Aspetti Avanzati dei Linguaggi
di Programmazione

Esercizi - Aspetti avanzati di Ling. di prog.

Autori:
Stefano Campese
Luca Costantino
Enrico Savoca

Il documento è stato realizzato da Stefano Campese, Luca Costantino 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 tutti gli autori del documento e la versione ritenuta corretta di ognuno di essi è stata riportata nel presente testo.

Contents

1 Il mini linguaggio funzionale (note 2)

Esercizio 1.1

La relazione di riduzione data definisce una strategia efficiente per la valutazione del termine 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

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

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

Regole

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

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

Esercizio 1.4

La valutazione, cioè l'esecuzione del programma, è deterministica. Se $M \rightarrow$ M' e $M \rightarrow$ 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):

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

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

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

  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

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

passo 2

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

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:

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

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

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

passo 2

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

passo 3

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

passo 4

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

passo 5

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

passo 6

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

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

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

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

CBN:
passo 1

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

passo 2

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

passo 3

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

passo 4

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

passo 5

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

passo 6

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

passo 7

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

passo 8

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

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 più 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 invoca prima la funzione, sostituendo ad x l'espressione. Essendo x presente 2 volte nella funzione, la valutazione viene 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 svolgono prima le due operazioni (somma e moltiplicazione) e poi si chiama la funzione. Con CBN invece, si evita di calcolare 2*4, tuttavia si calcola 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;
```

Il parametro b1 viene passato by-name nella nostra soluzione. Tuttavia, esso può essere passato anche by-value visto che in ogni caso viene valutato.

2 I tipi semplici (note 3)

Esempi di derivazione svolti

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

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

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

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

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

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

$$\text{(APP)}\ \dfrac{\text{(FUN)}\ \dfrac{\text{(VAR)}\ \dfrac{\text{x:Bool} \in \text{x:Bool}}{\text{x:Bool} \vdash \text{x :Bool}}}{\emptyset \vdash \mathsf{fn}\ \text{x:Bool.x :Bool}\to\text{Bool}} \quad \text{(TRUE)}\ \dfrac{\checkmark}{\emptyset \vdash \text{true :Bool}}}{\emptyset \vdash (\mathsf{fn}\ \text{x:Bool.x) true :Bool}}$$

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

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

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

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

Quindi ?=Bool.

Esercizio 2.1

Trovare un contesto Γ tale che $\Gamma \vdash$ f x y : Bool sia derivabile.

Svolgimento

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

Esiste un contesto per questo programma, in quanto il programma si può 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):

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

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

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

Esercizio 2.2

Svolgimento

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

No, non è derivabile. Nel nostro sistema di tipi x non può assumere contemporaneamente il tipo T e il tipo $T_1 \to$ T. Risulterebbe derivabile se fosse presente il tipo ricorsivo nel sistema di tipi.

3 I Tipi Semplici (note 45)

Esercizio 3.2

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

Svolgimento Per la risoluzione di questo esercizio è necessario utilizzare il metodo dell'induzione sui passi di derivazione, ovvero sull'altezza dell'albero di derivazione.
Se M è un termine ben tipato vuol dire che il giudizio di tipo $\Gamma \vdash M : T$ è derivabile con un albero di derivazione di altezza $h+1$ e quindi l'induzione viene fatta su $h+1$.

Caso Base $h = 1$ A questo livello il termine e il sottotermine coincidono. Questo perchè l'albero consiste nell'applicazione di un assioma e quindi il termine M non contiene sotto-termini, quindi la proprietà è verificata.

Caso Induttivo $h+1$ Questo caso è composto dall'applicazione della regola e dai vari sotto-alberi che provano le ipotesi della regola.
I sotto-alberi in questione, riguardano il giudizio dei sotto-termini di M e sono di altezza inferiore rispetto a questo albero (h). Possiamo quindi applicare l'ipotesi induttiva per definire che i sotto-termini dei sotto-termini di M sono ben tipati e quindi di conseguenza lo sono anche i sotto-termini di M. A questo punto la proprietà risulta essere verificata.

È possibile inoltre, utilizzare un altra dimostrazione che sfrutta la *struttura di M* e il *Lemma di Inversione*.

- ⋆ $M \equiv v$: In questo caso M è un valore, come ad esempio $true$, $false$ o n e quindi, poichè questi sono termini privi di sotto-termini la proprietà risulta essere verificata.
- ⋆ $M \equiv A + B$: In questo caso, per il Lemma di Inversione se M è ben tipato, allora lo sono anche A eB (tipo Nat)
- ⋆ $M \equiv A - B$: questo caso è analogo al precedente
- ⋆ $M \equiv if\ C\ then\ A\ else\ B$: In questo caso, per il Lemma di Inversione se M è ben tipato, allora C è ben tipato con tipo $Bool$ mentre A e B sono ben tipati con tipo T
- ⋆ $M \equiv A\ B$: In questo caso, per il Lemma di Inversione se M è ben tipato, allora A è ben tipato con tipo $T_1 \rightarrow T$ mentre B è ben tipato con tipo T
- ⋆ $M \equiv fn\ x.N$: In questo caso, per il Lemma di Inversione N è ben tipato con tipo T

Esercizio 3.12

Dimostrare subject reduction per induzione sulla derivazione di Γ ⊢ M : T.

Svolgimento Si vuole quindi dimostrare che:

$$\text{Se } \Gamma \vdash M : T \text{ e } M \to M', \text{ allora } \Gamma \vdash M' : T.$$

True $\Gamma \vdash M : T$ é $\Gamma \vdash \textsf{true} : \textsf{Bool}$. Il teorema é vacuamente vero per questa derivazione siccome $\nexists M'.M \to M'$.

False $\Gamma \vdash M : T$ é $\Gamma \vdash \textsf{false} : \textsf{Bool}$. Analogo a True.

Nat $\Gamma \vdash M : T$ é $\Gamma \vdash n : \textsf{Nat}$. Analogo a True.

Var $\Gamma \vdash M : T$ é $\Gamma \vdash x : T$. Analogo a True.

Fun $\Gamma \vdash M : T$ é $\Gamma \vdash fnx\, T_1C : T_1 \to T_2$. Analogo a True.

Sum $\Gamma \vdash M : T$ é $\Gamma \vdash A + B : \textsf{Nat}$.

Per quanto assunto dalla regola di tipo *Sum*, sappiamo per certo che anche A e B hanno tipo $\textsf{Nat}$ sotto contesto Γ.

Supponiamo che $A+B \to M'$ (sappiamo che M' é unico poiché l'esecuzione é deterministica).Tale riduzione puó essere:

- *Sum*: $M' \equiv n$. Per la regola di tipo *Nat*, si ha che $T \equiv \textsf{Nat}$ e che $\Gamma \vdash M' : \textsf{Nat}$;
- *Sum-Left*: $M' \equiv A' + B$. Per gli assunti della regola *Sum* ho che $\Gamma \vdash A : \textsf{Nat}$ e per *Sum-Left* $A \to A'$. Posso quindi applicare l'ipotesi induttiva[1] per ottenere il giudizio $\Gamma \vdash A' : \textsf{Nat}$. Sono quindi soddisfatte le premesse della regola *Sum* per M' e quindi $\Gamma \vdash M' : \textsf{Nat} \equiv T$;
- *Sum-Right*: $M' \equiv v + B'$. Il ragionamento é analogo al caso *Sum-Left*.

Minus $\Gamma \vdash M : T$ é $\Gamma \vdash A - B : \textsf{Nat}$. Analogo a *Sum*.

IfThenElse $\Gamma \vdash M : T$ é $\Gamma \vdash \textsf{if } M_1 \textsf{ then } M_2 \textsf{ else } M_3$.

Dal giudizio di tipo abbiamo che:

- $\Gamma \vdash M_1 : \textsf{Bool}$;
- $\Gamma \vdash M_2 : T$;
- $\Gamma \vdash M_3 : T$.

[1]perch é la derivazione $\Gamma \vdash A : \textsf{Nat}$ richiede meno passaggi

Supponiamo che $\text{if } M_1 \text{ then } M_2 \text{ else } M_3 \to M'$ (sappiamo che M' é unico poiché l'esecuzione é deterministica). Tale riduzione puó essere:

- (If): $M' \equiv \Gamma \vdash \text{if } M_1' \text{ then } M_2 \text{ else } M_3$. Dato che il giudizio $\Gamma \vdash M_1 : \text{ Bool }$ richiede una derivazione in meno e che $M_1 \to M_1'$, possiamo applicare l'ipotesi induttiva per ottenere il giudizio $\Gamma \vdash M_1' : \text{ Bool }$. Si ha quindi che la regola di tipo (IfThenElse) continua a valere per il termine M' e dato che i termini M_2 e M_3 non sono cambiati, anche M' ha tipo T;
- (If-True): $M' \equiv M_2$. M_2 ha tipo T e quindi anche M' ha tipo T;
- (If-False): $M' \equiv M_3$. Analogo a (If-True)

App $\Gamma \vdash M : T$ é $\Gamma \vdash A\ B : T_2$. Dal giudizio di tipo abbiamo che:

- $\Gamma \vdash A : T_1 \to T_2$;
- $\Gamma \vdash B : T_1$.

Supponiamo che $A\ B \to M'$ (sappiamo che M' é unico poiché l'esecuzione é deterministica). Tale riduzione puó essere:

- (Beta): $M' \equiv \Gamma \vdash Z[x \coloneqq B]$[2]. Per la regola di tipo (Fun) ho che $Z : T_2$ e quindi, per il Substitution Lemma, $\Gamma \vdash Z[x \coloneqq B] : T_2$. Per questo motivo si ha che $\Gamma \vdash M' : T_2$ con $T_2 \equiv T$;
- (App-1): $M' \equiv A'\ B$. Dato che il giudizio $\Gamma \vdash A : T_1 \to T_2$ richiede una derivazione con meno passaggi e che $A \to A'$ posso applicare l'ipotesi induttiva per ottenere il giudizio $A' : T_1 \to T_2$. Il termine M' soddisfa le premesse della regola di tipo (App) e quindi M' ha tipo $T_2 \equiv T$.
- (App-2): $M' \equiv v\ B'$. Analogo ad (App-1).

Esercizio 3.13

