20

Granularity via Non-Deterministic Computations

Vladik Kreinovich[1] and Bernadette Bouchon-Meunier[2]

[1]Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
email vladik@cs.utep.edu

[2]LAFORIA-IBP
University Paris VI, Case 169
4, place Jussieu
75252, Paris Ceex 05, France
email bouchon@laforia.ibp.fr

Abstract

We humans usually think in words; to represent our opinion about, e.g., the size of an object, it is sufficient to pick one of the few (say, five) words used to describe size ("tiny", "small", "medium", etc.). Indicating which of 5 words we have chosen takes 3 bits. However, in the modern computer representations of uncertainty, real numbers are used to represent this "fuzziness". A real number takes 10 times more memory to store, and therefore, processing a real number takes 10 times longer than it should. Therefore, for the computers to reach the ability of a human brain, Zadeh proposed to represent and process uncertainty in the computer by storing and processing the very words that humans use, without translating them into real numbers (he called this idea *granularity*).

If we try to define operations with words, we run into the following problem: e.g., if we define "tiny" + "tiny" as "tiny", then we will have to make a counter-intuitive conclusion that the sum of any number of tiny objects is also tiny. If we define "tiny" + "tiny" as "small", we may be overestimating the size. To overcome this problem, we suggest to use *non-deterministic* (probabilistic) operations with words. For example, in the above case, "tiny" + "tiny" is, with some probability, equal to "tiny", and with some other probability, equal to "small".

1 Granularity is Desirable: The Idea of Zadeh

In [4], L. Zadeh described the following idea:

- We humans usually think in words. When we want to describe the size of an object we use words like "tiny", "small", "medium", "large", "huge". For each quantity (like size, velocity, distance, etc.), there are usually few possible words, and we can describe the value by indicating which of these words best corresponds to the current situation.

  In principle, to represent one of, say, 5 possible values, we need only 3 bits.

- However, traditional methods of representing the uncertain ("fuzzy") expert knowledge use *real numbers* to describe uncertainty. A real number usually requires at least 4 bytes (32 bits). Thus, we use at least 10 times more memory than we should, and when we process these numbers, then, to process all these bits, we use at least 10 times more time than we should.

For the computers to reach the ability of a human brain, we thus need to develop *new* methods of representing and processing uncertainty, methods in which the representation is *granular*, in which the computer stores the words and operates directly with words.

2 The Main Problem of Computing with Words

At first glance, it may seem that computing with words is easy to implement. For example, to describe the expert's opinion about the size, we can use one of the five words given above; to process this data, all we have to do is to describe addition and other operations for these values.

And here comes a problem: For example, what is "tiny" + "tiny"?

- If we assume that "tiny" + "tiny" = "small", then we may be overestimating the resulting size.

- If we assume that "tiny" + "tiny" = "tiny", then we will inevitably conclude that no matter how many tiny objects we add, we will always end up with an object of tiny size. This conclusion is clearly not true, because every object (no matter how huge it is) can be subdivided (at least mentally) into tiny parts, and thus, the sum of tiny parts can be very huge.

So what value should we choose for "tiny" + "tiny"? We may try to resolve this problem by choosing some value intermediate between "tiny" and "small". The addition of this new value solves this problem, but creates similar new problems, for the results of adding more and more terms. We can, of course, solve these

new problems by adding new and new intermediate values, but then eventually, we will lose the very granularity that we are trying to maintain. So, adding a new intermediate value is not a good solution to this problem.

3 Our Solution: Non-Deterministic Operations

Our idea is as follows: since it is difficult to choose a *single, deterministic* value for "tiny" + "tiny", let us make addition (and other operations) *non-deterministic*. In other words, let us assume that with some probability p, "tiny" + "tiny" = "tiny", and with the remaining probability $1-p$, "tiny" + "tiny" = "small".

It is easy to implement probabilistic operations on modern computers, so this type of granularity is easy to implement: when, e.g., inputs are "tiny" and "tiny", we use a standard computer random number generator to generate "tiny" with probability p and "small" with probability $1-p$.

4 Formal Description

We start with a finite set of words $w_1, \ldots, w_n$. To define an arbitrary operation $\odot$, we must, for each $i \le n$ and $j \le n$, define the probabilities p_{ij}^k that the result of applying this operation to w_i and w_j will be equal to w_k. We will describe this probabilistic operation as

$$w_i \odot w_j = p_{ij}^1 w_1 \oplus \ldots \oplus p_{ij}^n w_n. \tag{1}$$

The total probability of getting some result must be equal to 1:

$$\sum_k p_{ij}^k = 1. \tag{2}$$

5 Examples

5.1 Example 1: "large" and "small"

Let us first give a simple example of a single positive quantity with two possible values: "small" (w_s) and "large" (w_l). In this case, we can take an arbitrary $p \in [0,1]$ and define the addition as follows:

$$w_l + w_l = w_l; \tag{3a}$$

$$w_l + w_s = w_s + w_l = w_l; \tag{3b}$$

$$w_s + w_s = p w_s \oplus (1-p) w_l. \tag{3c}$$

One can easily check that this operation is associative.

