Notes from "Haskell Programming from First Principles"

Ryan Batubara

February 25, 2025

1 All You Need is Lambda

1.1 Lambda Terms

Expressions include concrete values, variables, and functions. **Variables** have no meaning or value; they are just names for inputs. **Functions** are relations between a set of possible inputs with **purity** or **referential transparency**, meaning the same inputs always give the same output.

An **abstraction** is just a function. It has two parts: the **head** is a λ followed by a variable name; the **body** is another expression. The variable in the head is the **parameter** and **binds** all instances of that variable in the body of the function. This makes the paramater reserved for the body, but nowhere else.

1.2 Computing Lambda Terms

Syaing that a function is equivalent up to renaming is called **alpha equivalence**. The process of binding variables and removing heads is called **beta reduction**. The syntax **[x := z]** indicates z will be substituted for x in the body. **Beta normal form** is when you cannot beta reduce terms any further.

Free variables are variables in the body not named in the head. A combinator is a lambda term with no free variables; they combine the arguments given. **Currying** is the idea of nesting lambdas for multiple arguments:

$$\lambda xy.xy = \lambda x.(\lambda y.xy)$$

Reducing terms ordinarily **converges** to beta normal form. That is, it terminates. **Divergence** means the reduction never ends. For example:

$$(\lambda x.xx)(\lambda x.xx)$$

This is important because divergent programs don't produce an **answer**, i.e. a meaningful result.

2 Hello, Haskell!

2.1 Getting Started

GHCi provides a REPL environment for Haskell. Functions in Haskell consist of a name, parameters separated by whitespace, an equals sign, and then an expression. In general. modules and types are capitalized camel-case; functions lowercase camel-back; variable lowercase.

Index