Aspetti Avanzati dei Linguaggi di Programmazione

Esercizi - Aspetti avanzati di Linguaggi di programmazione

Autori:
Stefano Campese
Enrico Savoca

Contents

1 Introduzione

Il documento è stato realizzato da Stefano Campese ed Enrico Savoca. Lo scopo di esso è quello di raccogliere in un unico posto le soluzioni di tutti gli esercizi del corso di "Aspetti avanzati dei linguaggi di programmazione". Gli esercizi sono stati svolti da entrambi gli autori e la versione ritenuta corretta di ognuno di essi è stata riportata nel presente testo.

2 Il mini linguaggio funzionale (note 2)

Esercizio 1.1

La relazione di riduzione data definisce una strategia efficiente per la valutazione del ter- mine if-then-else, che permette cioè di valutare unicamente il ramo scelto dalla valutazione della guardia booleana. Ridefinire la semantica operazionale del linguaggio in modo che adotti una strategia non efficiente per il costrutto if-then-else, valutando entambi i rami del costrutto condizionale.

Svolgimento

Per ottenere una semantica meno efficiente, si cambiano gli assiomi (IF-TRUE) e (IF-FALSE) e si aggiungono le regole (THEN) ed (ELSE). (IF) si mantiene tale.

Assiomi

$$\text{(IF-TRUE)}\ \frac{}{if\ TRUE\ then\ v_1\ else\ v_2 \to\ v_1}$$

$$\text{(IF-FALSE)}\ \frac{}{if\ FALSE\ then\ v_1\ else\ v_2 \to\ v_2}$$

Regole

$$\text{(THEN)}\ \frac{M_2 \to\ M_2'}{if\ v_1\ then\ M_2\ else\ M_3 \to\ if\ v_1\ then\ M_2'\ else\ M_3}$$

$$\text{(ELSE)}\ \frac{M_3 \to\ M_3'}{if\ v_1\ then\ v_2\ else\ M_3 \to\ if\ v_1\ then\ v_2\ else\ M_3'}$$

Esercizio 1.4

La valutazione, cioè l'esecuzione del programma, è deterministica. Se $M \to$ M' e $M \to$ M" allora M'=M".
Dimostrare la proposizione precedente per induzione sulla struttura del termine M.

Dimostrazione

Si dimostra per induzione sulla derivazione.

Casi base:

- M=true
  Se true → M' e true → M" $\implies$ M' = M"
  Vero perchè true è un valore finale;
- M=false
  Se false → M' e false → M" $\implies$ M' = M"
  Vero perchè false è un valore finale;
- M=fn x.M
  Se fn x.M → M' e fn x.M → M" $\implies$ M' = M"
  Vero perchè fn x.M è un valore finale;
- M=n
  Se n → M' e n → M" $\implies$ M' = M"
  Vero perchè n è un valore finale.

Passo induttivo:

- Tesi: $M = M_1 + M_2 \;\rightarrow\; M'$, $M_1 + M_2 \;\rightarrow$ M" $\implies$ M'=M"
  Per ipotesi induttiva:
  - Se $M_1 \;\rightarrow\; M_1'$ e $M_1 \;\rightarrow\; M_1''$ allora $M_1' = M_1''$;
  - Se $M_2 \;\rightarrow\; M_2'$ e $M_2 \;\rightarrow\; M_2''$ allora $M_2' = M_2''$.

  3 Casi possibili:

  a) Derivazione tramite l'uso della regola (SUM-LEFT):

  $$\text{(SUM-LEFT)}\,\frac{M_1 \;\rightarrow\; M_1'}{M_1 + M_2 \;\rightarrow\; M_1' + M_2 = M'}$$

  In questo caso $M_1' = M_1''$ per ipotesi induttiva e quindi anche $M' = M''$;
  La derivazione risulta deterministica applicando questa regola.

  b) Derivazione tramite l'uso della regola (SUM-RIGHT):

  $$\text{(SUM-RIGHT)}\,\frac{M_2 \;\rightarrow\; M_2'}{v_1 + M_2 \;\rightarrow\; v_1 + M_2' = M'}$$

  In questo caso $M_2' = M_2''$ per ipotesi induttiva e quindi anche $M' = M''$;
  La derivazione risulta deterministica applicando questa regola.

c) Derivazione tramite l'uso della regola (SUM):

$$\text{(SUM)}\ \frac{\text{con n'}=n_1+n_2}{n_1+n_2\ \rightarrow\ n'}$$

Il risultato della somma è univoco e di conseguenza la derivazione è deterministica.

In tutti e 3 i casi, i soli possibili, la derivazione è deterministica, quindi la tesi è dimostrata.

- Tesi: $M = M_1 - M_2\ \rightarrow\ M'$, $M_1 - M_2\ \rightarrow$ M" $\implies$ M'=M"
Per ipotesi induttiva:
  - Se $M_1\ \rightarrow\ M_1'$ e $M_1\ \rightarrow\ M_1''$ allora $M_1' = M_1''$;
  - Se $M_2\ \rightarrow\ M_2'$ e $M_2\ \rightarrow\ M_2''$ allora $M_2' = M_2''$.

  3 Casi possibili:

  a) Derivazione tramite l'uso della regola (MINUS-LEFT):

$$\text{(MINUS-LEFT)}\ \frac{M_1\ \rightarrow\ M_1'}{M_1 - M_2\ \rightarrow\ M_1' - M_2 = M'}$$

  In questo caso $M_1' = M_1''$ per ipotesi induttiva e quindi anche $M' = M''$;
  La derivazione risulta deterministica applicando questa regola.

  b) Derivazione tramite l'uso della regola (MINUS-RIGHT):

$$\text{(MINUS-RIGHT)}\ \frac{M_2\ \rightarrow\ M_2'}{v_1 - M_2\ \rightarrow\ v_1 - M_2' = M'}$$

  In questo caso $M_2' = M_2''$ per ipotesi induttiva e quindi anche $M' = M''$;
  La derivazione risulta deterministica applicando questa regola.

  c) Derivazione tramite l'uso della regola (MINUS):

$$\text{(MINUS)}\ \frac{\text{con n'}=n_1-n_2}{n_1-n_2\ \rightarrow\ n'}$$

  Il risultato della differenza è univoco e di conseguenza la derivazione è deterministica.

  In tutti e 3 i casi, i soli possibili, la derivazione è deterministica, quindi la tesi è dimostrata.

- Tesi: $M = if\ M_1\ then\ M_2\ else\ M_3\ \rightarrow\ M'$, $if\ M_1\ then\ M_2\ else\ M_3\ \rightarrow\ M''$ $\implies$ M'=M"
Per ipotesi induttiva:
  - Se $M_1\ \rightarrow\ M_1'$ e $M_1\ \rightarrow\ M_1''$ allora $M_1' = M_1''$.

