A8: Induction

1) $xs.map(id) = xs$

Base Case

$[].map(id) = []$

Inductive Case

$xs.map(id) = xs$ (by IH)

LHS $\rightarrow (x : xs).map(id) = x : (xs.map(id))$

$= x : xs \leftarrow$ RHS

2) $(xs.append(ys)).map(f) = (xs.map(f)).append(ys.map(f))$

Base case

LHS $([].append(ys)).map(f) = ys.map(f)$

RHS $[].map(f).append(ys.map(f)) = [].append(ys.map(f))$

$= ys.map(f)$

Inductive case

LHS: $((x : xs).append(ys)).map(f) = (x : (xs.append(ys))).map(f)$

$= x.map(f) : (xs.append(ys).map(f))$

(by IH) $= x.map(f) : xs.map(f).append(ys.map(f))$

RHS: $(x : xs).map(f).append(ys.map(f)) = x.map(f) : xs.map(f).append(ys.map(f))$

3) $xs.append(ys).fold(g, a) = xs.fold(g, ys.fold(g, a))$

Base Case

LHS: $[].append(ys).fold(g, a) = ys.fold(g, a)$

RHS: $[].fold(g, ys.fold(g, a)) = ys.fold(g, a)$

Induction Case

LHS: $(x:xs).append(ys).fold(g, a) = x:(xs.append(ys).fold(g, a))$

$= x:(xs.fold(g, ys.fold(g, a)))$ (by IH)

RHS: $(x:xs).fold(g, ys.fold(g, a)) = x:(xs.fold(g, ys.fold(g, a)))$

4) $(xs.append(ys)).length = xs.length() + ys.length$

Base Case

LHS: $([].append(ys)).length() = ys.length()$

RHS: $[].length + ys.length = 0 + ys.length() = ys.length()$

Induction Case

LHS $((x:xs).append(ys)).length() = (x:(xs.append(ys)).length()$

$= 1 + (xs.append(ys)).length()$

$= 1 + xs.length() + ys.length()$

RHS $(x:xs).length() + ys.length() = 1 + xs.length() + ys.length()$

5) (xs.reverseH(ys)).length = xs.length() + ys.length()

Base Case

LHS: ([].reverseH(ys)).length() = ys.length()

RHS: [].length() + ys.length() = 0 + ys.length() = ys.length()

Induction Case

LHS: ((x:xs).reverseH(ys)).length() = x : (xs.reverseH(ys)).length()
= 1 + xs.length() + ys.length() (by IH)

RHS: x:xs.length() + ys.length = 1 + xs.length() + ys.length()

6) xs.length = (xs.reverse()).length()

Base Case

LHS: [].length() = 0

RHS: [].reverse().length() = [].length() = 0

Induction Case

LHS: (x:xs).length() = 1 + xs.length()

RHS: ((x:xs).reverse()).length() = xs.reverse().append(x).length
= xs.reversed().length + x.length()
= xs.length() + x.length() (by IH)
= 1 + xs.length()

7) (xs.append(ys)).reverse() = ys.reverse().append(xs.reverse())

Base Case

LHS: ([].append(ys)).reverse() = ys.reverse()

RHS: ys.reverse().append([].reverse()) = ys.reverse().append([])
= ys.reverse()

Induction Case

LHS: (x:xs).append(ys).reverse() =
(x: xs.append(ys)).reverse() = xs.append(ys).reverse().append(x)
= ys.reverse().append(xs.reverse()).append(x)

RHS: ys.reverse().append((x:xs).reverse()) = ys.reverse().append(xs.reverse().append(x))
= ys.reverse().append(x.reverse()).append(x)

8) (xs.reverse()).reverse() = xs

Base Case

LHS: ([].reverse()).reverse() = [].reverse() = []

RHS: [] = []

Induction Step

LHS: ((x:xs).reverse()).reverse() = (xs.reverse().append(x)).reverse()
= x.reverse().append(xs)
= x.append(xs)
= x:xs

RHS: x:xs = x:xs

9) $xs.reverseH(ys) = (xs.reverse()).append(ys)$

Base Case

LHS: $[].reverseH(ys) = ys$

RHS: $([].reverse()).append(ys) = [].append(ys) = ys$

Induction Case

LHS: $(x:xs).reverseH(ys) = xs.reverseH(x:ys)$

$= xs.reverse().append(x:ys)$ (by IH)

RHS: $((x:xs).reverse()).append(ys) = xs.reverse().append(x).append(ys)$

$= xs.reverse().append(x:ys)$

10) $(xs.reverseH(ys)).reverseH(zs) = ys.reverseH(xs.append(zs))$

Base Case

LHS: $([].reverseH(ys)).reverseH(zs) = ys.reverseH(zs)$

RHS: $ys.reverseH([].append(zs)) = ys.reverseH(zs)$

Induction Case

LHS: $((x:xs).reverseH(ys)).reverseH(zs) = xs.reverseH(x:ys).reverseH(zs))$

$= (x:ys).reverseH(xs.append(zs))$

$= ys.reverseH(x:(xs.append(zs)))$

RHS: $ys.reverseH((x:xs).append(zs)) = ys.reverseH(x:(xs.append(zs)))$

11) $xs.reverseH(ys.append(zs)) = (xs.reverseH(ys)).append(zs)$

Base Case

LHS: $[].reverseH(ys.append(zs)) = ys.append(zs)$

RHS: $([].reverseH(ys)).append(z) = (ys).append(z) = ys.append(zs)$

Induction Case

LHS: $(x:xs).reverseH(ys.append(zs)) = xs.reverseH(x:ys.append(zs))$

$= xs.reverseH((x:ys).append(zs)$

$= (xs.reverseH((x:ys))).append(zs)$ (by IH)

