# Functional Programming and Verification

### 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:

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

---

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

2. Big step tree:

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

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

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

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

3. Big step tree:

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

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

## 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`:

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

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

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

Here $\pi_{mul}$ 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.

$\square$

## 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 = []`)

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

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

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

$\square$

## 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
```

**Suggested Solution 11.4**

1. We need two new rules for the record evaluation $(RE)$ and record access $(RA)$:

   - $RE \dfrac{e_1 \Rightarrow v_1 \quad \ldots \quad e_n \Rightarrow v_n}{\{a_1 = e_1; \ldots a_n = e_n\} \Rightarrow \{a_1 = v_1; \ldots a_n = v_n\}}$

   - $RA \dfrac{e \Rightarrow \{\ldots a = v; \ldots\}}{e.a \Rightarrow v}$

2. The big-step tree:

$$\pi_0 = RE \cfrac{RE \cfrac{OP \cfrac{3 \Rightarrow 3 \quad 5 \Rightarrow 5 \quad 3+5 \Rightarrow 8}{3+5 \Rightarrow 8} \quad LI \cfrac{OP \cfrac{2 \Rightarrow 2 \quad 4 \Rightarrow 4 \quad 2+4 \Rightarrow 6}{2+4 \Rightarrow 6}}{[2+4] \Rightarrow [6]}}{\{~a=3+5;~b=[2+4]~\} \Rightarrow \{~a=8;~b=[6]~\}} \quad OP \cfrac{2 \Rightarrow 2 \quad 7 \Rightarrow 7 \quad 2*7 \Rightarrow 14}{2*7 \Rightarrow 14}}{\{~x=\{~a=3+5;~b=[2+4]~\};~y=2*7~\} \Rightarrow \{~x=\{~a=8;~b=[6]~\};~y=14~\}}$$

$$\pi_1 = RA \cfrac{RA \cfrac{\{~x=\{~a=8;~b=[6]~\};~y=14~\} \Rightarrow \{~x=\{~a=8;~b=[6]~\};~y=14~\}}{\{~x=\{~a=8;~b=[6]~\};~y=14~\} \Rightarrow \{~x=\{~a=8;~b=[6]~\};~y=14~\}}}{\{~x=\{~a=8;~b=[6]~\};~y=14~\}.x \Rightarrow \{~a=8;~b=[6]~\}}$$

$$\pi_2 = RA \cfrac{RA \cfrac{\{~x=\{~a=8;~b=[6]~\};~y=14~\} \Rightarrow \{~x=\{~a=8;~b=[6]~\};~y=14~\}}{\{~x=\{~a=8;~b=[6]~\};~y=14~\} \Rightarrow \{~x=\{~a=8;~b=[6]~\};~y=14~\}}}{\{~x=\{~a=8;~b=[6]~\};~y=14~\}.x \Rightarrow \{~a=8;~b=[6]~\}}$$

$$LD \cfrac{\pi_0 \quad LI \cfrac{\pi_1 \quad \pi_2}{\{~x=\{~a=8;~b=[6]~\};~y=14~\}.x.a :: \{~x=\{~a=8;~b=[6]~\};~y=14~\}.x.b \Rightarrow [8;6]}}{\textbf{let}~r~=~\{~x=\{~a=3+5;~b=[2+4]~\};~y=2*7~\}~\textbf{in}~r.x.a::r.x.b \Rightarrow [8;6]}$$

## 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]`

4

**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.