

# “Fondamenti logici dei linguaggi funzionali”

## Esercizi

Andrea Burattin

22 aprile 2008

### Sommario

Elenco di esercizi preparati per l'esame del corso di “Fondamenti logici dei linguaggi funzionali”. Anno accademico 2007/2008, docente Prof. Silvio Valentini.

## Indice

|              |    |
|--------------|----|
| Esercizio 2  | 1  |
| Esercizio 5  | 2  |
| Esercizio 8  | 3  |
| Esercizio 10 | 4  |
| Esercizio 12 | 4  |
| Esercizio 15 | 6  |
| Esercizio 21 | 6  |
| Esercizio 23 | 9  |
| Esercizio 24 | 10 |
| Esercizio 25 | 10 |
| Esercizio 27 | 11 |
| Esercizio 28 | 12 |
| Esercizio 31 | 13 |
| Esercizio 35 | 14 |
| Esercizio 42 | 18 |

## Esercizio 2

Scrivere il superesponenziale (*super-exp*).

Definizione della funzione **somma**:

$$\begin{cases} \text{Sum}(0, y) = y \\ \text{Sum}(x + 1, y) = 1 + \text{Sum}(x, y) \end{cases} \rightsquigarrow \begin{cases} \text{SUM}(0, \bar{y}) = P_1^1 \\ \text{SUM}(x + 1, \bar{y}) = C^3[S^1, P_3^3] \end{cases}$$

$$\text{SUM}^2 \equiv R^2[P_1^1, C^3[S^1, P_3^3]]$$

Definizione della funzione **prodotto**:

$$\begin{cases} \text{Prod}(0, y) = 0 \\ \text{Prod}(x + 1, y) = y + \text{Prod}(x, y) \end{cases} \rightsquigarrow \begin{cases} \text{PROD}(0, \bar{y}) = Z^1 \\ \text{PROD}(x + 1, \bar{y}) = C^3[\text{SUM}^2, P_2^3, P_3^3] \end{cases}$$

$$\text{PROD}^2 \equiv R^2[Z^1, C^3[\text{SUM}^2, P_2^3, P_3^3]]$$

Definizione della funzione **esponenziale**:

$$\begin{cases} \text{Exp}(0, y) = 1 \\ \text{Exp}(x + 1, y) = y * \text{Exp}(x, y) \end{cases} \rightsquigarrow \begin{cases} \text{EXP}(0, \bar{y}) = C^1[S^1, Z^1] \\ \text{EXP}(x + 1, \bar{y}) = C^3[\text{PROD}^2, P_2^3, P_3^3] \end{cases}$$

$$\text{EXP}^2 \equiv R^2[C^1[S^1, Z^1], C^3[\text{PROD}^2, P_2^3, P_3^3]]$$

Definizione della funzione per il calcolo dell'“**esponente del superesponenziale**”:

$$\begin{cases} \text{GetExp}(0, y) = 1 \\ \text{GetExp}(x + 1, y) = y^{\text{GetExp}(x, y)} \end{cases} \rightsquigarrow$$

$$\begin{cases} \text{GETEXP}(0, \bar{y}) = C^1[S^1, Z^1] \\ \text{GETEXP}(x + 1, \bar{y}) = C^3[\text{EXP}^2, P_2^3, P_3^3] \end{cases}$$

$$\text{GETEXP}^2 \equiv R^2[C^1[S^1, Z^1], C^3[\text{EXP}^2, P_2^3, P_3^3]]$$

Definizione della funzione **superesponenziale**:

$$\text{SUPEREXP}^3 \equiv C^3[\text{EXP}^2, P_1^3, C^3[\text{GETEXP}^2, P_3^3, P_2^3]]$$

## Esercizio 5

Trovare la cardinalità di  $\text{Bool}^n \rightarrow \text{Bool}$  e dimostrare che ogni funzione  $\text{Bool}^n \rightarrow \text{Bool}$  è definibile con *if thenelse*.

Per il primo punto è sufficiente ricordare che, le funzioni che hanno come dominio  $A$  e come codominio  $B$  sono:

$$|A \rightarrow B| = |B|^{|A|}$$

Quindi, considerando il caso particolare dell'esercizio, sappiamo che:

$$|\text{Bool}^n| = 2^n \text{ e che } |\text{Bool}| = 2$$

consegue che

$$|\text{Bool}|^{|\text{Bool}^n|} = 2^{2^n}$$

La seconda parte dell'esercizio, può essere fatta sapendo che tutte le possibili operazioni sui  $\text{Bool}$  sono programmabili tramite composizioni di:

*And Or Not*

i primi due sono operatori che vogliono due argomenti, mentre il terzo è un operatore unario. Tale affermazione può essere dimostrata per induzione sulla dimensione di  $n$  (cardinalità dell'input). Nel caso base consideriamo  $n = 1$  (con  $n = 0$  abbiamo solo le "costanti" *true* e *false*). Le funzioni che possiamo programmare sono quella che inverte l'input (*Not*) e quella che identifica (può essere vista come *Not* ◦ *Not*). Supponiamo quindi, per ipotesi induttiva, che per  $n$  valga la prova, e dimostriamo che vale per  $n + 1$ : il valore restituito dall'ipotesi è a sua volta booleano, possiamo quindi fare una tabella in cui mostriamo tutte le possibili combinazioni ed indichiamo l'operazione fatta (possiamo omettere metà tabella, in quanto tali operazioni sono commutative):

| Valore calcolato da $n$ | Valore $n + 1$ | Risultati | Operazione                 |
|-------------------------|----------------|-----------|----------------------------|
| 1                       | 1              | 1         | <i>And/Or</i>              |
| 1                       | 0              | 1         | <i>Or</i>                  |
| 0                       | 1              | 1         | <i>Or</i>                  |
| 0                       | 0              | 1         | <i>Not</i> ◦ <i>And/Or</i> |
| 1                       | 1              | 0         | <i>Not</i> ◦ <i>And/Or</i> |
| 1                       | 0              | 0         | <i>And</i>                 |
| 0                       | 1              | 0         | <i>And</i>                 |
| 0                       | 0              | 0         | <i>And</i>                 |

Dunque, se mostriamo che le tre operazioni sono programmabili con *ifthenelse* possiamo concludere positivamente:

$$\begin{aligned}
 \textit{And} &\equiv \text{if } x \text{ then if } y \text{ then } \top \text{ else } \perp \text{ else } \perp \\
 &\equiv \lambda x \lambda y. \textit{ifthenelse}(x)(\textit{ifthenelse}(y)(\textit{true})(\textit{false}))(\textit{false}) \\
 \\
 \textit{Or} &\equiv \text{if } x \text{ then } \top \text{ else if } y \text{ then } \top \text{ else } \perp \\
 &\equiv \lambda x \lambda y. \textit{ifthenelse}(x)(\textit{true})(\textit{ifthenelse}(y)(\textit{true})(\textit{false})) \\
 \\
 \textit{Not} &\equiv \text{if } x \text{ then } \perp \text{ else } \top \\
 &\equiv \lambda x. \textit{ifthenelse}(x)(\textit{false})(\textit{true})
 \end{aligned}$$

## Esercizio 8

Mostrare che la chiusura transitiva mantiene l'inclusione

A lezione, è stata data la seguente definizione induttiva per la relazione transitiva  $S^*$ , data la relazione  $S$ :

$$\begin{array}{ll}
 \text{se } \langle x, y \rangle \in S & \text{allora } \langle x, y \rangle \in S^* \\
 \text{se } \langle x, z \rangle \in S^* \text{ e } \langle z, y \rangle \in S^* & \text{allora } \langle x, y \rangle \in S^*
 \end{array}$$

Dobbiamo dimostrare che, sapendo  $R \subseteq S$ , preso un  $\langle a, b \rangle \in R^*$  qualsiasi, allora  $\langle a, b \rangle \in S^*$ .

