Elaborati

28 gennaio 2022

0.1 Modalità di consegna

L'esercizio risolto dovrà essere consegnato tramite email a marco.lucchese@univr.it e per conoscenza a massimo.merro@univr.it.

0.2 Descrizione

Scegliere e svolgere uno degli esercizi seguenti. L'estensione prevede l'aggiunta per: interprete, typechecker e prittyprinter visto a laboratorio dei costrutti elencati di seguito. Il file da consegnare dovrà contenere il seguente linguaggio:

$$\textsc{Linguaggio while} \tag{1}$$

$$op :: piu \mid maggioreouguale \tag{2}$$

$$e \in Expr :: n \mid b \mid e\ op\ e \mid if\ e\ then\ e\ else\ e \mid skip \mid e; e \mid while\ e\ do\ e \mid !e \mid e_1 := e_2 \mid ...estensioni... \mid \tag{3}$$

1. Estendere il linguaggio while con funzioni e applicazione di funzioni in CBV **oppure** in CBN. Mostrare un esempio. [**EXTRA**] Implementare operatori di punto fisso.

$$(\textsc{cbv-fn})\frac{-}{<(fn\ x:T\to e)v,s>\to<e\{v/x\},s>} \tag{4}$$

$$(\textsc{cbv-app1})\frac{<e_1,s>\to<e_1',s'>}{<e_1e_2,s>\to<e_1'e_2,s'>} \tag{5}$$

$$(\textsc{cbv-app2})\frac{<e_2,s>\to<e_2',s'>}{<ve_2,s>\to<ve_2',s'>} \tag{6}$$

$$(\textsc{cbv-fn})\frac{-}{<(fn\ x:T\to e)e_2,s>\to<e\{e_2/x\},s>} \tag{7}$$

$$(\textsc{cbn-app})\frac{<e_1,s>\to<e_1',s'>}{<e_1e_2,s>\to<e_1'e_2,s'>} \tag{8}$$

2. Estendere il linguaggio while con la composizione parallela, la somma non deterministica e i lock. Mostrare un esempio.

3. Estendere il linguaggio while con la composizione parallela, la somma nn deterministica e il costrutto await. Mostrare un esempio.

0.3 Esempio esercizio risolto

Estendere il linguaggio while con sintassi, semantica CBV, tipaggio per il costrutto let. Mostrare un esempio

$$(\textsc{cbv-let1})\frac{<e_1,s>\to<e_1',s'>}{<let\ x:T=e_1\ in\ e_2,s>\to<let\ x:T=e_1'\ in\ e_2,s'>} \tag{9}$$

$$(\textsc{cbv-let2})\frac{-}{<let\ x:T=e_1\ in\ e_2,s>\to<e_2\{v/x\},s>} \tag{10}$$

```
      | Var of int
      | Let of type_L1 * expr * expr
```

CODE 1: sintassi

```
  | reduce (Let(t,e1,e2),s) =
    (if is_value e1 then
        SOME(sostituisci e1 0 e2,s)                                    (*  (let2)  *)
        else (
            case reduce (e1,s) of                                  (*  (let1)  *)
                SOME(e1',s') => SOME(Let(t,e1',e2),s')
                |NONE => NONE
        )
```

CODE 2: semantica

```
fun is_value (Integer n) = true
  | is_value (Boolean b) = true
  | is_value (Skip) = true
  | is_value _ = false
```

CODE 3: Controllo se e è un valore

```
fun sostituisci e n (Integer n')      = Integer n'
  | sostituisci e n (Boolean b)       = Boolean b
  | sostituisci e n (Op (e1,opr,e2))  = Op (sostituisci e n e1,opr,sostituisci e n e2)
  | sostituisci e n (If (e1,e2,e3))   = If (sostituisci e n e1, sostituisci e n e2, sostituisci e n e3)
  | sostituisci e n (Assign (l,e1))   = Assign(l,sostituisci e n e1)
  | sostituisci e n (Deref l)         = Deref l
  | sostituisci e n (Skip)            = Skip
  | sostituisci e n (Seq (e1,e2))     = Seq (sostituisci e n e1,sostituisci e n e2)
```

CODE 4: funzione di sostituzione

```
  | infertype gamma (Let(t,e1,e2))
    = (case (infertype gamma e1, infertype gamma (#1 gamma, t::(#2 gamma)) e2) of
        (SOME t1, SOME t') => if t1=t then SOME t' else NONE
        | NONE => NONE)
```

CODE 5: typechecking

```
  | prettyprintexpr (Let (t,e1,e2)) =  "let val .:" ^ (prettyprintexpr t )  ^ "= " ^(prettyprintexpr e1 )
                            ^ " in " ^ (prettyprintexpr e2)^" end "
```

CODE 6: stampa

let val x:int = 3 in x end

CODE 7: esempio