Functional Programming and Verification

Prof. Dr. H. Seidl, N. Hartmann, R. Vogler
WS 2018/19

Exercise Sheet 12
Deadline: 27.01.2019

General Information
Detailed information about the lecture, tutorials and homework assignments can be found on the lecture website[1]. Solutions have to be submitted to Moodle[2]. Make sure your uploaded documents are readable. Blurred images will be rejected. Use Piazza[3] to ask questions and discuss with your fellow students.

Assignment 12.1 (L) What the fact

Consider the following function definitions:

```
let rec fact n = match n with 0 -> 1
  | n -> n * fact (n-1)

let rec fact_aux x n = match n with 0 -> x
  | n -> fact_aux (n*x) (n-1)

let fact_iter = fact_aux 1
```

Assume that all expressions terminate. Show that

```
fact_iter n = fact n
```

holds for all non-negative inputs $n \in \mathbb{N}_0$.

Assignment 12.2 (L) Arithmetic 101

Let these functions be defined:

```
let rec summa l = match l with [] -> 0
    | h::t -> h + summa t

let rec sum l a = match l with [] -> a
    | h::t -> sum t (h+a)

let rec mul i j a = if i <= 0 then a
  else mul (i-1) j (j+a)
```

Prove that, under the assumption that all expressions terminate, for arbitrary `l` and $c \geq 0$ it holds that:

```
mul c (sum l 0) 0 = c * summa l
```

[1] https://www.in.tum.de/i02/lehre/wintersemester-1819/vorlesungen/functional-programming-and-verification/
[2] https://www.moodle.tum.de/course/view.php?id=44932
[3] https://piazza.com/tum.de/fall2018/in0003/home

Assignment 12.3 (L) Counting nodes

A binary tree and two functions to count the number of nodes in such a tree are defined as follows:

```
type tree = Node of tree * tree | Empty

let rec nodes t = match t with Empty -> 0
    | Node (l,r) -> 1 + (nodes l) + (nodes r)

let rec count t =
  let rec aux t a = match t with Empty -> a
      | Node (l,r) -> aux r (aux l (acc+1))
  in
  aux t 0
```

Prove or disprove the following statement for arbitary trees `t`:

```
nodes t = count t
```

Assignment 12.4 (H) Len or nlen? [5 Points]

The following functions are defined:

```
let rec nlen n l = match l with [] -> 0
  | h::t -> n + nlen n t

let rec fold_left f a l = match l with [] -> a
  | h::t -> fold_left f (f a h) t

let rec map f l = match l with [] -> []
  | h::t -> f h :: map f t

let (+) a b = a + b
```

Show that the statement

```
nlen n l = fold_left (+) 0 (map (fun _ -> n) l)
```

holds for arbitrary `l` and `n`. Assume that all expressions do terminate.

Assignment 12.5 (H) Fun with fold [8 Points]

Given are the following functions with semantics as usual:

```
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)
let (+) a b = a + b
```

Prove that, if all expressions terminate, the statement

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

holds for all inputs `l`.

Assignment 12.6 (H) Trees [7 Points]

Once again, we define binary trees and some functions for them:

```
type tree = Empty | Node of int * tree * tree

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

let rec app l1 l2 = match l1 with [] -> l2
  | x::xs -> x::app xs l2

let rec tf f b t = match t with Empty -> b
  | Node (x, l, r) -> f (tf f b l) x (tf f b r)

let rec to_list t = match t with Empty -> []
  | Node (x, l, r) -> app (to_list l) (x::to_list r)

let add3 a b c = a + b + c
let (+) a b = a + b
```

Assume all expressions terminate, then proof for all trees `t`:

```
fl (+) 0 (to_list t) = tf add3 0 t
```

Hint: If you get stuck during your proof, try to formulate additional equalities that help to reach your goal. Don't forget to prove them, however!