

# I2I-2 Functional Verification Example Exercises

Dimitri Tabatadze

July 2, 2023

## 1 Treecount

The following OCaml functions are given:

```
1 type node = Leaf of int | Inner of node * node
2
3 let rec countleaves t = match t with
4   | Leaf _ -> 1
5   | Inner (l, r) -> countleaves l + countleaves r
6
7 let rec countinner t = match t with
8   | Leaf _ -> 0
9   | Inner (l, r) -> 1 + countinner l + countinner r
```

Prove that the equality

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

holds.

### Solution

- Case 1:  $t = \text{Leaf}$

$$\begin{aligned} \text{countleaves } t & \stackrel{\text{countleaves}}{=} \text{match } t \text{ with Leaf } \rightarrow 1 \mid \text{Inner } (l, r) \rightarrow \\ & \quad \text{countleaves } l + \text{countleaves } r \\ & \stackrel{\text{def } t, \text{match}}{=} 1 \\ & \stackrel{\text{arith}}{=} 0 + 1 \\ & \stackrel{\text{match, countinner}}{=} \text{countinner } t + 1 \end{aligned}$$

- Case 2:  $t = \text{Inner } (a, b)$

$$\begin{aligned} \text{countleaves } t & \stackrel{\text{countleaves}}{=} \text{match } t \text{ with Leaf } \rightarrow 1 \mid \text{Inner } (l, r) \rightarrow \\ & \quad \text{countleaves } l + \text{countleaves } r \\ & \stackrel{\text{def } t, \text{match}}{=} \text{countleaves } a + \text{countleaves } b \\ & \stackrel{\text{I.H.}}{=} (\text{countinner } a + 1) + (\text{countinner } b + 1) \\ & \stackrel{\text{arith}}{=} (1 + \text{countinner } a + \text{countinner } b) + 1 \\ & \stackrel{\text{match, countinner, def}}{=} \text{countinner } t + 1 \end{aligned}$$

## 2 Dacentrili

The following OCaml functions are given:

```
1 let rec damcentravi a lst = match lst with
2   | [] -> a
3   | h::t -> damcentravi (h::(List.rev a)) t
4
5 let rec get_nth n lst = match lst with
6   | [] -> 0
7   | h::t -> if n = 0 then h else get_nth (n-1) t
8
9 let pick_middle lst = get_nth ((List.length lst) / 2) lst;;
```

Prove that the equality

```
1 a = pick_middle (damcentravi [] (a::b))
```

holds.

### Solution

To prove that

$$a = \text{pick\_middle } (\text{damcentravi } [] \ (a::b))$$

### 3 Even list

The following OCaml functions are given:

```

1 let rec el a lst = match lst with
2   | h::_::t -> el (h::a) t
3   | [h] -> el (h::a) []
4   | [] -> a
5
6 let rec de i a lst = match lst with
7   | h::t -> de (i+1) (if i mod 2 = 0 then h::a else a) t
8   | [] -> a

```

Prove that the equality

```

1 de 0 [] 1 = el [] 1

```

holds.

### Solution

To prove that

$$\text{de } 0 \text{ [] } 1 = \text{el [] } 1$$

we generalize and try to prove

$$\text{de } (2*i) \text{ a } l = \text{el a } l$$

Induction on the list  $l$

- Base case 1:  $l = []$

$$\begin{aligned} \text{de } (2*i) \text{ a } l & \stackrel{\text{de, def } l, \text{ match}}{=} a \\ & \stackrel{\text{match, def } l, \text{ el}}{=} \text{el a } l \end{aligned}$$

- Base case 2:  $l = [h] = h::[]$

$$\begin{aligned} \text{de } (2*i) \text{ a } l & \stackrel{\text{de, def } l}{=} \text{match } h::[] \text{ with } | h::t \rightarrow \text{de } (2*i+1) \text{ (if } 2*i \bmod 2 = 0 \text{ then } h::a \text{ else a) } t \mid [] \rightarrow a \\ & \stackrel{\text{match}}{=} \text{de } (2*i+1) \text{ (if } 2*i \bmod 2 = 0 \text{ then } h::a \text{ else a) } [] \\ & \stackrel{\text{if}}{=} \text{de } (2*i+1) (h::a) [] \\ & \stackrel{\text{de, match}}{=} h::a \\ & \stackrel{\text{match, el}}{=} \text{el } (h::a) [] \\ & \stackrel{\text{match, el}}{=} \text{el a } [h] \\ & \stackrel{\text{def } l}{=} \text{el a } l \end{aligned}$$

- Induction step:  $l = h1::h2::t$

```

de (2*i) a l      de,def 1      match h1::h2::t with | h::t -> de (2*i+1) (if 2*i
                    ==          mod 2 = 0 then h::a else a) t | [] -> a
                    match
                    ==      de (2*i+1) (if 2*i mod 2 = 0 then h1::a else a)
                    ==      h2::t
if,de,match      de (2*i+2) (if 2*i+1 mod 2 = 0 then h2::h1::a
                    ==      else h1::a) t
                    if
                    ==      de (2*i+2) (h1::a) t
                    I.H.
                    ==      el (h1::a) t
match,el      el a (h1::h2::t)
                    ==
def 1      el a l)
==

```

## 4 Bigotree

The following OCaml functions are given:

```
1 type node = Empty | Inner of node * int * node
2
3 let rec insertintree v t = match t with
4   | Empty -> Inner (Empty, v, Empty)
5   | Inner (l, u, r) -> if v > u then
6     Inner (l, u, insertintree v r)
7   else
8     Inner (insertintree v l, u, r)
9
10 let rec totree a lst = match lst with
11   | [] -> a
12   | h::t -> insertintree h (totree a t)
13
14 let rec tolist t = match t with
15   | Empty -> []
16   | Inner (l, v, r) -> tolist l @ [v] @ tolist r
17
18 let rec insert n lst = match lst with
19   | [] -> [n]
20   | h::t -> if n > h then
21     h::(insert n t)
22   else
23     n::h::t
24
25 let rec sort lst = match lst with
26   | [] -> []
27   | h::t -> insert h (sort t)
```

Prove that the equality

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

holds.

### Solution

To prove that

$$\text{tolist } (\text{totree Empty } l) = \text{sort } l$$

we need to first prove that **Lemma 1**:

$$\text{tolist } t = \text{sort } (\text{tolist } t)$$

holds. We do this by induction on  $t$

- Case 1:  $t = \text{Empty}$

$$\text{tolist } t$$