FPV Week 12 *

Zixuan Fan

Jul 2022

A self-generatede answer for all 3 exercises are provided here.

1 Factorial

```
Proof: fact_aux acc n = acc * fact n
1. Base case: n = 0, fact_aux acc 0 = acc * fact 0

fact_aux acc 0
= match 0 with 0 -> acc | n -> ... (fact_aux.def)
= acc (match)
= acc * 1 (arith.)
= acc * (match 0 with 0 - > 1 | n -> ...) (match)
= acc * fact 0 (fact.def)
QED

2. Inductive case: choose a random but fixed n from nature numbers,
I.H. For an arbitrary value acc, fact_aux acc n = acc * fact n
```

*All contents are based on the Artemis exercises and lecture slides of Prof. Seidl. No content is guaranteed to be totally correct.

```
TO prove: For an arbitrary value acc,  fact_aux acc (n+1) = acc * fact (n+1)

fact_aux acc (n+1)
= match (n+1) with 0 -> acc | n’ -> fact_aux (n+1) * acc n (fact_aux.def)
= fact_aux (n+1) * acc n (match)
= (n+1) * acc * fact n (I.H.)
= acc * (n+1) * fact n (arith.)
= acc * (match (n+1) with 0 -> acc | n’ -> (n+1) * fact n) (match)
= acc * fact (n+1) (fact.def)

QED
```

2 List

```
2  Lemma to prove in this exercise: summa l = sum l 0 (1) and mul a b 0 = a * b (1)

Lemma 1: acc + summa l = sum l acc
Base case: skipped
Inductive case: I.H. acc + summa xs = sum xs acc
Prove: for arbitrary x, acc + summa (x::xs) = sum (x::xs) acc
Proof: acc + summa (x::xs)
= acc + (match x::xs with [] -> 0 | _ -> x + summa xs) (def)
= acc + x + summa xs (match)
= sum xs (acc + x) (I.H.)
= match (x::xs) with [] -> acc | _ -> summa xs (x+acc) (match)
```

```
= summa (x::xs) acc

QED

Lemma 2: mul a b acc = acc + a * b
Base case: skipped
Inductive case: I.H. mul a b acc = acc + a * b
Prove: mul (a+1) b acc = acc + (a + 1) * b
Proof: mul (a+1) b acc
= if (a+1) <= 0 then acc else mul a b (b + acc) (def)
= mul a b (b+acc) (if)
= acc + b + a * b (I.H.)
= acc + (a+1) * b (arith.)

QED

mul c (sum l 0) 0
= c * (sum l 0) (Lemma 2)
= c * summa l (Lemma 1)

QED
```

3 Tree

```
Nodes t = count t = aux t 0
-> aux : acc + nodes t = aux t acc

Base case: skipped
```

```

Inductive case:
I.H.: For arbitrary acc1 and acc2
assume acc1 + nodes l = aux l acc1
and acc2 + nodes r = aux r acc2
Prove: for arbitrary acc acc + nodes (Node (l,r)) = count (Node (l, r)) acc

acc + nodes (Node (l, r))
= acc + (match Node(l, r) with Empty -> 0 | _ -> 1 + nodes l + nodes r) (nodes.def)
= acc + 1 + nodes l + nodes r  (match)
= (acc + 1) + nodes l + nodes r  (arith)
= (aux l (acc+1) : int) + nodes r  (I.H.)
=  aux r (aux l (acc+1))  (I.H.)
= match (Node (l, r)) with Empty -> acc | _ -> aux r (aux l (acc+1))  (match)
= aux (Node (l, r)) acc  (def)

QED
```