3-2001

How to Make Sure that "~100" + 1 is ~100 in Fuzzy Arithmetic: Solution and its (Inevitable) Drawbacks

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

Hung T. Nguyen

Witold Pedrycz

Comments:
UTEP-CS-01-08.
Published in the *Proceedings of the Joint 9th World Congress of the International Fuzzy Systems Association and 20th International Conference of the North American Fuzzy Information Processing Society IFSA/NAFIPS 2001*, Vancouver, Canada, July 25-28, 2001, pp. 1653-1658.

How to Make Sure That "≈ 100" + 1 Is ≈ 100 in Fuzzy Arithmetic: Solution and Its (Inevitable) Drawbacks

Vladik Kreinovich
Comp. Science, Univ. of Texas at El Paso
El Paso, TX 79968, USA
vladik@cs.utep.edu

Hung T. Nguyen
Mathem. Sciences, New Mexico State Univ.
Las Cruces, NM 88003, USA
hunguyen@nmsu.edu

Witold Pedrycz
Electrical & Computer Engineering, University of Alberta
Edmonton T6G 2G7 Canada
pedrycz@ee.ualberta.ca

Abstract

From the commonsense viewpoint, if a person who weighs around 100 kilograms gains one more kilogram, his weight is still around 100 kilograms. Alas, not so in traditional fuzzy arithmetic. In this paper, we propose a modification of fuzzy arithmetic which does have this property. We gain the desired property, but there is no free lunch, we have to lose two important properties of the traditional fuzzy arithmetic: first, addition is no longer always associative; second, addition is no longer always easily computable.

1. Introduction

1.1. Intuitive Property of Commonsense Arithmetic

To explain the problem that we try to solve in this paper, let us start with a joke. A museum guide tells the visitors that a dinosaur that they are looking at is 14,000,005 years old. An impressed visitor asks how scientists can be so accurate in its predictions. "I don't know how they do it, – explains the guide – but 5 years ago, when I started working here, I was told that this dinosaur is 14,000,000 years old, so now it must be 5 years older".

This is clearly a joke, because from the common sense viewpoint, a dinosaur which was approximately 14,000,000 years old 5 years ago is still 14,000,000 years old. In more precise terms, if we add 5 to a "fuzzy" number "approximately 14,000,000", we should get the answer "approximately 14,000,000".

Similarly, if a person weighs, say, approximately 100 kg, and he gains 1 kg, he still weighs approximately 100 kg. So, if we add 1 to a "fuzzy" number "approximately 100", we should get the answer "approximately 100".

In general, if a is much larger than b ($a \gg b$), and we add b to "approximately a", we should get "approximately a". It is therefore natural to expect formal systems which formalize commonsense reasoning to have this property.

1.2. Fuzzy Arithmetic: A Natural Formalization of Commonsense Arithmetic

A natural way of dealing with approximately known values (such as "approximately a") is *fuzzy arithmetic*. In fuzzy arithmetic, each such value is represented by a membership function $\mu(x)$ describing, for each real number x, to what extent x matches the description (see, e.g., [10, 14]).

For example, if the value that we want to formalize is "approximately a" (for some given real number a), then the value $x = a$ matches the described property perfectly well ($\mu(a) = 1$), while the more distant the value x from a, the smaller the degree of matching. In other words, a natural way to represent a property "approximately a" is to have a membership function $\mu(x)$ which:

- attains its maximum value 1 for $x = a$,
- increase for $x < a$, and
- decreases for $x > a$.

In practical applications, researchers have used membership functions $\mu(x)$ of different shape to represent the property "approximately a": Gaussian, piece-wise linear, etc.; all these shapes have a clear maximum at $x = a$.

Vice versa, if we have a membership function $\mu(x)$ which:

- has a clear maximum at some point $x = a$,
- is increasing for $x < a$, and
- is decreasing for $x > a$,

it is natural to interpret this function as describing a property "approximately a".

When several numbers A, B, etc., are described by membership functions, we can use the extension principle to describe the result of applying an arithmetic operation to these numbers. For example, if a number A is described by a membership function $\mu_A(x)$, and the number B is described by a membership function $\mu_B(x)$, then their sum $C = A + B$ is described by the following membership function:

$$\mu_C(x) = \max_{y,z:y+z=x} \min(\mu_A(y), \mu_B(z)). \qquad (1)$$

We can also have a more general formula, if we use an arbitrary t-norm instead of the minimum.