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

Svolgimento La risposta è no:

È sufficiente pensare a $M \to M'$ con $\Gamma \vdash M' : T$ e che $\Gamma \nvdash M : T$
Quanto riportato sopra può essere semplificato tramite questo esempio:

[2] dato $A \equiv FNxT_1Z$

Supponiamo di avere $M' \equiv A : T$ e $M \equiv if\ true\ then\ A\ else\ B$, ovvero il caso in cui $M \to M'$ per la regola (If-True).
In questo caso la derivazione di $\Gamma \vdash M : T$ riesce ad ottenere usando la regola IF_THEN_ELSE, tuttavia questo impone che nel contesto ci siano questi giudizi:

- $\Gamma \vdash true{:}Bool$ che è un assioma
- $\Gamma \vdash A{:}T$ soddisfatto per l'ipotesi
- $\Gamma \vdash B{:}T$ purtroppo questa informazione non si riesce a reperire dall'ipotesi ne dal contesto e pertanto non c'è garanzia sull'uguaglianza di tipi di A e B
  Pertanto possiamo creare un termine N come $if\ true\ then\ 4\ else\ false$, da cui si riesce ad attenere $M' \equiv A \equiv 4$. Questo termine nonostante esegua risulta essere non ben-tipato.

Esercizio 3.14

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

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

sarebbe ancora vero il teorema di safety?

Svolgimento Si, lo si può dimostrare considerando che APP' non è altro che una specificazione di APP: se poniamo X uguale all'insieme dei tipi di APP, Y è un sottoinsieme di X corrispondente ai possibili tipi di APP'.
In sostanza la regola restringere l'insieme dei programmi realizzabili con il precedente type system

Di conseguenza il teorema di safety vale anche per APP' poichè la dimostrazioni continuano a valere.

Esercizio 3.15

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

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

e

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

sarebbe ancora vero il teorema di safety?

Svolgimento La risposta e' no:
In questo caso la FUN' non fa alcun controllo sul tipo dell'argomento e quindi esiste la possibilità di arrivare ad un passo della derivazione che produce un termine STUCK, invalidando il teorema stesso. Questo perchè la nuova regola risulta essere più stringente rispetto a prima visto che ammette solo tipi uguali al tipo di ritorno senza controllare effettivamente che la signatura dei parametri sia conforme a quanto richiesto.

Supponiamo di avere il termine:
$M = (fn\ x{:}Bool.if\ x\ then\ 1\ else\ 0)\ (1)$

e creiamo il suo albero di derivazione:

$$
(APP')\dfrac{
(FUN')\dfrac{
(IF-THEN-ELSE)\dfrac{
\dfrac{\dfrac{\checkmark}{x{:}Bool \in \Gamma}}{\Gamma \vdash x{:}Bool}(Var) \quad (Nat)\dfrac{\checkmark}{\Gamma \vdash 1{:}Nat} \quad (Nat)\dfrac{\checkmark}{\Gamma \vdash 0{:}Nat}
}{x{:}Bool \vdash if\ x\ then\ 1\ else\ 0 :Nat}
}{\vdash fn\ x{:}Bool.if\ x\ then\ 1\ else\ 0 :Nat \to :Nat}
\qquad (Nat)\dfrac{\checkmark}{\vdash 1{:}Nat}
}{\vdash (fn\ x{:}Bool.if\ x\ then\ 1\ else\ 0)\ (1)}
$$

In questo caso possiamo osservare che il termine M risulta essere ben-tipato secondo le nuove regole di tipo.
Tuttavia, è evidente che l'invocazione della funzione in questione porta M ad un termine stuck, questo perchè il sottotermine $N = if\ x\ then\ 1\ else\ 0$ si aspetta un $Bool$, infatti le regole della semantica operazionale che gestiscono l'if, permettono di avanzare nella computazione.
Nel caso in cui la guardia non sia un valore, la computazione procede con la valutazione della guardia, altrimenti procede il ramo $TRUE$ o con il ramo $FALSE$.
In questo caso, poichè il valore passato alla funzione è un intero, e quindi già un valore le uniche regole applicabili sarebbero $(IF - TRUE)$ e $(IF - FALSE)$ ma poichè la guardia é 1 non sono applicabili e quindi si otterrebbe un termine STUCK.

Esercizio 3.16

Se si aggiungessero al sistema di tipi i seguenti due assiomi

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

e

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

sarebbe ancora vero il teorema di safety?

Svolgimento La risposta è no:
Data la definizione di Teorema di Safety, è evidente che viene meno la stessa ipotesi ovvero che M sia un valore o un che non evolva in un termine STUCK.

Con queste regole di tipo, infatti, sarebbe permessa anche l'operazione SUM che genererebbe un termine STUCK, invalidando il teorema stesso. Supponiamo di avere il termine: $M = true + 4$

$$(SUM)\ \frac{(Nat)\ \frac{\checkmark}{\vdash 4{:}Nat} \qquad (Nat)\ \frac{\checkmark}{\vdash true{:}Nat}}{\vdash true + 4}$$

Anche in questo caso, come nel precedete, notiamo che le regole di tipo permettono la derivazione dell'albero.
Tuttavia, se applicassimo le regole della semantica operazionale non potremo avanzare perchè entrambi gli addendi sono giá dei valori e quindi non posso applicare $SUM - LEFT$ o $SUM - RIGHT$ ma non posso neppure applicare SUM perché genero un termine STUCK in quanto *true* non puó essere sommato a 4.

Esercizio 3.17

Dimostrare il seguente fatto: se $\Gamma \vdash M : T$ è derivabile, allora *fv(M)* $\subseteq$ *Dom(*Γ*)*

Svolgimento Si procede con la dimostrazione per induzione:

- Caso Base
  - ⋆ **True** $\Gamma \vdash M : T$ é $\Gamma \vdash \ \mathsf{true} \ : \ \mathsf{Bool}$. Il teorema é vacuamente vero per questa derivazione siccome $\nexists M'.M \to M'$.
    Il tutto si può riassumere con $fv(M) = fv(true) = \emptyset \subseteq Dom(\Gamma)$
  - ⋆ **False** $\Gamma \vdash M : T$ é $\Gamma \vdash \ \mathsf{false} \ : \ \mathsf{Bool}$. Analogo a True.
  - ⋆ **Nat** $\Gamma \vdash M : T$ é $\Gamma \vdash n : \ \mathsf{Nat}$. Analogo a True.
  - ⋆ **Var** Se $\Gamma \vdash x : T$, allora per la regola (Var) possiamo asserire che $x : T \in \Gamma$ e di conseguenza abbiamo che $fv(M) = fv(x) = \{x\} \subseteq Dom(\Gamma)$

  In questo passo il fatto è un assioma, questo perchè non esistono variabili libere quindi $fv(M)$ è $\emptyset$ e il vuoto e un sottoinsieme del $Dom(\Gamma)$

- Caso Induttivo
  - **SUM** in questo caso $M \equiv A + B$ e quindi per definizione le variabili libere di M sono f $v(A + B) = f\ v(A) \cup f\ v(B)$.
    Per ipotesi induttiva sappiamo che $fv(A)$ e $fv(B) \subseteq$ *Dom(*Γ*)*.
    Poichè, come anticipato, $fv(M) \equiv fv(A) \cup fv(B)$ possiamo dedurre che $fv(M) \subseteq$ *Dom(*Γ*)*.
  - **MINUS** in questo caso $M \equiv A - B$ e la sua dimostrazione risulta essere analoga a quella di SUM
  - **IF-THEN-ELSE** in questo caso $M \equiv if\ C\ then\ A\ else\ B$ e quindi per definizione, le variabili libere di M sono $fv(M) = fv(A) \cup fv(B) \cup fv(C)$.
    Per ipotesi induttiva sappiamo che $fv(A)$, $fv(B)$ e $fv(C) \subseteq$ *Dom(*Γ*)*.
    Poichè, come anticipato, $fv(M) \equiv fv(A) \cup fv(B) \cup fv(C)$ possiamo dedurre che $fv(M)$ $\subseteq$ *Dom(*Γ*)*.
  - **FUN** in questo caso $M \equiv fn\ x.N$ e quindi per definizione le variabili libere di M sono $fv(M) = fv(N) \backslash \{x\}$ $(fv(M) \subseteq fv(N))$.
    Per ipotesi induttiva sappiamo che $fv(N) \subseteq$ *Dom(*Γ*)*.
    Poichè, come anticipato, $fv(M) \equiv fv(N) \backslash \{x\}$ possiamo dedurre che $fv(M) \subseteq$ *Dom(*Γ*)*.
  - **APP** in questo caso $M \equiv A\ B$ e quindi per definizione, le variabili libere di M sono $fv(M)$ $= fv(A) \cup fv(B)$.
    Per ipotesi induttiva sappiamo che $fv(A)$ e $fv(B) \subseteq$ *Dom(*Γ*)*.
    Poichè, come anticipato, $fv(M) \equiv fv(A) \cup fv(B)$ possiamo dedurre che $fv(M) \subseteq$ *Dom(*Γ*)*.

Esercizio 3.18

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 $f(fn\ x : T4.x)$

Svolgimento

a) $fn\ x{:}T_1.fn\ y{:}T_2.if\ y$ then x else true

$$
\text{(FUN)}\dfrac{\text{(FUN)}\dfrac{\text{(IF-THEN-ELSE)}\dfrac{\text{(VAR)}\dfrac{\text{y:Bool} \in \Gamma}{\Gamma, x{:}T_1, y{:}T_2 \vdash y : Bool} \quad \text{(VAR)}\dfrac{x{:}T_3 \in \Gamma}{\Gamma, x{:}T_1, y{:}T_2 \vdash x : T_3} \quad \text{(TRUE)}\dfrac{\checkmark}{\Gamma, x{:}T_1, y{:}T_2 \vdash true : Bool}}{x{:}T_1, y{:}T_2 \vdash if\ y\ then\ x\ else\ true : T_3}}{x{:}T_1 \vdash fn\ y{:}T_2.if\ y\ then\ x\ else\ true : T_2 \to T_3}}{\emptyset \vdash fn\ x{:}T_1.fn\ y{:}T_2.if\ y\ then\ x\ else\ true : T_1 \to T_2 \to T_3}
$$

Il risultato finale è dato dal fatto che $T_1 = T_2 = T_3 = Bool$ e quindi il tipo finale risulta: $Bool \to Bool \to Bool$

b) $fn\ x : Nat :\to Bool.x$

$$
\text{(FUN)}\dfrac{\text{(VAR)}\dfrac{x : T_2 \in \Gamma,\ x{:}Nat :\to Bool}{\Gamma,\ x{:}Nat :\to Bool \vdash x{:}T_2}}{\emptyset \vdash fn\ x : Nat :\to Bool.x : T_1 \to T_2}
$$

Il risultato finale è dato dal fatto che $T_2 = Nat \to Bool$ e quindi il tipo finale risulta: $(Nat \to Bool) \to (Nat \to Bool)$

c) $fn\ f : T.fn\ x : T'.f(if\ true\ then\ x\ else\ f\ x)$

$$
\text{(FUN)}\dfrac{\text{(FUN)}\dfrac{\text{(APP)}\dfrac{\dfrac{\text{A}}{f{:}T, x{:}T' \vdash f{:}T'''' \to T''} \qquad \dfrac{\text{B}}{f{:}T, x{:}T' \vdash if true \text{ then } x \text{ else } f\ x : T'''}}{f{:}T, x{:}T' \vdash f\ (if\ true\ then\ x\ else\ f\ x)}}{f{:}T \vdash fn\ x{:}T'.f(if\ true\ then\ x\ else\ f\ x){:}T' \to T''}}{\emptyset \vdash fn\ f : T.fn\ x : T'.f(if\ true\ then\ x\ else\ f\ x) : T \to T' \to T''}
$$

- ⋆ Albero A

$$\text{(VAR)}\ \frac{f{:}T'''' \to T'' \vdash \Gamma}{f{:}T{,}x{:}T' \vdash f{:}T'''' \to T''}$$

⋆ Albero B

$$\text{(IF-THEN-ELSE)}\ \frac{\text{(TRUE)}\ \dfrac{\checkmark}{f{:}T{,}x{:}T' \vdash true : Bool} \quad \text{(VAR)}\ \dfrac{x : T'' \in \Gamma}{f{:}T{,}x{:}T' \vdash x : T''} \quad \text{(APP)}\ \dfrac{\text{C}}{f{:}T{,}x{:}T' \vdash fx : T''}}{f{:}T{,}x{:}T' \vdash if\ true\ then\ x\ else\ f\ x : T''}$$

⋆ Albero C (ramo else dell'if)

$$\frac{\text{(VAR)}\ \dfrac{f : T''' \in \Gamma}{f{:}T{,}x{:}T' \vdash f : T''' \to T''} \quad \text{(VAR)}\ \dfrac{x : T''' \in \Gamma}{f{:}T{,}x{:}T' \vdash x : T'''}}{f{:}T{,}x{:}T' \vdash fx : T''}$$

il risultato finale è supportato dalle seguenti sostituzioni:

- $T \to T' \to T''$
- $T' \to T''$
- $T''' \to T''$
- $T' = T'''$
- $T = T''''$
- $T'''' = T'''$
- $T = T'''$
- $T'' = T'''$

Il risultato finale è dato dalle sostituzioni precedenti e quindi il tipo è:
$(T \to T) \to (T \to T)$

d) $fn\ f : T1.fn\ g : T2.if\ (f\ (g\ true))\ then\ f\ (fn\ x : T3.true)\ else\ f(fn\ x : T4.x)$

$$\text{(FUN)}\,\dfrac{\text{(FUN)}\,\dfrac{\text{(IF-THEN-ELSE)}\,\dfrac{\dfrac{\text{A}}{f{:}T_1,g{:}T_2 \vdash f\ (g\ true))\ :\ Bool} \quad \dfrac{\text{B}}{f{:}T_1,g{:}T_2 \vdash f\ (fn\ x{:}T_3.true){:}W} \quad \dfrac{\text{C}}{f{:}T_1,g{:}T_2 \vdash f(fn\ x : T_4.x){:}W}}{f{:}T_1,g{:}T_2 \vdash if\ (f\ (g\ true))\ then\ f\ (fn\ x : T_3.true)\ else\ f(fn\ x : T_4.x){:}W}}{f{:}T_1 \vdash fn\ g : T_2.if\ (f\ (g\ true))\ then\ f\ (fn\ x : T_3.true)\ else\ f(fn\ x : T_4.x){:}T_2 \rightarrow W}}{\emptyset \vdash fn\ f : T_1.fn\ g : T_2.if\ (f\ (g\ true))\ then\ f\ (fn\ x : T_3.true)\ else\ f(fn\ x : T_4.x){:}T_1 \rightarrow T_2 \rightarrow W}$$

