



Ritmo di inferenza dei tipi (fase 1)

Luca Padovani

Linguaggi e Paradigmi di Programmazione

algoritmo di inferenza del tipo di una λ -espressione

Sommario delle fasi dell'algoritmo

- 1 costruzione dell'**albero sintattico** della λ -espressione
- 2 **annotazione** dell'albero + **generazione** dei vincoli
 - **variabile di tipo** = tipo sconosciuto
 - **espressione di tipo** = tipo (parzialmente) sconosciuto
 - **vincolo** = relazione di uguaglianza tra tipi espressa nelle regole di tipo
- 3 **risoluzione** dei vincoli
 - determinare se il sistema ammette almeno una **soluzione**
 - calcolare la soluzione "più generale", da cui derivare tutte le altre

fase 1: costruzione dell'albero sintattico

- ▶ Si rappresenta la λ -espressione come un albero
- ▶ I nodi interni e le foglie sono opportunamente etichettati
- ▶ Indichiamo con $T[M]$ l'albero corrispondente alla λ -espressione M

costruzione dell'albero sintattico

$$T[x] \stackrel{\text{def}}{=} x$$

x

foglia

$$T[c] \stackrel{\text{def}}{=} c$$

c

foglia

$$T[\lambda x.M] \stackrel{\text{def}}{=}$$

λx
|
 $T[M]$

$$T[M N] \stackrel{\text{def}}{=}$$

@
/ \
 $T[M]$ $T[N]$

$$T[\text{if } M N_1 N_2] \stackrel{\text{def}}{=}$$

if
/ | \
 $T[M]$ $T[N_1]$ $T[N_2]$

esercizi

Costruire l'albero sintattico delle seguenti espressioni:

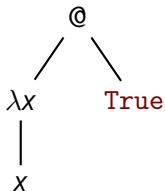
1 $(\lambda x.x) \text{ True}$

2 $\lambda x.\lambda y.\lambda z.z\ x\ y$

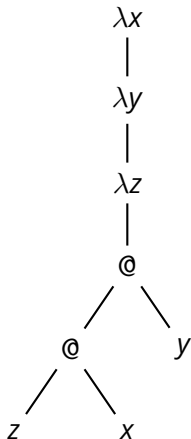
3 $\text{if True } (\lambda x.x) (\lambda x.\lambda y.y)$

soluzioni

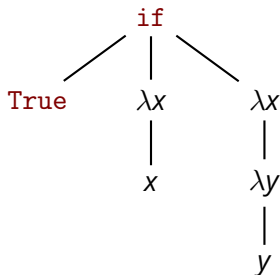
$(\lambda x.x) \text{ True}$



$\lambda x.\lambda y.\lambda z.z \ x \ y$



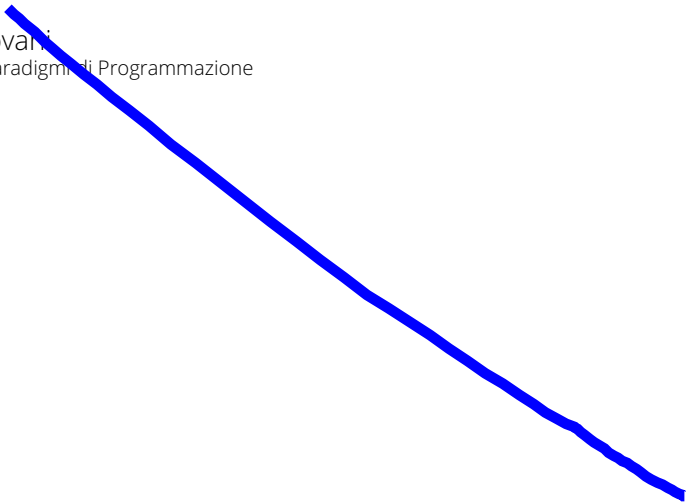
$\text{if True } (\lambda x.x) (\lambda x.\lambda y.y)$



Algoritmo di inferenza dei tipi (fase 2)