3 Casi possibili:

a) Derivazione tramite l'uso della regola (IF):

$$\text{(IF)}\ \frac{M_1 \ \rightarrow\ M_1'}{if\ M_1\ then\ M_2\ else\ M_3\ \rightarrow\ if\ M_1'\ then\ M_2\ else\ M_3 = M'}$$

In questo caso $M_1' = M_1''$ per ipotesi induttiva e quindi anche $M' = M''$;
La derivazione risulta deterministica applicando questa regola.

b) Derivazione tramite l'uso dell'assioma (IF-TRUE):

$$\text{(IF-TRUE)}\ \frac{}{if\ true\ then\ M_2\ else\ M_3\ \rightarrow\ M_2 = M'}$$

In questo caso $M_2 = M' = M''$ per applicazione dell'assioma e la derivazione risulta deterministica.

c) Derivazione tramite l'uso della regola (IF-FALSE):

$$\text{(IF-FALSE)}\ \frac{}{if\ false\ then\ M_2\ else\ M_3\ \rightarrow\ M_3 = M'}$$

In questo caso $M_3 = M' = M''$ per applicazione dell'assioma e la derivazione risulta deterministica.

In tutti e 3 i casi, i soli possibili, la derivazione è deterministica, quindi la tesi è dimostrata.

- Tesi: $M = M_1\, M_2\ \rightarrow\ M'$, $M_1\, M_2\ \rightarrow$ M" $\implies$ M'=M"
  Per ipotesi induttiva:
  - Se $M_1\ \rightarrow\ M_1'$ e $M_1\ \rightarrow\ M_1''$ allora $M_1' = M_1''$;
  - Se $M_2\ \rightarrow\ M_2'$ e $M_2\ \rightarrow\ M_2''$ allora $M_2' = M_2''$.

  2 Casi possibili:

  a) Derivazione tramite l'uso della regola (APP1):

$$\text{(APP1)}\ \frac{M_1\ \rightarrow\ M_1'}{M_1\, M_2\ \rightarrow\ M_1'\, M_2 = M'}$$

  In questo caso $M_1' = M_1''$ per ipotesi induttiva e quindi anche $M' = M''$;
  La derivazione risulta deterministica applicando questa regola.

  b) Derivazione tramite l'uso della regola (APP2):

$$\text{(APP2)}\ \frac{M_2\ \rightarrow\ M_2'}{v_1\, M_2\ \rightarrow\ v_1\, M_2' = M'}$$

In questo caso $M_2' = M_2''$ per ipotesi induttiva e quindi anche $M' = M''$;
La derivazione risulta deterministica applicando questa regola.

In tutti e 2 i casi, i soli possibili, la derivazione è deterministica, quindi la tesi è dimostrata.

- Tesi: $M = fn\, x.M\ v\ \rightarrow\ Mx := v = M'$, $fn\, x.M\ v\ \rightarrow\ Mx := v = M''\ \Longrightarrow$ M'=M"
  Derivazione tramite l'uso dell'assioma (BETA):

$$\text{(BETA)}\ \frac{}{fn\, x.M\ v\ \rightarrow\ Mx := v = M'}$$

  Tramite l'applicazione dell'assioma si ottiene Mx:=v=M'=M".
  La derivazione risulta quindi deterministica e la tesi risulta dimostrata.

La derivazione è determinstica all'applicazione di tutte le regole e gli assiomi del linguaggio, di conseguenza è deterministica l'esecuzione del programma.

□

Esercizio 1.5

Descrivere la valutazione del termine ((fn x.3) (fn y.y)) ((fn z.if z then 1 else 0) (false)). Modificare le regole di valutazione in modo tale che, mantenendo una strategia call-by-value, il termine precedente evolva in un termine stuck in meno passi di riduzione. Scrivere le regole di valutazione della strategia call-by-name e valutare il termine precedente secondo questa strategia.

Svolgimento

Valutazione tramite strategia call-by-value:
passo 1

$$\text{(APP1)}\ \frac{\text{(BETA)}\ \frac{}{(\mathsf{fn}\ x.3)(\mathsf{fn}\ y.y)\ \rightarrow\ 3}}{((\mathsf{fn}\ x.3)(\mathsf{fn}\ y.y))\ ((\mathsf{fn}\ z.if\ z\ then\ 1\ else\ 0)(false))\ \rightarrow\ 3\ ((\mathsf{fn}\ z.if\ z\ then\ 1\ else\ 0)(false))}$$

passo 2

$$\text{(APP2)}\ \frac{\text{(BETA)}\ \frac{}{(\mathsf{fn}\ z.if\ z\ then\ 1\ else\ 0)(false)\ \rightarrow\ if\ false\ then\ 1\ else\ 0}}{3\ ((\mathsf{fn}\ z.if\ z\ then\ 1\ else\ 0)(false))\ \rightarrow\ 3\ (if\ false\ then\ 1\ else\ 0)}$$

passo 3

$$\text{(APP2)}\ \dfrac{\text{(IF-FALSE)}\ \dfrac{}{if\ false\ then\ 1\ else\ 0\ \rightarrow\ 0}}{3\ (if\ false\ then\ 1\ else\ 0)\ \rightarrow\ 3\ 0}$$

3 0 costituisce uno stuck: non esistono regole o assiomi per continuare l'esecuzione di tale programma. Esecuzione terminata dopo 3 passi di derivazione.
Si può mantenere la strategia CBV e terminare l'esecuzione in un numero minore di passi modificando (APP2):

$$\text{(APP2)}\ \dfrac{N\ \rightarrow\ N'}{(\mathsf{fn}\ x.M)\ N\ \rightarrow\ \mathsf{fn}\ x.M\ N'}$$

In questo modo l'esecuzione del programma terminerebbe dopo un solo passo. Infatti, ci si ritroverebbe in una forma in cui non è possibile applicare alcuna regola o assioma (3 non è una funzione).

Con una strategia CBN (Call-by-name) non ho (APP2) e la funzione (BETA) è la seguente:

$$\text{(BETA)}\ \dfrac{\checkmark}{(\mathsf{fn}\ x.M)\ N\ \rightarrow\ Mx := N}$$

Con questa regola mostriamo l'esecuzione del programma:

passo 1

$$\text{(APP1)}\ \dfrac{\text{(BETA)}\ \dfrac{}{(\mathsf{fn}\ x.3)(\mathsf{fn}\ y.y)\ \rightarrow\ 3}}{((\mathsf{fn}\ x.3)(\mathsf{fn}\ y.y))\ ((\mathsf{fn}\ z.if\ z\ then\ 1\ else\ 0)(false))\ \rightarrow\ 3\ ((\mathsf{fn}\ z.if\ z\ then\ 1\ else\ 0)(false))}$$

Stuck! Non esistono regole o assiomi applicabili: APP1 non è applicabile in quanto 3 non è funzione.

Esercizio 1.6

Consideriamo le seguenti definizioni in Scala: def square(x:Int):Int = x*x def sumOfSquare(x:Int,y:Int):Int = square(x)+square(y) Descrivere i passi di riduzione dell'espressione sumOfSquare(3,4) secondo una strategia call- by-value analoga a quella definita nella sezione precedente. Descrivere inoltre la riduzione della stesa espressione secondo la strategia call-by-name. Descrivere i passi di riduzione dell'espressione sumOfSquare(3,2+2) secondo le strategie call-by-value e call-by-name.

