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)`

Suggested Solution 11.1

1. Big step tree:

$$\pi_0 = \text{LI}\,\dfrac{\text{TU}\,\dfrac{\text{OP}\,\dfrac{\texttt{7} \Rightarrow \texttt{7} \quad \texttt{1} \Rightarrow \texttt{1} \quad 7+1 \Rightarrow \texttt{8}}{\texttt{7+1} \Rightarrow \texttt{8}} \quad \text{OP}\,\dfrac{\texttt{7} \Rightarrow \texttt{7} \quad \texttt{1} \Rightarrow \texttt{1} \quad 7-1 \Rightarrow \texttt{6}}{\texttt{7-1} \Rightarrow \texttt{6}}}{\texttt{(7+1,7-1)} \Rightarrow \texttt{(8,6)}}}{\texttt{[(7+1,7-1)]} \Rightarrow \texttt{[(8,6)]}}$$

[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

$$\text{LD}\,\dfrac{\texttt{fun a -> [(a+1,a-1)]} \Rightarrow \texttt{fun a -> [(a+1,a-1)]} \qquad \text{APP'}\,\dfrac{\texttt{fun a -> [(a+1,a-1)]} \Rightarrow \texttt{fun a -> [(a+1,a-1)]} \quad \texttt{7} \Rightarrow \texttt{7} \quad \pi_0}{\texttt{(fun a -> [(a+1,a-1)]) 7} \Rightarrow \texttt{[(8,6)]}}}{\texttt{let f = fun a -> [(a+1,a-1)] in f 7} \Rightarrow \texttt{[(8,6)]}}$$

2. Big step tree:

$$\pi_f = \text{GD}\,\dfrac{\texttt{f} = \texttt{fun l -> match l with [] -> 1 | x::xs -> x+g xs} \quad \texttt{fun l -> match l with [] -> 1 | x::xs -> x+g xs} \Rightarrow \texttt{fun l -> match l with [] -> 1 | x::xs -> x+g xs}}{\texttt{f} \Rightarrow \texttt{fun l -> match l with [] -> 1 | x::xs -> x+g xs}}$$

$$\pi_g = \text{GD}\,\dfrac{\texttt{g} = \texttt{fun l -> match l with [] -> 0 | x::xs -> x*f xs} \quad \texttt{fun l -> match l with [] -> 0 | x::xs -> x*f xs} \Rightarrow \texttt{fun l -> match l with [] -> 0 | x::xs -> x*f xs}}{\texttt{g} \Rightarrow \texttt{fun l -> match l with [] -> 0 | x::xs -> x*f xs}}$$

$$\pi_0 = \text{PM}\,\dfrac{\texttt{[6]} \Rightarrow \texttt{[6]} \quad \text{OP}\,\dfrac{\texttt{6} \Rightarrow \texttt{6} \quad \text{APP'}\,\dfrac{\pi_f \quad \texttt{[]} \Rightarrow \texttt{[]} \quad \text{PM}\,\dfrac{\texttt{[]} \Rightarrow \texttt{[]} \quad \texttt{1} \Rightarrow \texttt{1}}{\texttt{match [] with [] -> 1 | x::xs -> x+g xs} \Rightarrow \texttt{1}}}{\texttt{f []} \Rightarrow \texttt{1}} \quad \texttt{6*1} \Rightarrow \texttt{6}}{\texttt{6*f []} \Rightarrow \texttt{6}}}{\texttt{match [6] with [] -> 0 | x::xs -> x*f xs} \Rightarrow \texttt{6}}$$

$$\text{APP'}\,\dfrac{\pi_f \quad \texttt{[3;6]} \Rightarrow \texttt{[3;6]} \quad \text{PM}\,\dfrac{\texttt{[3;6]} \Rightarrow \texttt{[3;6]} \quad \text{OP}\,\dfrac{\texttt{3} \Rightarrow \texttt{3} \quad \text{APP'}\,\dfrac{\pi_g \quad \texttt{[6]} \Rightarrow \texttt{[6]} \quad \pi_0}{\texttt{g [6]} \Rightarrow \texttt{6}} \quad \texttt{3+6} \Rightarrow \texttt{9}}{\texttt{3+g [6]} \Rightarrow \texttt{9}}}{\texttt{match [3;6] with [] -> 1 | x::xs -> x+g xs} \Rightarrow \texttt{9}}}{\texttt{f [3;6]} \Rightarrow \texttt{9}}$$