Dove gli alberi A, B e C sono:

- ⋆ Albero A (derivazione guardia del'if)

$$\text{(APP)}\,\dfrac{\text{(VAR)}\,\dfrac{f{:}U \rightarrow Bool \in f{:}T_1,g{:}T_2}{f{:}T_1,g{:}T_2 \vdash f{:}U \rightarrow Bool} \quad \text{(APP)}\,\dfrac{\text{(VAR)}\,\dfrac{g{:}Bool \rightarrow U \in f{:}T_1,g{:}T_2}{f{:}T_1,g{:}T_2 \vdash g{:}Bool \rightarrow U} \quad \text{(TRUE)}\,\dfrac{\checkmark}{f{:}T_1,g{:}T_2 \vdash true{:}Bool}}{f{:}T_1,g{:}T_2 \vdash g\ true{:}U}}{f{:}T_1,g{:}T_2 \vdash f\ (g\ true))\ :\ Bool}$$

- ⋆ Albero B (derivazione ramo then)

$$\text{(APP)}\,\dfrac{\text{(VAR)}\,\dfrac{f : Z \rightarrow W \in f{:}T_1,g{:}T_2}{f{:}T_1,g{:}T_2 \vdash f\ Z \rightarrow W} \quad \text{(FUN)}\,\dfrac{\text{(TRUE)}\,\dfrac{\checkmark}{f{:}T_1,g{:}T_2,x{:}T_3 \vdash true\ :\ Bool}}{f{:}T_1,g{:}T_2 \vdash fn\ x{:}T_3.true\ :\ T_3 \rightarrow Bool}}{f{:}T_1,g{:}T_2 \vdash f\ (fn\ x{:}T_3.true)\ {:}W}$$

- ⋆ Albero C (derivazione ramo else)

$$\text{(APP)}\,\dfrac{\text{(VAR)}\,\dfrac{f : Z_1 \rightarrow W \in f{:}T_1,g{:}T_2}{f{:}T_1,g{:}T_2 \vdash f\ Z_1 \rightarrow W} \quad \text{(FUN)}\,\dfrac{\text{(VAR)}\,\dfrac{x{:}T_4 \in \Gamma}{f{:}T_1,g{:}T_2,x{:}T_4 \vdash x\ :\ T_4}}{f{:}T_1,g{:}T_2 \vdash fn\ x{:}T_4.x\ :\ T_4 \rightarrow T_4}}{f{:}T_1,g{:}T_2 \vdash f(fn\ x : T_4.x){:}W}$$

Il risultato finale è supportato dalle seguenti sostituzioni:

- $T_1 = U \rightarrow Bool$
- $T_1 = Z \rightarrow Bool$
- $T_1 = Z_1 \rightarrow Bool$
- $T_2 = Bool \rightarrow U$
- $W = Bool$
- $U = Bool \rightarrow Bool$
- $T_3 = T_4 = Bool$
- quindi $T_2 = Bool \rightarrow (Bool \rightarrow Boool)$ (per sostituzione infatti f $(fn\ x{:}T_3.true)$ è $U \rightarrow W$ con $W = Bool$ e $U = (Bool \rightarrow Bool)$)
- quindi $T_1 = (Bool \rightarrow Bool) \rightarrow Boool$
- quindi da $T_1 \rightarrow T_2 \rightarrow W$ otteniamo $(Bool \rightarrow Bool) \rightarrow Boool \rightarrow (Bool \rightarrow (Bool \rightarrow Boool)) \rightarrow Bool$

e quindi il tipo del termine è quindi:
$(Bool \rightarrow Bool) \rightarrow Boool \rightarrow (Bool \rightarrow (Bool \rightarrow Boool)) \rightarrow Bool$

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:

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

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

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 perchè il termine M potrebbe per esempio essere il risultato di una funzione ed è 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 perchè la coppia, ammesso che contenga valori finali, è un buon termine finale anch'essa. Di conseguenza, non è necessario che venga estratto uno dei due valori perchè l'esecuzione si possa definire corretta e nel caso in cui ci si trovi come termine finale una coppia con valori non finali, è 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

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

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

$$\text{(PAIR 1)}\ \cfrac{}{\text{(3, false).\_1} \rightarrow 3}$$

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

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

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

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

$$\text{(PAIR 2)}\ \cfrac{}{(2, 4).\_2 \rightarrow 4}$$

Esercizio 4.3

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

Svolgimento

Per dimostrare che il teorema di Safety continua a valere anche per l'estensione del linguaggio con i records è necessario andare ad estendere i teoremi di Progressione e di Subject-Reduction.
Per rendere più semplice la dimostrazione, inoltre, é necessario anche aggiornare il Lemma di Inversione.

Teorema di Progressione Se M è un termine chiuso e ben tipato ($\emptyset \vdash M : T$), allora o M è un valore oppure esiste M' tale che $M \rightarrow M'$.

Per poter procedere con la dimostrazione del teorema di Progressione è necessario aggiungere alla dimostrazione due nuovi casi induttivi, mentre i casi per le altre regole di tipo restano invariati e continuano a valere.

- (Type-Record) Se $M = \{l_i = M_i\ ^{i=1...n}\}$ e $\emptyset \vdash M : T$, sappiamo che per la regola (Type-Record) valgono i giudizi di tipo $\emptyset \vdash M_i : T_i \,\forall\, i = 1 \ldots n$ e che gli alberi di derivazione relativi ai vari giudizi sono di altezza inferiore all'albero del giudizio principale.
  È quindi possibile applicare l'ipotesi induttiva sui sotto-termini M_i, i quali o sono un valore v_i di tipo T_i oppure possono avanzare in un termine M_i':
  - Se sono tutti dei valori si ha $M = \{l_i = v_i\ ^{i=1...n}\}$, ovvero M è un valore record di tipo $\{l_i : T_i\ ^{i=1...n}\}$ e quindi il teorema di Progressione continua a valere banalmente.
  - Se c'è almeno un M_i che non è un valore, è possibile identificare il termine M_j di indice minimo che non è un valore e per il quale continua a valere il giudizio di tipo $\emptyset \vdash M_j : T_j$. Vale quindi l'ipotesi induttiva, ovvero $\exists M_j'$ tale che $M_j \to M_j'$. Posso quindi applicare la regola (Eval-Record) per far avanzare il termine M al termine M', dove al posto di M_j compare M_j'. Anche in questo caso il teorema di Progressione continua a valere.
- (Type-Select) Se $M = N.l_j$ e $\emptyset \vdash M : T$, sappiamo che per la regola (Type-Select) vale il giudizio di tipo $\emptyset \vdash N : \{l_i : T_i\ ^{i=1...n}\}$ (con un albero di derivazione più piccolo, ovvero il passo precendete) e che $j \in \{1 \ldots n\}$. Si ha quindi che N può essere:
  - un valore di tipo record, ovvero $N = \{l_i = v_i\ ^{i=1...n}\}$ e quindi $M = \{l_i = v_i\ ^{i=1...n}\}.l_j$ con $j \in 1 \ldots n$. In questo caso il termine M può avanzare nel termine $M' = v_j$ perchè sono soddisfatte le premesse della regola (Select) e quindi il teorema di Progressione continua a valere.
  - un record non ancora completamente valutato, ovvero $N = \{l_i = v_i\ ^{i=1...x-1}, l_x = M_x, l_k = M_k\ ^{k=x+1...n}\}, \exists M_x' : M_x \to M_x'$ e quindi per (Eval-Record) $N \to N'$ e pertanto anche $M \to M' = N'.l_j$.
  - $N = \mathsf{if}\ M_1\ \mathsf{then}\ M_2\ \mathsf{else}\ M_3$ oppure $N = A\ B$. In entrambi i casi vale il giudizio $\emptyset \vdash N : \{l_i : T_i\ ^{i=1...n}\}$, ovvero N è un termine chiuso, ben tipato e non è un valore. Quindi per ipotesi induttiva ho che $\exists N'$ tale che $N \to N'$ e quindi per (Eval-Select) $\exists M' = N'.l_j$ tale che $M \to M'$. Pertanto il teorema di Progressione continua a valere.

Lemma di Inversione

- (Type-Record): Se $\Gamma \vdash M = \{l_i = M_i\ ^{i=1...n}\} : \{l_i : T_i\ ^{i=1...n}\}$ è derivabile, allora $\forall i \in 1 \ldots n . \Gamma \vdash M_i : T_i$ è derivabile.
- (Type-Select): Se $\Gamma \vdash M.l_j : T_j$ è derivabile, allora $\forall i \in 1 \ldots n . \exists T_i . \Gamma \vdash M : \{l_i : T_i\ ^{i=1...n}\}$ è derivabile e $j \in 1 \ldots n$.

La dimostrazione di questi casi è banale perchè derivano dall'applicazione delle regole di tipo.

Teorema di Subject-Reduction (Preservazione) Se $\Gamma \vdash M : T$ e $M \to M'$, allora $\Gamma \vdash M' : T$.

Anche per questo teorema è necessario aggiungere un nuovo caso base per l'assioma (SELECT) e due nuovi casi induttivi per (EVAL-SELECT) e (EVAL-RECORD).

(SELECT): $M \equiv N.l_j$ con $N = \{l_i = v_i\ ^{i=1\ldots n}\} : \{l_i : T_i\ ^{i=1\ldots n}\}$. Per la regola di tipo (TYPE-SELECT) M ha tipo $T \equiv T_j$ per $j \in 1 \ldots n$. Oltre a ciò $M' = v_j$ per $j \in 1 \ldots n$ e, per la regola (SELECT), v_j è il valore associato all'etichetta l_j di N. Quindi, per il Lemma di Inversione esteso, dato che $\Gamma \vdash N : \{l_i : T_i\ ^{i=1\ldots n}\}$ è derivabile, sono anche derivabili i giudizi $\Gamma \vdash M_i : T_i\ \forall i \in 1 \ldots n$. In particolare, è derivabile il giudizio $\Gamma \vdash v_j : T_j$ e quindi $M' : T_j$, con $T_j \equiv T$.

(EVAL-SELECT): $M \equiv N.l$ ed è derivabile il giudizio $\Gamma \vdash M \equiv N.l : T$. Inoltre, $N \to N'$ e $M' \equiv N'.l$ con l appartenente all'insieme delle etichette del record. Per il Lemma di Inversione esteso, ho che $\Gamma \vdash N : \{l_i : T_i\ ^{i=1\ldots n}\}$ è derivabile con un albero di derivazione di altezza inferiore. Posso quindi applicare l'ipotesi induttiva per dire che anche $\Gamma \vdash N' : \{l_i : T_i\ ^{i=1\ldots n}\}$ è derivabile, per poi concludere applicando la regola (TYPE-SELECT) a $M' \equiv N'.l$, ottenendo che $\Gamma \vdash N'.l : T$ è derivabile e quindi anche $\Gamma \vdash M' : T$ è derivabile.

(EVAL-RECORD): Essendo $M \equiv \{l_i = v_i\ ^{i=1\ldots j-1}, l_j = M_j, l_k = M_k\ ^{k=j+1\ldots n}\}$, è derivabile il giudizio $\Gamma \vdash M : T \equiv \{l_i : T_i^{\,i=1\ldots n}\}$ e $M_j \to M_j'$. Per il primo caso del Lemma di Inversione esteso, ho che tutti i giudizi $\Gamma \vdash M_i : T_i$ sono derivabili ed in particolare è derivabile $\Gamma \vdash M_j : T_j$. Per ipotesi induttiva sull'altezza dell'albero $M_j \to M_j'$, ottengo che è derivabile anche il giudizio $\Gamma \vdash M_j' : T_j$. Infine, dato che $M' \equiv \{l_i = v_i\ ^{i=1\ldots j-1}, l_j = M_j', l_k = M_k\ ^{k=j+1\ldots n}\}$, posso applicare la regola (TYPE-RECORD) per dire che $\Gamma \vdash M' : \{l_i : T_i\ ^{i=1\ldots n}\} \equiv T$ è derivabile e quindi anche $\Gamma \vdash M' : T$ è derivabile.

Teorema di Safety Se $\emptyset \vdash M : T$ e $M \to^* M'$ con M' tale che $M' \not\to$, allora M' è un valore.

La dimostrazione è una diretta conseguenza dei due teoremi precedenti: per il teorema di Preservazione anche $\emptyset \vdash M' : T$ è derivabile e, per il teorema di Progressione M' deve essere un valore, perché altrimenti esisterebbe un M'' al quale può ridursi.

5 Eccezioni (note 8)

Esercizio 5.1

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

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

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

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

Al passo 4 si effettua un salto non locale del flusso di controllo a causa dell'eccezione.

passo 5

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

passo 6

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

L'eccezione lanciata dal throw x in M viene raccolta dall'unico catch presente.

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

passo 1

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

passo 2

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

passo 3

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

passo 4

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

Al passo 4 si effettua un salto non locale del flusso di controllo a causa dell'eccezione.

passo 5

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

