A8
Ben Reichert

---

(1) xs.map(id) = xs

```
LHS = [].map(id) = []. (by def)
RHS = [] = LHS

Induction case: xs = (x:xs’)

Assume: xs’.map(id) = xs’

Prove: (x:xs’).map(id) = xs’

LHS = (x:xs’).map(id)
    = (x:xs’.map(id))
    = (x:xs’)

RHS = (x:xs’) = LHS
```

---

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

```
Base Case: xs = []
LHS = ([].append(ys)).map(f)
    = ys.map(f)
RHS = ([].map(f)).append(ys.map(f))
    = [].append(ys.map(f))
    = ys.map(f) = LHS

Induction case:
xs = (x:xs’)

Assume: (xs'.append(ys)).map(f) =
(xs'.map(f)).append(ys.map(f))

Prove: (x:xs'.append(ys)).map(f) =
(x:xs'.map(f)).append(ys.map(f))

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

```
RHS = ((x:xs’).map(f)).append(ys.map(f))
    = (x:xs’.map(f)).append(ys.map(f))
    = (x:xs’).append(ys.map(f))
    = x:(xs’.append(ys)).map(f)
    = LHS
```

---

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

```
Base Case: xs = []
LHS = ([].append(ys)).fold(g, a) = ys.fold(g, a)
RHS = [].fold(g, ys.fold(g, a)) = ys.fold(g, a)

Induction case: xs = (x:xs’)
Assume: (xs'.append(ys)).fold(g, a) = xs'.fold(g,
ys.fold(g, a))

Prove: ((x:xs’).append(ys)).fold(g, a) = (x:xs’).fold(g,
ys.fold(g, a))

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

RHS = (x:xs’).fold(g, ys.fold(g, a))
    = g.apply(x,xs’.fold(g, ys.fold(g, a))) = LHS
```

---

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

```
Base Case: xs = []
LHS = [].append(ys).length() = ys.length()
RHS = [].length() + ys.length() = ys.length() = LHS

Induction step: xs = (x:xs’)
Assume: (xs'.append(ys)).length() = xs'.length() +
ys.length()

Prove: ((x:xs’).append(ys)).length() = (x:xs’).length() +
ys.length()
```

```
(5)
Prove: (xs.reverseH (ys)).length() = xs.length() +
ys.length()

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

Base case: xs=[]

LHS = [].reverseH(ys).length() = ys.length()
RHS = [].length() + ys.length() = ys.length() = LHS

Inductive case: xs = (x:xs')

Assume this is true IH:
  forall ys. xs'.reverseH(ys).length() = xs'.length() +
ys.length()

To prove: (x:xs').reverseH(ys).length() = (x:xs').length()
+ ys.length()

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

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

```
(6)
 Prove: xs.length() = xs.reverse2().length()

    List reverse2 () { return reverseH(new EmptyL()); }

    LHS = xs.length()
    RHS = xs.reverseH([]).length()

    (5) forall ys.  xs.reverseH(ys).length() = xs.length()
     + ys.length()
    (5') xs.reverseH([]).length() = xs.length() +
```

```
     [].length() = xs.length()
    Done.
```

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

Base case: xs = []

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

Induction case: xs = (x:xs')
Assume: xs’.append(ys)).reverse() =
     ys.reverse().append(xs’.reverse())

Prove: (x:xs’).append(ys)).reverse() =
     ys.reverse().append((x:xs’).reverse())

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

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

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

Base Case: xs=[]

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

Induction step: xs=(x:xs’)
Assume: (xs'.reverse()).reverse() = xs’

Prove: ((x:xs').reverse()).reverse() = (x:xs’)

LHS = (x:xs’).reverse()).reverse()
```

```
    = xs’.reverse().append(x:[])).reverse90
    = (x:
[]).reverse().append((xs’reverse()).reverse()).reverse())
    = [].reverse().append(x:
[])).append(xs’.reverse().reverse())
    = (x:[]).append(xs’.reverse().reverse())
    = (x:[]).append(xs’) (IH)
    = (x:xs’)

RHS = (x:xs’) = LHS
```

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

Base Case: xs = []
LHS = [].reverseH (ys)
    = ys
RHS = ([].reverse()).append(ys)
    = [].append(ys)
    = ys = LHS

Induction case: xs = (x:xs’)
Assume: xs'.reverseH (ys) = (xs'.reverse()).append(ys)

Prove: (x:xs').reverseH (ys) =
((x:xs’).reverse()).append(ys)

LHS = (x:xs’).reverseH(ys)
    = xs’.reverseH(x:ys)
    = (xs’.reverse()).append(x:ys)(IH)
    = (xs’.reverse()).append(x:ys).reverse()))
    = (xs’.reverse ()).append(x:ys).reverse().reverse()))
    =
(((x:ys).reverse()).append((xs’.reverse()).reverse())).reve
rse()
    = (((x:ys).reverse()).append(xs’).reverse()
    = (ys.reverse().append(x:[]).append(xs’).reverse()
    = (ys.reverse()).append(x:
[]).append((xs’).append(xs’).reverse()
    = (ys.reverse()).append(x:xs’).reverse()
    = ((x:xs’).reversw()).append(ys.reverse().reverse())
    = (x:xs’).reverse()).append(ys)
```

```
RHS = ((x:xs’).reverse()).append(ys) = LHS
```