5.2 Example 2: "positive", "negative", and "close to 0"

As a more complicated example, let us take the set of *three* values w_- (meaning "negative"), w_+ (meaning "positive"), and w_0 (meaning something like "close to 0", or "sign unknown"). The sum of two positive numbers is positive, and the sum of two negative numbers is negative. Hence, we have

$$w_+ + w_+ = w_+; \tag{4a}$$

$$w_- + w_- = w_-. \tag{4b}$$

As for the other sums, symmetry requirements lead to the following formulas:

$$w_- + w_+ = w_+ + w_- = \alpha w_- \oplus \alpha w_+ \oplus (1 - 2\alpha) w_0; \tag{4c}$$

$$w_- + w_0 = w_0 + w_- = \beta w_- \oplus \gamma w_+ \oplus (1 - \beta - \gamma) w_0; \tag{4d}$$

$$w_+ + w_0 = w_0 + w_+ = \beta w_+ \oplus \gamma w_- \oplus (1 - \beta - \gamma) w_0; \tag{4e}$$

$$w_0 + w_0 = \delta w_- \oplus \delta w_+ \oplus (1 - 2\delta) w_0. \tag{4f}$$

for some real numbers α, β, γ, and δ.

When $\alpha = \beta = \gamma = \delta = 0$, we get associativity. Formulas corresponding to this case can be easily explained:

- When we know that one of the arguments is positive and the other is negative, then we do not know the sign of the sum, so, $w_+ + w_- = w_0$.
- When we know that one of the arguments is positive, and we do not know the sign of the other argument, then we do not know the sign of the sum, so $w_+ + w_0 = w_0$.
- Similarly, when we know that one of the arguments is negative, and we do not know the sign of the other argument, then we do not know the sign of the sum, so $w_- + w_0 = w_0$.

Comment. One can prove that this case $\alpha = \beta = \gamma = \delta = 0$ is the only case when formulas $(4a - 4f)$ define an associative operation. The proof is given in the [2].

5.3 Example 3: "positive" and "negative"

Let us now consider the example of a single quantity with only two values: positive w_+ and negative w_-, and a single operation: addition. Similarly to Example 2, we have

$$w_+ + w_+ = w_+; \tag{5a}$$

$$w_- + w_- = w_-. \tag{5b}$$

If we require that the addition operation be invariant with respect to changing w_+ and w_-, we get

$$w_+ + w_- = w_- + w_+ = 0.5w_+ \oplus 0.5w_-. \tag{5c}$$

Thus defined operation is not associative: namely,

$$w_+ + (w_+ + w_-) \neq (w_+ + w_+) + w_-. \tag{6}$$

Indeed, the left-hand side of (6) is equal to $w_+ + w_+$ $(= w_+)$ with probability 0.5, and to $w_+ + w_-$ with probability 0.5. In the second case, in half of the cases (i.e., in 25% of the total number of cases), we get w_+, and in half of the cases, we get w_-. Thus, we have w_+ with total probability $0.5 + 0.25 = 0.75$, and w_- with probability 0.25.

However, since the sum $w_+ + w_+$ is equal to w_+, the right-hand side is equal to w_+ and w_- with probability 0.5.

Comment. Summarizing: in some cases, we *cannot have both symmetry and associativity.* So, here is what we lose by using non-deterministic granularity: we either lose symmetry, or associativity.

In [2], we describe a general source of possible associative operations: lattice-ordered finite-dimensional algebras [1, 3].

Acknowledgments. This work was partially supported by NASA Grant No. NAG 9-757. This work was done when V. Kreinovich was an invited professor at Laforia, University Paris VI, Summer 1996.

The authors are thankful to R. R. Yager for the encouragement.

References

[1] L. Fuchs, *Partially Ordered Algebraic Systems*, Pergamon Press, New York, 1963.

[2] V. Kreinovich and B. Bouchon-Meunier, *Granularity via non-deterministic computations: what we gain and what we lose*, University of Texas at El Paso, Department of Computer Science, Technical Report No. UTEP-CS-96-29, August 1996. Available by `ftp` from `cs.utep.edu`, LaTeXfile *tr96-29.tex* in the directory `pub/reports`.

[3] S. Lang, *Algebra*, Addison-Wesley, Reading, MA, 1965.

[4] L. A. Zadeh, *Information Granularity, Fuzzy Logic, and Computing with Words*, unpublished talk at the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'96), Granada, Spain, July 1, 1996.