Svolgimento

Caso sumOfSquare(3,4).
CBV si comporta come CBN.

passo 1

$$(\text{BETA})\ \frac{}{\text{sumOfSquare}(3,\ 4) \to \text{square}(3) + \text{square}(4)}$$

passo 2

$$(\text{SUM-L})\ \frac{(\text{BETA})\ \frac{}{\text{square}(3) \to 3*3}}{\text{square}(3) + \text{square}(4) \to 3*3 + \text{square}(4)}$$

passo 3

$$(\text{SUM-L})\ \frac{(\text{MULT})\ \frac{}{3*3 \to 9}}{3*3 + \text{square}(4) \to 9 + \text{square}(4)}$$

passo 4

$$(\text{SUM-R})\ \frac{(\text{BETA})\ \frac{}{\text{square}(4) \to 4*4}}{9 + \text{square}(4) \to 9 + 4*4}$$

passo 5

$$(\text{SUM-R})\ \frac{(\text{MULT})\ \frac{}{4*4 \to 16}}{9 + 4*4 \to 9 + 16}$$

passo 6

$$(\text{SUM})\ \frac{}{9 + 16 \to 25}$$

Caso sumOfSquare(3,2+2)
CBV:
passo 1

$$(\text{APP2})\ \frac{(\text{SUM})\ \frac{}{2+2 \to 4}}{\text{sumOfSquare}(3,\ 2+2) \to \text{sumOfSquare}(3,\ 4)}$$

I passi successivi sono gli stessi del caso precedente, quindi i passi di esecuzione sono 7.

CBN:
passo 1

$$\text{(BETA)}\ \frac{}{\text{sumOfSquare(3, 2+2)} \rightarrow \text{square(3) + square(2+2)}}$$

passo 2

$$\text{(SUM-L)}\ \frac{\text{(BETA)}\ \frac{}{\text{square(3)} \rightarrow \text{3*3}}}{\text{square(3) + square(2+2)} \rightarrow \text{3*3 + square(2+2)}}$$

passo 3

$$\text{(SUM-L)}\ \frac{\text{(MULT)}\ \frac{}{\text{3*3} \rightarrow \text{9}}}{\text{3*3 + square(2+2)} \rightarrow \text{9 + square(2+2)}}$$

passo 4

$$\text{(SUM-R)}\ \frac{\text{(BETA)}\ \frac{}{\text{square(2+2)} \rightarrow \text{(2+2)*(2+2)}}}{\text{9 + square(2+2)} \rightarrow \text{9 + (2+2)*(2+2)}}$$

passo 5

$$\text{(SUM-R)}\ \frac{\text{(MULT-L)}\ \frac{\text{(SUM)}\ \frac{}{\text{2+2} \rightarrow \text{4}}}{\text{(2+2)*(2+2)} \rightarrow \text{(4)*(2+2)}}}{\text{9 + (2+2)*(2+2)} \rightarrow \text{9 + (4)*(2+2)}}$$

passo 6

$$\text{(SUM-R)}\ \frac{\text{(MULT-R)}\ \frac{\text{(SUM)}\ \frac{}{\text{2+2} \rightarrow \text{4}}}{\text{4*(2+2)} \rightarrow \text{4*4}}}{\text{9 + 4*(2+2)} \rightarrow \text{9 + 4*4}}$$

passo 7

$$\text{(SUM-R)}\ \frac{\text{(MULT)}\ \frac{}{\text{4*4} \rightarrow \text{16}}}{\text{9 + 4*4} \rightarrow \text{9 + 16}}$$

passo 8

$$\text{(SUM)}\ \frac{}{\text{9 + 16} \rightarrow \text{25}}$$

I passi di esecuzione sono 8 in questo caso (CBN).

Esercizio 1.7

Si consideri la seguente definizione in Scala: def test(x:Int, y:Int):Int = x*x confrontare la velocità (cioè il numero di passi) di riduzione delle seguenti espressioni secondo le strategie CBV e CBN, indicando quale delle due è piú veloce

1. test(2,3)
2. test(3+4,8)
3. test(7,2*4)
4. test(3+4,2*4)

Svolgimento

1. CBV=CBN: entrambi terminano l'esecuzione in 2 passi;
2. CBV<CBN: CBV termina prima perchè valuta l'espressione 3+4 subito (una sola volta). CBN invece invocherà prima la funzione, sostituendo ad x l'espressione. Essendo x presente 2 volte nella funzione, la valutazione dovrà essere effettuata 2 volte.
3. CBV>CBN: CBN termina prima perchè non valuta l'espressione 2*4 che non è utilizzata nella funzione test.
4. CBV=CBN: entrambi terminano l'esecuzione in 4 passi; la differenza sta nel fatto che con CBV si svolgeranno prima le due operazioni (somma e moltiplicazione) e poi si chiamerà la funzione. Con CBN invece, si eviterà di calcolare 2*4, tuttavia si calcolerà 3+4 due volte.

Esercizio 1.8

Definire in Scala una funzione and che si comporta come il costrutto logico $x\&\&y$.

Svolgimento

```
// and logico. b1 e b2 passati by-name
def and(b1:=> Boolean, b2:=> Boolean): Boolean =
   if(!b1) false
   else if(b2) true
       else false;
```

3 Il mini linguaggio funzionale (note 3)

Esempi di derivazione svolti

a) $\emptyset \vdash$ if true then $5 + 7$ else 2 :Nat

b) $\emptyset \vdash$ (fn x:T.x) x :T$\to$T

c) $\emptyset \vdash$ (fn x:Bool.x) true :Bool

d) f:Bool$\to$ Bool $\vdash$ f (if false then true else false) :Bool

e) f:Bool$\to$ Bool $\vdash$ fn x:Bool.f(if x then false else x) :Bool$\to$Bool

Svolgimento

a) $\emptyset \vdash$ if true then $5 + 7$ else 2 :Nat

$$(\text{IF-THEN-ELSE})\,\dfrac{(\text{TRUE})\,\dfrac{\checkmark}{\emptyset \vdash \text{true: Bool}} \qquad (\text{SUM})\,\dfrac{(\text{NAT})\,\dfrac{\checkmark}{\emptyset \vdash 5 :\text{Nat}} \quad (\text{NAT})\,\dfrac{\checkmark}{\emptyset \vdash 7 :\text{Nat}}}{\emptyset \vdash 5+7 :\text{Nat}} \qquad (\text{NAT})\,\dfrac{\checkmark}{\emptyset \vdash 2 :\text{Nat}}}{\emptyset \vdash \text{if true then } 5+7 \text{ else } 2 :\text{Nat}}$$

b) $\emptyset \vdash$ (fn x:T.x) x :T$\to$T

$$(\text{FUN})\,\dfrac{(\text{VAR})\,\dfrac{\text{x:T} \in \text{x:T}}{\text{x:T} \vdash \text{x :T}}}{\emptyset \vdash (\mathsf{fn}\ \text{x:T.x})\ \text{x :T}\to\text{T}}$$

c) $\emptyset \vdash$ (fn x:Bool.x) true :Bool

$$(\text{APP})\,\dfrac{(\text{FUN})\,\dfrac{(\text{VAR})\,\dfrac{\text{x:Bool} \in \text{x:Bool}}{\text{x:Bool} \vdash \text{x :Bool}}}{\emptyset \vdash \mathsf{fn}\ \text{x:Bool.x :Bool}/rightarrow\text{Bool}} \qquad (\text{TRUE})\,\dfrac{\checkmark}{\emptyset \vdash \text{true :Bool}}}{\emptyset \vdash (\mathsf{fn}\ \text{x:Bool.x})\ \text{true :Bool}}$$