RHS: $(x:xs).reverseH(ys)).append(zs) = xs.reverseH((x:ys))).append(zs)$

12) $(xs.append(ys)).reverseH(zs) = ys.reverseH(xs.reverseH(zs))$

Base Case

LHS: $([].append(ys)).reverseH(zs) = ys.reverseH(zs)$

RHS: $ys.reverseH([].reverseH(zs) = ys.reverseH(zs)$

Induction Case

LHS: $((x:xs).append(ys)).reverseH(zs) = (x:xs.append(ys)).reverseH(zs)$

$= (xs.append(ys)).reverseH(x:zs)$

$= ys.reverseH(xs.reverseH(x:zs))$ (by IH)

RHS: $ys.reverseH((x:xs).reverseH(zs)) = ys.reverseH(xs.reverseH(x:zs))$

13) $(xs.append(ys)).reverse2() = ys.reverse2().append(xs.reverse2())$

Base Case

LHS: $[].append(ys).reverse2() = ys.reverseH([]) =$

RHS: $ys.reverse2().append([].reverse2()) = ys.reverse2().append([].reverseH([]))$

$= ys.reverse2().append([])$

$= ys.reverseH([])$

Induction Case

LHS: $((x:xs).append(ys)).reverse2() = (x:xs.append(ys)).reverseH([])$

$= (xs.append(ys)).reverseH(x)$

RHS: $ys.reverse2().append((x:xs).reverse2()) = ys.reverseH([]).append(xs.reverseH(x))$

$(xs.append(ys)).reverseH(x)$

14) $(xs.reverse2()).reverse2() = xs$

Base Case

$([].reverse2()).reverse2() = ([].reverseH([])).reverseH([]) = [], reverseH([])$

$= []$

$[] = []$

Induction Case

$((x:xs).reverse2()).reverse2() = ((x:xs), reverseH([])).reverseH([])$

$= (xs.reverseH(x)):.reverseH([])$

$(x:xs) = ((x:xs).reverse2()).reverse2()$ (by IH)

$= ((x:xs).reverseH([])).reverseH([]) = (xs.reverseH(x)).reverseH([])$

15) $7.flattenH(xs) = 7.flatten().append(xs)$

Base Case

LHS: $d.flattenH(xs) = d:xs$

RHS: $d.flatten().append(xs) = (d:[]).append(xs) = d:xs$

Induction Case

LHS: $N(d, 7_1, 7_2).flattenH(xs) = 7_1.flattenH(d:7_2.flattenH(xs))$

$= 7_1.flatten().append(d:7_2.flattenH(xs))$ (by IH)

$= 7_1.flatten().append(d:7_2.flatten()).append(xs))$ (by IH)

RHS: $N(d, 7_1, 7_2).flatten().append(xs) = 7_1.flatten().append(d:7_2.flatten()).append(xs)$

16) $7.flatten2() = 7.flatten$

Base Case

LHS: $d.flatten2() = d.flattenH([]) = d:[]$

RHS: $d.flatten() = d:[]$

Induction Case

LHS: $N(d, 7_1, 7_2).flatten2() = N(d, 7_1, 7_2).flattenH([])$

$= 7_1.flattenH(d:7_2.flattenH([]))$

RHS: $N(d, 7_1, 7_2).flatten() = 7_1.flatten().append(d:7_2.flatten())$

17) $t.map(f1).sum() = t.nodes()$

Base Case

LHS: $d.map(f1).sum() = new\ leaf(1).sum() = 1$

RHS: $d.nodes() = 1$

Induction Case

LHS: $N(d, t_1, t_2).map(f1).sum() = N(f1.apply(d), t_1.map(f1), t_2.map(f_1)).sum()$

$= f1.apply(d) + t_1.map(f1).sum() + t_2.map(f1).sum()$

$= f_1.apply(d) + t_1.nodes() + t_2.nodes()$

$= 1 + t_1.nodes() + t_2.nodes$

RHS: $N(d, t_1, t_2).nodes() = 1 + t_1.nodes() + t_2.nodes()$

18) $t.nodes() = t.longestPath.length() + 1$

Base Case

LHS: $d.nodes() = 1$

RHS: $d.longestPath.length() + 1 = (d:[]).length() + 1 = 1 + 1 = 2$) Not possible

19) $t.internalNodes() + 1 = t.leaves()$

Base Case

LHS: $d.internalNodes() + 1 = 0 + 1 = 1$

RHS: $d.leaves() = 1$

Induction Case

LHS: $N(d, t_1, t_2).internalNodes() + 1 = (1 + t_1.internalNodes()) + (t_2.internalNodes() + 1)$

$= t_1.leaves() + t_2.leaves()$ (2x IH)

RHS $N(d, t_1, t_2).leaves() = t_1.leaves() + t_2.leaves()$