Whether we use min or a more general t-norm, in the simple case when the number B is crisp ($B = b$), the resulting membership function is equal to $\mu_C(x) = \mu_A(x - b)$; in other words, it has the same shape as the membership function for A – but it is shifted by b.

1.3. Problem: Traditional Fuzzy Arithmetic Does Not Have the Desired Property

In many practical applications, the traditional fuzzy arithmetic works well. Unfortunately, the traditional fuzzy arithmetic does not satisfy the desired intuitive property.

Indeed, let A mean "approximately a" (e.g., "approximately 100"). Then, the corresponding membership function $\mu_A(x)$ has a maximum at $x = a$, is increasing for $x < a$ and decreasing for $x > a$. When we add, to A, a crisp number $B = b$ (e.g., 1), we get a shifted membership function which has a maximum at $x = a + b$, is increasing for $x < a + b$ and decreasing for $x > a + b$. In accordance with the above interpretation, we thus interpret the sum $A + B$ as "approximately $a + b$". Thus, the sum "≈ 100"+1 is equal not to ≈ 100 as we would intuitively expect, but to ≈ 101.

How can we modify fuzzy arithmetic to make sure that the desired property is satisfied, and the sum of "≈ 100" and 1 is equal to ≈ 100?

2. Solution: Main Idea and Its Formalization

2.1. Main Idea

When we only know a (crisp of fuzzy) interval of possible values of a certain quantity (or a more general *set* of possible values), it is desirable to characterize this interval by supplying the user with the "simplest" element from this interval, and by characterizing how far away from this value we can get. For example, if, for some unknown physical quantity x, measurements result in the interval $[1.95, 2.1]$ of possible values, then, most probably, the physicist will publish this result as $y \approx 2$. Similarly, a natural representation of the measurement result $x \in [3.141592, 3.141593]$ is $x \approx \pi$.

So, intuitively, if we know the membership functions for A and for B, we should:

- compute the membership function $\mu_C(x)$ for $C = A + B$;
- find the interval of possible values of C (e.g., as all the values for which $\mu_C(x) \geq d_0$ for some value d_0);
- pick the simplest value c on this interval, and then
- return "approximately c" as the result of adding A and B.

In particular, when A is "approximately 14,000,000" – meaning that the interval of possible values is probably [13,500,000; 14,500,000] – and B is a crisp value 5, then for $A + B$, the interval of possible values is [13,500,005; 14,500,005]. On this interval, 14,000,000 is probably still the simplest value, so we conclude that the sum of "approximately 14,000,000" and 5 is – as we expected – equal to "approximately 14,000,000".

Similarly, in this new definition, if we add 1 kg to a weight of approximately 100 kg, we still get approximately 100 kg as the result.

2.2. How to Formalize This Definition?

In order to formalize the above definition, we must formalize what "simplest" means. Intuitively, the simpler the description of a real number, the simpler this number. Thus, to define relative complexity of different real numbers, we fix some logical theory T in which we will describe real numbers.

We will consider languages in which the list of sorts $\mathcal{S}$ contains two symbols: "integer" and "real", and which contain standard arithmetic predicates and function symbols such as 0, 1, $+$, $-$, $\cdot$, $/$, $=$, $<$, $\leq$, both

for integers and for reals. We will assume that this theory contains both the standard first order theory of integers (Peano arithmetic [1, 8, 16]) and a standard first order theory of real numbers [3, 7, 17, 18]. One of the possibilities is to consider, as the theory T, axiomatic set theory (e.g., ZF), together with explicit definitions of integers, real numbers, and standard operations and predicates in terms of set theory.

Once a theory T is fixed, we can define a *complexity* $D(x)$ of a real number x as the shortest length of a formula $F(y)$ in the language L which defines this particular number x, i.e., which is true for $y = x$ and false for $y \neq x$.

To clarify this definition, let us give examples of formulas which define different real numbers:

- A formula $(y \cdot y = 1 + 1)\,\&\,y \geq 0$ is true if and only if $y = \sqrt{2}$; thus, this formula defines the number $\sqrt{2}$.
- Similarly, a formula $\forall x\,(x \cdot y = x + x + x)$ defines a real number 3.
- If the language of the theory T contains the sine function $\sin$, and if the corresponding theory contains the standard definition of the sine function, then the formula $\sin(y) = 0\,\&\,3 \leq y \leq 4$ defines a real number π.