d) f:Bool$\to$ Bool $\vdash$ f (if false then true else false) :Bool
Sia Γ = f:Bool$\to$Bool

$$
\text{(APP)}\ \frac{\text{(VAR)}\ \dfrac{\text{f:Bool} \to \text{Bool} \in \Gamma}{\Gamma \vdash \text{f :Bool} \to \text{Bool}} \qquad \text{(IF-THEN-ELSE)}\ \dfrac{\text{(FALSE)}\ \dfrac{\checkmark}{\Gamma \vdash \text{false :Bool}} \quad \text{(TRUE)}\ \dfrac{\checkmark}{\Gamma \vdash \text{true :Bool}} \quad \text{(FALSE)}\ \dfrac{\checkmark}{\Gamma \vdash \text{false :Bool}}}{\Gamma \vdash \text{if false then true else false :Bool}}}{\Gamma \vdash \text{f (if false then true else false):Bool}}
$$

e) f:Bool→ Bool ⊢ fn x:Bool.f(if x then false else x) :Bool→Bool Sia Γ = f:Bool→Bool

$$
\text{(APP)}\ \frac{\text{(FUN)}\ \dfrac{\text{(VAR)}\ \dfrac{\text{f:Bool} \to \text{Bool} \in \Gamma\text{, x:Bool}}{\Gamma\text{, x:Bool} \vdash \text{f: Bool} \to \text{Bool}}}{\Gamma \vdash \text{fn x:Bool.f: ?} \to \text{Bool} \to \text{Bool}} \qquad \text{(IF-THEN-ELSE)}\ \dfrac{\text{(VAR)}\ \dfrac{\text{x:Bool} \in \Gamma}{\Gamma \vdash \text{x :?}} \quad \text{(TRUE)}\ \dfrac{\checkmark}{\Gamma \vdash \text{true :Bool}} \quad \text{(FALSE)}\ \dfrac{\checkmark}{\Gamma \vdash \text{false :Bool}}}{\Gamma \vdash \text{(if x then false else x):?} \to \text{Bool}}}{\Gamma \vdash \text{fn x:Bool.f(if x then false else x) :Bool} \to \text{Bool}}
$$

Quindi ?=Bool.

Esercizio 2.1

Trovare un contesto Γ tale che Γ' f x y : Bool sia derivabile.

Svolgimento

$$
\text{(APP)}\ \frac{\text{(APP)}\ \dfrac{\text{(VAR)}\ \dfrac{\text{f:}T_2 \to T_1 \to \text{Bool} \in \Gamma}{\Gamma \vdash \text{f :}T_2 \to T_1 \to \text{Bool}} \qquad \text{(VAR)}\ \dfrac{\text{x:}T_2 \in \text{x:}T_2}{\Gamma \vdash \text{x :}T_2}}{\Gamma \vdash \text{f x :}T_1 \to \text{Bool}} \qquad \text{(VAR)}\ \dfrac{\text{y :}T_1 \in \Gamma}{\Gamma \vdash \text{y :}T_1}}{\Gamma \vdash \text{f x y :Bool}}
$$

Esiste un contesto per questo programma, in quanto il programma si puo' ottenere applicando le regole e costruendo l'albero di derivazione.
Fanno parte del contesto le seguenti regole di tipo:

- f:$T_2 \to T_1 \to$ Bool
- x :T_2
- y :T_1

Caso f (x y):

$$\text{(APP)}\;\dfrac{\text{(VAR)}\;\dfrac{\text{f} : T_1 \rightarrow \text{Bool} \in \Gamma}{\Gamma \vdash \text{f} : T_1 \rightarrow \text{Bool}} \qquad \text{(APP)}\;\dfrac{\text{(VAR)}\;\dfrac{\text{x} : T_2 \rightarrow T_1 \in \Gamma}{\Gamma \vdash \text{x} : T_2} \qquad \text{(VAR)}\;\dfrac{\text{y} : T_2 \in \Gamma}{\Gamma \vdash \text{y} : T_2}}{\Gamma \vdash \text{x y} : T_1}}{\Gamma \vdash \text{f (x y)} : \text{Bool}}$$

Esiste un contesto anche per questo programma.
Fanno parte del contesto le seguenti regole di tipo:

- f:$T_1 \rightarrow$ Bool
- x :$T_2 \rightarrow T_1$
- y :T_2

Esercizio 2.2

Il giudizio Γ’ x x : T e’ derivabile? Se si’, trovare una derivazione per qualche Γ, T, altrimenti provare che non e’ derivabile.

Svolgimento

$$\text{(APP)}\;\dfrac{\Gamma \vdash \text{x} : T_1 \rightarrow \text{T} \qquad \Gamma \vdash \text{x} : T_1}{\Gamma \vdash \text{x x} : \text{T}}$$

No, non e’ derivabile. Sarebbe necessario avere un sistema di tipi che ammetta il tipo ricorsivo.

Esercizio 3.1

Provare che ogni sottotermine di un termine ben tipato e ben tipato.

Svolgimento Per la risoluzione di questo esericizio e necessario utilizzare il metodo dell'induzione sui passi di derivazione, ovvero sull'altezza dell'albero di derivazione.

Caso 0 A questo livello il termine e congruo con il sottotermine di conseguenza se il termine e bentipato lo e anche il sottotermine

Caso $n+1$ A questo livello il sottotermine e dato dai sottotermini dei passi $0 ... n$ e di conseguenza se questi sottotermini sono ben tipati allora lo sara anche questo. Per induzione infatti sappiamo che se applico:

SUM fdfdsf

Esercizio 3.2

Dimostrare subject reduction per induzione sulla derivazione di $\Gamma \vdash$ M : T.

Esercizio 3.4

Vale l'opposto di subject reduction, i.e. se $\Gamma \vdash M'$:T e M $\rightarrow$ M', allora $\Gamma \vdash M : T$ (detto subject expansion)? Dimostrarlo oppure dare un controesempio.

Esercizio 3.5

Se al posto della regola (APP) si definisse la regola seguente:

$$\text{(APP')}\ \frac{\Gamma \vdash M\ :\mathrm{T} \rightarrow \mathrm{T} \qquad \Gamma \vdash N\ :\mathrm{T}}{\Gamma \vdash M\ N : T}$$

sarebbe ancora vero il teorema di safety?

Esercizio 3.6

Se al posto delle regole (APP) e (FUN) si definissero le regole seguenti regole:

$$\text{(APP')}\ \frac{\Gamma \vdash M\ :\mathrm{T} \rightarrow \mathrm{T} \qquad \Gamma \vdash N\ :\mathrm{T}}{\Gamma \vdash M\ N : T}$$

e

$$\text{(FUN')}\ \frac{\Gamma \text{ , x : } T1 \vdash M\ :\mathrm{T}}{\Gamma \vdash fn \text{ x : } T1.M :\rightarrow \mathrm{T}}$$

sarebbe ancora vero il teorema di safety?

