mu(1) = 0.5$, $\mu(2) = 0$, and the function $\mu(x)$ is linear on each of four subintervals $[-2, -1]$, $[-1, 0]$, $[0, 1]$, and $[1, 2]$. One can easily check that this is a c-membership function, and that this function is computable. Here, for $\alpha > 0.5$, we have $\overline{x}(\alpha) < 0$, while for $\alpha = 0.5$, we get $\overline{x}(\alpha) = 1$. Thus, the function $\overline{x}$ is discontinuous for $\alpha = 0.5$. Since it is

known that every computable function on an interval is continuous [1, 5, 12, 13], this proves that the corresponding α-cuts are not computable.

Proof of Proposition 2. Let us define the following membership function μ on the interval $[-2,2]$: $\mu(x) = 0.5 \cdot (x+2)$ for $x \in [-2,-1]$, $\mu(x) = 1$ for $x \in [-1,1]$, and $\mu(x) = 0.5 \cdot (2-x)$ for $x \in (1,2]$. One can check that this is a c-membership function. Since this function is discontinuous at $x = -1$ and at $x = 1$, it is not computable. On the other hand, the corresponding α-cuts are computable: for $\alpha \geq 0.5$, we have $\underline{x}(\alpha) = -1$ and $\overline{x}(\alpha) = 1$, while for $\alpha < 0.5$, we have $\underline{x}(\alpha) = -1 - 2\alpha$ and $\overline{x}(\alpha) = 1 + 2\alpha$. Both piece-wise linear functions are clearly computable, so the α-cuts are indeed computable.

5 Second Result: Only α-Cuts Guarantee Algorithmic Fuzzy Data Processing

Since the two representations of fuzzy are not computationally equivalent, it is desirable to analyze which of them leads to an algorithmic fuzzy data processing. Here are the results of this analysis: fuzzy data processing is computable for α-cuts but, in general, not computable for membership functions.

Definition 5. *Let $\mu_1, \ldots, \mu_n$ be membership functions, and let $f(x_1, \ldots, x_n)$ be a function. By the* result *of applying f to fuzzy sets $\mu_1, \ldots, \mu_n$, we mean a membership function defined by the formula (1).*

Proposition 3. *There exists a computable c-membership function $\mu_1(x_1)$ and a computable function $f(x_1)$ for which the result μ of applying f to μ_1 is not computable.*

Proposition 4. *There exists an algorithm that, given n computable families of α-cuts corresponding to the membership functions $\mu_1, \ldots, \mu_n$ and a computable function $f(x_1, \ldots, x_n)$, returns computable α-cuts for the result μ of applying f to $\mu_1, \ldots, \mu_n$.*

Proof of Proposition 3. Let us take a computable c-membership function $\mu_1(x_1) = 1 - |x_1|$ on the interval $[-1,1]$, and a piece-wise linear computable function $f : [-1,1] \to [-1,1]$ for which $f(-1) = -1$, $f(0) = f(0.5) = 0$, and $f(1) = 1$. In other words, $f(x_1) = x_1$ for $x_1 \in [-1,0]$, $f(x_1) = 0$ for $x_1 \in [0,0.5]$, and $f(x_1) = 2x_1 - 1$ for $x_1 \in [0.5,1]$. Then, for the result μ of applying the function $f(x_1)$ the membership function $\mu_1(x)$, due to the formula (1), we have $\mu(0) = 1$, but for $y > 0$, the condition $f(x_1) = y$ leads to $2x_1 - 1 = y$, hence $x_1 = (y+1)/2$, and $\mu(y) = \mu_1(x_1) = 1 - (y+1)/2 = 0.5 - y/2$. When $y > 0$ and $y \to 0$, we get $\mu(y) \to 0.5 \neq 1$. Thus, the resulting membership function is discontinuous at $y = 0$ and is, thus, not computable.

