The Felix Language

John Skaller

December 21, 2021

Contents

Chapter 1

Introduction

The Felix programming language is based on algebra, in particular category theory.

The construction begins with a category $\mathcal{M}$ which is a cartesian closed category of monomorphic types, including a set of primitive types p_i, primitive functions f_i and the usual type combinators for products, coproducts, function types, and recursion.

Our objective is to construct a category $\mathcal{P}$ of polymorphic types. Initially, the objects of this category will be functors $\mathcal{P}^n \to \mathcal{P}$. Let $F : \mathcal{M}^m \to M$ and $G : \mathcal{M}^m \to M^n$ then application of the composite can be given by

$$F(G_1(t_1, t_2, ..t_m), G_2(t_1, ...t_m), ...G_n(t_1, ..t_m))$$

where G is split into component functors. in other words, compostion is just substitution, as is reduction of applications.

Let $\mathcal{K}$ be the category with objects $\mathcal{P}^i$ for finite natural numbers i, and arrows all the functors between them, then let $\mathcal{P}$ be the category with objects these functors, and arrows the natural transformnations of $\mathcal{K}$. In other words, the objects are polymorphic data types, and the arrows polymorphic functions.