| | |
|---|---|
| **CS421 : hw2** | **Summer 2015** |
| Dan McQuillan | *Handed In: June 13, 2015* |

1. **Problem 1**

   let p1 = (1., 3.);;
   let p2 = (2., 5.);;

   $\rho_1 = \{$

   $\quad p2 \rightarrow (2., 5.),$

   $\quad p1 \rightarrow (1., 3.)$

   $\}$

   let slope (x,y) = y /. x;;

   $\rho_2 = \{$

   $\quad slope \rightarrow < (x, y) \rightarrow y/.x, \rho_1 >,$

   $\quad p2 \rightarrow (2., 5.),$

   $\quad p1 \rightarrow (1., 3.)$

   $\}$

   let sub (x1,y1) (x2,y2) = (x2 - x1, y2 - y1);;

   $\rho_3 = \{$

   $\quad sub \rightarrow < (x1, y1) \rightarrow \text{fun } (x2, y2) \rightarrow (x2 - x1, y2 - y1), \rho_2 >,$

   $\quad slope \rightarrow < (x, y) \rightarrow y/.x, \rho_1 >,$

   $\quad p2 \rightarrow (2., 5.),$

   $\quad p1 \rightarrow (1., 3.)$

   $\}$

   let slope p1 p2 = slope (sub p1 p2);;

   $\rho_4 = \{$

   $\quad slope \rightarrow < p1 \rightarrow \text{fun } p2 \rightarrow \text{slope( sub p1 p2 )}, \rho_3 >,$

   $\quad sub \rightarrow < (x1, y1) \rightarrow \text{fun } (x2, y2) \rightarrow (x2 - x1, y2 - y1), \rho_2 >,$

   $\quad p2 \rightarrow (2., 5.),$

   $\quad p1 \rightarrow (1., 3.)$

   $\}$

let slope_p2 = slope p2;;

Evaluation:

$$\begin{aligned}
\text{Eval}(\text{slope p2}, \rho_4) &= \text{Eval}(\text{App}(< p1 \rightarrow \text{fun } p2 \rightarrow \text{slope}(\text{ sub p1 p2 }), \rho_3 >, (2., 5.)), \rho_4) \\
&= \text{Eval}(< \text{fun } p2 \rightarrow \text{slope}(\text{ sub p1 p2 }), \{p1 \rightarrow (2., 5.)\} + \rho_3 >, \rho_4) \\
&=< p2 \rightarrow \text{slope}(\text{sub p1 p2}), \{p1 \rightarrow (2., 5.)\} + \rho_3 >
\end{aligned}$$

$\rho_5 = \{$

$\quad slope\_p2 \rightarrow < p2 \rightarrow \text{slope}(\text{sub p1 p2}), \{p1 \rightarrow (2., 5.)\} + \rho_3 >$

$\quad slope \rightarrow < p1 \rightarrow \text{fun } p2 \rightarrow \text{slope}(\text{ sub p1 p2 }), \rho_3 >,$

$\quad sub \rightarrow < (x1, y1) \rightarrow \text{fun } (x2, y2) \rightarrow (x2 - x1, y2 - y1), \rho_2 >,$

$\quad p2 \rightarrow (2., 5.),$

$\quad p1 \rightarrow (1., 3.)$

$\}$

let p2 = (3., 9.);;

$\rho_6 = \{$

$\quad p2 \rightarrow (3., 9.),$

$\quad slope\_p2 \rightarrow < p2 \rightarrow \text{slope}(\text{sub p1 p2}), \{p1 \rightarrow (2., 5.)\} + \rho_3 >$

$\quad slope \rightarrow < p1 \rightarrow \text{fun } p2 \rightarrow \text{slope}(\text{ sub p1 p2 }), \rho_3 >,$

$\quad sub \rightarrow < (x1, y1) \rightarrow \text{fun } (x2, y2) \rightarrow (x2 - x1, y2 - y1), \rho_2 >,$

$\quad p1 \rightarrow (1., 3.)$

$\}$

slope_p2 p1;;

$$
\begin{aligned}
\text{slope\_p2 p1} &= \text{Eval}(\text{slope\_p2 p1}, \rho_6) \\
&= \text{Eval}(\text{App}(< p2 \to \text{slope}(\text{sub p1 p2}), \{p1 \to (2., 5.)\} + \rho_3 >, (1., 3.)), \rho_6) \\
&= \text{Eval}(\text{slope}(\text{sub p1 p2}), \{p2 \to (1., 3.), p1 \to (2., 5.)\} + \rho_3) \\
&= \text{Eval}(\text{App}(< (x, y) \to y/.x, \rho_1 >, \\
&\qquad\quad \text{Eval}(\text{sub p1 p2}, \{p2 \to (1., 3.), p1 \to (2., 5.)\} + \rho_3))) \\
&= \text{Eval}(\text{App}(< (x, y) \to y/.x, \rho_1 >, \\
&\qquad\quad \text{Eval}(\text{App}(\text{App}( \\
&\qquad\qquad\qquad < (x1, y1) \to \text{fun } (x2, y2) \to (x2 - x1, y2 - y1), \rho_2 >, \\
&\qquad\quad (2., 5.)), (1., 3.))))) \\
&= \text{Eval}(\text{App}(< (x, y) \to y/.x, \rho_1 >, \\
&\qquad\quad \text{Eval}((x2 - x1, y2 - y1), \{x2 \to 1, y2 \to 3, x1 \to 2, y1 \to 5\} + \rho_2))) \\
&= \text{Eval}(\text{App}(< (x, y) \to y/.x, \rho_1 >, (-1, -2))) \\
&= \text{Eval}(y/.x, \{x \to -1, y \to -2\} + \rho_1) \\
&= 2
\end{aligned}
$$