Abbiamo due casi possibili:

1. se  $\langle a, b \rangle \in R$  allora, per ipotesi, sappiamo che  $\langle a, b \rangle \in S$  e, di conseguenza  $\langle a, b \rangle \in S^*$ , dato che questo è costruito seguendo le regole illustrate;

2. se  $\langle a, b \rangle \notin R$  allora deve esistere  $c$  tale per cui  $\langle a, c \rangle \in R^*$  e  $\langle c, b \rangle \in R^*$ , ma allora questi ultimi due devono appartenere anche a  $S^*$  quindi, di conseguenza, anche  $\langle a, b \rangle \in S^*$  (per come, quest'ultimo è stato costruito).

## Esercizio 10

Dato un  $\lambda$ -termine, trovare  $\Gamma, \beta$  tali che nel contesto  $\Gamma$  il  $\lambda$ -termine abbia tipo  $\beta$ . Scrivere un algoritmo che risolva tale problema (fino a  $\rightarrow$ ).

La funzione principale è la funzione **TrovaTipo**, la quale utilizza altre funzioni d'appoggio:

---

**Algoritmo 1** Funzione per trovare il tipo di un termine

---

```

TrovaTipo( $b$ )
if  $typeof(b) = x : Var$  then
  if  $x \in \Gamma$  then
    return  $typeof(b)$ 
  else
     $t \leftarrow newType()$ 
     $\Gamma \leftarrow \Gamma \cup x : t$ 
     $M.add(t, t)$ 
  end if
else
  if  $typeof(b) = \lambda x.y$  then
    return  $TrovaTipo(x) \rightarrow TrovaTipo(y)$ 
  else
    if  $typeof(b) = x(y)$  then
       $a \leftarrow TrovaTipo(x)$ 
       $b \leftarrow TrovaTipo(y) \rightarrow dummy$ 
      return  $Tail(Unify(a, b, M))$ 
    end if
  end if
end if

```

---

## Esercizio 12

Vedere che  $\llbracket \alpha \rightarrow \beta \rrbracket$  è un saturo, per come è stato definito.

La definizione, che abbiamo dato a lezione, di *insieme saturo* prevede che, se  $S$  è un sottoinsieme saturo di  $\Lambda$ , valgano le seguenti:

$$\frac{x : Var \quad a_1 \dots a_n \in \mathcal{FN}}{x(a_1) \dots (a_n) \in S} \quad (1)$$

$$\frac{c[x := a](a_1) \dots (a_n) \in S \quad a \in \mathcal{FN}}{(\lambda x.c)(a)(a_1) \dots (a_n) \in S} \quad (2)$$

---

**Algoritmo 2** Funzione che fa l'unificazione

---

```
Unify( $\alpha, \beta, M$ )  
if  $count^{\rightarrow}(\alpha) \neq count^{\rightarrow}(\beta)$  then  
     $swap(\alpha, \beta)$   
end if  
if  $isSingleton(\alpha)$  then  
     $M.add(\alpha, \beta)$   
else  
     $Unify(Head(\alpha), Head(\beta), M)$   
     $Unify(Tail(\alpha), Tail(\beta), M)$   
end if  
 $t \leftarrow \alpha$   
for all  $e \in M$  do  
     $t \leftarrow t[e[1] := e[2]]$   
end for  
return  $t$ 
```

---

---

**Algoritmo 3** Funzione che corregge le sostituzioni nei termini in base alla tabella

---

```
M.adjust( $\alpha, \beta$ )  
for all  $e \in this$  do  
     $e[2] \leftarrow e[2][\alpha := \beta]$   
end for
```

---

---

**Algoritmo 4** Funzione che aggiunge un elemento alla “tabella di supporto”

---

```
M.add( $\alpha, \beta$ )  
for all  $e \in this$  do  
     $\alpha \leftarrow \alpha[[e[1] := e[2]]]$   
     $\beta \leftarrow \beta[[e[1] := e[2]]]$   
end for  
if  $isSingleton(\beta)$  then  
     $swap(\alpha, \beta)$   
end if  
if  $\alpha \in \{\beta\}$  then  
     $throw\ error$   
end if  
 $M.adjust(\alpha, \beta)$   
 $M.append(\alpha, \beta)$ 
```

---

Ora, l'interpretazione che è stata data per  $\llbracket \alpha \rightarrow \beta \rrbracket$  in  $\Lambda \rightarrow$  è:

$$\llbracket \alpha \rightarrow \beta \rrbracket \equiv \{t \in \Lambda \mid \forall u \in \llbracket \alpha \rrbracket t(u) \in \llbracket \beta \rrbracket\}$$

Sapendo che  $\llbracket \alpha \rrbracket \in \text{SAT}$  e  $\llbracket \beta \rrbracket \in \text{SAT}$ , dimostriamo i singoli passi:

1.  $\forall x : \text{Var}, \forall (a_1 \dots a_n) \in \mathcal{FN}$  vogliamo vedere se  $x(a_1) \dots (a_n) \stackrel{?}{\in} \llbracket \alpha \rightarrow \beta \rrbracket$ .  
Ciò vale poiché  $\forall u \in \llbracket \alpha \rrbracket$ , allora  $u \in \mathcal{FN}$  (perché  $\llbracket \alpha \rrbracket$  è SAT), quindi  $x(a_1) \dots (a_n)(u) \in \llbracket \beta \rrbracket$  poiché  $\llbracket \beta \rrbracket$  è SAT (e quindi soddisfa la prima condizione);
2. supponiamo  $c[x := a](a_1) \dots (a_n) \in \llbracket \alpha \rightarrow \beta \rrbracket$ , allora  $\forall u \in \llbracket \alpha \rrbracket$ ,  $c[x := a](a_1) \dots (a_n)(u) \in \llbracket \beta \rrbracket$  (poiché SAT). Ma dato che  $\llbracket \beta \rrbracket$  è SAT, anche la (2) è valida per questo insieme, e quindi  $(\lambda x.c)(a)(a_1) \dots (a_n)(u) \in \llbracket \beta \rrbracket$  e ciò implica che  $(\lambda x.c)(a)(a_1) \dots (a_n) \in \llbracket \alpha \rightarrow \beta \rrbracket$ .

## Esercizio 15

*Mostrare che l'intersezione e l'unione di saturi sono saturi.*

Dimostriamo singolarmente, prima l'unione e poi l'intersezione.

**L'unione di saturi è saturo.** Come prima cosa, mostriamo che vale la condizione (1). Per fare ciò dobbiamo mostrare che  $\forall x : \text{Var}$  e  $\forall (a_1 \dots a_n) \mathcal{FN}$ ,  $x(a_1) \dots (a_n) \in \bigcup_{\text{SAT}}$  ma, poiché ciò vale per ciascun  $S$ , a maggior ragione deve valere per l'insieme  $\bigcup_{\text{SAT}}$ .

Per la condizione (2), sappiamo che  $\forall a \mathcal{FN}$  se  $c[x := a](a_1) \dots (a_n) \in \bigcup_{\text{SAT}}$  allora  $(\lambda x.c)(a)(a_1) \dots (a_n) \in \bigcup_{\text{SAT}}$ . Ma se  $c[x := a](a_1) \dots (a_n) \in \bigcup_{\text{SAT}}$  allora  $\exists S_i$  tale che  $c[x := a](a_1) \dots (a_n) \in S_i$ , quindi  $(\lambda x.c)(a_1) \dots (a_n) \in S_i$  e, a maggior ragione,  $\in \bigcup_{\text{SAT}}$ .

**L'intersezione di saturi è saturo.** Anche qui, mostriamo che vale la (1):  $\forall x : \text{Var}$  e  $\forall (a_1 \dots a_n) \mathcal{FN}$ ,  $x(a_1) \dots (a_n) \in \bigcap_{\text{SAT}}$  se  $\forall S_i \in \bigcap_{\text{SAT}}$  abbiamo che  $x(a_1) \dots (a_n) \in S_i$ , e ciò accade per definizione di saturo.

Analogamente,  $c[x := a](a_1) \dots (a_n) \in \bigcap_{\text{SAT}}$  se  $\forall S_i \in \bigcap_{\text{SAT}}$  accade che  $c[x := a](a_1) \dots (a_n) \in S_i$  e ciò accade per definizione di saturo.

