The Derivative of a Regular Type is its Type of One-Hole Contexts

Extended Abstract

Conor McBride[*]

Abstract

Polymorphic regular types are tree-like datatypes generated by polynomial type expressions over a set of free variables and closed under least fixed point. The 'equality types' of Core ML can be expressed in this form. Given such a type expression T with x free, this paper shows a way to represent the one-hole contexts *for elements of x within elements of T, together with an operation which will plug an element of x into the hole of such a context. One-hole contexts are given as inhabitants of a regular type $\partial_x T$, computed* generically *from the syntactic structure of T by a mechanism better known as* partial differentiation. *The relevant notion of containment is shown to be appropriately characterized in terms of derivatives and plugging in. The technology is then exploited to give the one-hole contexts for sub-elements of recursive types in a manner similar to Huet's 'zippers'[Hue97].*

1 Introduction

Gérard Huet's delightful paper 'The Zipper' [Hue97] defines a representation of tree-like data decomposed into a subtree of interest and its surroundings. He shows us informally how to equip a datatype with an associated type of 'zippers'—one-hole contexts representing a tree with one subtree deleted. Zippers collect the subtrees forking off step by step from the path which starts at the hole and returns to the root. This type of contexts is thus independent of the particular tree being decomposed or the subtree in the hole. Decomposition is seen not as a kind of *subtraction*, an operation inherently troubled by the need to subtract only smaller things from larger, but as the *inversion* of a kind of *addition* (see figure 1).

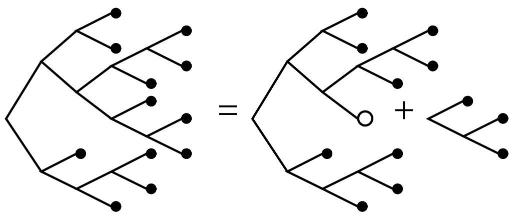

Figure 1: *a tree as a one-hole context 'plus' a subtree*

This paper exhibits a similar technique, defined more formally, which is generic for a large class of tree-like data structures—the 'regular datatypes'. These are essentially the 'equality types' of core ML, presented as polynomial type expressions closed under least fixed point. In particular, I will define and characterize two operations:

- on *types*, computing for each regular recursive datatype its type of one-hole contexts;
- on *terms*, computing a 'big' term by plugging a 'small' term into a one-hole context.

The first of these operations is given by recursion on the structure of the algebraic expressions which 'name' types. As I wrote down the rules corresponding to the empty type, the unit type, sums and products, I was astonished to find that Leibniz had beaten me to them by several centuries: they were exactly the rules of differentiation I had learned as a child.

[*]Department of Computer Science, University of Durham, `c.t.mcbride@durham.ac.uk`