

Title:

On Fold, the Ultimate Higher Order Function in Algebra of Programming

Student; Faculty:

Jonathan Gallagher; Dr. Siochi

Abstract

In this paper we showcase certainly one of the most profound and elegant approaches to deriving programs: the Algebra of Programming. We start by illustrating the tools behind the algebra: calculation and structural induction. Immediately we delve into an examination of higher-order structure. Higher-order proofs and higher-order programs are the tools that make algebraic programming possible. Functional programs are defined by a Cartesian Closed Category; functions exist between any tupling of appropriate types. A higher-order proof is much like a proof that can prove specific goals; similarly, higher-order programs are programs that make other programs given some constraints. We develop the fold function and prove its inductive soundness. The fold function is type dependent and defines a homomorphism between inductive types such as lists, trees, natural numbers, etc. Homomorphisms are structure preserving functions. We take fold and use it to derive two of the most powerful list functions there are: map and filter. We also develop the Ackermann function with only fold; therefore, we show fold is stronger than primitive recursion. This paper then serves a strong argument in favor of algebraic development techniques; a small amount of hard theoretical work has myriad practical applications.