Abstract and Concrete Models of Recursion

Martin Hyland
University of Cambridge, UK

The idea of recursion is essential to programming. For example a subroutine may be called many times in the running of a programme: each time it is called it does the same thing (though hopefully to different initial data). The idea that recursion amounts to repeating the same thing over and over is familiar enough from a range of computing practice. These lectures will introduce abstract mathematical ways to think about this basic phenomenon, and discuss applications.

The focus of the treatment of recursion will be the idea of what is called a trace on a (symmetric monoidal) category. Variants on this idea capture notions of feedback. The force of the lectures will be that the abstract notions introduced to analyse recursion can themselves be given a concrete diagrammatic representation. I shall start by turning that on its head, and shall explain how to regard flow diagrams from basic imperative programming in terms of a notion of trace.

I shall go on to explain connections with other topics in theoretical computer science: finite automata, the simulation of recursion in the lambda calculus, fixed points, game semantics. I hope to finish by saying something about the relation between feedback as trace and other abstract ideas related to the idea of recursion: I shall discuss abstract treatments of algebraic data types, streams, relational trees and so on.

The stress throughout will be on examples; and I hope by use of examples to convey to those with little background in abstract mathematics. I shall draw lots of pictures.

There is nothing in the literature which provides a good preparation for what I shall say, and I shall try to do without prerequisites. However it would be of value to have some idea of the mathematical notion of a category. Mac Lane's book, see [3], remains a standard reference: one might try to read the elementary material on monoidal categories (chapter VII). The book by Barr and Wells, see [1], focuses on the needs of computer scientists. A third edition and much electronic material can be found on the web. The book by Hasegawa, see [2], provides an introduction to the notion of trace in a computer science context. There is a little overlap with the proposed lectures. Unfortunately the book is out of print, but some information can be extracted from the author's home page.

References

1. M. Barr and C. Wells. *Category Theory for Computing Science*, Prentice-Hall, 1990.
2. M. Hasegawa. *Models of Sharing Graphs: A Categorial Semantics of let and letrec*, Springer-Verlag, 1999.
3. S. Mac Lane, *Categories for the Working Mathematician*, Springer-Verlag, 1971.