Bag Language Speeds Up Interval Computations

Daniel E. Cooke, Richard Duran, and Ann Gates

***Abstract*— As applications of interval computations in the real world become more common, it will be desirable that new software performs as quickly and efficiently as possible. We will use related findings to show how such speed and efficiency can be achieved in bag languages.**

I. Formulation of the Main Problem

Interval data are more difficult to process that numerical data, because we have two numbers instead of one, which makes our computational problems even more time-consuming. So, what can we do?

II. Main Idea

Our idea is as follows: if we have an *interval* of possible values, e.g., $[0.75, 0.85]$, this means that we know the value of the corresponding quantity with an accuracy ± 0.05. Therefore, it makes no sense to perform computations with these values with the typical computer accuracy of 2^{-16} or 2^{-32}.

If we store these numbers in fewer bits, and perform operations with fewer bits, then the number of operations with numbers does not change, but the number of operations with *bits* will be thus smaller. Can we use that fact to speed up interval computations?

III. Operations With Long Words (OLWs)

Some operations are running very fast on the computer, much faster than if we were doing them word-by-word. For example, moving a file on a PC, or copying it into another place. Such operations use special Operations with Long Words (OLW) that are hardware supported on the majority of pre-RISC computers (they allow operations with up to 256 or more bytes at a time).

Unfortunately, the potential behind them has not been fully realized. This is primarily due to the fact that the compilers for high level languages did not take advantage of these operations. We will show that within a new paradigm of programming languages: bag languages, these operation can be efficiently used on the language level.

The authors are with Department of Computer Science, University of Texas at El Paso, El Paso, TX 79968. This research was sponsored by the Air Force Office of Scientific Research (AFSC), under contracts F49620-89-C-0074 and F49620-93-1-0152, by NSF grant No. CDA-9015006, and NASA Research Grant No. No. NAG 2-670 Supplement No. 2.

IV. How to Use OLWs

Computational speed on computers with OLW hardware can be improved by reorganizing the data to be computed. The particulars of this are that, rather than grouping bits by the value they are associated, we group them by their place value. As an example, suppose we have n values, each represented by k bits. The data is organized as follows:

The 1st bit of the 1st value, followed by
the 1st bit of the 2nd value, followed by
...
the 1st bit of the nth value, followed by
the 2nd bit of the 1st value, followed by
the 2nd bit of the 2nd value, followed by
...
the 2nd bit of the nth value, followed by
...
...
the ith bit of the 1st value, followed by
the ith bit of the 2nd value, followed by
...
the ith bit of the nth value, followed by
...
...
the kth bit of the 1st value, followed by
the kth bit of the 2nd value, followed by
...
the kth bit of the nth value.

It is possible to vary the precision of data by varying the number k of bits used to represent each number. Reducing the number of bits could give us a larger interval, and increasing the number of bits could give us a smaller, more precise interval.

If this precision could be specified, we could save space by not having to store meaningless data (like the tenth bit of the value of a membership function that is known only with accuracy ± 0.05).

In addition to saving space, bitwise logical operations with long words can be used on the packed data to increase speed of computations over the array. For example, how can we add two arrays? To add two numbers in a binary format, we must add their last bits, then ad the previous ones (taking carry into consideration), etc. Now, that we have all last bits together, we can add the last bits in a single OLW. Similarly, other operations can be done extremely fast.

V. Why Bags?

A *bag* (*multiset*) is a generalization of a set that is used in different areas of computer science. Namely, a *bag* is a collection of elements over some domain. Unlike sets, bags allow *multiple* occurencies of elements. For example $\{a, a, b\}$ is a bag but not a set. Bags are used in computing:

- The main sorting algorithms (see, e.g., [12]) actually sort bags, not sets of data.
- Bags are used to describe Petri nets ([4], [13], [14]): namely, a state of a Petri net at any given moment of time is described by specifying how many tokens are there in each location. So, a state is a bag of locations.
- Bags are not only a good way to describe algorithms, but also a good way to describe specifications (see, e.g., [11]), because unsorted collection of elements with possible repetitions is a frequent example of input. Moreover, other, more complicated data structures can be naturally expressed in bag terms, to an extent that bags have been proposed as a basic data type for a new generation of high-level programming languages [1], [2], [3], [10], [5], [6], [7].