## Esercizio 21

*Introdurre il tipo "unione disgiunta".*

Come prima cosa, diamo le **regole di introduzione**:

$$\frac{\Gamma \vdash a : \alpha}{\Gamma \vdash f(a) : \alpha \oplus \beta} \quad \frac{\Gamma \vdash b : \beta}{\Gamma \vdash s(b) : \alpha \oplus \beta} \quad (3)$$

Ora possiamo indicare la **regola di eliminazione**:

$$\frac{\Gamma \vdash c : \alpha \oplus \beta \quad \Gamma \vdash d : \alpha \rightarrow \gamma \quad \Gamma \vdash e : \beta \rightarrow \gamma}{\Gamma \vdash \text{when}(c, d, e) : \gamma} \quad (4)$$

A questo punto, sappiamo come si deve “comportare” la funzione, dandone le **regole computazionali**:

$$\begin{aligned} \text{when}(f(a), d, e) &\rightsquigarrow d(a) \\ \text{when}(s(b), d, e) &\rightsquigarrow e(b) \end{aligned}$$

L’interpretazione del tipo  $\alpha \oplus \beta$  nei saturi è la seguente:

$$\begin{aligned} \llbracket f(a) \rrbracket &\equiv \lambda x \lambda y. x(\llbracket a \rrbracket) \\ \llbracket s(b) \rrbracket &\equiv \lambda x \lambda y. y(\llbracket b \rrbracket) \\ \llbracket \text{when}(c, d, e) \rrbracket &\equiv (\lambda x \lambda y \lambda z. x(y)(z))(\llbracket c \rrbracket)(\llbracket d \rrbracket)(\llbracket e \rrbracket) \\ &\equiv \llbracket c \rrbracket(\llbracket d \rrbracket)(\llbracket e \rrbracket) \end{aligned}$$

A questo punto, è necessario fornire delle dimostrazioni che ci diano la certezza che le regole di introduzione e quella di eliminazione sono contenute nell’interpretazione del tipo  $\alpha \oplus \beta$ . Costruiamo quindi due interpretazioni differenti, dette “massima” e “minima” che si pongono nella seguente relazione:

$$\boxed{\llbracket \alpha \oplus \beta \rrbracket_{min} \subseteq \llbracket \alpha \oplus \beta \rrbracket \subseteq \llbracket \alpha \oplus \beta \rrbracket_{max}}$$

Diamo ora la definizione delle due interpretazioni.

L’**interpretazione minima** è costruita a partire dalle regole di introduzione (3), ed è la seguente:

$$\llbracket \alpha \oplus \beta \rrbracket_{min} \equiv \bigcap_{S \in \mathbb{SAT}} S \cup \{ \lambda x \lambda y. x(a) \mid a \in \llbracket \alpha \rrbracket \} \cup \{ \lambda x \lambda y. y(b) \mid b \in \llbracket \beta \rrbracket \}$$

L’**interpretazione massima** è invece definita tramite la regola di eliminazione (4), ed è la seguente:

$$\begin{aligned} \llbracket \alpha \oplus \beta \rrbracket_{max} &\equiv \{ t \in \Lambda \mid (\forall S \in \mathbb{SAT})(\forall d \in \llbracket \alpha \rrbracket \rightarrow S)(\forall e \in \llbracket \beta \rrbracket \rightarrow S) t(d)(e) \in S \} \\ &\equiv \bigcap_{S \in \mathbb{SAT}} ((\llbracket \alpha \rrbracket \rightarrow S) \rightarrow ((\llbracket \beta \rrbracket \rightarrow S) \rightarrow S)) \end{aligned}$$

Ora, ciò che dobbiamo verificare è che  $\llbracket \alpha \oplus \beta \rrbracket_{min} \subseteq \llbracket \alpha \oplus \beta \rrbracket_{max}$  e per fare ciò devo controllare che  $\llbracket \alpha \oplus \beta \rrbracket_{max}$  contenga i termini usati per generare il “minimo”, ovvero  $\lambda x \lambda y. x(\llbracket a \rrbracket)$  e  $\lambda x \lambda y. y(\llbracket b \rrbracket)$ . Ciò basta poiché  $\llbracket \alpha \oplus \beta \rrbracket_{max}$ , se è  $\mathbb{SAT}$  e contiene  $\llbracket f(a) \rrbracket$  e  $\llbracket s(b) \rrbracket$ , allora vuol dire che ho considerato anche il “massimo” quando ho fatto l’intersezione per generare  $\llbracket \alpha \oplus \beta \rrbracket_{min}$  che, di conseguenza, lo contiene.

Dobbiamo quindi verificare che:

$$(\lambda x \lambda y. x(\llbracket a \rrbracket))(d)(e) \stackrel{?}{\in} S$$

ciò è equivalente a chiedersi se

$$d(\llbracket a \rrbracket) \stackrel{?}{\in} S$$

e ciò è vero poiché sappiamo che  $d \in \llbracket \alpha \rrbracket \rightarrow S$ , gli applichiamo un elemento di tipo  $\alpha$  e, conseguentemente, otteniamo qualcosa di  $S$ .

Con un procedimento del tutto analogo, si può dimostrare anche l'altra regola di introduzione genera un sottoinsieme del “massimo”.

Vediamo ora la dimostrazione del **Teorema di Validità**: dobbiamo verificare che:

$$\begin{array}{ccccccc} x_1 : \alpha_1 & \dots & x_n : \alpha_n & \vdash & a : \alpha \\ \downarrow & & \downarrow & & \\ g_1 \in \llbracket \alpha_1 \rrbracket & \dots & g_n \in \llbracket \alpha_n \rrbracket & \vdash & \llbracket a \rrbracket[\bar{x} := \bar{g}] \in \llbracket \alpha \rrbracket \end{array}$$

Dimostriamo, per induzione, la prima regola di introduzione:

$$\frac{\Gamma \vdash a : \alpha}{\Gamma \vdash f(a) : \alpha \oplus \beta}$$

devo mostrare che  $\llbracket f(a) \rrbracket[\bar{x} := \bar{g}] \in \llbracket \alpha \oplus \beta \rrbracket$ :

$$\begin{aligned} \llbracket f(a) \rrbracket[\bar{x} := \bar{g}] &\equiv \lambda x \lambda y. x(\llbracket a \rrbracket[\bar{x} := \bar{g}]) \\ &\equiv \lambda x \lambda y. x(\llbracket a \rrbracket[\bar{x} := \bar{g}]) \end{aligned}$$

ora, sappiamo che per ipotesi induttiva,  $\llbracket a \rrbracket[\bar{x} := \bar{g}] \in \llbracket \alpha \rrbracket$ , e quindi

$$\lambda x \lambda y. x(\llbracket a \rrbracket[\bar{x} := \bar{g}]) \in \llbracket \alpha \oplus \beta \rrbracket$$

La dimostrazione per l'altra regola di introduzione è del tutto analoga, e viene omessa.

Dimostriamo ora il teorema per la regola di eliminazione:

$$\frac{\Gamma \vdash c : \alpha \oplus \beta \quad \Gamma \vdash d : \alpha \rightarrow \gamma \quad \Gamma \vdash e : \beta \rightarrow \gamma}{\Gamma \vdash \text{when}(c, d, e) : \gamma}$$

ovvero, dobbiamo verificare che  $\llbracket \text{when}(c, d, e) \rrbracket[\bar{x} := \bar{g}] \in \llbracket \gamma \rrbracket$ , sapendo che:

$$\begin{aligned} \llbracket \text{when}(c, d, e) \rrbracket[\bar{x} := \bar{g}] &\equiv \llbracket c \rrbracket(\llbracket d \rrbracket)(\llbracket e \rrbracket)[\bar{x} := \bar{g}] \\ &\equiv \llbracket c \rrbracket[\bar{x} := \bar{g}](\llbracket d \rrbracket[\bar{x} := \bar{g}])(\llbracket e \rrbracket[\bar{x} := \bar{g}]) \end{aligned}$$

ora, per ipotesi induttiva so che:

$$\begin{aligned} \llbracket c \rrbracket[\bar{x} := \bar{g}] &\in \llbracket \alpha \oplus \beta \rrbracket \\ \llbracket d \rrbracket[\bar{x} := \bar{g}] &\in \llbracket \alpha \rrbracket \rightarrow \llbracket \gamma \rrbracket \\ \llbracket e \rrbracket[\bar{x} := \bar{g}] &\in \llbracket \beta \rrbracket \rightarrow \llbracket \gamma \rrbracket \end{aligned}$$

dove, sappiamo che  $\llbracket \gamma \rrbracket$  è saturo. Quindi, deduciamo che  $\llbracket c \rrbracket[\bar{x} := \bar{g}] \in (\llbracket \alpha \rrbracket \rightarrow \llbracket \gamma \rrbracket) \rightarrow ((\llbracket \beta \rrbracket \rightarrow \llbracket \gamma \rrbracket) \rightarrow \llbracket \gamma \rrbracket)$ , che è un elemento di  $\llbracket \alpha \oplus \beta \rrbracket_{max}$ .

