I2I-2 Functional Verification Example Excercises

Dimitri Tabatadze

July 2, 2023

1 Dacentrili

The followign OCaml functions are given:

```
let rec damcentravi a lst = match lst with
  | [] -> a
  | h::t -> damcentravi (h::(List.rev a)) t

let rec get_nth n lst = match lst with
  | [] -> 0
  | h::t -> if n = 0 then h else get_nth (n-1) t

let pick_middle lst = get_nth ((List.length lst) / 2) lst;;
```

Prove that the equality

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

holds.

Solution

To prove that

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

2 Even list

The followign 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

$$\texttt{de (le a - 1) a l = el a l}$$

Induction on the list `l`

- Base case: `l = []`

$$\begin{aligned}\texttt{de (le a - 1) a l} &\overset{\texttt{de,def,match}}{=} \texttt{a} \\ &\overset{\texttt{match,el}}{=} \texttt{el a l}\end{aligned}$$

- Induction step: `l = h::t`

$$\begin{aligned}\texttt{de (le a - 1) a l} &\overset{\texttt{de,def,match}}{=} \texttt{if (le a - 1) mod 2 = 0 then de (le a) (h::a) t} \\ &\qquad \texttt{else de (le a) a t}\end{aligned}$$

  - case 1: `le a` is even

$$\begin{aligned}&\overset{\texttt{if}}{=} \texttt{de (le a) (h::a) t} \\ &\overset{\texttt{I.H.}}{=} \texttt{el (h::a) t}\end{aligned}$$

3 Bigotree

The following OCaml functions are given:

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

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

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

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

```
to_list (to_tree Empty a) = sort a
```

holds.