VI. How to Use OLW in Bag Language?

The main reason why bit-by-bit-layer representation of a real array is not possible in standard computer languages is that in these languages, the real number is a basic type. The only basic structure of the bag language BagL is an ordered bag. A fixed-point real number can be represented as a bag of its bits. In this representation, an array is a bag of bags. To get the above-mentioned layer-by-layer representation from the standard one, we must group together all first bits, all second bits, etc. Such an operation is needed in other cases, e.g., if we represent a matrix as a bag of columns, and each column is a bag of elements, then the transposition of this matrix is exactly what we need: first all first elements of all the columns, then all second elements, etc. This operation has therefore been implemented in BagL under the name of *rotation.*

Bit-wise operations (e.g., adding all last bits of an array) can also be implemented by using global operations with bags.

VII. Potential Applications

We have applied bag languages to parallel interval computations [8] and to processing fuzzy data with interval-valued membership functions [9].

References

[1] J.-P. Banâtre, A. Courant, D. Le Métayer, "A parallel machine for multiset transformation and its programming style, *Future Generation Computer Systems*, vol. 4, pp. 133–144, 1988.

[2] J.-P. Banâtre, D. Le Métayer, "The GAMMA model and its discipline of programming", *Sci. Comput. Program.*, vol. 15, pp. 55–77, 1990.

[3] J.-P. Banâtre, D. Le Métayer, "Programming by multiset transformation", *Communications of the ACM*, vol. 36, No. 1, pp. 98–111, 1993.

[4] V. Cerf, E. Fernandez, K. Gostelow, and S. Volansky. *Formal Control Flow Properties as a Model of Computation*, Report ENG-7178, Computer Science Department, University of California at Los Angeles, December 1971.

[5] D. Cooke, "Arithmetic over multisets leading to a high level language", *Proceedings of the Computers in Engineering Symposium*, Houston, TX, 1993, pp. 31–36.

[6] D. Cooke, "Possible effects of the next generation programming language on the software process model", *International Journal of Software Engineering and Knowledge Engineering*, 1993.

[7] D. Cooke, "A high level computer language based upon ordered multisets", *IEEE Fifth International Conference on Software Engineering and Knowledge Engineering*, San Francisco, 1993.

[8] D. E. Cooke, "An informal introduction to a high level language with applications to interval mathematics", *Interval Computations*, 1995 (to appear).

[9] D. E. Cooke, R. Duran, and A. Gates, "Bag Language speeds up fuzzy interval computations", In: L. Hall, H. Ying, R. Langari, and J. Yen (eds.), *NAFIPS/IFIS/NASA'94, Proceedings of the First International Joint Conference of The North American Fuzzy Information Processing Society Biannual Conference, The Industrial Fuzzy Control and Intelligent Systems Conference, and The NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic, San Antonio, December 18–21, 1994*, IEEE, Piscataway, NJ, pp. 398–399.

[10] D. E. Cooke, A. Gutierrez, "An introduction to BagL", *IEEE Fourth International Conference on Software Engineering and Knowledge Engineering*, Capri, Italy, 1992, pp. 479-486.

[11] G. Dromey, *Program Derivation. The Development of Programs from Specifications*, Addison-Wesley, Sydney, 1989.

[12] D. Knuth, *The Art of Computer Programming. Seminumerical Algorithms*, Addison-Wesley, Reading, MA, 1969.

[13] J. L. Petersen, "Computation sequence sets", *Journal of Computer and System Sciences*, vol. 13, No. 1, pp. 1–24, 1976.

[14] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Inc., 1981.