Esercizio 3.6

Se si aggiungessero al sistema di tipi i seguenti due assiomi

$$\text{(TRUE')}\ \frac{}{\Gamma \vdash \text{true} : Nat}$$

e

$$\text{(FALSE')}\ \frac{}{\Gamma \vdash \text{false} : Nat}$$

sarebbe ancora vero il teorema di safety?

Esercizio 3.7

Dimostrare il seguente fatto: se $\Gamma \vdash M\ :\mathrm{T}$ e derivabile allora *fv(M)* $\subseteq$ *Dom(*Γ*)*

Esercizio 3.8

Ricostruire il tipo dei seguenti termini:

- $fn\ x{:}T1.fn\ y{:}T2.if\ y$ then x else true
- $fn\ x : Nat :\rightarrow$ Bool.x
- $fn\ f : T.fn\ x : T'.f$(if true then x else $f\ x$)
- $fn\ f : T1.fn\ g : T2.$if (f (g true))then f ($fn\ x$:T3.true) else $f(fn\ x : T4.x)$

4 Estensioni del linguaggio (note 6)

Esercizio 4.1

Discutere quali altri regole/strategie di riduzione sono possibili per i termini coppia.

Svolgimento

Si potrebbero modificare le regole PAIR 1 e PAIR 2 e mantenere le altre:

$$\text{(PAIR-NOT-EVAL 1)}\ \frac{}{(\mathrm{M}_1,\ \mathrm{M}_2).\_1 \to \mathrm{M}_1}$$

$$\text{(PAIR-NOT-EVAL 2)}\ \frac{}{(\mathrm{M}_1,\ \mathrm{M}_2).\_2 \to \mathrm{M}_2}$$

In questo modo non serve valutare i termini M_1 e M_2 prima di fare la proiezione. A seguito della proiezione, solo il termine estratto viene valutato. Le regole PROJECT 1 e PROJECT 2 vanno mantenute perche' il termine M potrebbe per esempio essere il risultato di una funzione ed e' necessario che ci siano delle regole di derivazione che portino ad uno stato in cui sono applicabili le due aggiunte. EVAL PAIR 1 ed EVAL PAIR 2, invece, vanno mantenute perche' la coppia, ammesso che contenga valori finali, e' un buon termine finale anch'essa. Di conseguenza, non e' necessario che venga estratto uno dei due valori perche' l'esecuzione si possa definire corretta e nel caso in cui ci si trovi come termine finale una coppia con valori non finali, e' necessaria una regola di valutazione che porti avanti l'esecuzione.

Esercizio 4.2

Scrivere la valutazione dei termini (4 − 1, if true then false else false)._1 e (fn x : Nat ∗ Nat.x._2) (4 − 2, 3 + 1).

Svolgimento

Termine 1: (4 − 1, if true then false else false)._1

$$\text{(PROJECT 1)}\ \frac{\text{(EVAL PAIR 1)}\ \dfrac{\text{(MINUS)}\ \dfrac{}{4-1 \to 3}}{(4-1,\ \text{if true then false else false}) \to (3,\ \text{if true then false else false})}}{(4-1,\ \text{if true then false else false}).\_1 \to (3,\ \text{if true then false else false}).\_1}$$

$$\text{(PROJECT 2)}\ \dfrac{\text{(EVAL PAIR 2)}\ \dfrac{\text{(IF-TRUE)}\ \dfrac{}{\text{if true then false else false} \to \text{false}}}{(3,\ \text{if true then false else false}) \to (3,\ \text{false})}}{(3,\ \text{if true then false else false}).\_1 \to (3,\ \text{false}).\_1}$$

$$\text{(PAIR 1)}\ \dfrac{}{(3,\ \text{false}).\_1 \to 3}$$

Termine 2: (fn x : Nat ∗ Nat.x._2) (4 − 2, 3 + 1)

$$\text{(APP2)}\ \dfrac{\text{(EVAL PAIR 1)}\ \dfrac{\text{(MINUS)}\ \dfrac{}{4 - 2 \to 2}}{(4 - 2,\ 3 + 1) \to (2,\ 3 + 1)}}{(\text{fn x : Nat} * \text{Nat.x.\_2})\ (4 - 2,\ 3 + 1) \to (\text{fn x : Nat} * \text{Nat.x.\_2})\ (2,\ 3 + 1)}$$

$$\text{(APP2)}\ \dfrac{\text{(EVAL PAIR 1)}\ \dfrac{\text{(SUM)}\ \dfrac{}{3 + 1 \to 4}}{(2,\ 3 + 1) \to (2,\ 4)}}{(\text{fn x : Nat} * \text{Nat.x.\_2})\ (2,\ 3 + 1) \to (\text{fn x : Nat} * \text{Nat.x.\_2})\ (2,\ 4)}$$

$$\text{(BETA)}\ \dfrac{}{(\text{fn x : Nat} * \text{Nat.x.\_2})\ (2,\ 4) \to (2,\ 4).\_2}$$

$$\text{(PAIR 2)}\ \dfrac{}{(2,\ 4).\_2 \to 4}$$

Esercizio 4.3

Dimostrare il teorema di safety per il linguaggio contenente interi, booleani, funzioni e records.

DA FARE

5 Estensioni del linguaggio (note 7)

Esercizio 5.1

Si ridimostri il teorema di safety per il linguaggio contenente interi, booleani, funzioni, records e tipi varianti. **DA FARE**

6 Eccezioni (note 8)

Esercizio 6.1

Si dia la semantica operazionale dei termini indicati sopra, osservando come il sollevamento delle eccezioni comporti un salto non locale del flusso di controllo. Sia M = fn x.(if pari(x) then x/2 else throw x)

- try (M 3) catch fn y.y + y
- try (fn y.y - 2 (M 5)) catch fn z.print(z)
- try (fn y.y - 2 (M throw 2)) catch fn z.print(z)
- try (fn y.y - 2 (try (M 5) catch fn z.z+z)) catch fn z.print(z)
- try ((fn x.x throw 1) (try (M 5) catch fn z.z+z)) catch fn z.print(z)

Svolgimento

a) try (M 3) catch fn y.y + y

passo 1

$$(\text{TRY})\ \dfrac{\dfrac{(\text{BETA})}{\text{M 3 = fn x.(if pari(x) then x/2 else throw x) 3} \rightarrow \text{if pari(3) then 3/2 else throw 3}}}{\text{try (M 3) catch fn y.y + y} \rightarrow \text{try (if pari(3) then 3/2 else throw 3) catch fn y.y + y}}$$

passo 2

$$(\text{TRY})\ \dfrac{(\text{IF})\ \dfrac{\dfrac{(\text{BETA})}{\text{pari(3)} \rightarrow \text{false}}}{\text{if pari(3) then 3/2 else throw 3} \rightarrow \text{if false then 3/2 else throw 3}}}{\text{try (if pari(3) then 3/2 else throw 3) catch fn y.y + y} \rightarrow \text{try (if false then 3/2 else throw 3) catch fn y.y + y}}$$

passo 3

$$(\text{TRY})\ \dfrac{\dfrac{(\text{IF-FALSE})}{\text{if false then 3/2 else throw 3} \rightarrow \text{throw 3}}}{\text{try (if false then 3/2 else throw 3) catch fn y.y + y} \rightarrow \text{try (throw 3) catch fn y.y + y}}$$