passo 6

$$\frac{\text{(BETA)}}{\text{fn z.print(z) 5} \to \text{print(5)}}$$

passo 7

$$\frac{\text{(BETA)}}{\text{print(5)} \to \text{"stampa 5"}}$$

L'eccezione lanciata dal throw x in M viene raccolta dall'unico catch presente.

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

passo 1

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

passo 2

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

Ai passi 1 e 2 si effettuano dei salti non locali del flusso di controllo a causa dell'eccezione.

passo 3

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

passo 4

$$\frac{\text{(BETA)}}{\text{fn z.print(z) 2} \to \text{print(2)}}$$

passo 5

$$\frac{(\text{BETA})}{\text{print(2)} \to \text{"stampa 2"}}$$

L'eccezione lanciata da throw 2 viene raccolta dall'unico catch presente.

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

passo 1

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

passo 2

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

passo 3

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

passo 4

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

Al passo 4 si effettua un salto non locale del flusso di controllo a causa dell'eccezione.

passo 5

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

passo 6

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

passo 7

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

passo 8

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

L'eccezione lanciata da throw x in M viene raccolta dal primo catch (quello più vicino).

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

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

Ai passi 1 e 2 si effettuano dei salti non locali del flusso di controllo a causa dell'eccezione.

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

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

L'eccezione lanciata da throw 1 viene raccolta dall'ultimo catch (in ordine di lettura).

Esercizio 5.2

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, visto che sono state aggiunti solo un valore e un termine.

Record:

$$\frac{\text{(RAISE EVAL SELECT)}}{\text{throw v.i} \rightarrow \text{throw v}}$$

La regola SELECT va bene così e basta per tirare fuori i lanci di eccezione che comunque vengono valutati in seguito tramite altre regole.

$$\frac{\text{(RAISE EVAL RECORD)}}{\{\ l_i = x_i{}^{\text{i} \in \text{1..j-1}},\ l_j\text{=throw v},\ l_k = x_k{}^{\text{k} \in \text{j+1..n}}\ \} \rightarrow \text{throw v}}$$

Varianti:

La regola RAISE MATCH non è necessaria, in quanto si valuta il valore del variante. Se contiene un lancio di un'eccezione si valuta con RAISE RED MATCH. Se il lancio è all'interno del match, allora dev'essere necessariamente all'interno di M_i e viene valutato solo quando si esegue il contenuto di M_i.

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

$$\frac{\text{(RAISE VARIANT)}}{<\text{l=throw v}> \rightarrow \text{throw v}}$$

6 Subtyping (note 9)

Esercizio 6.1

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':{}}

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

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

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

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

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

Esercizio 6.2

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

Svolgimento

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

Esercizio 6.3

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

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

Esistono altre derivazioni di questo giudizio; infatti, in ogni momento posso applicare la regola SUBSUMPTION. Esempio di alternativa:

$$\text{(SUBSUMPTION)}\ \dfrac{\dfrac{\text{(A)}}{\emptyset \vdash \text{(fn r:}\{\text{l:Nat}\}\text{.r.l + 2) }\{\text{l= 0, l'= 1}\}\text{:Nat}} \qquad \dfrac{\text{(REFLEX)}}{\text{Nat<:Nat}}}{\emptyset \vdash \text{(fn r:}\{\text{l:Nat}\}\text{.r.l + 2) }\{\text{l= 0, l'= 1}\}\text{:Nat}}$$

Dove A è la derivazione vista precedentemente. Potendo applicare SUBSUMPTION in qualsiasi punto della derivazione, anche più volte, ne consegue che il numero di derivazioni possibili è infinito.

Esercizio 6.4

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 profondità (DEPTH) basta usare SUBSUMPTION.

Esercizio 6.5

Quali proprietà 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? Se avessimo definito le regole in modo tale che {l : Nat} <: {l : Nat, l' : Nat}, quale proprietà del sistema non sarebbe più vera? Identificarla e darne un controesempio.

Svolgimento

Regola di troppo Aggiungendo una regola in più, come ad esempio la regola per che rendere derivabile il giudizio:

$\{l{:}Nat\} <: \{l{:}Nat, l_1{:}Nat\}$

ovvero la regola

$$\text{(SUB-WIDTH-INVERSE)}\ \frac{}{\{l_i : T_i\ i \in 1, ..., n\} < :\{l_i : T_i\ i \in 1, ..., n+k\}}$$

Tale regola può portare alla generazione di un termine STUCK invalidando il teorema di Safety.

Regola in meno Gli effetti che si otterrebbero con una regola in meno sono diversi e dipendono dalla regola che viene tolta.

- (SUBSUMPTION): la rimozione di questa regola comporta la perdita della possibilità di usare un sottotipo dove viene utilizzato il sopra-tipo.
- (REFLEX): la rimozione di questa regola comporterebbe degli assurdi, come ad esempio l'impossibilità di poter derivare la regola SUB-DEPTH per record i cui tipi dei valori siano uguali, in quanto un tipo T generico non sarebbe più sotto-tipo di se stesso.In sostanza avremo una situazione di questo tipo: $T \not<: T$.
- (TRANS): la rimozione di questa regola comporterebbe la rimozione della possibilità di rendere transitiva la proprietà e la relazione di subtyping impedendo di avere realazioni di subtyping a più livelli. Poniamo U e S sottotipi di un tipo generico T, se mancasse tale regola non si riuscirebbe a derivare un giudizio del tipo $S <:T$ sulla base di $S <:U$ e $U <:T$. In sostanza la derivazione tramite l'utilizzo di tipi intermedi non sarebbe più sopportata e quindi la gerarchia di tipo verrebbe meno.
- (SUB-WIDTH): la rimozione di questa regola renderebbe limiterebbe la relazione di subtyping per quanto riguarda i record. Questa regola infatti permette di una relazione di subtyping tra il record $\{l{:}Nat, l_1{:}Nat\}$ e il record $\{l{:}Nat\}$ permettendo di utilizzare il primo (specifico) al posto del secondo (generale). La mancanza di tale regola, come detto, limita fortemente il

subtyping per i record perchè obbliga sempre l'utilizzo di record della stessa *larghezza*, tuttavia la relazione di subtyping tra record, continua ad esistere in altre forme, come quelle garantite dall SUB-DEPTH.

- (SUB-DEPTH): la rimozione di questa regola, come nel caso precedente, limita la relazione di subtyping dei record, anche se più debolmente rispetto alla precedente. Senza questa regola, la relazione di subtyping tra record non potrebbe più appoggiarsi sulla relazione di tipo esistente tra i corrispettivi tipi dei valori dei record. Supponendo di avere $S <: T$, un record $\{l{:}S\}$ e un record $\{l{:}T\}$, nonstante il primo record possa potenzialmente essere utilizzato ovunque venga utilizzato il secondo, la mancanza di questa regola ne impedisce di derivare il giudizio che ne garantirebbe il funzionamento.
- (PERMUTE): la rimozione di questa regola, elimina la possibilità di avere giudizi come ad esmepio $\{a{:}Nat, b{:}Nat\} <: \{b{:}Nat\}$, questo perchè senza questa regola la relazione di suptyping risulta essere posizionale e non gestirebbe il sottotipo con permutazione degli elementi del record.
- (ARROW): la rimozione di questa regola le funzioni accetterebbero solo il tipo T scritto nella signatura della funzione stessa senza accettare un sottotipo del tipo specificato. Inoltre il tipo di ritorno della funzione sarebbe sempre quello piú specifico. In sostanza il sottotipaggio per le funzioni non potrebbe esistere.

Regola invertita Se fosse definita la relazione secondo quanto scritto, esisterebbe la regola di SUB-WIDTH-INVERSE ovvero:

$$\text{(SUB-WIDTH-INVERSE)}\ \frac{}{\{l_i : T_i\ i \in 1, ..., n\} < :\{l_i : T_i\ i \in 1, ..., n+k\}}$$

Tale regola permetterebbe di usare record del tipo $\{l{:}Nat\}$ al posto di $\{l{:}Nat, l_1{:}Nat\}$ cosi facendo si potrebbero creare situazioni come ad esempio:

$(fn\ x : \{l{:}Nat, l_1{:}Nat\}.x.l_1 + 1)\ (\{l = 1\})$

Questa funzione andrebbe a generare un termine STUCK invalidando di fatto il teorema di Safety:

Sia M un termine chiuso e ben tipato. Allora M non evolve ad un termine stuck, ma $\exists v$ tale che $M \to * v$ oppure $M \to *$ throw v.

Esercizio 6.12

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

Svolgimento

(Lemmi di inversione)

- Se $\Gamma \vdash x : T$ allora $\exists S$ tale che $x : S \in \Gamma$e $S <: T$.
- Se $\Gamma \vdash fnx : S1M : T$ è derivabile, allora $\exists T_1, T_2$ tali che $T = T_1 \rightarrow T_2$, con $T_1 <: S_1$ e $\Gamma, x : S_1 \vdash M : T_2$.
- Se $\Gamma \vdash \{k_r = M_r \ ^{r \in 1 \ldots m}\} : T$ è derivabile allora $\exists T_i, i = 1, \ldots, n$ tali che $T = \{\ell_i : T_i \ ^{i \in 1 \ldots n}\}$, con $\{\ell_i \ ^{i \in 1 \ldots n}\} \subseteq \{k_r \ ^{r \in 1 \ldots m}\}$ e $\Gamma \vdash M_r : T_i$ per ogni etichetta comune $k_r = \ell_i$.

Inoltre sempre a lezione è stato definito il lemma di inversione della relazione di subtyping: (Lemma di inversione della relazione di subtyping)

1. Se $S <: T_1 \rightarrow T_2$ allora S è della forma $S_1 \rightarrow S_2$ con $T_1 <: S_1$ e $S_2 <: T_2$.
2. Se $S <: \{\ell_i : T_i \ ^{i \in 1 \ldots n}\}$ allora S è della forma $\{k_j : S_j \ ^{j \in 1 \ldots m}\}$ tale che $\{\ell_i \ ^{i \in 1 \ldots n}\} \subseteq \{k_j \ ^{j \in 1 \ldots m}\}$ e $S_j <: T_i$ per ogni etichetta comune $\ell_i = k_j$

(Lemma delle Forme Canoniche)

- Se v è un valore di tipo $T_1 \rightarrow T_2$ allora v è della forma $fnx : S1M$.
- Se v è un valore di tipo $\{\ell_i : T_i \ ^{i \in 1 \ldots n}\}$, allora v è della forma $\{k_j = v_j \ ^{j \in 1 \ldots m}\}$ tale che $\{\ell_i \ ^{i \in 1 \ldots n}\} \subseteq \{k_j \ ^{j \in 1 \ldots m}\}$

Dimostrazione del substitution Lemma Per provare il substitution Lemma devo estendere la definizione induttiva di $fv(M)$ e la definizione di sostituzione per i tipi record e i tipi select. (Variabili libere) Vengono riportati solo i casi che estendono la definizione fornita in Note 2:

- $fv(\{\ell_i = M_i \ ^{i \in 1 \ldots n}\}) = \bigcup_{i=1 \ldots n} fv(M_i)$
- $fv(M.l) = fv(M)$

(Sostituzione) Vengono riportati solo i casi che estendono la definizione fornita in Note 2:

- $\{\ell_i = M_i \ ^{i \in 1 \ldots n}\}\{x := N\} = \{\ell_i = M_i\{x := N\} \ ^{i \in 1 \ldots n}\}$
