Esercitazione
28 Novembre 2018

Esercizio 1

Una funzione a dominio finito è una funzione che è definita solo per un numero finito di elementi. Ad esempio si consideri la seguente funzione con una sintassi nello stile di OCaML

```
let sum = fun y -> 50 + y for y in [0; 1; 2; 3; 4];;
```

La funzione `sum` è definita solamente per valori del parametro attuale che appartengono all'insieme $\{0, 1, 2, 3, 4\}$, insieme che è calcolato al momento della definizione della funzione stessa.

1. Si estenda la sintassi astratta del linguaggio didattico funzionale senza funzioni ricorsive in modo da includere tali funzioni.

```
type exp = ...
   | DFun of ide * exp * exp list

type evT = ...
   | DFunVal of ide * exp * evT list * evT env
```

2. Si definiscano le regole OCaML dell'interprete per trattare la valutazione di dichiarazione e la chiamata di funzioni a dominio finito.

```
let rec eval (e : exp) (r : evT env) : evT = match e with
   ...
   | DFun(i, a, exL) -> let vL = (evalList exL r) in DFunVal(i, a, vL, r)
   ...
   | FunCall(f, eArg) ->
         let fClosure = (eval f r) in
            (match fClosure with
                ...
                DFunVal(arg, fBody, fDom, fDecEnv) ->
                      let v = (eval eArg r) in
                          if (check v fDom) then eval fBody (bind fDecEnv arg v)
                                            else Unbound |
                _ -> failwith("non functional value")) |
   ...
and evalList (lst : exp list) (r : evT env) : evT list = match lst with
     [ ] -> [ ] |
     e :: rest ->
         let v = (eval e r) in v :: evalList rest r |
and check (v : evT) (lst : evT list) : bool = match lst with
     [ ] -> false |
     val :: rest -> if (evTeq v val) then true else (check v rest)
and evTeq (v1 : evT) (v2 : evT) : bool = match lst with
     ... (* adeguata nozione di eguaglianza su evT *);;
```

Esercizio 2

Si consideri il seguente programma scritto in OCaML.

```
let n = 4;;

let f = (fun x -> n + x);;

let g x y = (x + y) * x in
  let rec h m n = if n < 2 then m 2
                  else g (m 2) (h (fun x -> n) (n - 2)) in
    h f 4;;
```

1. Si determini il tipo inferito dall'interprete OCaML per gli identificatori di funzione (`f`, `g` e `h`) che compaiono nel programma scelto.

```
val f : int -> int = <fun>
val g : int -> int -> int = <fun>
val h : (int -> int) -> int -> int = <fun>
```

2. Si simuli la valutazione del programma mostrando la struttura della pila dei record di attivazione.

3. Si determini il valore calcolato dal programma. `180`

Esercizio 3

Si consideri il seguente programma OCaML

```
let rec map2 f l1 l2 =
   match (l1,l2) with
        ([], _) -> []
      | (_, []) -> []
      | (x::l1s,y::l2s) -> (f x y) :: map2 f l1s l2s;;

let pick n m =
   if n > m then (fun b x -> if b then x+1 else x*2)
            else (fun b x -> if b then x+2 else x*4);;

map2 (pick 7 9) [true;false] [3;4;5;6];;
```

1. Si determini il tipo inferito dall'interprete OCaML per gli identificatori di funzione (`map2` e `pick`) che compaiono nel programma scelto.

```
val map2 : ('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
val pick : 'a -> 'a -> bool -> int -> int = <fun>
```

2. Si simuli la valutazione del programma mostrando la struttura della pila dei record di attivazione.

3. Si determini il valore calcolato dal programma. `- : int list = [5; 16]`