Comment 1. This definition is similar to the so-called *Kolmogorov complexity* $C(x)$ (invented independently by Chaitin, Kolmogorov, and Solomonoff), which is defined as the smallest length of the program that *computes* x (for a current survey on Kolmogorov complexity, see, e.g., [12]). In our case, however, we do not care that much about how to compute: computing 3.141592 may be easier than computing π; we are more interested in how easy it is to *describe* x. Due to this difference, we cannot simply use the original Kolmogorov's definition: we have to modify it.

Comment 2. It is worth mentioning that not all real numbers are definable: indeed, there are only countably many formulas, so there can be no more than countably many definable real numbers, while the total cardinality of the set of all real numbers is known to be larger ($\aleph_1 > \aleph_0$).

This new definition solves the above problem, but – in full accordance with the saying "there is no free lunch" – it comes with drawbacks. We will see that these drawbacks do not mean that our solution is bad, they seem to be implied (surprisingly) by the very properties that we try to retain.

3. First Drawback: Addition is No Longer Always Associative

This drawback is the easiest to describe and to explain. Both standard arithmetic and traditional fuzzy arithmetic are *associative*: if we add several numbers $A_1 + \ldots + A_n$, the resulting sum does not depend on the order in which we add them; in particular,

$$(\ldots((A_1 + A_2) + A_3) + \ldots) + A_n =$$

$$A_1 + (A_2 + (A_3 + (\ldots + A_n)\ldots)). \qquad (2)$$

Let us show that for the newly defined addition, this formula is no longer always true.

Indeed, suppose now that we want to formalize the idea that, say "$\approx$ 100" + 1 is equal to $\approx$ 100 (this is just an example, but any other example can be used to illustrate non-associativity). Let us take $n = 101$, "approximately 100" as A_1, and $A_2 = \ldots = A_n = 1$ (crisp numbers). In terms of the newly defined numbers A_i, the desired property takes the form $A_1 + A_2 = A_1$ (similarly, $A_1 + A_3 = A_1$, etc.). Thus, $A_1 + A_2 = A_1$, hence $(A_1 + A_2) + A_3 = A_1 + A_3 = A_1$, etc., and hence the left-hand side of the formula (2) is equal to "approximately 100":

$$(\ldots((A_1 + A_2) + A_3) + \ldots) + A_n = A_1.$$

On the other hand, since $A_2, \ldots, A_n$ are crisp numbers (equal to 1 each), their sum $A_2 + (A_3 + (\ldots + A_n)\ldots)$ is simply a crisp number $1 + \ldots + 1 = 100$. Thus, the right-hand side of the formula (2) is equal to

"approximately 100" + 100

which, intuitively, should be rather "approximately 200" than "approximately 100". Thus, the left-hand side of (2) is clearly different from its right-hand side. Hence, the newly defined addition is not associative.

4. Second Drawback: Addition Is No Longer Always Easily Computable

Traditional fuzzy arithmetic – defined by the extension principle – provides an explicit formula for computing the sum $C = A + B$ of two fuzzy numbers A and B. So, we can still find the interval of possible values for C. Unfortunately, as we will now show, the next step – finding the simplest possible real number on this interval – is no longer easily computable.

Theorem 1. *No algorithm is possible that, given an interval with definable endpoints, would return the simplest real number from this interval.*

Proof. This proof is similar to proofs from [9]. Let us prove the desired impossibility of an algorithm by reduction to a contradiction.

In this proof, for simplicity, we will identify each definable real number with the property which defines this number. Let $F(y)$ be a definable real number which is *not* the simplest possible real number. Let us assume that there exists an algorithm U that, given any other definable real number $F'(y)$:

- chooses the simplest representative s from the corresponding interval ($[F(y), F'(y)]$ or $[F'(y), F(y)]$) ; and
- if this simplest representative coincides with one of the endpoints, returns $-$ or $+$ depending on whether s is the left or the right endpoint.

The fact that $F(y)$ is not the simplest possible number means that there exist other definable real numbers whose complexity is smaller than $D(F)$, i.e., that are defined by formulas shorter than $D(F)$.

Since for every length l, there are only finitely many formulas of this length, these formulas can only define finitely many different numbers. Thus, for every length l, there exist finitely many definable real numbers of complexity l. Hence, there exist finitely many definable real numbers that are simpler than $F(y)$. From these numbers, let us pick the formula $G(y)$ for which the number defined by it is the closest to the number $F(y)$ (if there are two such numbers, let us pick the one that is greater than the number defined by the formula $F(y)$).