- $M.l\{x := N\} = M\{x := N\}.l$

(Substitution Lemma) Se $\Gamma, x : S \vdash M : T$ e $\Gamma \vdash N : S$, allora $\Gamma \vdash M\{x := N\} : T$.

Proof. Procedo per induzione sulla lunghezza della derivazione del giudizio $\Gamma, x : S \vdash M : T$. La dimostrazione procede distinguendo l'ultima regola di tipo che è stata applicata per derivare tale giudizio:

- **Casi Base** (h = 1): Quelli in cui è stato applicato un assioma per le regole dei tipi.
  - $\boxed{\text{(TRUE)}}$. $\Gamma, x : S \vdash \mathsf{true} : T \equiv \mathsf{Bool}$. La tesi $\Gamma \vdash \mathsf{true}\{x := N\} : \mathsf{Bool}$, visto che $\mathsf{true}\{x := N\} \equiv \mathsf{true}$, si ottiene che la tesi è: $\Gamma \vdash \mathsf{true} : \mathsf{Bool}$ che è vera per l'assioma (TRUE).
  - $\boxed{\text{(FALSE)}}$: analogo al caso (TRUE).
  - $\boxed{\text{(TRUE)}}$: analogo al caso (TRUE).
  - $\boxed{\text{(VAR)}}$. Questo caso differisce leggermente da quello visto a lezione, in quanto il lemma di inversione per le variabili è stato modificato e asserisce che se $\Gamma \vdash x : T$, allora $x : S \in \Gamma$ con $S <: T$. Tuttavia, se il giudizio con il tipo preciso non è nel contesto (ovvero $S \neq T$), l'albero non può essere composto dal solo assioma (VAR) ma deve essere presente anche l'applicazione di (SUBSUMPTION). Quindi, per ricondurre questo caso a quello mostrato a lezione dimostro un lemma:

    Se il giudizio $\Gamma \vdash z : T$ è stato ottenuto dall'assioma (VAR), allora $z : T \in \Gamma$. (Vale il lemma di inversione del capitolo 45)

    Proof. In generale dall'assunzione $\Gamma \vdash z : T$ segue per il lemma di inversione (di note 9) che $\exists S$ tale che $z : S \in \Gamma$ e $S <: T$. Se $S <: T$ possono darsi solo due casi:

    * $\boxed{S = T}$ la regola (VAR) può essere applicata perchè il giudizio $z : T \in \Gamma$
    * $\boxed{S \neq T}$ la regola (VAR) non può essere applicata perchè il giudizio $z : T \notin \Gamma$.

    □

    Quindi per questo caso vale il lemma di inversione (più forte) del capitolo 45. E quindi la dimostrazione fornita a lezione risulta essere ancora valida in questa estensione del linguaggio.

    Ovvero se $\Gamma, x : S \vdash y : T$ viene derivato con la regola (VAR) possono verificarsi due casi:

    1. $y = x$: $\Gamma, x : S \vdash x : T$. Per il lemma di inversione (delle note 45), il giudizio deriva da un'asserzione presente nel contesto e quindi per forza $S = T$. Si ha quindi che considerando la sostituzione $\Gamma \vdash x\{x := N\} : T$. Ma N ha tipo S e $S = T$, quindi la tesi continua a valere.
    2. $y \neq x$: in questo caso la sostituzione $\Gamma \vdash y\{x := N\} : T$ non altera il termine, in più per l'assioma (VAR) $y : T \in \Gamma, x : S$ e quindi è derivabile il giudizio $\Gamma \vdash y : T$.

  Ipotesi induttiva: Da $\Gamma, x : S \vdash_k M : T$ e $\Gamma \vdash N : S$ implica che $\Gamma \vdash M\{x := N\} : T$.

  - **Passo Induttivo**: I casi (SUM), (MINUS), (FUN) sono analoghi a quelli visti a lezione. Visto che questo teorema vuole dimostrare giudizi del tipo $\Delta \vdash M : T$ le regole di sottotipo non potranno essere mai l'ultima regola applicata (poichè dimostrano giudizi solo del tipo $\Delta \vdash M <: T$) e quindi non devono essere considerate. Queste hanno infatti un ruolo indiretto nel caso (SUBSUMPTION).

* $\boxed{\text{(SUM)}}$: Invariato rispetto a quanto visto a lezione.

* $\boxed{\text{(MINUS)}}$: Invariato rispetto a quanto visto a lezione.

* $\boxed{\text{(FUN)}}$: Invariato rispetto a quanto visto a lezione.

* $\boxed{\text{(APP)}}$: Invariato rispetto a quanto visto a lezione

* $\boxed{\text{(TYPE-RECORD)}}$ In questo caso $M \equiv \{\ell_i = M_i \ ^{i \in 1 \ldots n}\}$. Assumo le due ipotesi per questo termine quindi

  1. $\Gamma, x : S \vdash_{k+1} \{\ell_i = M_i \ ^{i \in 1 \ldots n}\} : \{\ell_i : T_i\}$
  2. $\Gamma \vdash N : S$

  Siccome l'ultima regola applicata è (TYPE-RECORD) le sue premesse (al passo precedente) devono essere verificate quindi $\forall i \in 1 \ldots n \ \Gamma, x : S \vdash_{k_i} M_i : T_i$. Siccome le derivazioni dei giudizi per i sottotermini sono strettamente inferiori (di almeno un'unità) rispetto a quella di partenza e vale $\Gamma \vdash N : S$, ottengo per ipotesi induttiva (applicata agli n M_i) che $\forall i \in 1 \ldots n \ \Gamma \vdash M_i\{x := N\} : T_i$. Applicando la regola (TYPE-RECORD) ottengo $\Gamma \vdash \{\ell_i : M_i\{x := N\} \ ^{i \in 1 \ldots n}\} : \{\ell_i : T_i \ ^{i \in 1 \ldots n}\}$ che è proprio la tesi da dimostrare visto che per definizione:

$$\{\ell_i : M_i \ ^{i \in 1 \ldots n}\}\{x := N\} = \{\ell_i : M_i\{x := N\} \ ^{i \in 1 \ldots n}\}$$

* $\boxed{\text{(TYPE-SELECT)}}$: $M \equiv M.\ell_j$ Assumo che valgano le due ipotesi ovvero:

  1. $\Gamma, x : S \vdash_{k+1} M.\ell_j : T$, lo rinomino T_j per comodità.
  2. $\Gamma \vdash N : S$

  Ora siccome vale la prima ipotesi e l'ultima regola applicata è (TYPE-SELECT) le premesse devono essere soddisfatte quindi $\Gamma, x : S \vdash M : \{\ell_i : T_i \ ^{i \in 1 \ldots n}\}$ con $j \in 1 \ldots n$. Siccome questo giudizio ha una derivazione di lunghezza strettamente inferiore rispetto a quello di partenza e vale $\Gamma \vdash N : S$ posso applicare l'ipotesi induttiva e ottenere che $\Gamma \vdash M\{x := N\} : \{\ell_i : T_i \ ^{i \in 1 \ldots n}\}$ quindi riapplicando (TYPE-SELECT) ottengo:

$$\Gamma \vdash M\{x := N\}.\ell_j : T_j$$

  che è proprio la definizione di $M.\ell_j\{x := N\}$

* $\boxed{\text{(SUBSUMPTION)}}$: Assumo le due ipotesi per questa derivazione ovvero:

1. $\Gamma, x : S \vdash_{k+1} M : T$
2. $\Gamma \vdash N : S$

Visto che l'utima regola applicata è stata (Subsumption) posso dire che valgono le sue premesse:

$$(\textsc{Subsumption})\Gamma, x : S \vdash_k M : UU <: T\Gamma, x : S \vdash_{k+1} M : T$$

Ora siccome vale $\Gamma, x : S \vdash_k M : U$ e $\Gamma \vdash N : S$ posso applicare l'ipotesi induttiva e ottenere che $\Gamma \vdash M\{x \coloneqq N\} : U$. E visto che U è un sottotipo di T l'asserto è dimostrato per la regola (Subsumption).

$$(\textsc{Subsumption})\Gamma \vdash M\{x \coloneqq N\} : UU <: T\Gamma \vdash M\{x \coloneqq N\} : T$$

□

Dimostrazione del teorema di preservazione (per induzione su $M \to M'$) Se $\Gamma \vdash M : T$ e $M \longrightarrow M'$, allora $\Gamma \vdash M' : T$

Procedo per induzione sull'altezza della derivazione di $M \longrightarrow M'$. Inizio dai casi base, che sono quelli in cui $M \longrightarrow M'$ viene da un assioma.

- **Caso Base** I casi base per le regole di derivazione (Sum), (Minus), (If-True), (If-False) rimangono invariati rispetto a quelli mostrati a lezione. Il caso (Beta) è leggermente diverso rispetto a quello mostrato a lezione per via del lemma di inversione modificato, e infine va aggiunto il nuovo assioma (Select).
  - (Beta) $M \equiv (fnx : S1M_1\ v) : T \longrightarrow M_1\{x \coloneqq v\}$ Dall'ipotesi $\Gamma \vdash fnx : S1M_1\ v : T$ segue per il lemma di inversione che esiste T_A tale che $\Gamma \vdash v : T_A$ e $\Gamma \vdash fnx : S1M_1 : T_A \to T$.

    Da $\Gamma \vdash fnx : S1M_1 : T_A \to T$ segue per il lemma di inversione di note 9:

    1. $T_A <: S_1$
    2. $\Gamma, x : S_1 \vdash M_1 : T$

    Visto che

    $$(\textsc{Subsumption})\Gamma \vdash v : T_A T_A <: S_1 \Gamma \vdash v : S_1$$

    ho ricavato che $\Gamma \vdash v : S_1$. Considerato che vale 2 posso applicare il lemma di sostituzione e ottenere la tesi $\Gamma \vdash M_1\{x \coloneqq v\} : T$.
  - (Select) $\{\ell_i = v_i^{i\in 1\ldots n}\}.\ell_j \longrightarrow v_j$, grazie all'assioma (Select) (con $j \in 1 \ldots n$). Per ipotesi $\Gamma \vdash \{\ell_i = v_i^{i\in 1\ldots n}\}.\ell_j$ ha tipo T. Per comodità e in linea con la regola (Type-Select)

rinomino T in T_j. Quindi per il lemma di inversione della regola (Type-Select) ottengo che: $\Gamma \vdash M : \{\ell_i : T_i \ ^{i\in 1\ldots n}\}$ con $j \in 1 \ldots n$. Ora, siccome per la regola (Select) v_j è stato ottenuto dalla selezione del valore della label ℓ_j che ha tipo T_j, deduco che v_j ha il tipo T_j cercato. Visto che T_j non è altro che un alias per T la tesi è dimostrata.

- **Passo Induttivo** Procedo ora con i casi induttivi, quelli cioè che in cui $M \longrightarrow M'$ è stato derivato con una derivazione di altezza $k+1$. Distinguendo i vari casi a seconda dell'ultima regola usata: i casi (Sum-Left), (Sum-Right), (Minus-Left), (Minus-Right), (If) sono invariati visto che vale il lemma di inversione di note 2.
  - (Eval-Select) $M \equiv N.\ell_j \longrightarrow N'.\ell_j \equiv M'$. Assumo le due ipotesi per questo termine:
    1. $N.\ell_j \longrightarrow N'.\ell_j$: Visto che ho assunto che $N.\ell_j \longrightarrow N'.\ell_j$ è stato derivato con un albero di derivazione alto $k+1$ e l'ultima regola applicata è stata (Eval-Select) la premessa della regola $N \longrightarrow N'$ deve essere verificata. Graficamente:

    $$(\text{Eval-Select}) N \longrightarrow N' N.\ell_j \longrightarrow N'.\ell_j$$

    2. $\Gamma \vdash N.\ell_j : T$ rinomino T in T_j. Quindi per il lemma di inversione della regola (Type-Select) $\Gamma \vdash N : \{\ell_i : T_i \ ^{i\in 1\ldots n}\}$ con $j \in 1 \ldots n$.

    Quindi applicando l'ipotesi induttiva sulla derivazione $N \longrightarrow N'$ (che è alta k), si ottiene che:

    $$\Gamma \vdash N' : \{\ell_i : T_i \ ^{i\in 1\ldots n}\}$$

    Applicando la regola (Type-Select) ottengo $N'.\ell_j : T_j$ che è la tesi visto che T_j è un altro nome per T.
  - (Eval-Record) Assumo come di consueto le due ipotesi:
    1.

    $$(\text{Eval-Record}) M_x \longrightarrow M'_x \{\ell_i : v_i \ ^{i\in 1\ldots x-1}, \ell_x : M_x, \ell_j : M_j \ ^{j\in x+1\ldots m}\} \longrightarrow \{\ell_i : v_i \ ^{i\in 1\ldots x-1}, \ell_x :$$

    Con $M_x \rightarrow M'_x$ che viene derivato con un albero di derivazione di altezza inferiore rispetto a quello per $M \rightarrow M'$. Si può quindi applicare l'ipotesi induttiva ad M'_x, ottenendo che è derivabile il giudizio $\Gamma \vdash M'_x : T_x$.

    2. $\{\ell_i : v_i \ ^{i\in 1\ldots x-1}, \ell_x : M_x, \ell_j : M_j \ ^{j\in x+1\ldots m}\} : T$ è derivabile. Quindi per il lemma di inversione $\exists T_i$, $i = 1 \ldots n$ tale che $T = \{\ell_i : T_i \ ^{i\in 1\ldots n}\}$ con $\{\ell_i \ ^{i\in 1\ldots n}\} \subseteq \{k_r \ ^{r\in 1\ldots m}\}$ e per ogni etichetta in comune $k_r = \ell_i$. $\Gamma \vdash M_r : T_i$. Sostanzialmente stiamo dicendo che M può essere un'istanza di un sottotipo S di T e non necessariamente il tipo T.

