HW 3 **Prof. Caldwell**
Due: 2 September 2010 **COSC 3015**

1 Equality of functions

Recall from class that functions are equal if and only if they are equal on all inputs (this equality is called extensionality.)

Definition 1.1. (extensionality) If $f, g :: a \to b$ then they are defined to be (extensionally) equal as follows:

$$f = g \stackrel{\text{def}}{=} \forall x\!:\!a.\ f(x) = g(x)$$

So, we can prove two functions f and g are equal by choosing an arbitrary x of type a and showing $f(x) = g(x)$.

For example, if $f(x) = |x|$ (the absolute value) and $g(x) = x$ then, $f \neq g$ when we consider them as functions in the type $\mathbb{Z} \to \mathbb{Z}$ since $f(-2) = 2$ and $g(-2) = -2$. But, if we think of these functions as elements of $\mathbb{N} \to \mathbb{N}$, they are equal. To see this, choose an arbitrary $x \in \mathbb{N}$ and argue that $f(x) = g(x)$ *i.e.* that $|x| = x$. But this is trivially true when $x \geq 0$, which follows because $x \in \mathbb{N}$.

Recall the following Haskell definitions.

Definition 1.2. plus

$$\begin{array}{l} plus :: (Integer, Integer) \to Integer \\ plus(x, y) = x + y \end{array}$$

Definition 1.3. plusc

$$\begin{array}{l} plusc :: Integer \to (Integer \to Integer) \\ plusc\ x\ y = x + y \end{array}$$

Definition 1.4. curry

$$\begin{array}{l} curry :: ((a, b) \to c) \to (a \to (b \to c)) \\ curry\ f\ x\ y = f\ (x, y) \end{array}$$

Definition 1.5. uncurry

$$\begin{array}{l} uncurry :: (a \to (b \to c)) \to ((a, b) \to c) \\ uncurry\ f\ (x, y) = f\ x\ y \end{array}$$

In class we proved the following theorem:

Theorem 1.1.

$$curry\ \ plus = plusc$$

Proof: Note that both *curry plus* and *plusc* have the type $Integer \to (Integer \to Integer)$ *i.e.* they are functions mapping an $Integers$ to a function of type $Integer \to Integer$. This means we can use extensionality to prove they are equal as functions. We must show the following.

$$\forall x : Integer.\ curry\ \ plus\ x = plusc\ x$$

Assume x is an arbitrary $Integer$. Then we must show

$$curry\ \ plus\ x = plusc\ x$$

But $curry\ \ plus\ x$ and $plusc\ x$ are functions of type $Integer \to Integer$. To show they are equal we use extensionality a second time, to show:

$$\forall y : Integer.\ curry\ \ plus\ x\ y = plusc\ x\ y$$

We choose an arbitrary y of type $Integer$ and show the following

$$curry\ \ plus\ x\ y = plusc\ x\ y$$

Starting with the left side of the equality we get the following:

$$curry\ \ plus\ x\ y \stackrel{\langle\langle def.\, of\, curry\rangle\rangle}{=} plus\ (x, y) \stackrel{\langle\langle def.\, of\, plus\rangle\rangle}{=} x + y$$

On the right side of the equality, we have the following:

$$plusc\ x\ y \stackrel{def.\, of\, plusc}{=} x + y$$

Since both sides of the equality are equal to $x + y$ we see that the functions are equal.
□

Problem 1.1. Prove the following theorem using extensionality.

Theorem 1.2. [uncurry-plusc]

$$uncurry\ plusc = plus$$

Hint: The functions ($uncurry\ plusc$) and $plus$ have the type $(Integer, Integer) \to Integer$. Extensionality for functions f and g of this type can most conveniently be written as

$$\forall (x, y) : (Integer, Integer).\ f(x, y) = g(x, y)$$

2 Function Composition

Now, consider the following two definitions.

Definition 2.1. Function Composition

$$\begin{aligned}&compose :: (b \to c) \to (a \to b) \to (a \to c)\\&compose\ f\ g\ x = f(g\ x)\end{aligned}$$

In Haskell, $(compose\ f\ g)$ is written $(f\ .\ g)$, we will write $(f \circ g)$ here.

Definition 2.2. Identity function

$$\begin{aligned}&id :: a \to a\\&id\ x = x\end{aligned}$$

Theorem 2.1. Compose-id-right

$$\forall f : a \to b.\ (f \circ id) = f$$

Proof: Choose an arbitrary function $f :: a \to b$ and show that

$$f \circ id = f$$

Since f has type $a \to b$ and id has type $a \to a$ we can see that $f \circ id$ has the same type. (why?) We use extensionality to show that these two functions are equal, *i.e.* we must show:

$$\forall x : a. (f \circ id)\,x = f\ x$$

Chose an arbitrary x in type a and show

$$(f \circ id)\,x = f\ x$$

By definition of compose and definition of id we get the following sequence of equalities.

$$(f \circ id)x = compose\ f\ id\ x = f\,(id\ x) = f\ x$$

This completes the proof.
□

Problem 2.1. Prove the following theorem.

Theorem 2.2. **[compose-id-left]**

$$\forall f : a \to b.\ id \circ f = f$$

Hint: First argue that $id \circ f$ and f have the same type (note that $id :: b \to b$) and then use extensionality.