The Importance of Algebraic Properties

Lokesh Podipireddy

April 2017

1 Abstract

Many algebraic properties that are foundational to mathematics play a significant role in the world of programming. They reach beyond the world of pure mathematics and a play in scientific computation, parallel programming and cryptography. In this document, we discuss the importance of algebraic structure and properties of inverse element, commutativity and associativity.

2 Introduction

In this paper, we discuss the various properties of commutativity and associativity and its basic definitions and the many uses in simple mathematics. We then extend these properties to the world of computation to see where the properties are being used for benefit or where they are creating a nuisance for programmers and how they are being dealt with.

3 Associativity

Formal Defination: an expression containing two or more occurrences in a row of the same associative operator, the order in which the operations are performed does not matter as long as the sequence of the operands is not changed

Simple Definition: Associativity is a simple mathematical property. Let's say you are adding numbers "1+2+3". When you add these numbers, you can add 1 + 2 first, and use this result and add it to 3 to get 6 (1 + 2) + 3. You can also add 1 to the result of 2 + 3 to get the answer 6 by doing 1 + (2 + 3). Either way you choose to add up those numbers you, get the same result. Now this rule can be applied for multiplication as well but not for division and subtraction.

4 Impact of Associativity in Computation

The property of associativity plays a huge role in the world of computation. In compilers, it allows for optimizations to take place when data dependency is not

an issue. There are some areas that associativity property is a disadvantage. An example of this is in the calculation of floating point numbers. Let's say that you are adding three floating point numbers represented in 4-bit mantissa. Below is an example of this error.

$(1.000_2 \times 2^0 + 1.000_2 \times 2^0) + 1.000_2 \times 2^4 = 1.000_2 2^1 + 1.000_2 \times 2^4 = 1.001_2 \times 2^4$
$(1.000_2 \times 2^0 + 1.000_2 \times 2^0) + 1.000_2 \times 2^4 = 1.000_2 \times 2^1 + 1.000_2 \times 2^4 = 1.001_2 \times 2^4$

As you can see this rounding error can be significant especially when calculating many numbers that has data dependency from previous calculations. One of the ways to minimize this is to use the Kahan Summation algorithm. [1]

5 Commutativity

This is another basic property of mathematical operations. What this tells us is that we are adding three numbers, it doesn't matter in which order we add them, the result is the same. For example, let's say you are adding 1+2+3, the result is the same as adding 3+2+1 or 2+1+3.

6 Commutativity in Computation

Commutativity plays a big role in computation, specifically parrallel computation. The defining trait of Commutativity is the fact that you can rearrange the operands in any order as long as all the binary operators in between these operands is the same.

Now lets say you have a sum calculation for the population of a country. Now in order to calculate the population of a total country you add up the population of all the cities. Since these are all addition operations you know that they are commutative. now Let's say you have n threads calculating the population of each city for n cities. Now because these operations are commutative the order doesn't matter. The sum of the country calculation can accept whichever thread is done calculating their cities population. This means that the total sum calculation doesn't have to wait for a specific thread to be done before another thread is done.[2]

7 Conclusion

In conclusion, associativity and commutativity allow for more freedom in terms of evaluation but this freedom can also create problems when dealing with parallelism. Associativity can also create problems when dealing with floating point representations in 4-bit mantissa.

References

[1] Donald Knuth. *The Art of Computer Programming, Volume 3.*

[2] M C. Rinard. *Commutativity Analysis: A New Analysis Framework for Parallelizing Compilers.* University of California, Santa Barbara, 1996.