    Dal momento che $M' = \{\ell_i : v_i \ ^{i\in 1\ldots x-1}, \ell_x : M'_x, \ell_j : M_j \ ^{j\in x+1\ldots m}\}$ posso applicare la regola di tipo (Type-Record) per derivare il giudizio $\Gamma \vdash M' : T$ con $T = \{\ell_i : T_i \ ^{i\in 1\ldots n}\}$.

Sono infatti soddisfatte le premesse della regola perchè i termini M_i associati alle ℓ_i con $i \in 1 \ldots n$, $i \neq x$ non sono cambiati e quindi anche il loro tipo non è cambiato, mentre per ipotesi induttiva $\Gamma \vdash M'_j : T_j$. La tesi è quindi provata.

Dimostrazione del teorema di preservazione (per induzione su $\Gamma \vdash M : T$) Se $\Gamma \vdash M : T$ e $M \longrightarrow M'$, allora $\Gamma \vdash M' : T$

Proof. Procedo in analogia all'esercizio 3.12, che dimostra $\Gamma \vdash M : T$ per induzione sull'altezza dell'albero della prova.

- $\boxed{\textsf{Casi Base}}$ $(h = 1)$: Casi in cui è stato applicato uno degli assiomi. Si veda l'esercizio 3.12.
- $\boxed{\textsf{Passo Induttivo}}$ Ora siccome l'ipotesi è verificata per i casi base dimostro che vale anche per i casi induttivi a seconda dell'ultima regola usata. Rispetto all'esercizio 3.12 sono state aggiunte solo 3 regole di tipo:
  - $\boxed{(\textsc{Type-Record})}$ assumendo le due ipotesi, ovvero:
    1. $\Gamma \vdash M : T$ visto che l'ultima regola applicata è $(\textsc{Type-Record})$, allora $M \equiv \{\ell_i = M_i \ ^{i \in 1 \ldots n}\}$ e $T = \{\ell_i : T_i \ ^{i \in 1 \ldots n}\}$. Dal momento che la sua premessa deve essere verificata si ha quindi $\forall i \in 1 \ldots n.\ \Gamma \vdash M_i : T_i$.
    2. $M \longrightarrow M'$

    Per il lemma delle forme canoniche dalla premessa 1 segue che M potrebbe essere un valore, ma in questo caso la seconda ipotesi non verrebbe verificata.

    Visto che M non è un valore, l'unica regola che fa ridurre un M di tipo $\{\ell_i : M_i \ ^{i \in 1 \ldots n}\}$ è:

    $(\textsc{Eval-Record}) M_j \longrightarrow M'_j \{\ell_i = v_i \ ^{i \in 1 \ldots j-1}, \ell_j = M_j, \ell_p = M_p \ ^{p \in j+1 \ldots n}\} \longrightarrow \{\ell_i = v_i \ ^{i \in 1 \ldots j-1}, \ell_j = M'_j, \ell_p$

    Siccome il giudizio $\Gamma \vdash M_j : T_j$ è più corto di un passo rispetto a $\Gamma \vdash M_j : T_j$ e $M_j \longrightarrow M'_j$, allora per ipotesi induttiva concludo che $\Gamma \vdash M'_j : T_j$. Ora devo dimostrare che $\{\ell_i = v_i \ ^{i \in 1 \ldots j-1}, \ell_j = M'_j, \ell_p = M_p \ ^{p \in j+1 \ldots n}\}$ ha tipo T. Dal momento che $M' = \{\ell_i : v_i \ ^{i \in 1 \ldots j-1}, \ell_j : M'_j, \ell_p : M_p \ ^{p \in j+1 \ldots m}\}$ posso applicare la regola di tipo $(\textsc{Type-Record})$ per derivare il giudizio $\Gamma \vdash M' : T$ con $T = \{\ell_i : T_i \ ^{i \in 1 \ldots n}\}$. Sono infatti soddisfatte le premesse della regola perchè i termini M_i associati alle ℓ_i con $i \in 1 \ldots n$, $i \neq j$ non sono cambiati e quindi anche il loro tipo non è cambiato, mentre per ipotesi induttiva $\Gamma \vdash M'_j : T_j$. La tesi è quindi provata.
  - $\boxed{(\textsc{Type-Select})}$ $M = N.\ell_j$
    1. $\Gamma \vdash N.\ell_j : T_j$. Visto che l'ultima regola applicata è stata $(\textsc{Type-Select})$ le sue premesse devono essere verificate quindi $\Gamma \vdash N : \{\ell_i : T_i \ ^{i \in 1 \ldots n}\}$ con $j \in 1 \ldots n$.

2. $M \longrightarrow M'$. Per questi tipi possono valere due opzioni:

   a) $N = \{\ell_i = v_i\ ^{i \in 1 \ldots n}\}$ è un valore. Ma questo caso è immediato (non ha bisogno dell'ipotesi induttiva) perchè per l'assioma (Select) si conclude che $N.\ell_j \longrightarrow v_j$, per quanto mostrato dall'ipotesi 1, ha tipo T_j.

   b) N non è un valore valutato e quindi $M' = N'.\ell_j$.

Considero dunque solo il caso in cui N non è un valore.

Visto che la lunghezza della derivazione del giudizio di tipo per N è strettamente inferiore di quella per $N.\ell_j$ e vale $N.\ell_j \longrightarrow N'.\ell_j$ (perchè N non è un valore) vale anche la premessa alla regola (Eval-Select) e quindi $N \longrightarrow N'$. Quindi posso applicare l'ipotesi induttiva e concludere che N' ha lo stesso tipo di N, ovvero $\{\ell_i : T_i\ ^{i \in 1 \ldots n}\}$. Applicando la regola (Type-Select) ed usando $N' : \{\ell_i : T_i\ ^{i \in 1 \ldots n}\}$ come premessa ottengo il risultato cercato, ovvero $N'.\ell_j : T_j$.

– $\boxed{\text{(Subsumption)}}$ $M = \{\ell_i = M_i\ ^{i \in 1 \ldots n}\}$ Assumo le due ipotesi:

1. $\Gamma \vdash M : T$
2. $M \longrightarrow M'$

Visto che l'ultima regola applicata è stata (Subsumption) posso assumere che le sue premesse siano verificate:

$$\text{(Subsumption)} \Gamma \vdash M : S S <: T \Gamma \vdash M : T$$

Visto che il giudizio $\Gamma \vdash M : S$ è più corto di almeno un'unità rispetto a $\Gamma \vdash M : T$ e $M \longrightarrow M'$ allora per ipotesi induttiva concludo che $\Gamma \vdash M' : S$. Quindi ho ottenuto le premesse per poter applicare la regola (Subsumption) a M':

$$\text{(Subsumption)} \Gamma \vdash M' : S S <: T \text{(vero per quanto mostrato prima)} \Gamma \vdash M' : T$$

$\Gamma \vdash M' : T$ era la tesi da dimostrare, perciò la dimostrazione del teorema di preservazione è conclusa.

□

Dimostrazione del teorema di progressione (Progressione). Sia M un termine chiuso e ben tipato, i.e. $\emptyset \vdash M : T$, allora o M è un valore, oppure esiste un termine M' tale che $M \longrightarrow M'$.

Proof. Procedo per induzione sull'altezza dell'albero di prova che dimostra $\varnothing \vdash M : T$.

- $\boxed{\textbf{Casi Base}(h = 1)}$: I casi base rimangono quelli mostrati a lezione, visto che gli assiomi per i giudizi di tipo non sono stati estesi.

Ipotesi induttiva (h = k): Se $\varnothing \vdash_k \bar{M} : T$ allora $\bar{M}$ è un valore oppure $\exists \bar{M}' : \bar{M} \longrightarrow \bar{M}'$

- $\boxed{\textsf{\textbf{Passo induttivo}}(h = k+1)}$ I casi che rimangono invariati rispetto a lezione vengono omessi. Per via del lemma di inversione delle forme canoniche per il subtyping, la dimostrazione per (Type-Select) e (Type-Record) è leggermente differente a quella mostrata nell'esercizio 6.3. Mentre la regola (Subsumption) è “inedita":
  - (Type-Record) Se $M = \{\ell_i = M_i\ ^{i=1\dots n}\}$ e $\emptyset \vdash M : T$, sappiamo che per la regola (Type-Record) valgono i giudizi di tipo $\emptyset \vdash M_i : T_i \ \ \forall\, i = 1\dots n$ e che gli alberi di derivazione relativi ai vari giudizi sono di altezza inferiore all'albero del giudizio principale. è quindi possibile applicare l'ipotesi induttiva sui sotto-termini M_i, i quali o sono un valore v_i di tipo T_i oppure possono avanzare in un termine M_i':
    * Se sono tutti dei valori si ha $M = \{\ell_i = v_i\ ^{i=1\dots n}\}$, ovvero M è un valore record di tipo $\{\ell_i : T_i\ ^{i=1\dots n}\}$ e quindi il teorema di Progressione continua a valere banalmente.
    * Se c'è almeno un M_i che non è un valore, è possibile identificare il termine M_j di indice minimo che non è un valore e per il quale continua a valere il giudizio di tipo $\emptyset \vdash M_j : T_j$. Vale quindi l'ipotesi induttiva, ovvero $\exists M_j'$ tale che $M_j \to M_j'$. Posso quindi applicare la regola (Eval-Record) per far avanzare il termine M al termine M', dove al posto di M_j compare M_j'. Anche in questo caso il teorema di Progressione continua a valere.
  - (Type-Select) Se $M = N.\ell_j$ e $\emptyset \vdash M : T$, sappiamo che per la regola (Type-Select) vale il giudizio di tipo $\emptyset \vdash N : \{\ell_i : T_i\ ^{i=1\dots n}\}$ (con un albero di derivazione più piccolo) e che $j \in \{1\dots n\}$. Si ha quindi che N o è un buon valore finale oppure $\exists N' : N \longrightarrow N'$.
    * Se N è un valore v per il lemma delle forme canoniche di note 9 ottengo che v è della forma $\{k_q = v_q\ ^{q\in 1\dots m}\}$ tale che $\{\ell_i\ ^{i\in 1\dots n}\} \subseteq \{k_q\ ^{q\in 1\dots m}\}$. Siccome l'etichetta $\ell_j \in \{k_q\ ^{q\in 1\dots m}\}$ allora $\exists$ un indice $p \in 1\dots m : \ell_j = k_p$ e quindi $M = N.\ell_j$ si riscrive grazie alla regola (Select) nel valore v_p.
    * Un record non ancora completamente valutato, ovvero $N = \{\ell_i = v_i\ ^{i=1\dots x-1}, l_x = M_x, l_k = M_k\ ^{k=x+1\dots n}\}, \exists M_x' | M_x \to M_x'$ e quindi per (Eval-Record) $N \to N'$ e pertanto anche $M \to M' = N'.\ell_j$.
    * Un termine generico di tipo record, ovvero $N = \textsf{if } M_1 \textsf{ then } M_2 \textsf{ else } M_3$ oppure $N = A\ B$. In entrambi i casi vale il giudizio $\emptyset \vdash N : \{\ell_i : T_i\ ^{i=1\dots n}\}$, cioè N è un termine chiuso, ben tipato e non è un valore. Quindi per ipotesi induttiva ho che $\exists N'$ tale che $N \to N'$ e quindi per (Eval-Select) $\exists M' = N'.\ell_j$ tale che $M \to M'$. Pertanto il teorema di Progressione continua a valere.
  - (Subsumption)

$$(\textsc{Subsumption})\vdots \varnothing \vdash_k M : S \vdots S <: T \varnothing \vdash_{k+1} M : T$$

Ora, siccome la derivazione del giudizio $\varnothing \vdash M : S$ è più corta di (almeno) un passo di derivazione rispetto a quella di partenza, posso concludere per ipotesi induttiva che M o è un buon valore finale o che $\exists M' : M \longrightarrow M'$. Siccome questo M corrisponde a quello di partenza, la tesi è dimostrata.

$\square$

Teorema di Safety Se $\emptyset \vdash M : T$ e $M \rightarrow^* M'$ con M' tale che $M' \not\rightarrow$, allora M' è un valore.

La dimostrazione è una diretta conseguenza dei due teoremi precedenti: per il teorema di Progressione anche $\emptyset \vdash M' : T$ è derivabile e, per il teorema di Subject-Reduction M' deve essere un valore, perchè altrimenti esisterebbe un M'' al quale può ridursi.

Esercizio 6.16

Trovare due termini M e N tali che M → N, Γ ⊢ M : T, Γ ⊢ N : S con S<::T e T$\not<$::S, cioè esibire un caso in cui il tipo di un termine decresce durante la computazione.

Svolgimento

Una possibile soluione potrebbe essere la seguente:

$(fn\ x{:}\{l{:}Nat\}.\{l_i = x.l_i^{i\in 1,\ldots,x.length}, l_j^{j=x.lenght+1} = 0\})\ (l = 4)$

Questa funzione, in sostanza non fa altro che ritornare un record con gli stessi campi del primo più un ulteriore campo. Così facendo il tipo del paramentro della funzione è T ma accetta qualsiasi tipo S tale che $S <: T$, tuttavia il tipo di ritorno è sempre più specifico (a causa della nuova etichetta aggiunta) e di conseguenza, per come abbiamo definito la regola di subtyping, il tipo di ritorno U non sarà mai sopra-tipo di T.

Quindi abbiamo trovato due termini, $M{:}T$ e $N{:}S$ tali che $M \rightarrow N$ con le caratteristiche richieste.

Inoltre, per come definita, questa funzione rispetta quanto richiesto anche nel subtyping non algoritmico.

7 Featherweight Java (note 11)

Esercizio 7.1

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

Svolgimento:
Definiamo le due seguenti Classi

