HW 9 **Prof. Caldwell**
Due: 1 October 2013 **COSC 3015**

1 Datatypes and Structural Induction

As discussed in class, Haskell datatypes have corresponding structural induction princples.

1.1 Nat

The datatype definition for natural numbers was given as:

datatype Nat = Zero | Succ Nat

The induction principle for the type Nat is

$$(P(Zero) \wedge \forall k :: Nat.\ P(k) \Rightarrow P(Succ\, k)) \Rightarrow \forall n :: Nat.\ P(n)$$

Thus to show a property P holds for every finite natural number, show two things:

Case *Zero:* Show $P(Zero)$.

Case *Succ:* For arbitrary $k :: Nat$ assume $P(k)$ and show $P(Succ\, k)$.

Note, if we want to reason about possibly infinte elements of the type we add the case to show $P(\bot)$. This is true for each of the induction principles below.

1.2 Lists

Recall the defintion of lists containing elements of type a.

datatype List a = Nil | Cons a (List a)

The induction principle for finite lists is

$$(P(Nil) \quad \wedge \quad \forall x :: a.\ \forall xs :: List\ a.\ P(xs) \Rightarrow P(Cons\, x\, xs)) \quad \Rightarrow \forall ys :: List\ a.\ P(ys)$$

Thus to show a property P holds for every list, show two things:

Case *Nil***:** Show $P(Nil)$.

Case *Cons***:** For arbitrary $x :: a$ and $xs :: List\ a$ assume $P(xs)$ and show $P(Cons\ x\ xs)$.

1.3 Binary Trees

Recall the definition of the datatype for binary trees containing elements of type a.

datatype BTree a = Leaf | Node a (BTree a) (BTree a)

The induction principle for finite binay trees is

$$\begin{array}{l}(P(Leaf) \\ \ \wedge\ \forall x :: a.\ \forall t_l, t_r :: BTree\ a.\ (P(t_l) \wedge P(t_r)) \Rightarrow P(Node(v, t_l, t_r))) \\ \quad \Rightarrow \forall t :: BTree\ a.\ P(t)\end{array}$$

Thus to show a property P holds for every $BTree$, show two things:

Case *Leaf***:** Show $P(Leaf)$.

Case *Node***:** For arbitrary $x :: a$ and $t_l, t_r :: BTree\ a$ assume $P(t_l)$ and assume $P(t_r)$ and show $P(Node(v, t_l, t_r))$.

2 Assignment

Write the structural induction principles for the finite instances of the following types.

1. Trees having nodes with both one and with two children.

   $$OneTwoTree\ a = Empty \mid Single\ a\ (OneTwoTree\ a) \mid Branch\ a\ (OneTwoTree\ a)\ (OneTwoTree\ a)$$

2. Trees whose nodes have exactly three children.

   $$ThreeTree\ a = Leaf \mid Node\ a\ (ThreeTree\ a)\ (ThreeTree\ a)\ (ThreeTree\ a)$$

3. A data type for propositional formulas.

   $$Formula = Bottom\ \mid Not\ Formula \mid Implies\ Formula\ Formula$$