Exploiting Unique Fixed Points

Ralf Hinze

Computing Laboratory, University of Oxford
Wolfson Building, Parks Road, Oxford, OX1 3QD, England
ralf.hinze@comlab.ox.ac.uk
http://www.comlab.ox.ac.uk/ralf.hinze/

Abstract. Functional programmers happily use equational reasoning and induction to prove properties of recursive programs. To show properties of corecursive programs they employ coinduction, per perhaps less enthusiastically. Coinduction is often considered as a rather low-level proof method, especially, as it seems to depart rather radically from equational reasoning. In this talk we introduce an alternative proof technique based on unique fixed points. To make the idea concrete, consider the simplest example of a coinductive type: the type of streams, where a stream is an infinite sequence of elements. In a lazy functional language, such as Haskell, streams are easy to define and many textbooks on Haskell reproduce the folklore examples of Fibonacci or Hamming numbers defined by recursion equations over streams. One has to be a bit careful in formulating a recursion equation basically avoiding that the sequence defined swallows its own tail. However, if this care is exercised, the equation even possesses a unique solution, a fact that is not very widely appreciated. Uniqueness can be exploited to prove that two streams are equal: if they satisfy the same recursion equation, then they are! We will use this proof technique to infer some intriguing facts about particular streams and to develop the basics of finite calculus. Quite attractively, the resulting proofs have a strong equational flavour. In a nutshell, the proof method brings equational reasoning to the coworld. Of course, it is by no means restricted to streams and can be used equally well to prove properties of infinite trees or the observational equivalence of instances of an abstract datatype.