Vediamo, infine, la corretta **terminazione dei calcoli**, ovvero verifichiamo che un calcolo nel sistema tipato richiede almeno un calcolo nell'interpretazione:

$$\begin{array}{ccc}
f(a) & \llbracket f(a) \rrbracket & s(a) & \llbracket s(a) \rrbracket \\
\downarrow & \downarrow & \downarrow & \downarrow \\
a \rightarrow a' & \text{1 passo} & a \rightarrow a' & \text{1 passo} \\
\downarrow & \downarrow & \downarrow & \downarrow \\
f(a') & \llbracket f(a') \rrbracket & s(a') & \llbracket s(a') \rrbracket
\end{array}$$

Infine, per la regola di eliminazione:

$$\begin{array}{c}
\text{when}(c, d, e) \\
\swarrow \quad \downarrow \quad \searrow \\
\text{when}(c', d, e) \quad \text{when}(c, d', e) \quad \text{when}(c, d, e') \quad d(a) \quad e(b)
\end{array}$$
  

$$\begin{array}{c}
\llbracket \text{when}(c, d, e) \rrbracket \\
\swarrow \quad \downarrow \quad \searrow \\
\llbracket \text{when}(c', d, e) \rrbracket \quad \llbracket \text{when}(c, d', e) \rrbracket \quad \llbracket \text{when}(c, d, e') \rrbracket \quad d(\llbracket a \rrbracket) \quad e(\llbracket b \rrbracket)
\end{array}$$

Quindi, con un numero finito di passi, riesco a terminare anche nel “versante” dell’interpretazione. Ricordiamo che un numero  $\infty$  di passi nel calcolo tipato implica  $\infty$  passi nell’interpretazione, ma poiché qui non possiamo avere  $\infty$  passi, allora possiamo essere sicuri della terminazione del calcolo.

## Esercizio 23

Vedere che per le macchine categoriali la sostituzione è associativa.

Siano  $f$ ,  $h$  e  $g$  tre funzioni così definite:

$$\begin{aligned}
f &\equiv x : \alpha \vdash a : \beta \\
g &\equiv y : \beta \vdash b : \gamma \\
h &\equiv z : \gamma \vdash c : \delta
\end{aligned}$$

due loro composizioni possono essere scritte nella seguente maniera:

$$\begin{aligned}
f; g &\equiv x : \alpha \vdash b[y := a] : \gamma \\
g; h &\equiv y : \beta \vdash c[z := b] : \delta
\end{aligned}$$

e queste, a loro volta composte sono:

$$\begin{aligned}
(f; g); h &\equiv x : \alpha \vdash c[z := b[y := a]] : \delta \\
f; (g; h) &\equiv x : \alpha \vdash c[z := b][y := a] : \delta
\end{aligned}$$

A questo punto, il problema si sposta nel chiedersi se i tipi dei “codomini” delle funzioni sono gli stessi. Consideriamo il caso in cui  $y \notin FV(c)$ , allora è possibile spostare la seconda sostituzione della seconda composizione di funzioni all’interno, ottenendo:

$$c[z := b[y := a]] \stackrel{?}{=} c[z := b][y := a]$$

Anche facendo le opportune sostituzioni ai tipi delle due composizioni originarie risulta che:

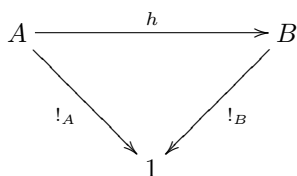
$$\begin{aligned}(\lambda z.c)((\lambda y.b)(a)) &\equiv c \\ (\lambda y.((\lambda z.c)(b)))(a) &\equiv c\end{aligned}$$

In effetti, si può osservare che  $y \notin FV(c)$ , in quanto appartenenti a funzioni diverse, e non essendoci variabili globali, se  $y$  compare in  $c$  allora deve essere astratta.

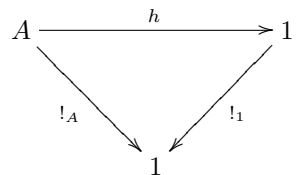
## Esercizio 24

Dimostrare che, in virtù delle regole  $f; !_B = !_A$  e  $id_1 = !_1$ , si può dimostrare che per ogni freccia  $h$  da  $A$  in  $1$ :  $!_A = h$ .

La relazione che si chiede di dimostrare può essere rappresentata nella seguente maniera:



Dato che le premesse valgono per un qualsiasi  $B$  consideriamo il caso in cui  $B = 1$ , abbiamo quindi la seguente rappresentazione:



e, sappiamo valere le seguenti:

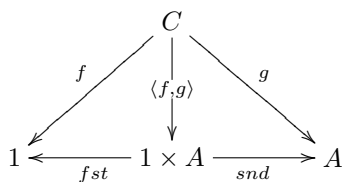
$$h; !_1 = !_A \equiv h; id_1 = !_A \equiv h = !_A$$

che conclude l'esercizio.

## Esercizio 25

Dimostrare che  $1 \times A \cong A$ .

Dal punto di vista grafico, la relazione  $1 \times A$  può essere rappresentata in questa maniera:



al fine di avere un isomorfismo è necessario che esistano

$$\begin{aligned} F &: A \rightarrow 1 \times A \\ G &: 1 \times A \rightarrow A \end{aligned}$$

tali per cui valgano le seguenti:

$$F; G = id_A \quad (5)$$

$$G; F = id_{1 \times A} \quad (6)$$

la cui rappresentazione grafica è:

$$id_A \circlearrowleft A \begin{array}{c} \xrightarrow{F} \\ \xleftarrow{G} \end{array} 1 \times A \circlearrowright id_{1 \times A}$$

Ora, poiché nell'esempio precedente, con  $C$  ci si riferisce a qualsiasi categoria, adeguiamo il disegno precedente al nostro particolare caso:

$$\begin{array}{ccccc} & & A & & \\ & \swarrow !_A & \downarrow \langle !_A, id_A \rangle & \searrow id_A & \\ 1 & \xleftarrow{fst} & 1 \times A & \xrightarrow{snd} & A \end{array}$$

In questa situazione, le funzioni  $F$  e  $G$  valgono rispettivamente:

$$\begin{aligned} F &\equiv \langle !_A, id_A \rangle \\ G &\equiv snd \end{aligned}$$

Qui osserviamo che

$$\langle !_A, id_A \rangle; snd = id_A$$

vale “per definizione”, in quanto, tramite la funzione  $snd$  andiamo a considerare solo la seconda componente del prodotto cartesiano, ed in questa maniera abbiamo dimostrato la (5).

Per dimostrare la (6), osserviamo che:

$$\begin{aligned} snd; \langle !_A, id_A \rangle &= \langle snd; !_A, snd; id_A \rangle \\ &= \langle fst, snd \rangle \\ &= id_{1 \times A} \end{aligned}$$

In questa maniera, abbiamo dimostrato l'isomorfismo.

## Esercizio 27

Mostrare che  $A \times B \cong B \times A$ .