Without loss of generality, we can assume that the number x_F defined by the formula $F(y)$ is smaller that the number x_G defined by the formula $G(y)$ (the case $x_G < x_F$ can be considered similarly). Now, let $f(n)$ be any algorithmic function from natural numbers to natural numbers. It is known that every algorithmic sequence is definable in Peano arithmetic, and therefore, since out theory T includes Peano arithmetic, $f(n)$ is definable in T as well.

For every such function, we can define a new definable number z_f as follows:

- If $\forall n (f(n) = 0)$, then $z_f = x_G$.
- If $\exists n (f(n) \neq 0)$, then

$$z_f = x_G - 2^{-n_{\min}} \cdot (x_G - x_F),$$

where $n_{\min}$ is the smallest natural number n for which $f(n) \neq 0$.

(We have used words to define z_f, but this definition can be easily reformulated in terms of formulas, so, the number z_f is indeed definable.)

For each function f, it is easy to see which element from the interval $[x_F, z_f]$ is the simplest:

- If $\exists n (f(n) \neq 0)$, then $x_F < z_f < x_G$. Since we have chosen x_G as the closest of all definable real numbers that are simpler than x_F, and since all the elements of the semi-open interval $(x_F, z_f]$ are closer to x_F than x_G, we can conclude that none of the real numbers from the interval $(x_F, z_f]$ is simpler than x_F. Thus, x_F is the simplest of all real numbers from the interval $[x_F, z_f]$.
- If $\forall n (f(n) = 0)$, then $z_f = x_G$. Since we have chosen x_G as the closest of all definable real numbers that are simpler than x_F, and since all the elements of the open interval (x_F, x_G) are closer to x_F than x_G, we can conclude that none of the real numbers from the open interval (x_F, x_G) is simpler than x_F. Thus, x_G is the simplest of all real numbers from the interval $[x_F, x_G] = [x_F, z_f]$.

In both cases, the simplest element coincides with one of the endpoints, so, the algorithm U will return either $-$ or $+$:

- If $\exists n (f(n) \neq 0)$, then the lower endpoint (x_F) is the simplest, and hence, the algorithm U will return $-$.
- If $\forall n (f(n) = 0)$, then the upper endpoint (z_f) is the simplest, and hence, the algorithm U will return $+$.

Thus, by checking whether the sign returned by the algorithm U is $-$ or $+$, we will be able to check, for a given computable function f, whether $\forall n (f(n) = 0)$ is true or not.

However, it is known (see, e.g., [11, 13, 15]) that there exists *no* algorithm for deciding whether a program (to be more precise, a program that always finishes its computations) always returns 0. In other words, there exists no algorithm, that, given an algorithmic (everywhere defined) function $f(n)$ from natural numbers to natural numbers would check whether $\forall n (f(n) = 0)$. This contradiction shows that our initial assumption — that the problem of choosing the representative from an interval is algorithmically solvable — is false. Hence, this problem is not algorithmically solvable. The theorem is proven.

5. A Similar Result Holds for Computable Real Numbers

A similar result holds if we restrict ourselves to *computable* real numbers, i.e., real numbers that can be

computed with an arbitrary accuracy (see, e.g., [2, 4, 5, 6]). To be more precise, a real number x is called computable if there exists an algorithm (program) that transforms an arbitrary integer k into a rational number x_k that is $2^{-k}-$close to x. It is said that this algorithm *computes* the real number x.

Every computable real number is uniquely determined by the corresponding algorithm and is, therefore, definable.

Theorem 2. *No algorithm is possible that, given an interval with computable endpoints, returns the simplest computable real number from this interval.*

Proof. If x is not the simplest possible computable real number, then we can use the same construction as in the proof of Theorem 1. To complete the proof, we must now prove only the following two additional statements:

- First, we need to prove that z_f is a computable real number (and that, given a program f, we can construct a program (algorithm) for computing z_f).
- Second, in our definition, we no longer require the algorithm to return $-$ or $+$. Therefore, to complete the proof, we must show that if an algorithm returns a computable real number s that is equal to one of the endpoints (i.e., to x or to z_f), then we can algorithmically check whether this computable real number coincides with the left endpoint or with the right endpoint.

Both statements are (relatively) easy to prove:

- To compute z_f with an accuracy $(z-x)\cdot 2^{-k}$, it is sufficient to compute first k values of f, and take:
  - If $\forall n_{\le k}(f(n)=0)$, then $a_k = z$.
  - If $\exists n_{n\le k}(f(n)\neq 0)$, then

    $$a_k = z - 2^{-n_{\min}}\cdot(z-x),$$

    where $n_{\min}$ is the smallest natural number $n \le k$ for which $f(n)\neq 0$.

