NOTES ON LAMBDA CALCULUS

VAIBHAV KARVE

These notes were last updated September 9, 2018. They are notes taken from my reading of *Haskell Programming from First Principles* by *Chris Allen, Julie Moronuki*.

1. Basics and definitions

(1) Lambda calculus has three basic components or *lambda terms* – expressions, variables and abstractions.

(2) *Expressions* are variable names, abstractions, or combinations of other expression. *Variables* have no meaning or value, they are only names for potential inputs to functions. An *abstraction* is a function – it is a lambda term that has a head (a lambda) and a body and is applied to an argument. An *argument* is an input value.

(3) Abstractions have two parts – a *head* and a *body*. The head of the function is a λ followed by a variable name. The body of the function is another expression. For example: $\lambda x..x^2$

Lambda abstractions are anonymous functions.

(4) The variable named in the head is the *parameter* and *binds* all instances of that same variable in the body of the function. The dot (.) separates the parameters of the lambda from the function body.

2. Equivalences and reductions

(1) *Alpha equivalence* states that $\lambda x..x$ is the same as $\lambda y..y$, that is, the variables x and y are not semantically meaningful except in their role in their single expressions.

(2) *Beta reduction:* when applying a function to an argument, substitute the input expression for all instances of bound variables within the body of the abstraction.

$$\lambda x.x^2\ 3 = 3^2 = 9$$

Hence, Beta reduction is the process of applying a lambda term to an argument, replacing the bound variables with the value of the argument,

and eliminating the head.

$$\begin{aligned} \lambda x.x\ \lambda y.y &= x[x := (\lambda y.y)] \\ &= \lambda y.y \end{aligned}$$

(3)