passo 4

$$\dfrac{(\text{TRY HANDLE})}{\text{try (throw 3) catch fn y.y + y} \rightarrow \text{fn y.y + y 3}}$$

passo 5

$$\frac{\text{(BETA)}}{\text{fn y.y + y 3} \to 3 + 3}$$

passo 6

$$\frac{\text{(SUM)}}{3 + 3 \to 6}$$

b) try (fn y.y - 2 (M 5)) catch fn z.print(z)

passo 1

$$\text{(TRY)}\ \dfrac{\text{(APP 2)}\ \dfrac{\dfrac{\text{(BETA)}}{\text{M 5 = fn x.(if pari(x) then x/2 else throw x) 5} \to \text{if pari(5) then 5/2 else throw 5}}}{\text{fn y.y - 2 (M 5)} \to \text{fn y.y - 2 (M}\{x=5\})}}{\text{try (fn y.y - 2 (M 5)) catch fn z.print(z)} \to \text{try (fn y.y - 2 (M}\{x=5\})) \text{ catch fn z.print(z)}}$$

passo 2

$$\text{(TRY)}\ \dfrac{\text{(APP 2)}\ \dfrac{\text{(IF)}\ \dfrac{\dfrac{\text{(BETA)}}{\text{pari(5)} \to \text{false}}}{(\text{M}\{x=5\}) = \text{if pari(5) then 5/2 else throw 5} \to \text{if false then 5/2 else throw 5}}}{\text{fn y.y - 2 (M}\{x=5\}) \to \text{fn y.y - 2 (M'}\{x=5\})}}{\text{try (fn y.y - 2 (M}\{x=5\})) \text{ catch fn z.print(z)} \to \text{try (fn y.y - 2 (M'}\{x=5\})) \text{ catch fn z.print(z)}}$$

passo 3

$$\text{(TRY)}\ \dfrac{\text{(APP 2)}\ \dfrac{\dfrac{\text{(IF-FALSE)}}{(\text{M'}\{x=5\}) = \text{if false then 5/2 else throw 5} \to \text{throw 5}}}{\text{fn y.y - 2 (M'}\{x=5\}) \to \text{fn y.y - 2 (throw 5)}}}{\text{try (fn y.y - 2 (M'}\{x=5\})) \text{ catch fn z.print(z)} \to \text{try (fn y.y - 2 (throw 5)) catch fn z.print(z)}}$$

passo 4

$$\text{(TRY)}\ \dfrac{\dfrac{\text{(RAISE APP 2)}}{\text{fn y.y - 2 (throw 5)} \to \text{(throw 5)}}}{\text{try (fn y.y - 2 (throw 5)) catch fn z.print(z)} \to \text{try (throw 5) catch fn z.print(z)}}$$

passo 5

$$\dfrac{\text{(TRY HANDLE)}}{\text{try (throw 5) catch fn z.print(z)} \rightarrow \text{fn z.print(z) 5}}$$

passo 6

$$\dfrac{\text{(BETA)}}{\text{fn z.print(z) 5} \rightarrow \text{print(5)}}$$

passo 7

$$\dfrac{\text{(BETA)}}{\text{print(5)} \rightarrow \text{"stampa 5"}}$$

c) try (fn y.y - 2 (M throw 2)) catch fn z.print(z)

passo 1

$$\text{(TRY)}\ \dfrac{\text{(APP 2)}\ \dfrac{\dfrac{\text{(RAISE APP 2)}}{\text{M throw 2} = \text{fn x.(if pari(x) then x/2 else throw x) throw 2} \rightarrow \text{throw 2}}}{\text{fn y.y - 2 (M throw 2)} \rightarrow \text{fn y.y - 2 (throw 2)}}}{\text{try (fn y.y - 2 (M throw 2)) catch fn z.print(z)} \rightarrow \text{try (fn y.y - 2 (throw 2)) catch fn z.print(z)}}$$

passo 2

$$\text{(TRY)}\ \dfrac{\dfrac{\text{(RAISE APP 2)}}{\text{fn y.y - 2 (throw 2)} \rightarrow \text{throw 2}}}{\text{try (fn y.y - 2 (throw 2)) catch fn z.print(z)} \rightarrow \text{try (throw 2) catch fn z.print(z)}}$$

passo 3

$$\dfrac{\text{(TRY HANDLE)}}{\text{try (throw 2) catch fn z.print(z)} \rightarrow \text{fn z.print(z) 2}}$$

passo 4

$$\dfrac{\text{(BETA)}}{\text{fn z.print(z) 2} \rightarrow \text{print(2)}}$$

passo 5

$$\dfrac{\text{(BETA)}}{\text{print(2)} \rightarrow \text{"stampa 2"}}$$

d) try (fn y.y - 2 (try (M 5) catch fn z.z+z)) catch fn z.print(z)

passo 1

$$\text{(TRY)}\ \dfrac{\text{(APP 2)}\ \dfrac{\text{(TRY)}\ \dfrac{\dfrac{\text{(BETA)}}{\text{M 5} = \text{fn x.(if pari(x) then x/2 else throw x) 5} \rightarrow \text{if pari(5) then 5/2 else throw 5}}}{\text{try (M 5) catch fn z.z+z} \rightarrow \text{try (M\{x=5\}) catch fn z.z+z}}}{\text{(fn y.y - 2 (try (M 5) catch fn z.z+z))} \rightarrow \text{(fn y.y - 2 (try (M\{x=5\}) catch fn z.z+z))}}}{\text{try (fn y.y - 2 (try (M 5) catch fn z.z+z)) catch fn z.print(z)} \rightarrow \text{try (fn y.y - 2 (try (M\{x=5\}) catch fn z.z+z)) catch fn z.print(z)}}$$

passo 2

$$\text{(TRY)}\ \dfrac{\text{(APP 2)}\ \dfrac{\text{(TRY)}\ \dfrac{\text{(IF)}\ \dfrac{\dfrac{\text{(BETA)}}{\text{pari(5)} \rightarrow \text{false}}}{\text{M\{x=5\}} = \text{if pari(5) then 5/2 else throw x} \rightarrow \text{if false then 5/2 else throw 5}}}{\text{try (M\{x=5\}) catch fn z.z+z} \rightarrow \text{try (M'\{x=5\}) catch fn z.z+z}}}{\text{(fn y.y - 2 (try (M\{x=5\}) catch fn z.z+z))} \rightarrow \text{(fn y.y - 2 (try (M'\{x=5\}) catch fn z.z+z))}}}{\text{try (fn y.y - 2 (try (M\{x=5\}) catch fn z.z+z)) catch fn z.print(z)} \rightarrow \text{try (fn y.y - 2 (try (M'\{x=5\}) catch fn z.z+z)) catch fn z.print(z)}}$$

