COSC 3015 Name: Jacob Tuttle
Functional Programming

# Homework 6
**October 11, 2019**

**1.** $\forall xs : [a].map(\backslash x \to x)\ \ xs = xs$

First, let $f$ stand for $(\backslash x \to x)$. We must then show that both the base case and the induction case hold for the above function in order to prove it.

Base:

$$map\ \ f\ \ [] = [] \quad \{\text{def. map}\}$$

Induction:

$$\begin{aligned} & map\ \ f\ \ x : xs = \\ = & (f\ \ x) : map\ \ f\ \ xs && \{\text{def. map}\} \\ = & x : map\ \ f\ \ xs && \{\text{apply f}\} \\ = & x : xs && \{\text{induction hypothesis}\} \\ = & xs \end{aligned}$$

□

**2.** $\forall xs : [a].map(f.g)\ \ xs = ((map\ \ f).(map\ \ g))\ \ xs$

Base:

$$\begin{aligned} & map(f.g)\ \ [] = [] && \{\text{def. map}\} \\ \hline & ((map\ \ f).(map\ \ g))[] = \\ = & map\ \ f\ \ (map\ \ g\ \ []) && \{\text{def. compose}\} \\ = & map\ \ f\ \ [] && \{\text{def. map}\} \\ = & [] && \{\text{def. map}\} \end{aligned}$$

Induction:

$$\begin{aligned} & map\ \ (f.g)\ \ x : xs = \\ = & (f.g)\ \ x : map\ \ (f.g)\ \ xs && \{\text{def. map}\} \\ = & (f.g)\ \ x : ((map\ \ f).(map\ \ g))\ \ xs && \{\text{induction hypothesis}\} \\ = & (f.g)\ \ x : (map\ \ f\ \ (map\ \ g\ \ xs)) && \{\text{def. compose}\} \\ = & ((map\ \ f).(map\ \ g))\ \ x : xs && \{\text{def. map}\} \end{aligned}$$

□

**3.** $\forall xs : [a].head(reverse\ \ xs) = last\ \ xs$

Base:

$$\begin{aligned} & head(reverse\ \ x : []) \\ = & \ head(reverse\ \ [] + +[x]) && \{\text{def. reverse}\} \\ = & \ head([] + +[x]) && \{\text{def. reverse}\} \\ = & \ head([x]) && \{\text{def. ++}\} \\ = & \ x && \{\text{def. head}\} \end{aligned}$$

Induction[1]:

$$\begin{aligned} & head(reverse\ \ x : xs) = \\ = & \ head((reverse\ \ xs) + +[x]) && \{\text{def. reverse}\} \\ = & \ head(reverse\ \ xs) && \{\text{def. head}^1\} \\ = & \ last\ \ xs && \{\text{induction hypothesis}\} \end{aligned}$$

□

**4.** $\forall xs : [a].last(reverse\ \ xs) = head\ \ xs$

Base:

$$\begin{aligned} & last(reverse\ \ x : []) = \\ = & \ last(revere\ \ [] + +[x]) && \{\text{def. reverse}\} \\ = & \ last([] + +[x]) && \{\text{def. reverse}\} \\ = & \ last([x]) && \{\text{def. ++}\} \\ = & \ x && \{\text{def. last}\} \end{aligned}$$

Induction[2]:

$$\begin{aligned} & last(reverse\ \ x : xs) = \\ = & \ last((reverse\ \ xs) + +[x]) && \{\text{def. reverse}\} \\ = & \ [x] && \{\text{def. last}^2\} \\ = & \ head([x]) && \{\text{def. head}\} \\ = & \ head([xs]) && \{\text{induction hypothesis}\} \end{aligned}$$

□

[1] Since append places the first element at the end of the list it constructs, and head discards everything except the first element, the second list is ignored in step 3 below.

[2] Similar to the above proof, since the function reverse will construct the rest of the list by adding things to the front, the first element peeled of (the head of xs) will be the last element within the reversed list.

**5.** $\forall xs, ys : [a].len(xs + +ys) = (len\ \ xs) + (len\ \ ys)$
Base:

$$
\begin{array}{cc}
len([] + +[]) = & \\
= len\ \ [] & \{\text{def ++}\} \\
= 0 & \{\text{def. len}\} \\
= 0 + 0 & \{\text{0 additive identity}\} \\
= len\ \ [] + len\ \ [] & \{\text{def. len}\} \\
\hline
len\ \ ([] + +xs) = & \\
= len\ \ xs & \{\text{def. ++}\} \\
= len\ \ xs + 0 & \{\text{0 additive identity}\} \\
= len\ \ xs + len\ \ [] & \{\text{def. len}\}
\end{array}
$$

Induction[3]:

$$
\begin{array}{cc}
len(x : xs + +ys) = & \\
= len(x : (xs + +ys)) & \{\text{def. ++}\} \\
= 1 + len(xs + +ys) & \{\text{def. len}\} \\
= 1 + (len\ \ xs + len\ \ ys) & \{\text{induction hypothesis}[3]\}
\end{array}
$$

□

[3]Additionally, the length of a single element is 1, so we can intuitively understand that the length of two lists with a single element consed on would be 1 plus the length of those lists.