```
Class A extends B {A(){super();} }

Class B extends A {B(){super();} }
```

A e B sono muatamente ricorsive. La relativa Class Table non è però ben fatta, in quanto contiene una relazione di sottotipo indotta con cicli.

Esercizio 7.2

Dato il seguente codice:

```
class A extends Object { A(){ super(); } }

class B extends Object { B(){ super(); } }

class Pair extends Object {
   Object fst; Object snd;

   Pair(Object fst, Object snd){
      super(); this.fst=fst; this.snd=snd;
   }

   Pair setfst(Object newfst){ return new Pair(newfst, this.snd); }
}
```

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())

Svolgimento:

Per motivi di spazio i seguenti termini saranno abbreviati con delle lettere

* K ≡ new Pair(new A(),new B())
* H ≡ Pair setfst(Object newfst) { return new Pair(newfst, this.snd)
* J ≡ CT(Pair)= class Pair extends Object Object fst, Object snd; Pair(Object fst, Object snd),Pair setfst(Object newfst)

- $\boxed{\text{new Pair(new A(), new B()).snd}}$

$$(\text{PROJNEW})\ \dfrac{\dfrac{J \qquad \text{fields(Object)} = \emptyset}{\text{fields(Pair)} = (\text{fst} : \text{Object}, \text{snd} : \text{Object})} \qquad \text{snd} \in \{\text{fst} , \text{snd}\}}{\text{new Pair(new A(), new B()).snd} \rightarrow \text{new B()}}$$

- $\boxed{\text{(Pair) new Pair(new A(), new B())}}$

$$(\text{CASTNEW})\ \dfrac{\dfrac{}{\text{Pair} <: \text{Pair}}\,(\text{REFLEX})}{\text{(Pair) new Pair(new A(), new B())} \rightarrow \text{new Pair(new A(), new B())}}$$

- $\boxed{\text{new Pair(new A(),new B()).setfst(new B())}}$

$$(\text{INVKNEW})\ \dfrac{\dfrac{J \quad H}{\text{mbody(setfst, Pair)} = (\text{newfst, new Pair(newfst, this.snd)})} \qquad |\text{fst}| = |\text{newB()}|}{\text{K.setfst(new B())} \rightarrow \text{new Pair(newfst,this.snd)}\{\text{fst=new B(),this=K}\}}$$

- $\boxed{\text{((Pair) (new Pair(new Pair(new A(), new B()), new A()).fst)).snd}}$

  Passo 1

$$\text{(FIELD)}\ \dfrac{\text{(CAST)}\ \dfrac{\text{(PROJNEW)}\ \dfrac{\dfrac{\text{J} \qquad \text{field(Object)} = \emptyset}{\text{fields (Pair)} = \{\text{Object fst,Object snd}\}} \qquad fst \in \tilde{f}}{\text{(new Pair(K, new A()).fst)} \to \text{K}}}{\text{((Pair) (new Pair(K, new A()).fst))} \to \text{(Pair) K}}}{\text{((Pair) (new Pair(K, new A()).fst)).snd} \to \text{((Pair) K).snd}}$$

  Passo 2

$$\text{(FIELD)}\ \dfrac{\dfrac{\text{(REFLEX)}\ \dfrac{}{\text{Pair <: Pair}}}{\text{((Pair) K} \to \text{K}}\ \text{(CASTNEW)}}{\text{((Pair) K).snd} \to \text{K.snd}}$$

  Passo 3

$$\text{((PROJNEW))}\ \dfrac{\dfrac{\text{J} \qquad \text{field(Object)} = \emptyset}{\text{fields (Pair)} = \{\text{Object fst,Object snd}\}} \qquad \text{snd} \in \widetilde{f}}{\text{(new Pair(new A(), new B()).snd} \to \text{new B()}}$$

- $\boxed{\text{(B) ((A)new C())}}$

  Possiamo fare due ipotesi di subtyping:

  - C <: A <: B

    Passo 1

$$\text{(CAST)}\ \dfrac{\dfrac{\dfrac{\text{CT(C)=class C extends A\{...\}}}{\text{C <: A}}}{\text{(A)new C()} \to \text{new C()}}\ \text{(CASTNEW)}}{\text{(B) ((A)new C())} \to \text{(B) new C()}}$$

    Passo 2

$$\text{(CASTNEW)}\ \cfrac{\cfrac{\cfrac{CT(A)=\text{class A extends B}\{...\}}{A <: B} \qquad \cfrac{CT(C)=\text{class C extends A}\{...\}}{C <: A}}{C <: B}}{\text{(B) new C()} \rightarrow \text{new C()}}$$

- C <: B <: A

  Passo 1

$$\text{(CAST)}\ \cfrac{\cfrac{\cfrac{\cfrac{CT(B)=\text{class B extends A}\{...\}}{B <: A} \qquad \cfrac{CT(C)=\text{class C extends B}\{...\}}{C <: B}}{C <: A}}{\text{(A)new C()} \rightarrow \text{new C()}}\ \text{(CASTNEW)}}{\text{(B) ((A)new C())} \rightarrow \text{(B) new C()}}$$

  Passo 2

$$\text{(CASTNEW)}\ \cfrac{\cfrac{CT(C)=\text{class C extends B}\{...\}}{C <: B}}{\text{(B) new C()} \rightarrow \text{new C()}}$$

Esercizio 7.3

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

Svolgimento:

```
class A extends Object{

   Nat fst; Nat snd;
```

```
   A (Nat fst, Nat snd){
      super(); this.fst=fst; this.snd=snd;
   }

   Nat print() { return fst; }
}

class B extends A{
   B (Nat fst, Nat snd){

   super(); this.fst=fst; this.snd=snd;
   }
}

Nat print() { return snd; }
```

consideriamo ora il seguente termine:

((A) new B(10,4)).print()

Per motivi di spazio il seguente termine sarà abbreviato con una lettera

- J ≡ CT(B)= class B extends A Object fst, Object snd; B(Nat fst, Nat snd),Nat print()

Passo 1

$$\text{(INVKRECV)}\ \dfrac{\dfrac{\dfrac{\text{J}}{\text{B} <: \text{A}}}{\text{(A) new B(10,4)} \rightarrow \text{new B(10,4)}}\ \text{(CASTNEW)}}{\text{((A) new B(10,4)).print()} \rightarrow \text{new B(10,4).print()}}$$

Passo 2

$$\text{(INVKNEW)}\ \dfrac{\dfrac{\text{J} \qquad \text{Nat print() \{ return snd \}} \in \widetilde{M}}{\text{mbody}(\emptyset,\text{B})=\text{(snd, return snd)}} \qquad \emptyset = \emptyset}{\text{new B(10,4).print()} \rightarrow 4}$$

Esercizio 7.4

Perchè c'è una regola di tipo sia per l'Up-cast che per il Downcast, mentre c'è la sola regola di valutazione per Up-cast, nella semantica operazionale?

Svolgimento:

L'operazione di Up-cast, applicata su termini non stuck, non produrrà mai termini stuck in quanto effettua una conversione da un tipo più informativo(sottoclasse) ad un tipo meno informativo(superclasse). Ciò non è sempre valido con l'operazione di Down-cast, la quale effettua invece una conversione da un tipo meno informativo(superclasse) ad uno che richiede più informazione (sottoclasse), con conseguente mancanza di valori da assegnare ai campi dati della sottoclasse stessa.

La regola di typing di Down-cast, è inserita per permettere la valutazioni di termini, che il compilatore non considerebbe ben tipati anche se a run-time non evolvono in un termine stuck.

Consideriamo ad esempio il seguente termine:

- (B)((C) new A())

con la seguente relazione di sottotipo (A<:B<:C)

Valutazione di tipo:

$$\dfrac{\text{(UP-CAST)}\ \dfrac{\text{(NEW)}\ \dfrac{|\emptyset| = |\emptyset| \quad fields(A) = \emptyset}{\emptyset \vdash \text{new A(): A}} \qquad \text{A <: C}}{\emptyset \vdash \text{(C)new A(): C}} \qquad \text{B <: C} \qquad \text{C} \neq \text{B}}{\emptyset \vdash \text{(B) ((C)new A()): B}}\ \text{(DOWN-CAST)}$$

Applicazione regole di semantica operazionale:

Passo 1

$$\text{(CAST)}\ \dfrac{\dfrac{\dfrac{\dfrac{\text{CT(A)=class A extends B\{...\}}}{\text{A <: B}} \qquad \dfrac{\text{CT(B)=class B extends C\{...\}}}{\text{B <: C}}}{\text{A <: C}}}{\text{(C)new A()} \rightarrow \text{new A()}}\ \text{(CASTNEW)}}{\text{(B) ((C)new A())} \rightarrow \text{(B) (new A())}}$$

Passo 2

$$\text{(CASTNEW)}\ \frac{\dfrac{\text{CT(A)=class A extends B\{...\}}}{\text{A <: B}}}{\text{(B) new A()} \rightarrow \text{new A()}}$$

Risulta essere ben tipato grazie anche all'utilizzo della regola di tipo DOWN-CAST, ed a Run-time effettivamente non evolve in un termine stuck.

Esercizio 7.5

Ha senso aggiungere a FJ la regola di sub-typing per i tipi freccia A→B?

Svolgimento:

In FJ le funzioni non appartengono all'insieme Termini t della sintassi del linguaggio.

I tipi freccia li troviamo solamente all'interno di metodi, i quali hanno già una regola di typing (INVK) che, essendo il sistema di regole sintax-direct (o algoritmico), integra direttamente la regola di sub-typing.

Esercizio 7.12

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 possibilità di gestire le eccezioni.

Svolgimento:

8 Imperative Featherweight Java (note 12)

Esercizio 8.1

Scrivere una regola per eliminare i riferimenti non più utilizzati.

Svolgimento:

I riferimenti agli oggetti che non vengono più utilizzati all'interno di un termine e, possono essere eliminati.

consideriamo la seguente regola:

$$\frac{\text{o} \notin \text{fref}(e) \qquad \text{o} \notin \text{Codom}(\sigma)}{\langle \sigma\, [o \mapsto (C, \tilde{f:v})]\, ,\ e\rangle \to \langle \sigma\ ,\ e\rangle}$$

- $\text{fref}(e)$ = Insieme dei riferimenti a tutti gli oggetti utilizzati nel termine e
- $\text{Codom}(\sigma)$ = Insieme degli oggetti che sono riferiti da altri oggetti presenti nella memoria.

Esercizio 8.2

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();}

   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();
```

Risoluzione:

Definiamo la configurazione iniziale sostituendo allo store σ l'insieme vuoto($\emptyset$) ed all'espressione e l'insieme delle istruzioni del programma. Otteniamo dunque la seguente configurazione iniziale:

$\langle \emptyset \, , \, Object\ z = new\ Object(); \ C\ x = new\ C(z); \ .....\rangle$

