Foundations of Haskell

Vaibhav Pujari

January 22, 2020

Contents

1 Lambda calculus and Category theory

- Everything is a function
- There is no error reporting - not that practical because there is no help from compiler
- If we introduce error checking and types, we deviate from lambda calculus because not every expression is correct. But we also gain a lot of practical value
- This then becomes a category - because in category composition is not arbitrary, it is constrained by morphisms

2 The language

2.1 Data types

```
x :: Int
x = 5
```

2.2 Function types

Types are sets and in category of sets, a function is a member of Hom-set between two objects, so its a member of some other object in the same category (every set is an object)

2.3 Partial functions

- Recursion can create programs that never terminate
- Well-founded recursion - compiler can prove that it terminates. But its a theoretical concept because for every algorithm that claims to find out if a recursion is well-founded or not, there always exists a particual recursion that breaks the algorithm
- Therefore, in practice we need partial functions. A partial function gives result when it terminates, but there is no guarantee that it will terminate

2.4 Composition

```
(.) :: (b->c) -> (a->b) -> a -> c
(g . f) x = g ( f x )
```

Polymorphic types and higher order functions are foundational in haskell as opposed to a later added feature in most common languages

2.5 Functors

- A structure-preserving mapping between two categories
  - To preserve objects, we just need a type constructor

  F: Ob(Hask) $\to$ Ob(Hask)

  - To preserve morphism structure, we need `fmap`. `fmap` maps morphism in first category to a morphism in second category such that it is still composable in the same way as original morphism was composable

  fmap: $\text{Hom}_{\text{Hask}} \to \text{Hom}_{\text{Hask}}$
- Shrinking the structure is a special case of preserving. When objects collapse, the morphisms must follow

- There is a category `Cat`, in which objects are smaller categories and morphisms are functors