I2I-2 Functional Verification Example Excercises

Dimitri Tabatadze

July 3, 2023

1 Treecount

The following OCaml functions are given:

```ocaml
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.

Solution

- Case 1: `t = Leaf`

$$
\begin{aligned}
\texttt{countleaves t} \quad &\overset{\text{countleaves}}{=} && \texttt{match t with Leaf -> 1 | Inner (l, r) ->} \\
& && \texttt{countleaves l + countleaves r} \\
&\overset{\text{def t,match}}{=} && \texttt{1} \\
&\overset{\text{arith}}{=} && \texttt{0 + 1} \\
&\overset{\text{match,countinner}}{=} && \texttt{countinner t + 1}
\end{aligned}
$$

- Case 2: `t = Inner (a, b)`

$$
\begin{aligned}
\texttt{countleaves t} \quad &\overset{\text{countleaves}}{=} && \texttt{match t with Leaf -> 1 | Inner (l, r) ->} \\
& && \texttt{countleaves l + countleaves r} \\
&\overset{\text{def t,match}}{=} && \texttt{countleaves a + countleaves b} \\
&\overset{\text{I.H.}}{=} && \texttt{(countinner a + 1) + (countinner b + 1)} \\
&\overset{\text{arith}}{=} && \texttt{(1 + countinner a + countinner b) + 1} \\
&\overset{\text{match,couninner,def}}{=} && \texttt{countinner t + 1}
\end{aligned}
$$

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.

Solution

To prove the given equality, we first generalize and try to prove that

$$fl\ (+)\ b\ (rev\_map\ (fun\ x \to x * 2)\ l\ a) = fr\ (fun\ x\ a \to a + 2 * x)\ (l\ @\ a)\ b$$

We prove the equality by induction on the length of `l`

- Base case: `l = []`

$$fl\ (+)\ 0\ (rev\_map\ (fun\ x \to x * 2)\ l\ [])\ \stackrel{\text{rev\_map,def l,match}}{=}\ fl\ (+)\ b\ a$$

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.

Solution

To prove that

$$\texttt{de 0 [] l = el [] l}$$

we generalize and try to prove

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

Induction on the list `l`

- Base case 1: `l = []`

$$\texttt{de (2*i) a l} \quad \overset{\texttt{de,def l,match}}{=} \quad \texttt{a}$$

$$\overset{\texttt{match,def l,el}}{=} \quad \texttt{el a l}$$

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

$$\texttt{de (2*i) a l} \quad \overset{\texttt{de,def l}}{=} \quad \texttt{match h::[] with | h::t -> de (2*i+1) (if 2*i mod 2 = 0 then h::a else a) t | [] -> a}$$

$$\overset{\texttt{match}}{=} \quad \texttt{de (2*i+1) (if 2*i mod 2 = 0 then h::a else a) []}$$

$$\overset{\texttt{if}}{=} \quad \texttt{de (2*i+1) (h::a) []}$$

$$\overset{\texttt{de,match}}{=} \quad \texttt{h::a}$$

$$\overset{\texttt{match,el}}{=} \quad \texttt{el (h::a) []}$$

$$\overset{\texttt{match,el}}{=} \quad \texttt{el a [h]}$$

$$\overset{\texttt{def l}}{=} \quad \texttt{el a l}$$

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

```
de (2*i) a l    de,def l    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 l       el a l)
                ≡
```

4 Bigotree

The following OCaml functions are given:

```ocaml
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.

Solution

There is no solution.

To prove that

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

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

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

holds. We do this by induction on `t`

- Case 1: `t = Empty`

    tolist t