- Creazione oggetto o_1 da memorizzare nella variabile z

  $\rightarrow \langle \emptyset \, , \, Object\ z = o_1; \ C\ x = new\ C(z); \ .....\rangle$

$$\text{(CONG)}\ \dfrac{\dfrac{o_1 \notin Dom(\emptyset) \qquad \text{field(Object)} = \emptyset \qquad |\emptyset| = |\emptyset|}{\langle \emptyset,\ newObject()\rangle \rightarrow \langle \emptyset\,[o_1 \mapsto (Object)]\,,\ o_1\rangle}\ \text{(NEW)}}{\langle \emptyset\,,E[t]\rangle \rightarrow \langle \emptyset\,[o_1 \mapsto (Object)]\,,\ E[o_1]\rangle}$$

  - E[] $\equiv$ Object z = []
  - $t \equiv$ new Object()

- Associazione dell'oggetto o_1 a z

  $\rightarrow \langle \emptyset \ [o_1 \ \mapsto \ (Object)] \ , \, C\ x = new\ C(z); \ .....\rangle$

$$(\text{CONG})\ \dfrac{\dfrac{z \notin Dom(\sigma)}{\langle \sigma\ ,\ Object\ z =\ o_1\ ;\ e\rangle \rightarrow \langle \sigma\ [z \mapsto o_1]\ ,\ e\rangle}\ (\text{NEW})}{\langle \sigma\ , E[t]\rangle \rightarrow \langle \sigma\ [z\ \mapsto\ o_1]\ ,\ E[o_1]\rangle}$$

- E[] $\equiv$ []
- t $\equiv$ Object z=o_1
- $\sigma\ \equiv\ \emptyset\ [o_1\ \mapsto\ (Object)]$

- Dereferenzazione della varibile z per creare l'oggetto o_2

  $\rightarrow \langle \emptyset\ [o_1\ \mapsto\ Object]\ [z\ \mapsto: o_1]\ ,\ C\ x = new\ C(o_1);\ .....\rangle$

- Creazione dell'oggetto o_2 da associare nella variabile x

  $\rightarrow \langle \emptyset\ [o_1\ \mapsto\ Object]\ [z\ \mapsto\ o_1]\ [o_2\ \mapsto\ (C, f : o_1)]\ ,\ C\ x = new\ C(z);\ .....\rangle$

- Associazione dell'oggetto o_2 ad x

  $\rightarrow \langle \emptyset\ [o_1\ \mapsto\ Object]\ [z\ \mapsto\ o_1]\ [o_2\ \mapsto\ (C, f : o_1)]\ [x\ \mapsto\ o_2]\ ,\ C\ y = new\ D(x);\ .....\rangle$

- Dereferenzazione della variabile x per creare l'oggetto o_2

  $\rightarrow \langle \emptyset\ [o_1\ \mapsto\ Object]\ [z\ \mapsto\ o_1]\ [o_2\ \mapsto\ (C, f : o_1)]\ [x\ \mapsto\ o_2]\ ,\ C\ y = new\ D(o_2);\ .....\rangle$

- Creazione dell'oggetto o_3 di tipo D da associare nella variabile y di tipo C. La classe C è un sottotipo della classe D, però non aggiunge nessun nuovo campo dati e nessun nuovo metodo, effettua solamente l'override del metodo m().

  $\rightarrow \langle \emptyset\ [o_1\ \mapsto\ Object]\ [z\ \mapsto\ o_1]\ [o_2\ \mapsto\ (C, f : o_1)]\ [x\ \mapsto\ o_2]\ [o_3\ \mapsto\ (D, f : o_2)]\ ,\ C\ y = o_3;\ .....\rangle$

- Associazione dell'oggetto o_3 ad y

  $\rightarrow \langle \emptyset\ [o_1 \mapsto Object]\ [z \mapsto o_1]\ [o_2 \mapsto (C, f : o_1)]\ [x \mapsto o_2]\ [o_3 \mapsto (D, f : o_2)]\ [y \mapsto o_3]\ , x.m(); .....\rangle$

- Dereferenzazione di x per effettuare l'invocazione del metodo m().

  $\rightarrow \langle \emptyset\ [o_1 \mapsto Object]\ [z \mapsto o_1]\ [o_2 \mapsto (C, f : o_1)]\ [x \mapsto o_2]\ [o_3 \mapsto (D, f : o_2)]\ [y \mapsto o_3]\ , o_2.m(); .....\rangle$

- Invocazione del metodo m() della classe C perchè la funzione mbody effettua la scelta di tale metodo in base alla class-table dell'oggetto di invocazione, il metodo restituisce l'oggetto stesso quindi o_2.

  Lo store rimane inalterato e si ha solo il passaggio all'istruzione successiva.

  $\rightarrow \langle \emptyset\, [o_1 \mapsto Object]\, [z \mapsto o_1]\, [o_2 \mapsto (C, f : o_1)]\, [x \mapsto o_2]\, [o_3 \mapsto (D, f : o_2)]\, [y \mapsto o_3]\, , x = y; .....\rangle$

- Dereferenzazione della variabile y.

  $\rightarrow \langle \emptyset\, [o_1 \mapsto Object]\, [z \mapsto o_1]\, [o_2 \mapsto (C, f : o_1)]\, [x \mapsto o_2]\, [o_3 \mapsto (D, f : o_2)]\, [y \mapsto o_3]\, , x = o_3; .....\rangle$

- Associazione dell'oggetto o_3 di tipo D alla variabile x. Si effetua una modifica allo store in quanto la variabile x già esisteva ed era associata all'oggetto o_2.

  $\rightarrow \langle \emptyset\, [o_1 \mapsto Object]\, [z \mapsto o_1]\, [o_2 \mapsto (C, f : o_1)]\, \underline{[x \mapsto o_3]}\, [o_3 \mapsto (D, f : o_2)]\, [y \mapsto o_3]\, , x.m(); .....\rangle$

- Dereferenzazione di x per effettuare l'invocazione del metodo m().

  $\rightarrow \langle \emptyset\, [o_1 \mapsto Object]\, [z \mapsto o_1]\, [o_2 \mapsto (C, f : o_1)]\, [x \mapsto o_3]\, [o_3 \mapsto (D, f : o_2)]\, [y \mapsto o_3]\, , o_3.m(); .....\rangle$

- Invocazione del metodo m() della classe D perchè la funzione mbody effettua la scelta di tale metodo in base alla class-table dell'oggetto di invocazione, il metodo restituisce l'oggetto stesso quindi o_3.

  Lo store rimane inalterato e si ha solo il passaggio all'istruzione successiva.

  $\rightarrow \langle \emptyset\, [o_1 \mapsto Object]\, [z \mapsto o_1]\, [o_2 \mapsto (C, f : o_1)]\, [x \mapsto o_3]\, [o_3 \mapsto (D, f : o_2)]\, [y \mapsto o_3]\, , y.f = new\, Object(); .....\rangle$

- Dereferenzazione di y

  $\rightarrow \langle \emptyset\, [o_1 \mapsto Object]\, [z \mapsto o_1]\, [o_2 \mapsto (C, f : o_1)]\, [x \mapsto o_3]\, [o_3 \mapsto (D, f : o_2)]\, [y \mapsto o_3]\, , o_3.f = new\, Object(); .....\rangle$

- Creazione dell'oggetto o_4 di tipo Object da associare alla struttura dati f dell'oggetto o_3, associato alla variabile y, che referenziava l'oggetto o_2.

  $\rightarrow \langle \emptyset\, [o_1 \mapsto Object]\, [z \mapsto o_1]\, [o_2 \mapsto (C, f : o_1)]\, [x \mapsto o_3]\, \underline{[o_3 \mapsto (D, f : o_4)]}\, [y \mapsto o_3]\, [o_4 \mapsto Object]\, , o_3.f = o_4; .....\rangle$

- Associazione alla struttura dati dell'oggetto o_3 associato alla variabile y.

  Lo store rimane inalterato e si ha solo il passaggio all'istruzione successiva.

  $\rightarrow \langle\emptyset\ [o_1 \mapsto Object]\,[z \mapsto o_1]\,[o_2 \mapsto (C, f : o_1)]\,[x \mapsto o_3]\,[o_3 \mapsto (D, f : o_4)]\,[y \mapsto o_3]\,[o_4 \mapsto Object]\ ,\ z = null; .....\rangle$

- Associazione di **null** alla variabile z.

  $\rightarrow \langle\emptyset\ [o_1 \mapsto Object]\,\underline{[z \mapsto null]}\,[o_2 \mapsto (C, f : o_1)]\,[x \mapsto o_3]\,[o_3 \mapsto (D, f : o_4)]\,[y \mapsto o_3]\,[o_4 \mapsto Object]\ ,\ x.f = z; .....\rangle$

- Dereferenzazione di z.

  $\rightarrow \langle\emptyset\ [o_1 \mapsto Object]\,\underline{[z \mapsto null]}\,[o_2 \mapsto (C, f : o_1)]\,[x \mapsto o_3]\,[o_3 \mapsto (D, f : o_4)]\,[y \mapsto o_3]\,[o_4 \mapsto Object]\ ,\ x.f = null; .....\rangle$

- Dereferenzazione di x.

  $\rightarrow \langle\emptyset\ [o_1 \mapsto Object]\,\underline{[z \mapsto null]}\,[o_2 \mapsto (C, f : o_1)]\,[x \mapsto o_3]\,[o_3 \mapsto (D, f : o_4)]\,[y \mapsto o_3]\,[o_4 \mapsto Object]\ ,\ o_3.f = null; .....\rangle$

- Assegnazione di *null* ad $o_3.f$. Considerando che la variabile y si riferisce all'oggetto o_3, abbiamo che anche y.f = *null*

  $\rightarrow \langle\emptyset\ [o_1 \mapsto Object]\,[z \mapsto null]\,[o_2 \mapsto (C, f : o_1)]\,[x \mapsto o_3]\,[o_3 \mapsto (D, f : null)]\,[y \mapsto o_3]\,[o_4 \mapsto Object]\ ,\ y.f.m(); \rangle$

- Dereferenzazione di y.

  $\rightarrow \langle\emptyset\ [o_1 \mapsto Object]\,[z \mapsto null]\,[o_2 \mapsto (C, f : o_1)]\,[x \mapsto o_3]\,[o_3 \mapsto (D, f : null)]\,[y \mapsto o_3]\,[o_4 \mapsto Object]\ ,\ o_3.f.m(); \rangle$

- Il programma termina restituendo NPE, in quanto si tenta di invocare il metodo m() su *null*.

Esercizio 8.3

Dare una derivazione di tipo per il termine C x=o_1; o_1.f=y; x=o_2.

$$\text{(DICHIAR)}\ \dfrac{\dfrac{o_1\ :\ F\ \in\ \Gamma}{\Gamma\ \vdash\ o_1 : F}\text{(OID)}\qquad \dfrac{\text{(A)}}{F\ <:\ C}\text{(CLASS)}\qquad \dfrac{\text{(B)}\qquad \dfrac{\text{(C)}}{\Gamma\ ,\ x = o_2\ : T}}{\Gamma\ ,\ x : C\ \vdash\ o_1.f = y,\ x = o_2\ :\ T}\text{(SEQ)}}{C\ x = o_1;\ o_1.f = y;\ x = o_2\ :\ T}$$

(A) $\quad (CT(F)\ =\ class\, F\, extends\, C\,\{..\})$

(B)

$$\text{(Fld Ass)}\ \dfrac{\dfrac{\dfrac{o_1\ :\ F\ \in\ \Gamma}{\Gamma\ ,\ x : C\ \vdash\ o_1\ :\ F}\text{(Oid)}\qquad f\ \in\ field(N)}{\Gamma\ ,\ x : C\ \vdash\ o_1.f\ :\ N}\text{(Fld)}\qquad \dfrac{CT(N)..}{N\ <:\ M}\text{(CL)}\qquad \dfrac{y\ :\ M\ \in\ \Gamma}{\Gamma\ ,\ x : C\ \vdash\ y\ :\ M}\text{(Var)}}{\Gamma\ ,\ x : C\ \vdash\ o_1.f = y\ :\ M}$$

(C)

$$\text{(Ass)}\ \dfrac{\dfrac{o_2\ :\ T\ \in\ \Gamma}{\Gamma\ ,\ x : C\ \vdash\ o_2\ :\ T}\text{(OID)}\qquad \dfrac{CT(T) = classT\ extends\ C\{..\}}{T\ <:\ C}\text{(CL)}\qquad \dfrac{x\ :\ C\ \in\ \Gamma}{\Gamma\ \vdash\ x\ :\ C}\text{(Var)}}{\Gamma\ ,\ x : C\ \vdash\ x = o_2\ :\ T}$$