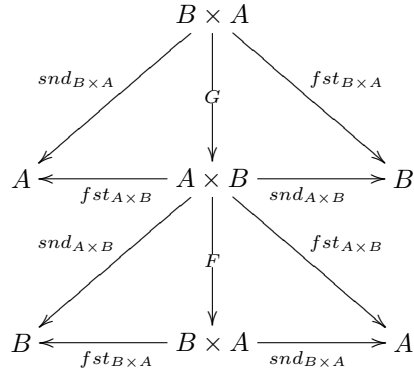
Come fatto per l'Esercizio 25, dobbiamo verificare l'esistenza delle funzioni

$$\begin{aligned} F &: A \times B \rightarrow B \times A \\ G &: B \times A \rightarrow A \times B \end{aligned}$$

tali per cui valgano le seguenti:

$$\begin{aligned} F;G &= id_{A \times B} \\ G;F &= id_{B \times A} \end{aligned}$$

Vogliamo dimostrare, innanzitutto, la vericità dell'affermazione  $G;F = id_{B \times A}$ . Dal punto di vista grafico, la relazione che si vuole rappresentare è la seguente:



La verifica dell'equivalenza è data da:

$$\begin{aligned} G;F &= \langle snd_{B \times A}, fst_{B \times A} \rangle; \langle snd_{A \times B}, fst_{A \times B} \rangle \\ &= \langle \langle snd_{B \times A}, fst_{B \times A} \rangle; snd_{A \times B}, \langle snd_{B \times A}, fst_{B \times A} \rangle; fst_{A \times B} \rangle \\ &= \langle fst_{B \times A}, snd_{B \times A} \rangle \\ &= id_{B \times A} \end{aligned}$$

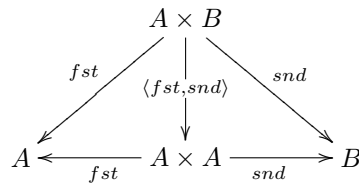
Per l'altra equivalenza, lo schema grafico è del tutto analogo, si riportano qui solamente i calcoli:

$$\begin{aligned} F;G &= \langle snd_{A \times B}, fst_{A \times B} \rangle; \langle snd_{B \times A}, fst_{B \times A} \rangle \\ &= \langle \langle snd_{A \times B}, fst_{A \times B} \rangle; snd_{B \times A}, \langle snd_{A \times B}, fst_{A \times B} \rangle; fst_{B \times A} \rangle \\ &= \langle fst_{A \times B}, snd_{A \times B} \rangle \\ &= id_{A \times B} \end{aligned}$$

## Esercizio 28

Mostrare che  $\langle fst, snd \rangle = id_{A \times B}$ .

Dal punto di vista grafico, l'uguaglianza che si chiede di dimostrare può essere visualizzata nel seguente grafico:



Ci chiediamo quindi:

$$\langle fst, snd \rangle \stackrel{?}{=} id_{A \times B}$$

in virtù della proprietà delle categorie che ci garantisce la seguente uguaglianza:

$$h = \langle h; fst, h; snd \rangle$$

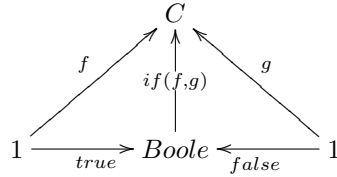
possiamo scrivere:

$$\begin{aligned} id_{A \times B} &= \langle id_{A \times B}; fst, id_{A \times B}; snd \rangle \\ &= id_{A \times B}; \langle fst, snd \rangle \\ &= \langle fst, snd \rangle \end{aligned}$$

## Esercizio 31

Aggiungere alla macchina categoriale le istruzioni per il tipo *Boole*.

Dal punto di vista grafico, il tipo *Boole* può essere rappresentato in questa maniera:



Le regole di calcolo sono quindi le seguenti:

$$\begin{aligned} true; if(f, g) &\equiv f \\ false; if(f, g) &\equiv g \end{aligned}$$

La cui interpretazione nelle categorie è la seguente:

$$\begin{aligned} \llbracket true \rrbracket_{env} &= true \\ \llbracket false \rrbracket_{env} &= false \\ \llbracket ifthenelse(c, d, e) \rrbracket_{env} &= \llbracket c \rrbracket_{env}; if(\llbracket d \rrbracket_{env}, \llbracket e \rrbracket_{env}) \end{aligned}$$

Dalla quale seguono le interpretazioni macchina:

$$\begin{aligned} \llbracket true \rrbracket_{env} &= \top \\ \llbracket false \rrbracket_{env} &= \perp \\ \llbracket ifthenelse(c, d, e) \rrbracket_{env} &= PUSH; \llbracket c \rrbracket_{env}; if(\llbracket d \rrbracket_{env}, \llbracket e \rrbracket_{env}) \end{aligned}$$

Le regole di calcolo sono quindi:

| Env.                | Code          | Stack       |
|---------------------|---------------|-------------|
| <b>True</b>         |               |             |
| $\Delta$            | $\top; C$     | $\$$        |
| $\Delta.\top$       | $C$           | $\$$        |
| <b>False</b>        |               |             |
| $\Delta$            | $\perp; C$    | $\$$        |
| $\Delta.\perp$      | $C$           | $\$$        |
| <b>If Then Else</b> |               |             |
| $\top$              | $if(d, e); C$ | $\Delta.\$$ |
| $\Delta$            | $d; C$        | $\$$        |
| $\perp$             | $if(d, e); C$ | $\Delta.\$$ |
| $\Delta$            | $e; C$        | $\$$        |

Un semplice esempio, di esecuzione del codice è il seguente:

$\llbracket ifthenelse(true, a, b) \rrbracket$

| Env.       | Code   | Stack       |
|------------|--|-------------|
| $\Delta$   | PUSH; $\llbracket true \rrbracket; if(a, b)$ | $\$$        |
| $\Delta$   | $\llbracket true \rrbracket; if(a, b)$       | $\Delta.\$$ |
| $\Delta.T$ | $if(a, b)$                                   | $\Delta.\$$ |
| $\Delta$   | $a$  | $\$$        |

## Esercizio 35

Definire il tipo ricorsivo lista e studiarne la normalizzazione forte.

Come prima cosa, diamo le **regole di introduzione**:

$$\Gamma \vdash \emptyset_\alpha : List_\alpha \quad \frac{\Gamma \vdash a : \alpha \quad \Gamma \vdash l : List_\alpha}{\Gamma \vdash a.l : List_\alpha} \quad (7)$$

Ora possiamo indicare la **regola di eliminazione** ricorsiva:

$$\frac{\Gamma \vdash c : List_\alpha \quad \Gamma \vdash d : \gamma \quad \Gamma \vdash e : \alpha \rightarrow (List_\alpha \rightarrow (\gamma \rightarrow \gamma))}{\Gamma \vdash lrec(c, d, e) : \gamma} \quad (8)$$

A questo punto, sappiamo come si deve “comportare” la funzione, dandone le **regole computazionali**:

$$\begin{aligned} lrec(\emptyset_\alpha, d, e) &\rightsquigarrow d \\ lrec(a.l, d, e) &\rightsquigarrow e(a, l, lrec(l, d, e)) \end{aligned}$$

Tramite queste regole di calcolo sono in grado di risolvere le equazioni in questa forma:

$$\begin{cases} f(\emptyset_\alpha) = k : \gamma \\ f(a.l) = g(a, l, f(l)) : \gamma \end{cases}$$

e, a partire da queste, definisco uno schema iterativo nella seguente maniera:

$$\begin{cases} F(\emptyset_\alpha) = K : \gamma' \\ F(a.l) = G(a, F(l)) : \gamma' \end{cases}$$

si deve verificare che lo schema ricorsivo e quello iterativo siano equivalenti. In particolare, se so risolvere quello iterativo allora conosco la risoluzione anche del ricorsivo.