Proof of Proposition 4. To compute the α-cut $[\underline{y}(\alpha), \overline{y}(\alpha)] = \mathbf{y}(\alpha)$, we can use the formula (2), according to which $\underline{y}(\alpha)$ is the minimum of the computable function $f(x_1, \ldots, x_n)$ on a computable box $[\underline{x}_1(\alpha), \overline{x}_1(\alpha)] \times \ldots \times [\underline{x}_1(\alpha), \overline{x}_1(\alpha)]$, and $\overline{y}(\alpha)$ is the maximum of the function $f(x_1, \ldots, x_n)$ on this box. It is known [1, 5, 12, 13] that we can algorithmically compute both the minimum and the maximum of a computable function on a computable box, so the α-cuts are indeed computable.

6 Auxiliary Result: Why min and Not Any Other And-Operation

Idea. We want all the property to satisfy the "convexity" condition, that if $a \le b \le c$, then $\mu(b) \ge \min(\mu(a), \mu(c))$. Sometimes, we know that the actual value x satisfies *two* properties S' and S'' characterized by membership functions $\mu'(x)$ and $\mu''(x)$; then, the degree $\mu(x)$ to which a real number x is consistent with this information can be described as $\mu(x) = f_{\&}(\mu'(x), \mu''(x))$. It is reasonable to require that this combined property should also be "convex" (in the above sense). It turns out that the only and-operation which preserves convexity is $f_{\&}(a, b) = \min(a, b)$.

To prove this result, we do not need to use all the properties of an and-operation; it turns out that it is sufficient to require that $f(a, 1) = f(1, a) = a$ and that this operation is $\le$-monotonic in each of the variables, i.e., $a \le a'$ and $b \le b'$ implies $f(a, b) \le f(a', b')$.

Let us describe our result in precise terms.

Definition 6. *A function $\mu : \mathbb{R} \to [0, 1]$ is called* f-convex *if $a \le b \le c$ implies that $\mu(b) \ge \min(\mu(a), \mu(c))$.*

Definition 7. *By a* generalized and-operation*, we mean a function $f : [0, 1] \times [0, 1] \to [0, 1]$ which satisfies the following two properties:*

- *for all a, a', b, and b', if $a \le a'$ and $b \le b'$, then $f(a, b) \le f(a', b')$ (monotonicity);*
- *for all a, we have $f(a, 1) = f(1, a) = a$.*

Proposition 5. *Let $f(a, b)$ be a generalized and-operation. Then, the following two conditions are equivalent to each other:*

- *for every two f-convex functions $\mu'(x)$ and $\mu''(x)$, the function $\mu(x) = f(\mu'(x), \mu''(x))$ is also f-convex;*
- *$f(a, b) = \min(a, b)$.*

Proof. Let us first prove that if $f(a, b) = \min(a, b)$, then the function $\mu(x) = f(\mu'(x), \mu''(x))$ is f-convex. Indeed, if $a \le b \le c$, then, since both functions $\mu'(x)$ and $\mu''(x)$ are f-convex, we get $\mu'(b) \ge \min(\mu'(a), \mu'(c))$ and $\mu''(b) \ge \min(\mu''(a), \mu''(c))$. Thus, the smallest of the left-hand sides is larger than or equal that the smallest of the right-hand sides:

$$\min(\mu'(b), \mu''(b)) \ge \min(\mu'(a), \mu'(c), \mu''(a), \mu''(c)).$$

The left-hand side of this new inequality is $\mu(b)$, and its right-hand side can be rewritten as $\min(\min(\mu'(a), \mu''(a)), \min(\mu'(c), \mu''(c))$, i.e., as $\min(\mu(a), \mu(c))$. Thus, indeed $\mu(b) \ge \min(\mu(a), \mu(c))$.

Vice versa, let us assume that for some generalized and-operation $f(a, b)$, the function $\mu(x) = f(\mu'(x), \mu''(x))$ is always f-convex, let us then prove that for all a and b, we have $f(a, b) = \min(a, b)$. Indeed:

- Let us take a function $\mu'(x)$ which is equal to a for $x \le 0$, to 1 for $x \ge 1$, and is linear for $0 \le x \le 1$, i.e., has the form $\mu'(x) = a + x \cdot (1 - a)$ on this interval $[0, 1]$. One can easily check that this function is f-convex.
- Similarly, let us take a function $\mu''(x)$ which is equal to 1 for $x \le -1$, to b for $x \ge 0$, and which is linear on the interval $[-1, 0]$, i.e., has the form $\mu''(x) = b + |x| \cdot (1 - b)$ on this interval. This function is also f-convex.