slope p1 p2;;

$$
\begin{aligned}
\text{slope p1 p2} &= \text{Eval}(\text{slope p1 p2}, \rho_6) \\
&= \text{Eval}(\text{App}(\text{App}(< p1 \to \text{fun } p2 \to \text{slope}(\text{sub p1 p2}), \rho_3 >, (1., 3.)), (3., 9.))) \\
&= \text{Eval}(\text{slope}(\text{sub p1 p2}), \{p2 \to (3., 9.), p1 \to (1., 3.)\} + \rho_3) \\
&= \text{Eval}(\text{App}(< (x, y) \to y/.x, \rho_1 >, \\
&\qquad\quad \text{Eval}(\text{sub p1 p2}, \{p2 \to (3., 9.), p1 \to (1., 3.)\} + \rho_3))) \\
&= \text{Eval}(\text{App}(< (x, y) \to y/.x, \rho_1 >, \\
&\qquad\quad \text{Eval}(\text{App}(\text{App}( \\
&\qquad\qquad\qquad < (x1, y1) \to \text{fun } (x2, y2) \to (x2 - x1, y2 - y1), \rho_2 >, \\
&\qquad\quad (1., 3.)), (3., 9.))))) \\
&= \text{Eval}(\text{App}(< (x, y) \to y/.x, \rho_1 >, \\
&\qquad\quad \text{Eval}((x2 - x1, y2 - y1), \{x2 \to 3, y2 \to 9, x1 \to 1, y1 \to 3\} + \rho_2))) \\
&= \text{Eval}(\text{App}(< (x, y) \to y/.x, \rho_1 >, (2, 6))) \\
&= \text{Eval}(y/.x, \{x \to 2, y \to 6\} + \rho_1) \\
&= 3
\end{aligned}
$$

2. **Problem 2**

```
let f g x =
  (let r =
    if ((print_string "a"; x > 5) && (g(); x > 10))
    then
      (print_string "b"; x - 7)
    else
      let z = (print_string "c"; 15) in (print_string "d"; z)
    in (g(); r));;
let u = (f (fun () -> print_string "e\n")
           (f (fun () -> print_string "f\n") 3));;
```

The first command creates a curried function, f. This function has the following type information: val f : (unit → 'a) → int → int = <fun>

The instantiation of u prints the following: acdf
ae
be
val u : int = 8

u is initialized by f which is done by first invoking a call to the function, f. f has 2 parameters that need to be passed to it, g and x.

g is a function with no parameters and simply prints out "e" and then a trailing new line and will finally return unit. x will represent a second call to the function and will ultimately return a value, to be utilized by the function that is initializing u as mentioned above. The first parameter prints out "f" and then a trailing new line and will then return unit.

The logic within the function, f, will first print out the character "a". Then if x is greater then 5 it will invoke the function g and then if x is also greater than 10 it will print out the character "b" as well as return the value x reduced by 7 which will then be the value of r. If the number is less than 10, then it will create a new variable z and in it's initialization it will print out "c" then that block will return 15 and the value for z will be 15 in the block: (print_string "d"; z). When that block is evaluated it will print the character "d" and then return the value z which will be the value of r.

Finally, once we have the value of r from one of the two paths it will invoke the function g and then return the value of r.

Therefore the order of execution is the following for our script:

(a) Evaluate **(f (fun () → print_string "f\n") 3)**
    i. Call function f with params (fun () → print_string "f\n"), 3
        A. print out string "a"
        B. 3 ¡ 5 therefore we go to else block

C. print out string "c"
D. initialize z to 15
E. print out string "d"
F. initialize r to z which is 15
G. call function g which will print out "f" and a new line
H. return the value of r which is 15 from the function f

After the above flows we will have on the console:
"acdf\n"

ii. Get the return value of the function call to f which will be 15

(b) Evaluate **f (fun () → print_string "e\n") 15**

i. Call function f with params (fun() → print_string "e\n") 15

A. print out string "a"
B. 15 ¿ 5 therefore we continue by evaluating the expression after the &&
C. invoke g which will print out the string "e\n"
D. 15 ¿ 10 therefore we continue into the primary block of the ternary expression
E. print out string "b" and return 15 - 7 from the ternary expression
F. initialize r to 8
G. invoke g which will print out the string "e\n"
H. return the value of r which will be 8 from the function f

After the above steps we will have on the console:
acdf
ae
be

(c) Evaluate let u = 8;; This will set up a variable u in our environment with a value of 8.

Our final content in our console will be what was described above:
acdf
ae
be
val u : int = 8