

Numeral Systems in the λ -Calculus

(Abstract)

Ian Mackie

We investigate numeral systems in the λ -calculus; specifically in the linear λ -calculus where copying and erasing are not permitted. Our interest is in finding efficient (and where possible, in constant time) representations for successor, predecessor, addition, subtraction and test for zero. We begin by recalling some well-known systems, before going on to the linear case where we give several systems with different properties. We conclude with a characterisation of linear numeral systems.

Summary

Representing data in the λ -calculus is as old as the calculus itself, and numbers are the most basic example. Church numerals are probably the best known example, but there are many others. We are interested in taking a new look at representing numbers in the λ -calculus, together with the usual operations: successor, predecessor, addition, subtraction and a test for zero. Our focus is on representing numbers efficiently: specifically, we are interested in the following questions:

- Can we define numeral systems in the linear λ -calculus—a proper subset of the λ -calculus where copy and erase operations are not possible? In this constrained system we investigate how to represent numbers and the usual operations over these as linear functions.
- Can we obtain efficient representations and computations—constant time operations?

In this talk we recall some results in this area, and add some new ideas. Specifically, we define several representations, and give a characterisation of linear numeral systems. Representing the calculus as a system of interaction nets will also give a useful graphical interpretation, as well as an implementation of linear β -reduction as a constant time operation.

One aspect that we will particularly focus on is subtraction. No matter which representation is chosen, one of the problems of representing numerals in the λ -calculus is that subtraction is not naturally captured. Specifically, numbers are like lists in some way, so:

- addition is like concatenation of lists, and it is well-known that constant time append is possible.
- subtraction is like a split operation, and the unwanted part of the number must be erased. This will never be a constant time operation.

Therefore, subtraction is always going to be a more expensive operation than addition. Nevertheless, it is possible to design data structures such as difference lists that can help, but this just postpones the extra work until we need to read out the answer. In this talk we give a solution to this, and indicate some directions for further study.