Definiamo la funzione ricorsiva in questa maniera:

$$F(l) = \langle l, f(l) \rangle$$

Verifichiamo quindi la seguente equivalenza:

$$F(l) \stackrel{?}{\equiv} f(l)$$

che è vera, in quanto:

$$\begin{cases} f(\emptyset_\alpha) = \Pi_2(F(\emptyset_\alpha)) \\ f(l) = \Pi_2(F(l)) \end{cases}$$

A questo punto, ci risulta possibile dare una definizione per  $G$ :

$$\begin{aligned} F(a.l) &= \langle a.l, f(a.l) \rangle \\ &= \langle a.l, g(a, l, f(l)) \rangle \\ &= \langle a.\Pi_1(F(l)), g(a, \Pi_1(F(l))), \Pi_2(F(l)) \rangle \\ &= G(a, F(l)) \end{aligned}$$

La **regola di eliminazione** definita secondo lo schema iterativo diventa quindi:

$$\frac{\Gamma \vdash c : List_\alpha \quad \Gamma \vdash d : \gamma \quad \Gamma \vdash e : \alpha \rightarrow (\gamma \rightarrow \gamma)}{\Gamma \vdash lite(c, d, e) : \gamma} \quad (9)$$

Le nuove **regole computazionali** diventano:

$$\begin{aligned} lite(\emptyset_\alpha, d, e) &\rightsquigarrow d \\ lite(a.l, d, e) &\rightsquigarrow e(a, lite(l, d, e)) \end{aligned}$$

L'interpretazione nei saturi è la seguente:

$$\begin{aligned} \llbracket \emptyset_\alpha \rrbracket &\equiv \lambda x \lambda y. x \\ \llbracket a.l \rrbracket &\equiv \lambda x \lambda y. y(\llbracket a \rrbracket)(\llbracket l \rrbracket(x)(y)) \\ \llbracket lite(c, d, e) \rrbracket &\equiv (\lambda x \lambda y \lambda z. x(y)(z))(\llbracket c \rrbracket)(\llbracket d \rrbracket)(\llbracket e \rrbracket) \\ &\equiv \llbracket c \rrbracket(\llbracket d \rrbracket)(\llbracket e \rrbracket) \end{aligned}$$

A questo punto, è necessario fornire delle dimostrazioni che ci diano la certezza che le regole di introduzione e quella di eliminazione sono contenute nell'interpretazione del tipo  $List_\alpha$ . Costruiamo quindi due interpretazioni differenti, dette "massima" e "minima" che si pongono nella seguente relazione:

$$\boxed{\llbracket List_\alpha \rrbracket_{min} \subseteq \llbracket List_\alpha \rrbracket \subseteq \llbracket List_\alpha \rrbracket_{max}}$$

Diamo ora la definizione delle due interpretazioni.

L'**interpretazione minima** è costruita a partire dalle regole di introduzione (7), ed è la seguente:

$$\begin{aligned} \llbracket List_\alpha \rrbracket_{min} &\equiv \bigcap S \\ &\quad \{ \lambda x \lambda y. x \} \cup \\ &\quad \{ \lambda x \lambda y. y(a)(l(x)(y)) \mid a \in \llbracket \alpha \rrbracket, l \in \llbracket List_\alpha \rrbracket \} \subseteq \\ &\quad S \in \text{SAT} \end{aligned}$$

L'interpretazione massima è invece definita tramite la regola di eliminazione (9), ed è la seguente:

$$\begin{aligned} \llbracket List_\alpha \rrbracket_{max} &\equiv \{t \in \Lambda \mid (\forall S \in \mathbb{SAT})(\forall d \in S)(\forall e \in \llbracket \alpha \rrbracket \rightarrow (S \rightarrow S)) t(d)(e) \in S\} \\ &\equiv \bigcap_{S \in \mathbb{SAT}} S \rightarrow ((\llbracket \alpha \rrbracket \rightarrow (S \rightarrow S)) \rightarrow S) \end{aligned}$$

Ora, ciò che dobbiamo verificare è che  $\llbracket List_\alpha \rrbracket_{min} \subseteq \llbracket List_\alpha \rrbracket_{max}$  e per fare ciò devo controllare che  $\llbracket List_\alpha \rrbracket_{max}$  contenga i termini usati per generare il “minimo”, ovvero  $\lambda x \lambda y. x$  e  $\lambda x \lambda y. y(\llbracket a \rrbracket)(\llbracket l \rrbracket)(x)(y)$ . Ciò basta poiché  $\llbracket List_\alpha \rrbracket_{max}$ , se è  $\mathbb{SAT}$  e contiene  $\llbracket \emptyset_\alpha \rrbracket$  e  $\llbracket a.l \rrbracket$ , allora vuol dire che ho considerato anche il “massimo” quando ho fatto l'intersezione per generare  $\llbracket List_\alpha \rrbracket_{min}$  che, di conseguenza, lo contiene.

Dobbiamo quindi verificare che:

$$\lambda x \lambda y. x(d)(e) \stackrel{?}{\in} S$$

ciò è equivalente a chiedersi se

$$d \stackrel{?}{\in} S$$

e ciò è vero per ipotesi ( $d \in \mathbb{SAT}$ ).

Vediamo ora l'altra regola introduttiva:

$$(\lambda x \lambda y. y(\llbracket a \rrbracket)(\llbracket l \rrbracket)(x)(y))(d)(e) \stackrel{?}{\in} S$$

ciò è equivalente a chiedersi se

$$e(\llbracket a \rrbracket)(\llbracket l \rrbracket)(d)(e) \stackrel{?}{\in} S$$

che è anch'essa vera poiché  $a \in \llbracket \alpha \rrbracket$  e  $l \in \llbracket List_\alpha \rrbracket \in \llbracket List_\alpha \rrbracket_{max}$ , per ipotesi induttiva. Concludiamo quindi, verificando che  $\llbracket List_\alpha \rrbracket_{min} \subseteq \llbracket List_\alpha \rrbracket_{max}$ .

Vediamo ora la dimostrazione del **Teorema di Validità**: dobbiamo verificare che:

$$\begin{array}{ccccccc} x_1 : \alpha_1 & \dots & x_n : \alpha_n & \vdash & a : \alpha \\ \downarrow & & \downarrow & & \\ g_1 \in \llbracket \alpha_1 \rrbracket & \dots & g_n \in \llbracket \alpha_n \rrbracket & \vdash & \llbracket a \rrbracket[\bar{x} := \bar{g}] \in \llbracket \alpha \rrbracket \end{array}$$

Dimostriamo, per induzione, la prima per la prima regola di introduzione:

$$\Gamma \vdash \emptyset_\alpha : List_\alpha$$

devo mostrare che  $\llbracket \emptyset_\alpha \rrbracket[\bar{x} := \bar{g}] \in \llbracket List_\alpha \rrbracket$ :

$$\begin{aligned} \llbracket \emptyset_\alpha \rrbracket[\bar{x} := \bar{g}] &\equiv \lambda x \lambda y. x[\bar{x} := \bar{g}] \\ &\equiv \lambda x \lambda y. x \end{aligned}$$

ora, la sostituzione non interessa la  $x$ , in quanto astratta, e quindi:

$$\lambda x \lambda y. x \in \llbracket List_\alpha \rrbracket_{min} \subseteq \llbracket List_\alpha \rrbracket$$

Dimostriamo ora, sempre per induzione, la seconda regola di introduzione:

$$\frac{\Gamma \vdash a : \alpha \quad \Gamma \vdash l : List_\alpha}{\Gamma \vdash a.l : List_\alpha}$$

devo mostrare che  $\llbracket a.l \rrbracket [\bar{x} := \bar{g}] \in \llbracket List_\alpha \rrbracket$ :

$$\begin{aligned} \llbracket a.l \rrbracket [\bar{x} := \bar{g}] &\equiv \lambda x \lambda y. y(\llbracket a \rrbracket)(\llbracket l \rrbracket(x)(y))[\bar{x} := \bar{g}] \\ &\equiv \lambda x \lambda y. y(\llbracket a \rrbracket[\bar{x} := \bar{g}])(\llbracket l \rrbracket[\bar{x} := \bar{g}](x)(y)) \end{aligned}$$

ora, per ipotesi induttiva so che  $a[\bar{x} := \bar{g}] \in \llbracket \alpha \rrbracket$  e che  $l[\bar{x} := \bar{g}] \in \llbracket List_\alpha \rrbracket$ , e quindi:

$$\lambda x \lambda y. y(\llbracket a \rrbracket[\bar{x} := \bar{g}])(\llbracket l \rrbracket[\bar{x} := \bar{g}](x)(y)) \in \llbracket List_\alpha \rrbracket_{min} \subseteq \llbracket List_\alpha \rrbracket$$

