10 Variant Type Definitions

1. Yes.

   Proof:

   $\llcorner$add(e1, add(e2, add(e3, empty))) $\equiv$ add(e1, add(e3, add(e2, add(e1,empty))))$\lrcorner$

   unordered:

   $\llcorner$add(e1, add(e2, add(e3, empty))) $\equiv$ add(e1, add(e3, add(e1, add(e2,empty))))$\lrcorner$

   unordered:

   $\llcorner$add(e1, add(e2, add(e3, empty))) $\equiv$ add(e1, add(e1, add(e3, add(e2,empty))))$\lrcorner$

   unordered:

   $\llcorner$add(e1, add(e2, add(e3, empty))) $\equiv$ add(e1, add(e1, add(e2, add(e3,empty))))$\lrcorner$

   no_duplicates:

   $\llcorner$add(e1, add(e2, add(e3, empty))) $\equiv$ add(e1, add(e2, add(e3,empty)))$\lrcorner$

   is_annihilation:

   $\llcorner$**true**$\lrcorner$

2. (a) **type** Figure

   value

   box : **Real** $\times$ **Real** $\rightarrow$ Figure,

   circle : **Real** $\rightarrow$ Figure,

   length : Figure $\stackrel{\sim}{\rightarrow}$ **Real**,

   width : Figure $\stackrel{\sim}{\rightarrow}$ **Real**,

   radius : Figure $\stackrel{\sim}{\rightarrow}$ **Real**

   axiom

   [disjoint] $\forall$ r1, r2, r3 : **Real** $\bullet$ box(r1,r2) $\neq$ circle(r3),

   [induction]

   $\forall$ p : Figure $\rightarrow$ **Bool** $\bullet$

   ($\forall$ r1, r2: **Real** $\bullet$ p(box(r1,r2))) $\wedge$ ($\forall$ r : **Real** $\bullet$ p(circle(r)))

   $\Rightarrow$

   ($\forall$ f : Figure $\bullet$ p(f)),

   [length_box] $\forall$ r1, r2 : **Real** $\bullet$ length(box(r1,r2)) $\equiv$ r1,

   [width_box] $\forall$ r1, r2 : **Real** $\bullet$ width(box(r1,r2)) $\equiv$ r2,

   [radius_circle] $\forall$ r3 : **Real** $\bullet$ radius(circle(r3)) $\equiv$ r3

   (b) **type** Figure

   value

   box : **Real** $\times$ **Real** $\rightarrow$ Figure,

   circle : **Real** $\rightarrow$ Figure,

   length : Figure $\stackrel{\sim}{\rightarrow}$ **Real**,

   width : Figure $\stackrel{\sim}{\rightarrow}$ **Real**,

   radius : Figure $\stackrel{\sim}{\rightarrow}$ **Real**

   axiom

   [disjoint] $\forall$ r1, r2, r3 : **Real** $\bullet$ box(r1,r2) $\neq$ circle(r3),

   [length_box] $\forall$ r1, r2 : **Real** $\bullet$ length(box(r1,r2)) $\equiv$ r1,

   [width_box] $\forall$ r1, r2 : **Real** $\bullet$ width(box(r1,r2)) $\equiv$ r2,

   [radius_circle] $\forall$ r3 : **Real** $\bullet$ radius(circle(r3)) $\equiv$ r3

(c) **type** Figure
value
box : **Real** × **Real** → Figure,
circle : **Real** → Figure,
length : Figure $\tilde{\to}$ **Real**,
width : Figure $\tilde{\to}$ **Real**,
radius : Figure $\tilde{\to}$ **Real**,
base_line : Figure $\tilde{\to}$ **Real**
axiom
[disjoint] ∀ r1, r2, r3 : **Real** • box(r1,r2) ≠ circle(r3),
[length_box] ∀ r1, r2 : **Real** • length(box(r1,r2)) ≡ r1,
[width_box] ∀ r1, r2 : **Real** • width(box(r1,r2)) ≡ r2,
[radius_circle] ∀ r3 : **Real** • radius(circle(r3)) ≡ r3

(d) **type** Tree
value
empty : Tree,
node : Tree × Elem × Tree → Tree,
left : Tree $\tilde{\to}$ Tree,
val : Tree $\tilde{\to}$ Elem,
right : Tree $\tilde{\to}$ Tree
axiom
[disjoint] ∀ t1, t2 : Tree, e : Elem • empty ≠ node(t1, e, t2),
[induction]
∀ p : Tree → **Bool** •
(
p(empty) ∧
(∀ t1, t2 : Tree, e : Elem • p(t1) ∧ p(t2) ⇒ p(node(t1,e,t2)))
) ⇒
(∀ t : Tree • p(t)),
[left_node] ∀ t1, t2 : Tree, e : Elem • left(node(t1, e, t2)) ≡ t1,
[val_node] ∀ t1, t2 : Tree, e : Elem • val(node(t1, e, t2)) ≡ e,
[right_node] ∀ t1, t2 : Tree, e : Elem • right(node(t1, e, t2)) ≡ t2

3. (a) It is true in all models.