3. Big step tree:

$$\pi_0 = \text{APP'}\,\dfrac{\texttt{fun x -> x 3} \Rightarrow \texttt{fun x -> x 3} \quad \texttt{fun y z -> z y} \Rightarrow \texttt{fun y z -> z y} \quad \text{APP'}\,\dfrac{\texttt{fun y z -> z y} \Rightarrow \texttt{fun y z -> z y} \quad \texttt{3} \Rightarrow \texttt{3} \quad \texttt{fun z -> z 3} \Rightarrow \texttt{fun z -> z 3}}{\texttt{(fun y z -> z y) 3} \Rightarrow \texttt{fun z -> z 3}}}{\texttt{(fun x -> x 3) (fun y z -> z y)} \Rightarrow \texttt{fun z -> z 3}}$$

$$\text{APP'}\,\dfrac{\pi_0 \quad \texttt{fun w -> w+w} \Rightarrow \texttt{fun w -> w+w} \quad \text{APP'}\,\dfrac{\texttt{fun w -> w+w} \Rightarrow \texttt{fun w -> w+w} \quad \texttt{3} \Rightarrow \texttt{3} \quad \text{OP}\,\dfrac{\texttt{3} \Rightarrow \texttt{3} \quad \texttt{3} \Rightarrow \texttt{3} \quad \texttt{3+3} \Rightarrow \texttt{6}}{\texttt{3+3} \Rightarrow \texttt{6}}}{\texttt{(fun w -> w+w) 3} \Rightarrow \texttt{6}}}{\texttt{(fun x -> x 3) (fun y z -> z y) (fun w -> w+w)} \Rightarrow \texttt{6}}$$

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 $a, b \geq 0$.

Suggested Solution 11.2

We prove by induction on `a` that `mul a b` terminates with $a * b$:

- Base case: `a = 0`:

$$\text{APP}\,\dfrac{\pi_{mul} \quad \text{PM}\,\dfrac{}{\texttt{match 0 with 0 -> 0 | \_ -> b + mul (-1) b} \Rightarrow \texttt{0}}}{\texttt{mul 0 b} \Rightarrow \texttt{0}}$$

- Inductive case: Assume `mul a b` terminates for an $a \geq 0$. Now, we show that it also terminates for $a + 1$:

$$\text{APP}\ \dfrac{\pi_{mul} \quad \text{PM}\ \dfrac{\text{OP}\ \dfrac{\text{APP}\ \dfrac{\text{by I.H.}}{\texttt{mul (a+1-1) b} \Rightarrow a*b} \quad b + (a*b) \Rightarrow (a+1)*b}{\texttt{b + mul (a+1-1) b} \Rightarrow (a+1)*b}}{\texttt{match a+1 with 0 -> 0 | \_ -> b + mul (a+1-1) b} \Rightarrow (a+1)*b}}{\texttt{mul (a+1) b} \Rightarrow (a+1)*b}$$

Here π_{mul} is the GD-tree of `mul`. Note one important thing here: When reducing to the induction hypothesis, we do not apply the operator rule for the `a+1-1` term, since $a+1$ is not really an OCaml expression, but the successor of a. We silently simplify $a+1-1$ to a and apply the induction hypothesis.

□

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.

Suggested Solution 11.3

We define:

$$\pi_{ts} = \text{GD}\ \dfrac{\texttt{threesum = fun l -> match l with [] -> 0 | x::xs -> 3*x + threesum xs}}{\texttt{threesum} \Rightarrow \texttt{fun l -> match l with [] -> 0 | x::xs -> 3*x + threesum xs}}$$

Now, we do an induction on the length n of the list.

- Base case: $n = 0$ (`l = []`)

$$\text{APP}\ \dfrac{\pi_{ts} \quad \texttt{[]} \Rightarrow \texttt{[]} \quad \text{PM}\ \dfrac{\texttt{[]} \Rightarrow \texttt{[]} \quad \texttt{0} \Rightarrow \texttt{0}}{\texttt{match [] with [] -> 0 | x::xs -> 3*x + threesum xs} \Rightarrow \texttt{0}}}{\texttt{threesum []} \Rightarrow \texttt{0}}$$

- Inductive step: We assume `threesum xs` terminates with $3\sum_{i=1}^{n} x_i$ for an input `xs` = $[x_n; \ldots; x_1]$ of length $n \geq 0$. Now, show that `threesum` x_{n+1}`::xs` terminates with $3\sum_{i=1}^{n+1} x_i$:

$$\text{APP}\ \dfrac{\pi_{ts} \quad x_{n+1}\texttt{::xs} \Rightarrow x_{n+1}\texttt{::xs} \quad \text{PM}\ \dfrac{x_{n+1}\texttt{::xs} \Rightarrow x_{n+1}\texttt{::xs} \quad \text{OP}\ \dfrac{\text{OP}\ \dfrac{\texttt{3} \Rightarrow \texttt{3} \quad x_{n+1} \Rightarrow x_{n+1} \quad 3*x_{n+1} \Rightarrow 3x_{n+1}}{3*x_{n+1} \Rightarrow 3x_{n+1}} \quad \text{APP}\ \dfrac{\texttt{by I.H.}}{\texttt{threesum xs} \Rightarrow 3\sum_{i=1}^{n} x_i} \quad 3x_{n+1} + 3\sum_{i=1}^{n} x_i \Rightarrow 3\sum_{i=1}^{n+1} x_i}{\texttt{3*}x_{n+1}\texttt{ + threesum xs} \Rightarrow 3\sum_{i=1}^{n+1} x_i}}{\texttt{match } x_{n+1}\texttt{::xs with [] -> 0 | x::xs -> 3*x + threesum xs} \Rightarrow 3\sum_{i=1}^{n+1} x_i}}{\texttt{threesum (}x_{n+1}\texttt{::xs)} \Rightarrow 3\sum_{i=1}^{n+1} x_i}$$

□

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
```