Dimostriamo ora il teorema per la regola di eliminazione iterativa:

$$\frac{\Gamma \vdash c : List_\alpha \quad \Gamma \vdash d : \gamma \quad \Gamma \vdash e : \alpha \rightarrow (\gamma \rightarrow \gamma)}{\Gamma \vdash lite(c, d, e) : \gamma}$$

ovvero, dobbiamo verificare che  $\llbracket lite(c, d, e) \rrbracket [\bar{x} := \bar{g}] \in \llbracket \gamma \rrbracket$ , sapendo che:

$$\begin{aligned} \llbracket lite(c, d, e) \rrbracket [\bar{x} := \bar{g}] &\equiv \llbracket c \rrbracket(\llbracket d \rrbracket)(\llbracket e \rrbracket)[\bar{x} := \bar{g}] \\ &\equiv \llbracket c \rrbracket[\bar{x} := \bar{g}](\llbracket d \rrbracket[\bar{x} := \bar{g}])(\llbracket e \rrbracket[\bar{x} := \bar{g}]) \end{aligned}$$

ora, per ipotesi induttiva, so che:

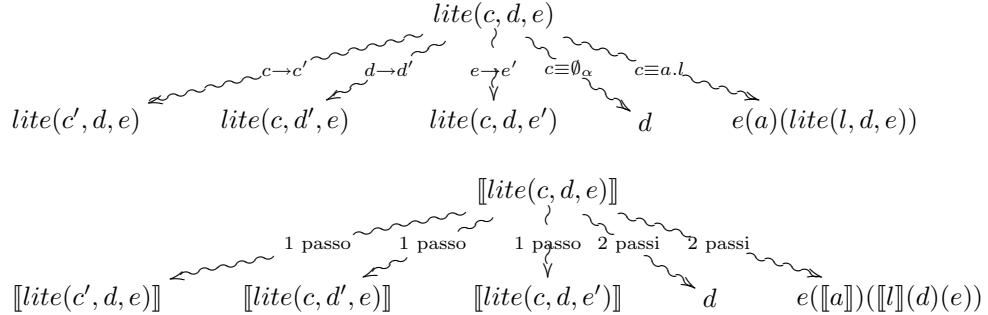
$$\begin{aligned} \llbracket c \rrbracket[\bar{x} := \bar{g}] &\in \llbracket List_\alpha \rrbracket \\ \llbracket d \rrbracket[\bar{x} := \bar{g}] &\in \llbracket \gamma \rrbracket \\ \llbracket e \rrbracket[\bar{x} := \bar{g}] &\in \llbracket \alpha \rrbracket \rightarrow (\llbracket \gamma \rrbracket \rightarrow \llbracket \gamma \rrbracket) \end{aligned}$$

dove, sappiamo che  $\llbracket \gamma \rrbracket$  è saturo. Quindi, deduciamo che  $\llbracket c \rrbracket[\bar{x} := \bar{g}] \in \llbracket \gamma \rrbracket \rightarrow (\llbracket a \rrbracket \rightarrow (\llbracket \gamma \rrbracket \rightarrow \llbracket \gamma \rrbracket)) \rightarrow \llbracket \gamma \rrbracket$ , che è un elemento di  $\llbracket List_\alpha \rrbracket_{max}$ .

Vediamo, infine, la corretta **terminazione dei calcoli**, ovvero verifichiamo che un calcolo nel sistema tipato richiede almeno un calcolo nell'interpretazione. Per quanto riguarda  $\emptyset_\alpha$ , questo è un valore ed è già in forma normale, vediamo per le altre:



Per la regola di eliminazione:



Quindi, con un numero finito di passi, riesco a terminare anche nel “versante” dell’interpretazione. Ricordiamo che un numero  $\infty$  di passi nel calcolo tipato implica  $\infty$  passi nell’interpretazione, ma poiché qui non possiamo avere  $\infty$  passi, allora possiamo essere sicuri della terminazione del calcolo.

Per quanto riguarda la definizione del secondo ordine del tipo  $List_\alpha$ , si è fatto riferimento ad un documento di Franck Binar<sup>1</sup>.

Per formare le liste, come prima cosa, necessitiamo della formalizzazione delle regole di introduzione, in particolare, avremo bisogno di due costruttori:

$$\begin{aligned} \emptyset_\alpha &\rightarrow S_1 = X \\ a.l &\rightarrow S_2 = U \rightarrow (X \rightarrow X) \end{aligned}$$

dove, nella definizione del tipo  $S_2$ , la  $U$  sta ad identificare l’elemento  $a$ , testa della lista, mentre la prima  $X$ , indica il resto della lista.

L’interpretazione del tipo lista può quindi essere rappresentata nella seguente maniera:

$$\llbracket List_\alpha \rrbracket = \Pi X. X \rightarrow ((\alpha \rightarrow X \rightarrow X) \rightarrow X)$$

L’interpretazione degli elementi risulta quindi essere:

$$\begin{aligned} \llbracket \emptyset_\alpha \rrbracket &= \Lambda X. \lambda x : X. \lambda y : \alpha \rightarrow (X \rightarrow X). x \\ \llbracket a.l \rrbracket &= \Lambda X. \lambda x : X. \lambda y : \alpha \rightarrow (X \rightarrow X). y(\llbracket \alpha \rrbracket)(\llbracket l \rrbracket^X(x)(y)) \end{aligned}$$

A questo punto, tuttavia, è possibile generalizzare il tipo “lista”, rendendolo indipendente da  $\alpha$ . Per fare ciò, dobbiamo introdurre una nuova variabile per il tipo della lista:

$$\llbracket List \rrbracket = \Pi Y. (\Pi X. X \rightarrow ((Y \rightarrow (X \rightarrow X)) \rightarrow X))$$

## Esercizio 42

*Dimostrare il teorema di validità per  $\Pi$ .*

All’interno del sistema di tipi di  $\Pi$ , un *tipo* è stato definito in questa maniera:

$$Type : TypeVar \mid Type \rightarrow Type \mid \Pi X. Type$$

Definiamo ora, le interpretazioni che possiamo dare, dopo aver definito la funzione  $\sigma$ :

$$\sigma : TypeVar \rightarrow \mathbb{SAT}$$

Con l’aiuto di questa nuova funzione, mostriamo le interpretazioni:

$$\begin{aligned} \llbracket TypeVar_x \rrbracket_\sigma &= \sigma(TypeVar_x) \in \mathbb{SAT} \\ \llbracket Type_1 \rightarrow Type_2 \rrbracket_\sigma &= \{t \in \Lambda \mid (\forall u \in \llbracket Type_1 \rrbracket) t(u) \in \llbracket Type_2 \rrbracket\} \\ \llbracket \Pi X. Type \rrbracket_\sigma &= \bigcap_{S \in \mathbb{SAT}} \llbracket Type \rrbracket_{(X/S)} \end{aligned}$$

<sup>1</sup>Documento reperibile all’URL: <http://www.site.uottawa.ca/~fbinard/Intuitionism/TypeTheory/SystemF/>.

Una interpretazione generica, per il tipo può quindi essere formulata nella seguente maniera:

$$\llbracket Type \rrbracket_\sigma = \{ \sigma(x) \mid \forall x \in TypeVar \} \cup \{ \llbracket \alpha \rightarrow \beta \rrbracket_\sigma \} \cup \{ \llbracket \Pi X. \alpha \rrbracket_\sigma \mid \forall X \in TypeVar \}$$

A questo punto, avendo a disposizione una interpretazione, è possibile andare a dimostrare il Teorema di Validità. Tale procedimento verrà effettuato per tutte le regole di  $\Pi$ .

### Prima regola

$$\Gamma, X : TypeVar \vdash X : Type$$

Si vuole verificare che:

$$\llbracket X \rrbracket_\sigma \in \llbracket TypeVar \rrbracket_\sigma \stackrel{?}{\vdash} \llbracket X \rrbracket_\sigma [\bar{x} := \bar{\gamma}] \in \llbracket Type \rrbracket_\sigma$$

che è equivalente a:

$$\sigma(X) \in \llbracket TypeVar \rrbracket_\sigma \stackrel{?}{\vdash} \sigma(X) [\bar{X} := \bar{\gamma}] \in \llbracket Type \rrbracket_\sigma$$

E questa vale, in quanto, per come è stato costruito  $Type$ , esso contiene elementi di  $TypeVar$ .