   [assumption] t1 $\neq$ t2

   $\llcorner$node(t1,e,t2) $\neq$ node(t2,e,t1)$\lrcorner$
   /* this is true if we can show */
   $\llcorner$left(node(t1,e,t2)) $\neq$ left(node(t2,e,t1))$\lrcorner$
   left_node:
   $\llcorner$t1 $\neq$ t2$\lrcorner$
   assumption:
   $\llcorner$**true**$\lrcorner$
   qed

   (b) It is true in all models.

   [assumption] e1 $\neq$ e2.

   $\llcorner$node(t1,e1,t2) $\neq$ node(t1,e2,t2)$\lrcorner$
   /* this is true if we can show */
   $\llcorner$val(node(t1,e1,t2)) $\neq$ val(node(t1,e2,t2))$\lrcorner$
   val_node:
   $\llcorner$e1 $\neq$ e2$\lrcorner$
   assumption:
   $\llcorner$**true**$\lrcorner$
   qed

   (c) It is true in all models.

   $\llcorner$node(empty,e,empty) $\neq$ node(node(empty,e,empty),e,empty)$\lrcorner$
   /* this is true if we can show */
   $\llcorner$left(node(empty,e,empty)) $\neq$ left(node(node(empty,e,empty),e,empty))$\lrcorner$
   left_node:
   $\llcorner$empty $\neq$ node(empty,e,empty)$\lrcorner$
   disjoint:
   $\llcorner$**true**$\lrcorner$
   qed

4. PEANO =
   class
   type N == zero | succ(N)
   axiom
   [linear_order] $\forall$ n1,n2 : N • (succ(n1) $\equiv$ succ(n2)) $\Rightarrow$ (n1 $\equiv$ n2),
   end

   alternatively

   PEANO =
   class
   type N == zero | succ(pred : N)
   end

5. **scheme**
ORIENTATION =
class
type Orientation == north | south | east | west

value
turnleft : Orientation $\to$ Orientation,
turnright : Orientation $\to$ Orientation,
opposite : Orientation $\to$ Orientation

axiom
[turnleft_north] turnleft(north) $\equiv$ west,

[turnleft_south] turnleft(south) $\equiv$ east,

[turnleft_east] turnleft(east) $\equiv$ north,

[turnleft_west] turnleft(west) $\equiv$ south,

[turnright_turnleft] $\forall$ d : Orientation • turnright(turnleft(d)) $\equiv$ d,

[opposite_ax] $\forall$ d : Orientation • opposite(d) $\equiv$ turnleft(turnleft(d))
end

6.
```
   ⌞∀ v : Orientation • turnleft(v) ≠ v⌟
all_variant_induction :
    ⌞true⌟
  since
    •  ⌞turnleft(north) ≠ north⌟
       turnleft_north :
         ⌞west ≠ north⌟
       /* using disjointness axiom */
       west_north :
         ⌞true⌟
       qed
    •  ⌞turnleft(south) ≠ south⌟
       turnleft_south :
         ⌞east ≠ south⌟
       /* using disjointness axiom */
       east_south :
         ⌞true⌟
       qed
    •  ⌞turnleft(east) ≠ east⌟
       turnleft_east :
         ⌞north ≠ east⌟
       inequality_commutativity :
         ⌞east ≠ north⌟
       /* using disjointness axiom */
       east_north :
         ⌞true⌟
       qed
    •  ⌞turnleft(west) ≠ west⌟
       turnleft_west :
         ⌞south ≠ west⌟
       inequality_commutativity :
         ⌞west ≠ south⌟
       /* using disjointness axiom */
       west_south :
         ⌞true⌟
       qed
  end qed
```

7.
```
scheme STACK ==
  class
    type Elem, Stack == empty | push(top : Elem, pop : Stack)
  end
```