passo 3

$$\text{(TRY)}\ \dfrac{\text{(APP 2)}\ \dfrac{\text{(TRY)}\ \dfrac{\dfrac{\text{(IF-FALSE)}}{\text{M'\{x=5\}} = \text{if false then 5/2 else throw x} \rightarrow \text{throw 5}}}{\text{try (M'\{x=5\}) catch fn z.z+z} \rightarrow \text{try (throw 5) catch fn z.z+z}}}{\text{(fn y.y - 2 (try (M'\{x=5\})))} \rightarrow \text{(fn y.y - 2 (try (throw 5) catch fn z.z+z))}}}{\text{try (fn y.y - 2 (try (M'\{x=5\}) catch fn z.z+z)) catch fn z.print(z)} \rightarrow \text{try (fn y.y - 2 (try (throw 5) catch fn z.z+z)) catch fn z.print(z)}}$$

passo 4

$$\text{(TRY)}\ \dfrac{\text{(APP 2)}\ \dfrac{\dfrac{\text{(TRY HANDLE)}}{\text{try (throw 5) catch fn z.z+z} \rightarrow \text{5+5}}}{\text{(fn y.y - 2 (try (throw 5) catch fn z.z+z))} \rightarrow \text{fn y.y - 2 (5+5)}}}{\text{try (fn y.y - 2 (try (throw 5) catch fn z.z+z)) catch fn z.print(z)} \rightarrow \text{try (fn y.y - 2 (5+5)) catch fn z.print(z)}}$$

passo 5

$$\text{(TRY)}\ \dfrac{\text{(APP 2)}\ \dfrac{\dfrac{\text{(SUM)}}{\text{5+5} \rightarrow \text{10}}}{\text{fn y.y - 2 (5+5)} \rightarrow \text{fn y.y - 2 (10)}}}{\text{try (fn y.y - 2 (5+5)) catch fn z.print(z)} \rightarrow \text{try (fn y.y - 2 (10)) catch fn z.print(z)}}$$

passo 6

$$\text{(TRY)}\ \dfrac{\dfrac{\text{(BETA)}}{\text{fn y.y - 2 (10)} \rightarrow \text{10-2}}}{\text{try (fn y.y - 2 (10)) catch fn z.print(z)} \rightarrow \text{try (10-2) catch fn z.print(z)}}$$

passo 7

$$\text{(TRY)}\ \dfrac{\dfrac{\text{(MINUS)}}{\text{10-2} \rightarrow \text{8}}}{\text{try (10-2) catch fn z.print(z)} \rightarrow \text{try (8) catch fn z.print(z)}}$$

passo 8

$$\frac{\text{(TRY VAL)}}{\text{try (8) catch fn z.print(z)} \rightarrow 8}$$

e) try ((fn x.x throw 1) (try (M 5) catch fn z.z+z)) catch fn z.print(z)

passo 1

$$\text{(TRY)}\ \frac{\text{(APP 1)}\ \dfrac{\dfrac{\text{(RAISE APP 2)}}{\text{fn x.x throw 1} \rightarrow \text{throw 1}}}{\text{(fn x.x throw 1) (try (M 5) catch fn z.z+z)} \rightarrow \text{(throw 1) (try (M 5) catch fn z.z+z)}}}{\text{try ((fn x.x throw 1) (try (M 5) catch fn z.z+z)) catch fn z.print(z)} \rightarrow \text{try ((throw 1) (try (M 5) catch fn z.z+z)) catch fn z.print(z)}}$$

passo 2

$$\text{(TRY)}\ \frac{\dfrac{\text{(RAISE APP 1)}}{\text{(throw 1) (try (M 5) catch fn z.z+z)} \rightarrow \text{throw 1}}}{\text{try ((throw 1) (try (M 5) catch fn z.z+z)) catch fn z.print(z)} \rightarrow \text{try (throw 1) catch fn z.print(z)}}$$

passo 3

$$\frac{\text{(TRY HANDLE)}}{\text{try (throw 1) catch fn z.print(z)} \rightarrow \text{fn z.print(z) 1}}$$

passo 4

$$\frac{\text{(BETA)}}{\text{fn z.print(z) 1} \rightarrow \text{print(1)}}$$

passo 5

$$\frac{\text{(BETA)}}{\text{print(1)} \rightarrow \text{"stampa 1"}}$$

Esercizio 6.2

Si definisca la semantica operazionale per il liguaggio esteso con i costrutti visti in precedenza: unit, records e varianti.

Svolgimento

Unit: Non si aggiungono regole per gestire questo tipo.

Record:

$$\frac{\text{(RAISE PROJECTION I)}}{\text{throw v.i} \rightarrow \text{throw v}}$$

Varianti:

$$\frac{\text{(RAISE MATCH)}}{< l_j = v_j > \text{match } \{ \text{ throw v } \} \rightarrow \text{throw v}}$$

$$\frac{\text{(RAISE RED MATCH)}}{\text{throw v match } \{ \text{ case } l_i = x_i => M_i^{\text{i} \in 1..\text{n}} \ \} \rightarrow \text{throw v}}$$

DA FINIRE?

Esercizio 6.3

teorema di safety DA FARE? Non lo so, Giacomo mi ha detto che non gli sembra

7 Subtyping (note 9)

Esercizio 7.1

Scrivere le derivazioni dei giudizi:

- {l:{a:Nat, b:Nat}, l':{m:Nat}} <: {l:{a:Nat}, l':{}}
- {l:{a:Nat, b:Nat}, l':{m:Nat}} <: {l:{a:Nat}, l':{m:Nat}}
- {l:{a:Nat, b:Nat}, l':{m:Nat}} <: {l:{a:Nat}}

Svolgimento

A): {l:{a:Nat, b:Nat}, l':{m:Nat}} <: {l:{a:Nat}, l':{}}

$$\text{(SUB-DEPTH)}\,\dfrac{\dfrac{\text{(SUB-WIDTH)}}{\emptyset \vdash \text{l:\{a:Nat, b:Nat\}} <: \text{l:\{a:Nat\}}} \qquad \dfrac{\text{(SUB-WIDTH)}}{\emptyset \vdash \text{l':\{m:Nat\}} <: \text{l':\{\}}}}{\emptyset \vdash \text{\{l:\{a:Nat, b:Nat\}, l':\{m:Nat\}\}} <: \text{\{l:\{a:Nat\}, l':\{\}\}}}$$

B): {l:{a:Nat, b:Nat}, l':{m:Nat}} <: {l:{a:Nat}, l':{m:Nat}}

$$\text{(SUB-DEPTH)}\,\dfrac{\dfrac{\text{(SUB-WIDTH)}}{\emptyset \vdash \text{l:\{a:Nat, b:Nat\}} <: \text{l:\{a:Nat\}}} \qquad \dfrac{\text{(REFLEX)}}{\emptyset \vdash \text{l':\{m:Nat\}} <: \text{l':\{m:Nat\}}}}{\emptyset \vdash \text{\{l:\{a:Nat, b:Nat\}, l':\{m:Nat\}\}} <: \text{\{l:\{a:Nat\}, l':\{m:Nat\}\}}}$$