### Seconda regola

$$\frac{\Gamma \vdash \alpha : Type \quad \Gamma \vdash \beta : Type}{\Gamma \vdash \alpha \rightarrow \beta : Type}$$

Quindi, sapendo che:

$$\llbracket \alpha \rrbracket_\sigma [\bar{X} := \bar{\gamma}] \in \llbracket Type \rrbracket_\sigma \quad \llbracket \beta \rrbracket_\sigma [\bar{X} := \bar{\gamma}] \in \llbracket Type \rrbracket_\sigma$$

ci si chiede se:

$$\llbracket \alpha \rightarrow \beta \rrbracket_\sigma [\bar{X} := \bar{\gamma}] \stackrel{?}{\in} \llbracket Type \rrbracket_\sigma$$

che è equivalente a chiedersi

$$\llbracket \alpha \rrbracket_\sigma [\bar{X} := \bar{\gamma}] \rightarrow \llbracket \beta \rrbracket_\sigma [\bar{X} := \bar{\gamma}] \stackrel{?}{\in} \llbracket Type \rrbracket_\sigma$$

ma, dato che  $\llbracket \alpha \rrbracket_\sigma \in \llbracket Type \rrbracket_\sigma$  e ciò vale anche per  $\beta$ , per come è stato definito  $\llbracket Type \rrbracket_\sigma$  (in particolare, sfruttando la seconda “regola”  $Type : Type \rightarrow Type$ ), possiamo concludere affermativamente anche la dimostrazione per la seconda regola.

### Terza regola

$$\frac{\Gamma, X : TypeVar \vdash \alpha : Type}{\Gamma \vdash \Pi X. \alpha : Type}$$

Quindi, sapendo che:

$$\llbracket X \rrbracket_\sigma[\bar{X} := \bar{\gamma}] \in \llbracket TypeVar \rrbracket_\sigma \quad \llbracket \alpha \rrbracket_\sigma[\bar{X} := \bar{\gamma}] \in \llbracket Type \rrbracket_\sigma$$

ci si chiede se:

$$\llbracket \Pi X. \alpha \rrbracket_\sigma[\bar{X} := \bar{\gamma}] \stackrel{?}{\in} \llbracket Type \rrbracket_\sigma$$

che è equivalente a chiedersi

$$\bigcap_{S \in \text{SAT}} \llbracket \alpha \rrbracket_{\sigma(X/S)}[\bar{X} := \bar{\gamma}] \stackrel{?}{\in} \llbracket Type \rrbracket_\sigma$$

che è a sua volta equivalente a

$$\bigcap_{S \in \text{SAT}} \sigma(\alpha)[X := S][\bar{X} := \bar{\gamma}] \stackrel{?}{\in} \llbracket Type \rrbracket_\sigma$$

Anche in questo caso, possiamo concludere positivamente, per come si è costruito  $\llbracket Type \rrbracket_\sigma$ .

#### Quarta regola

$$\frac{\Gamma \vdash \alpha : Type}{\Gamma, x : \alpha \vdash x : \alpha}$$

Ci si chiede se:

$$\llbracket x \rrbracket_\sigma[\bar{x} := \bar{g}] \in \llbracket \alpha \rrbracket_\sigma \stackrel{?}{\vdash} \llbracket x \rrbracket_\sigma[\bar{x} := \bar{g}] \in \llbracket \alpha \rrbracket_\sigma$$

che è banalmente verificata.

#### Quinta regola

$$\frac{\Gamma \vdash x : \alpha \rightarrow \beta \quad \Gamma \vdash y : \alpha}{\Gamma \vdash x(y) : \beta}$$

Quindi, sapendo che:

$$\llbracket x \rrbracket_\sigma[\bar{x} := \bar{g}] \in \llbracket \alpha \rightarrow \beta \rrbracket_\sigma \quad \llbracket y \rrbracket_\sigma[\bar{x} := \bar{g}] \in \llbracket \alpha \rrbracket_\sigma$$

ci si chiede se:

$$\llbracket x(y) \rrbracket_\sigma[\bar{x} := \bar{g}] \stackrel{?}{\in} \llbracket \beta \rrbracket_\sigma$$

che è già stata essere verificata.

#### Sesta regola

$$\frac{\Gamma, x : \alpha \vdash y : \beta}{\Gamma \vdash \lambda x. y : \alpha \rightarrow \beta}$$

Quindi, sapendo che:

$$\llbracket x \rrbracket_\sigma[\bar{x} := \bar{g}] \in \llbracket \alpha \rrbracket_\sigma \quad \llbracket y \rrbracket_\sigma[\bar{x} := \bar{g}] \in \llbracket \beta \rrbracket_\sigma$$

ci si chiede se:

$$\llbracket \lambda x. y \rrbracket_\sigma[\bar{x} := \bar{g}] \stackrel{?}{\in} \llbracket \alpha \rightarrow \beta \rrbracket_\sigma$$

che è già stata essere verificata.

### Settima regola

$$\frac{\Gamma, X : TypeVar \vdash a : \alpha}{\Gamma \vdash \Lambda X.a : \Pi X.\alpha}$$

Ricordando che l'ultimo elemento del contesto  $\Gamma$  deve essere proprio  $X$ .

Quindi, sapendo che:

$$\llbracket X \rrbracket_\sigma [\bar{X} := \bar{\gamma}] \in \llbracket TypeVar \rrbracket_\sigma \vdash \llbracket a \rrbracket_\sigma [\bar{x} := \bar{g}] \in \llbracket \alpha \rrbracket_\sigma$$

ci si chiede se:

$$\llbracket \Lambda X.a \rrbracket_\sigma [\bar{X} := \bar{\gamma}] \stackrel{?}{\in} \llbracket \Pi X.\alpha \rrbracket_\sigma$$

che è equivalente a chiedersi

$$\llbracket a \rrbracket_\sigma [\bar{X} := \bar{\gamma}] \stackrel{?}{\in} \bigcap_{S \in \text{SAT}} \llbracket a \rrbracket_\sigma (X/S)$$

ma la sostituzione  $[\bar{X} := \bar{\gamma}]$  fatta in  $\llbracket a \rrbracket_\sigma$  non ha effetti, se non nel tipo di  $a$ . Per questo motivo, quando si fa l'intersezione sui SAT, l' $\llbracket a \rrbracket_\sigma$  è sempre presente, e ciò verifica l'appartenenza.

### Ottava regola

$$\frac{\Gamma \vdash \Lambda X.a : \Pi X.\alpha \quad \Gamma \vdash \beta : Type}{\Gamma \vdash a(\beta) : \alpha[X := \beta]}$$

Dove, si ricorda che  $a(\beta)$  è da considerarsi come l'“istanziamento” e non l'applicazione (istanzio  $a$  al tipo  $\beta$ ).

Quindi, sapendo che:

$$\llbracket \Lambda X.a \rrbracket_\sigma [\bar{X} := \bar{\gamma}] \in \llbracket \Pi X.\alpha \rrbracket_\sigma \quad \llbracket \beta \rrbracket_\sigma [\bar{X} := \bar{\gamma}] \in \llbracket Type \rrbracket_\sigma$$

ci si chiede se:

$$\llbracket a(\beta) \rrbracket_\sigma [\bar{X} := \bar{\gamma}] \stackrel{?}{\in} \llbracket \alpha[X := \beta] \rrbracket_\sigma$$

che è equivalente a chiedersi

$$\llbracket a(\beta) \rrbracket_\sigma [\bar{X} := \bar{\gamma}] \stackrel{?}{\in} \llbracket \alpha \rrbracket_\sigma (X/\llbracket \beta \rrbracket_\sigma)$$

in cui

$$a \in \bigcap_{S \in \text{SAT}} \llbracket \alpha \rrbracket_\sigma (X/S)$$

che dimostra l'appartenenza richiesta, in quanto quando si è fatta l'intersezione sui saturi si sarà necessariamente tenuto conto anche di  $\llbracket \beta \rrbracket_\sigma$ .