  Then, as one can easily see,

  $$|a_k - z_f| \le 2^{-k}\cdot|z_f - x| \le 2^{-k}\cdot|z-x|.$$

  From these values a_k, we can easily compute the desired rational approximations z_{fk} to z_f.
- If an algorithm returns a computable real number s that coincides with one of the computable endpoints of the interval $[x, z_f]$, then, by computing x, z_f, and s with sufficient accuracy (namely, with accuracy $\varepsilon < (z_f - x)/4$), and comparing the corresponding rational numbers, we will be able to check whether $s = x$ or $s = z_f$. Indeed, in this case, from $|s_k - s| \le \varepsilon$, and $|z_{fk} - z_f| \le \varepsilon$, we can conclude that

  $$|z_{fk} - s_k| \ge |z_f - s| - |s_k - s| - |z_{fk} - z_f| >$$

  $$|z_f - s| - 2\cdot(1/4)\cdot|z_f - s| > (1/2)\cdot|z_f - x|.$$

  Hence:
  - If $s = x$, then, similarly,

    $$|s_k - z_{fk}| > (1/2)\cdot|z_f - x|.$$

    On the other hand, in this case,

    $$|s_k - x_k| \le |s_k - s| + |x_k - x| \le$$

    $$2\varepsilon < (1/2)\cdot(z_f - x).$$

    Therefore, in this case, $|s_k - x_k| < |s_k - z_{fk}|$.
  - Similarly, if $s = z_f$, then

    $$|s_k - x_k| > |s_k - z_{fk}|.$$

  Thus, comparing two rational numbers $|s_k - x_k|$ and $|s_k - z_{fk}|$, we can tell with which of the endpoints s coincides.

The theorem is proven.

6. Conclusion

From the commonsense viewpoint, if 5 years ago, a dinosaur was approximately 14,000,000 years old, it is still approximately 14,000,000 years years old. Unfortunately, when we formalize the notion "approximately 14,000,000" in traditional fuzzy arithmetic, we do not get this property. In this paper, we have described a natural modification of fuzzy arithmetic which does have this property. This modification is closer to commonsense reasoning, but this closeness comes at a cost: addition is no longer always associative and no longer always easily computable.

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCC5-209, by Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-00-1-0365, and by Grant No. W-00016 from the U.S.-Czech Science and Technology Joint Fund.

References

[1] J. Barwise (ed.), *Handbook of Mathematical Logic*. North-Holland, Amsterdam, 1977.

[2] M.J. Beeson, *Foundations of computable mathematics*, Springer-Verlag, N.Y., 1985.

[3] N. Ben-Or, D. Kozen, and J. Reif, "The complexity of elementary algebra and geometry", *Journal of Computer and System Sciences*, 1986, Vol. 32, pp. 251–264.

[4] E. Bishop, *Foundations of Computable Analysis*, McGraw-Hill, 1967.

[5] E. Bishop, D.S. Bridges, *Computable Analysis*, Springer, N.Y., 1985.

[6] D.S. Bridges, *Computable Functional Analysis*, Pitman, London, 1979.

[7] J. Canny, "Improved algorithms for sign determination and existential quantifier elimination", *The Computer Journal*, 1993, Vol. 36, No. 5, pp. 409–418.

[8] H.B. Enderton. *A mathematical introduction to logic*. Academic Press, N.Y., 1972.

[9] G. Heindl, V. Kreinovich, and M. Rifqi, "In case of interval (or more general) uncertainty, no algorithm can choose the simplest representative", Reliable Computing, 2001 (to appear).

[10] G. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall, Upper Saddle River, NJ, 1995.

[11] L.R. Lewis and C.H. Papadimitriou, *Elements of the theory of computation*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[12] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, N.Y., 1997.

[13] J.C. Martin, *Introduction to languages and the theory of computation*, McGraw-Hill, N.Y., 1991.

[14] H.T. Nguyen and E.A. Walker, *First Course in Fuzzy Logic*, CRC Press, Boca Raton, FL, 1999.

[15] C.H. Papadimitriou, *Computational Complexity*, Addison Wesley, San Diego, 1994.

[16] J.R. Schoenfield. *Mathematical logic*. Addison-Wesley, 1967.

[17] A. Seidenberg, "A new decision method for elementary algebra", *Annals of Math.*, 1954, Vol. 60, pp. 365–374.

[18] A. Tarski, *A Decision Method for Elementary Algebra and Geometry*, University of California Press, Berkeley, 1948.