Functional Programming and Verification

Prof. Dr. H. Seidl, N. Hartmann, R. Vogler

WS 2018/19

Exercise Sheet 11

Deadline: 20.01.2019

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.

Big-step proofs

Unless specified otherwise, all rules used in a big-step proof tree must be annotated and all axioms ($v \Rightarrow v$) must be written down.

Assignment 11.1 (L) Big Steps

We define these functions:

```
let rec f = fun l ->
  match l with [] -> 1 | x::xs -> x + g xs
and g = fun l ->
  match l with [] -> 0 | x::xs -> x * f xs
```

Consider the following expressions. Find the values they evaluate to and construct a big-step proof for that claim.

1. `let f = fun a -> (a+1,a-1)::[] in f 7`
2. `f [3;6]`
3. `(fun x -> x 3) (fun y z -> z y) (fun w -> w + w)`

Assignment 11.2 (L) Multiplication

Prove that the function

```
let rec mul a b =
  match a with 0 -> 0 | _ -> b + mul (a-1) b
```

terminates for all inputs $a, b \geq 0$.

Assignment 11.3 (L) Threesum

Use big-step operational semantics to show that the function

```
let rec threesum = fun l ->
  match l with [] -> 0 | x::xs -> 3*x + threesum xs
```

terminates for all inputs and computes three times the sum of the input list's elements.

---

[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 11.4 (L) Records

Let MiniOCaml++ be an extended version of MiniOCaml that comes with records. Perform these tasks:

1. Extend the operational big-step semantics of MiniOCaml for these new expressions.
2. Construct a big-step proof for the value of this expression:

```
let r = { x={ a=3+5; b=2+4::[] }; y=2*7 } in r.x.a::r.x.b
```

Assignment 11.5 (H) More Big Steps [12 Points]

Globally defined are these functions:

```
let rec map = fun f l ->
  match l with [] -> [] | x::xs -> f x :: map f xs
and fold_left = fun f a l ->
  match l with [] -> a | x::xs -> fold_left f (f a x) xs
and comp = fun f g x -> f (g x)
and mul = fun a b -> a * b
and id = fun x -> x
```

Give big-step proofs for the following claims:

1. `fold_left mul 3 [10]` $\Rightarrow$ `30`
2. `(let a = comp (fun x -> 2 * x) in a (fun x -> x + 3)) 4` $\Rightarrow$ `14`
3. `map (id id) [8]` $\Rightarrow$ `[8]`

Assignment 11.6 (H) Computing Zero [4 Points]

Consider the function `foo`:

```
let rec foo = fun l ->
  match l with [] -> 0
  | 0::xs -> foo xs
  | x::xs -> if x > 0 then foo (x-1::xs) else foo (x+1::xs)
```

Prove that `foo` terminates for all inputs. Axioms ($v \Rightarrow v$) may be omitted.

Assignment 11.7 (H) Raise the bar! [4 Points]

Given are these definitions:

```
let rec impl = fun n a ->
  match n with 0 -> a | _ -> impl (n-1) (a * n * n)
and bar = fun n -> impl n 1
```

Prove that `bar n` terminates with $n! * n!$ for all non-negative inputs n. Axioms ($v \Rightarrow v$) may be omitted.