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

Base Case: xs = []

LHS = ([].reverseH (ys)).reverseH(zs) = ys.reverseH(zs)
RHS = ys.reverseH ([].append(zs)) = ys.reverseH(zs) = LHS

Induction step: xs = (x:xs’)
Assume: (xs'.reverseH (ys)).reverseH (zs) = ys.reverseH
(xs'.append(zs))

Prove: ((x:xs’).reverseH (ys)).reverseH (zs) = ys.reverseH
((x:xs’).append(zs))

LHS = ((x:xs’).reverseH (ys)).reverseH (zs)
    = (xs’.reverseH(x:ys)).reverseH(zs)
    = (x:ys).reverseH(xs’.append(zs)) (IH)
    = ys.reverseH(xs’.append(zs)))
RHS = ys.reverseH(x:xs’).append(zs))
    = ys.reverseH(xs’.append(zs))) = LHS
```

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

Invalid argument: xs.reverseH(ys.append(zs)) ≠
     (xs.reverseH(ys)).append(zs)
```

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

Base Case: xs = []

LHS = [].append(ys).reverseH(zs) = ys.reverseH(zs)
RHS = ys.reverseH([].reverseH(zs)) = ys.reverseH(zs)
```

```
Induction step: xs = (x:xs’)
Assume: (xs’.append(ys)).reverseH(zs) =
ys.reverseH(xs’.reverseH(zs))

Prove: ((x:xs’).append(ys)).reverseH(zs) =
ys.reverseH((x:xs’).reverseH(zs))

LHS = (xs’.append(ys)).reverseH(zs)
    = (xs’.append(ys)).reverse()).append(zs)
    = ys.reverse.append(xs’,reverse())).append(zs)
    = ys.reverse().append((xs’.reverse().append(zs)
    = ys.reverseH(xs.reverseh(zs))

RHS = ys.reverseH(xs’.reverseH(zs))
```

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

Base Case: xs = []

LHS = [].append(ys).reverse2() = ys.reverse2()
RHS = ys.reverse2().append([].reverse2()) = ys.reverse2() =
LHS

Induction step: xs = (x:xs’)

LHS = xs’.append(ys)).reverse2()
    = xs’append(ys)).reverseh([])
    = ys.reverseH(XS'.reverseH([]))
    = ys.reverseH(xs’.reverse2())
    = ys.reverseH({}.append(xs’, reverse2())
    = ys.reverse2().append(xs’,reverse2())
RHS = ys,reverse().append(xs’,reverse2()) = LHS
```

```
(14) (xs.reverse2()).reverse2() = xs
```

```
(15) t.flattenH(xs) = t.flatten().append(xs)
```

```
By induction on t:
Base case: t = d

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

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

Inductive case: t = N(d,t1,t2)

   Lemma:
      xs.append(ys.append(zs)) = (xs.append(ys)).append(zs)

   To prove: N(d,t1,t2).flattenH(xs) =
     N(d,t1,t2).flatten().append(xs)

   IH1:  t1.flattenH(xs) = t1.flatten().append(xs)
   IH2:  t2.flattenH(xs) = t2.flatten().append(xs)

   LHS = N(d,t1,t2).flattenH(xs)
         = t1.flattenH(d:t2.flattenH(xs))
      = t1.flatten().append(d:t2.flattenH(xs))
      = t1.flatten().append(d:t2.flatten().append(xs))

   RHS = N(d,t1,t2).flatten().append(xs)
       = (t1.flatten().append(d:t2.flatten())).append(xs)
      = t1.flatten().append(d:t2.flatten().append(xs))
       = LHS
```

```
(16) t.flatten2 () = t.flatten()

Base Case: t = d

LHS =  d.flatten2()
    = d.flattenH([])
    = d:[]
RHS = d.flatten()
    = d:[]

Induction step: t = N(d,t1,t2)
Prove: n(d,t1,t2).flatten2() = N(d,t1,t2).flatten()
```

```
IH1:t1.flatten2() = t1.flatten()
IH2:t2.flatten2()= t2.flatten()

LHS =
   = N(d,t1,t2).flatten2()
   = N(d,t1,t2).flattenH([])
   = N(d,t1,t2).flatten().append([])
   = N(d,t1,t2).flatten()
RHS = N(d,t1,t2).flatten.append(XS)
```

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

Base case: t = d

Invalid argument: t.map(f1).sum() ≠ t.nodes()
```

```
(18) t.nodes() = t.longestP ath().length() + 1

Invalid argument: t.nodes() ≠ t.longestP ath().length() + 1
```

```
(19)
Prove: For non-empty trees t, it is the case that
    t.internalNodes() + 1 = t.leaves().