C): {l:{a:Nat, b:Nat}, l':{m:Nat}} <: {l:{a:Nat}}

$$\text{(TRANS)}\,\dfrac{\text{(SUB-DEPTH)}\,\dfrac{\dfrac{\text{(SUB-WIDTH)}}{\emptyset \vdash \text{l:\{a:Nat, b:Nat\}} <: \text{l:\{a:Nat\}}} \qquad \dfrac{\text{(SUB-WIDTH)}}{\emptyset \vdash \text{l':\{m:Nat\}} <: \text{l':\{\}}}}{\emptyset \vdash \text{\{l:\{a:Nat, b:Nat\}, l':\{m:Nat\}\}} <: \text{\{l:\{a:Nat\}, l':\{\}\}}} \qquad \dfrac{\text{(SUB-WIDTH)}}{\emptyset \vdash \text{\{l:\{a:Nat\}, l':\{\}\}} <: \text{\{l:\{a:Nat\}\}}}}{\emptyset \vdash \text{\{l:\{a:Nat, b:Nat\}, l':\{m:Nat\}\}} <: \text{\{l:\{a:Nat\}\}}}$$

Esercizio 7.2

Si scriva la derivazione di {a:Nat, b:Bool, c:Nat} <: {b:Bool}

Svolgimento

$$\text{(TRANS)}\ \dfrac{\text{PERMUTE}\ \dfrac{\{\text{a:Nat, b:Bool, c:Nat}\}\ \text{e' permutazione di}\ \{\text{b:Bool, a:Nat, c:Nat}\}}{\emptyset \vdash \{\text{a:Nat, b:Bool, c:Nat}\} <: \{\text{b:Bool, a:Nat, c:Nat}\}} \qquad \dfrac{\text{(SUB-WIDTH)}}{\emptyset \vdash \{\text{b:Bool, a:Nat, c:Nat}\} <: \{\text{b:Bool}\}}}{\emptyset \vdash \{\text{a:Nat, b:Bool, c:Nat}\} <: \{\text{b:Bool}\}}$$

Esercizio 7.3

Dare la derivazione del giudizio $\emptyset \vdash$ (fn r:{l:Nat}.r.l + 2) {l= 0, l'= 1}:Nat. Esiste una sola derivazione di questo giudizio?

Svolgimento

$$\text{(APP)}\ \dfrac{\text{(FUN)}\ \dfrac{\text{(SUM)}\ \dfrac{\text{(VAR)}\ \dfrac{\text{r.l:Nat} \in \Gamma}{\emptyset, \text{r:}\{\text{l:Nat}\} \vdash \text{r.l:Nat}} \qquad \dfrac{\text{(NAT)}}{\emptyset, \text{r:}\{\text{l:Nat}\} \vdash \text{2:Nat}}}{\emptyset, \text{r:}\{\text{l:Nat}\} \vdash \text{(r.l + 2):Nat}}}{\emptyset \vdash \text{(fn r:}\{\text{l:Nat}\}\text{.r.l + 2):}\{\text{l:Nat}\} \to \text{Nat}} \qquad \text{(SUBSUMPTION)}\ \dfrac{\text{(TYPE-RECORD)}\ \dfrac{\dfrac{\text{(NAT)}}{\emptyset \vdash \text{0:Nat}} \qquad \dfrac{\text{(NAT)}}{\emptyset \vdash \text{1:Nat}}}{\emptyset \vdash \{\text{l=0, l'=1}\}\text{:}\{\text{l:Nat, l':Nat}\}} \qquad \dfrac{\text{(SUBWIDTH)}}{\{\text{l:Nat, l':Nat}\} <: \{\text{l:Nat}\}}}{\emptyset \vdash \{\text{l=0, l'=1}\}\text{:}\{\text{l:Nat}\}}}{\emptyset \vdash \text{(fn r:}\{\text{l:Nat}\}\text{.r.l + 2)}\ \{\text{l= 0, l'= 1}\}\text{:Nat}}$$

Esercizio 7.4

Quale potrebbbe essere la relazione di sottotipo dei variant types?

Svolgimento

$$\text{(SUBTYPE VARIANT)}\ \dfrac{\Gamma \vdash \text{M} : T_j \qquad \text{j} \in \text{1..n+k}}{\Gamma \vdash <l_j = \text{M}> \ : \ <l_i \ : \ T_i^{\text{i} \in \text{1..n}}>}$$

In questo modo, modificando TYPE VARIANT, si ottiene il subtyping in larghezza (WIDTH). Per ottenere il subtyping in profondita' (DEPTH) basta usare SUBSUMPTION.

Esercizio 7.5

Quali proprieta' del sistema di tipi si perderebbero se avessimo definito la relazione di subtyping con una regola di troppo? E se l'avessimo definita con una regola in meno? E se l'avessimo definita con una regola in meno? Se avessimo definito le regole in modo tale che {l : Nat} <: {l : Nat, l' : Nat}, quale proprieta' del sistema non sarebbe piu' vera? Identificarla e darne un controesempio. **DA FARE**

Esercizio 7.12

Ridimostrare il teorema di progressione, preservazione, il substitution lemma e il teorema di safety per il linguaggio con i record e il subtyping. **DA FARE**

Esercizio 7.16

Trovare due termini M e N tali che $M \to N$, $\Gamma \vdash M : T$, $\Gamma \vdash N : S$ con S<::T e T$\not<$::S, cioe' esibire un caso in cui il tipo di un termine decresce durante la computazione. **DA FARE**

8 Featherweight Java (note 11)

Esercizio 8.1

Si noti che una class table puo' contenere definizioni di classi mutuamente ricorsive. Scrivere un esempio.

DA FARE

Esercizio 8.2

Descrivere la semantica operazionale dei seguenti termini:

- new Pair(new A(), new B()).snd
- (Pair)new Pair(new A(), new B())
- new Pair(new A(), new B()).setfst(new B())
- ((Pair) (new Pair(new Pair(new A(), new B()), new A()).fst)).snd
- (B) ((A)new C())

DA FARE

Esercizio 8.3

Scrivere un programma con override di un metodo e descriverne la valutazione, evidenziando il binding dinamico per la chiamata del metodo riscritto.

DA FARE

Esercizio 8.4

Perche' c'e' una regola di tipo sia per l'upcast che per il downcast, mentre c'e' la sola regola di valutazione per upcast, nella semantica operazionale? **DA FARE**

Esercizio 8.5

Ha senso aggiungere a FJ la regola di subtyping per i tipi freccia A $\rightarrow$ B?

DA FARE

Esercizio 8.12

Aggiungere a FJ il termine ClassCastException: come cambia la semantica operazionale? Le regole di tipo? Il teorema di Safety e i teoremi di Preservazione e Progressione? Aggiungere in seguito anche la possibilita' di gestire le eccezioni.

DA FARE

9 Imperative Featherweight Java (note 12)

Esercizio 9.2

Scrivere una regola analoga per eliminare i riferimenti non piu' usati. **DA FARE**

Esercizio 9.3

Descrivere il comportamento del seguente programma:

```
class D extends Object {
   Object f;
   D(Object f) { super(); this.f=f;}
   Object m() { return this; }
}

class C extends D {
   C(Object f) { super(f); }
   Object m() {return this;}
}

Object z=new Object();
C x=new C(z);
C y=new D(x);
x.m(); x=y; x.m()
y.f=new Object();
z=null; x.f=z; y.f.m();
```

DA FARE

Esercizio 9.4

Dare una derivazione di tipo per il termine C X=O_1; O_1.f=y; X=O_2 **DA FARE**