For the function $\mu(x) = f(\mu'(x), \mu''(x))$, we get $\mu(-1) = f(a, 1) = a$, $\mu(1) = f(b, 1) = b$, and $\mu(0) = f(a, b)$. Since the function $\mu(x)$ is f-convex, we get $\mu(0) \geq \min(\mu(-1), \mu(1))$, i.e., $f(a, b) \geq \min(a, b)$.

On the other hand, due to monotonicity, we have $f(a, b) \leq f(a, 1) = a$ and $f(a, b) \leq f(1, b) = b$, so $f(a, b) \leq \min(a, b)$. From $f(a, b) \geq \min(a, b)$ and $f(a, b) \leq \min(a, b)$, we conclude that $f(a, b) = \min(a, b)$. The proposition is proven.

Comment. The selection of max as an or-operation is even easier to explain. In general, there are infinitely many triples $(x_1, \ldots, x_n)$ for which $f(x_1, \ldots, x_n) = y$. Thus, in general, we need to apply the or-operation to infinitely many terms. The classification of all possible or-operations (t-conorms) is known [4, 11], and for all operations except for $\max(a, b)$, infinite application to non-zero values leads to a meaningless degree 1 (or other values a for which $a = f_\vee(a, a)$).

Acknowledgments

This work was supported in part by the National Science Foundation grants HRD-0734825 (Cyber-ShARE Center of Excellence) and DUE-0926721, by Grant 1 T36 GM078000-01 from the National Institutes of Health, and by a grant on F-transforms from the Office of Naval Research.

References

[1] E. Bishop and D. S. Bridges, *Constructive Analysis*, Springer, N.Y., 1985.

[2] M. Gehrke, V. Kreinovich, and B. Bouchon-Meunier, "Propositional fuzzy logics: decidable for some (algebraic) operators, undecidable for more complicated ones", *International Journal of Intelligent Systems*, 1999, Vol. 14, No. 9, pp. 935-947.

[3] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer-Verlag, London, 2001.

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

[5] B. A. Kushner, *Lectures on Constructive Mathematical Analysis*, Amer. Math. Soc., Providence, Rhode Island, 1984.

[6] S. N. Manukian, "On some properties of recursively enumerable fuzzy sets", *Proceedings of the Conference "Computer Science and Information Technologies" CSIT'99*, Yerevan, Armenia, August 1999, pp. 5–6.

[7] S. N. Manukian, "Algorithmic operators on recursively enumerable fuzzy sets", *Proceedings of the Conference "Computer Science and Information Technologies" CSIT'01* (September, 2001), Yerevan, Armenia, September 2001, pp. 125–126.

[8] S. N. Manukian, "On binary recursively enumerable fuzzy sets", *Abstracts of the International Conference "21st Days of Weak Arithmetics"*, St. Petersburg, Russia, June 2002, pp. 13–15.

[9] S. N. Manukian, "Algebras of recursively enumerable sets and their applications to fuzzy logic", *J. Math. Sci.*, 2005, Vol. 130, No. 2, pp. 4598–4606.

[10] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*, SIAM Press, Philadelphia, Pennsylvania, 2009.

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

[12] M. Pour-El and J. Richards, *Computability in Analysis and Physics*, Springer-Verlag, New York, 1989.

[13] K. Weihrauch, *Computable Analysis*, Springer-Verlag, Berlin, 2000.

[14] I. D. Zaslavsky, "Fuzzy constructive logic", In: *Studies in constructive mathematics and mathematical logic. Part XI, Zap. Nauchn. Sem. POMI*, 2008, Vol. 358, pp. 130–152; English translation in *Journal of Mathematical Sciences*, 2009, Vol. 158, No. 5, pp. 677–688.