Functional Programming with Bananas, Lenses, Envelopes and Barbed Wire

Erik Meijer * Maarten Fokkinga † Ross Paterson ‡

Abstract

We develop a calculus for lazy functional programming based on recursion operators associated with data type definitions. For these operators we derive various algebraic laws that are useful in deriving and manipulating programs. We shall show that all example functions in Bird and Wadler's "Introduction to Functional Programming" can be expressed using these operators.

1 Introduction

Among the many styles and methodologies for the construction of computer programs the Squiggol style in our opinion deserves attention from the functional programming community. The overall goal of Squiggol is to *calculate* programs from their specification in the way a mathematician calculates solutions to differential equations, or uses arithmetic to solve numerical problems.

It is not hard to state, prove and use laws for well-known operations such as addition, multiplication and —at the function level— composition. It is, however, quite hard to state, prove and use laws for arbitrarily recursively defined functions, mainly because it is difficult to refer to the recursion scheme in isolation. The algorithmic structure is obscured by using unstructured recursive definitions. We crack this problem by treating various recursion schemes as separate higher order functions, giving each a notation of its own independent of the ingredients with which it constitutes a recursively defined function.

*University of Nijmegen, Department of Informatics, Toernooiveld 6525 ED Nijmegen, e-mail: erik@cs.kun.nl

†CWI, Amsterdam & University of Twente

‡Imperial College, London