Base case: t = d

LHS = d.internalNodes() + 1 = 1
RHS = d.leaves() = 1 = LHS

Induction case: t = N(d,t1,t2)
Where t1 = d, t2 = d

    To prove: N(d,t1,t2).internalNodes() + 1 =
    N(d,t1,t2).leaves()

    LHS = N(d,t1,t2).internalNodes() + 1
        = d.internalNodes() + t1.internalNodes() +
    t2.internalNodes() + 1
        = 1 + 0 + 0 + 1 = 2
```

```
    RHS = N(d,t1,t2).leaves()
        = t1.leaves() + t2.leaves()
        = 2 = LHS
```

---

(20)
Prove: A full m-ary with n nodes has (n - 1)/m internal nodes and ((m - 1)n + 1)/m leaves.

```
  Leaf = d
  Leaves = 1
  check: 1 = ((1 - 1)1 + 1)/1 = 1 YES
  internal nodes = 0
  check: 0 = (1-1)/1 YES

  Node(d,t1,t2,...tm)
  leaves = l1+l2+...+lm
  internal nodes = i1+i2+...+im+1
  For each subtree k, we have ik = (nk-1)/m
  For each subtree k, we have nk = ((m-1)nk + 1)/m

  check:
    LHS = i1+i2+...+im+1
        = (n1-1)/m + (n2-1)/m + ... + (nm-1)/m + 1
        = (n1-1 + n2-1 + ... + nm-1 + m) / m
        = (n1+n2 + ... + nm) / m

    RHS = l1+l2+...+lm
        = ( ((m-1)n1 + 1) + (m-1)n2 + 1) + ... + (m-1)nm +
    1) + 1 )/m
        = ( (m*n1 - n1 + 1) + (m*n2 - n2 + 1) + ... +
    (m*nm - nm + 1) )/m
        = (m*n1-n1 + m*n2-n2 + ... + m*nk-nk) / m
        = (n1+n2+...+nm) / m
        = LHS
```

---

(21) A full m-ary with i internal nodes has mi + 1 nodes and (m - 1)i + 1 leaves.

```
  Leaf = d
```

```
  Internal nodes = 0
  nodes = 1
  check: 1 = mi + 1 = 1 YES
  leaves = 1
  check: 1 = (m - 1)i + 1 = 1 YES

  Node(d,t1,t2,...tm)
  nodes = n1+n2+...+nm+1
  leaves = l1+l2+...+lm
  For each subtree k, we have nk = mi*k + 1
  For each subtree k, we have lk = (m - 1)i*k + 1

  check:
    LHS = n1+n2+...+nm+1
          = (mi1 + 1) + (mi2 + 1) + ... + (mim + 1)
          =
          = (n1+n2 + ... + nm) / m

     RHS = l1+l2+...+lm
          = ((m-1)i1 + 1) + ((m-1)i2 + 1) + ... + ((m-1)im +
    1)
          = (mi1-i1 + mi2-i2 + ... + mim-im) + m
          =
          = LHS
```

---

(22) A full m-ary with l leaves has (ml - 1)/(m - 1) nodes
     and (l - 1)/(m - 1) internal nodes.

```
  Leaf = d
  Internal nodes = 0
  nodes = 1
  check: 1 = mi + 1 = 1 YES
  leaves = 1
  check: 1 = (m - 1)i + 1 = 1 YES

  Node(d,t1,t2,...tm)
  nodes = n1+n2+...+nm+1
  leaves = l1+l2+...+lm
  For each subtree k, we have nk = mi*k + 1
  For each subtree k, we have lk = (m - 1)i*k + 1

  check:
```

```
LHS = n1+n2+...+nm+1
     = (mi1 + 1) + (mi2 + 1) + ... + (mim + 1)
     =
     = (n1+n2 + ... + nm) / m

RHS = l1+l2+...+lm
     = ((m-1)i1 + 1) + ((m-1)i2 + 1) + ... + ((m-1)im +
1)
     = (mi1-i1 + mi2-i2 + ... + mim-im) + m
     =
     = LHS
```

---

(23)
l = 100
n = ?
i = ?

n = ((m-1)n + 1)/m
100 = ((4-1)n + 1)/4
100 = (3n + 1)/4
400 = 3n+1
399 = 3n
n = 133

133 people have read the letter

---

(24)
l = 100
n = 133
i = ?

i = (l-1)/(m-1)
i = (100-1)/(4-1)
i = 99/3
i = 33

33 people sent out the letter