I2I-2 Functional Verification Example Excercises

Dimitri Tabatadze

July 3, 2023

1 Treecount

The following OCaml functions are given:

```
type node = Leaf of int | Inner of node * node

let rec countleaves t = match t with
  | Leaf _ -> 1
  | Inner (l, r) -> countleaves l + countleaves r

let rec countinner t = match t with
  | Leaf _ -> 0
  | Inner (l, r) -> 1 + countinner l + countinner r
```

Prove that the equality

```
count_leaves t = count_inner t + 1
```

holds.

2 Fun With Fold

The following OCaml functions are given:

```
let rec fl f a l = match l with [] -> a
  | x::xs -> fl f (f a x) xs
let rec fr f l a = match l with [] -> a
  | x::xs -> f x (fr f xs a)
let rec rev_map f l a = match l with [] -> a
  | x::xs -> rev_map f xs (f x :: a)
```

Prove that the equality

```
f1 (+) 0 (rev_map (fun x -> x * 2) l []) = fr (fun x a -> a + 2 * x) l 0
```

holds.

3 Even list

The following OCaml functions are given:

```
let rec el a lst = match lst with
  | h::_::t -> el (h::a) t
  | [h] -> el (h::a) []
  | [] -> a

let rec de i a lst = match lst with
  | h::t -> de (i+1) (if i mod 2 = 0 then h::a else a) t
  | [] -> a
```

Prove that the equality

```
de 0 [] l = el [] l
```

holds.

4 Bigotree

The following OCaml functions are given:

```
type node = Empty | Inner of node * int * node

let rec insertintree v t = match t with
  | Empty -> Inner (Empty, v, Empty)
  | Inner (l, u, r) -> if v > u then
      Inner (l, u, insertintree v r)
    else
      Inner (insertintree v l, u, r)

let rec totree a lst = match lst with
  | [] -> a
  | h::t -> insertintree h (totree a t)

let rec tolist t = match t with
  | Empty -> []
  | Inner (l, v, r) -> tolist l @ [v] @ tolist r

let rec insert n lst = match lst with
  | [] -> [n]
  | h::t -> if n > h then
      h::(insert n t)
    else
      n::h::t

let rec sort lst = match lst with
  | [] -> []
  | h::t -> insert h (sort t)
```

Prove that the equality

```
tolist (totree Empty l) = sort l
```

holds.