Luca Padovani

Linguaggi e Paradigmi di Programmazione



È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza. Ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

algoritmo di inferenza del tipo di una λ -espressione

Sommario delle fasi dell'algoritmo

- 1 costruzione dell'**albero sintattico** della λ -espressione
- 2 **annotazione** dell'albero + **generazione** dei vincoli
 - **variabile di tipo** = tipo sconosciuto
 - **espressione di tipo** = tipo (parzialmente) sconosciuto
 - **vincolo** = relazione di uguaglianza tra tipi espressa nelle regole di tipo
- 3 **risoluzione** dei vincoli
 - determinare se il sistema ammette almeno una **soluzione**
 - calcolare la soluzione "più generale", da cui derivare tutte le altre

fase 2: annotazione dell'albero e generazione dei vincoli

- ▶ ogni nodo dell'albero viene annotato con un'espressione di tipo
- ▶ si procede in modo bottom-up, dalle foglie verso la radice

Variabili di tipo

- ▶ Sia $TVar = \{\alpha, \beta, \gamma, \dots\}$ un insieme *infinito* di **variabili di tipo**
- ▶ α rappresenta un tipo sconosciuto, ancora da determinare

Sintassi delle espressioni di tipo

Espressioni di tipo

τ, σ	$::=$	α	variabile di tipo
		Bool	booleani
		$\tau \rightarrow \sigma$	funzioni

Un **vincolo** è una coppia di espressioni di tipo scritta come

$$\tau = \sigma$$

annotazione bottom-up e generazione dei vincoli

Albero	Note
$x : \alpha$	α è nuova, avendo cura di usare la stessa α per tutte le occorrenze della stessa x
$c : \text{Bool}$	Altre costanti richiederanno tipi diversi
$\lambda x : \alpha \rightarrow \tau$ $T : \tau$	α è la variabile di tipo usata per annotare x in T se x compare in T o è nuova altrimenti
$@ : \alpha$ / \ $T_1 : \tau$ $T_2 : \sigma$	α è nuova Generare il vincolo $\tau = \sigma \rightarrow \alpha$
$@ : \tau_2$ / \ $T_1 : \tau_1 \rightarrow \tau_2$ $T_2 : \sigma$	Ottimizzazione facoltativa del caso precedente che evita l'introduzione di una nuova α Generare il vincolo $\tau_1 = \sigma$
$\text{if} : \tau_2$ / \ $T_1 : \tau_1$ $T_2 : \tau_2$ $T_3 : \tau_3$	Generare il vincolo $\tau_1 = \text{Bool}$ Generare il vincolo $\tau_2 = \tau_3$

esercizi

Generare i vincoli di tipo per le seguenti espressioni:

1 $(\lambda x.x) \text{ True}$

2 $\lambda f.\lambda x.f (f x)$

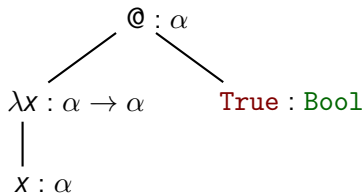
3 $(\lambda x.x) (\lambda x.x) \text{ True}$

4 $\lambda x.\lambda y.\lambda z.z x y$

5 $\text{if True } (\lambda x.x) (\lambda x.\lambda y.y)$

 SOL

Albero annotato



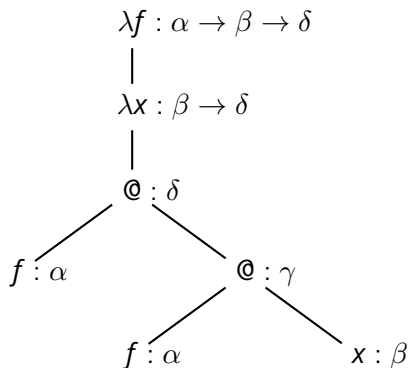
Vincoli generati

► $\alpha = \text{Bool}$

Nota

- usiamo l'ottimizzazione per evitare di introdurre una variabile di tipo

Albero annotato



Vincoli generati

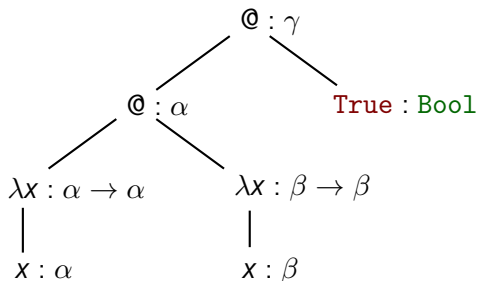
► $\alpha = \beta \rightarrow \gamma$

► $\alpha = \gamma \rightarrow \delta$

Nota

- le due occorrenze di f hanno la stessa annotazione α

Albero annotato



Vincoli generati

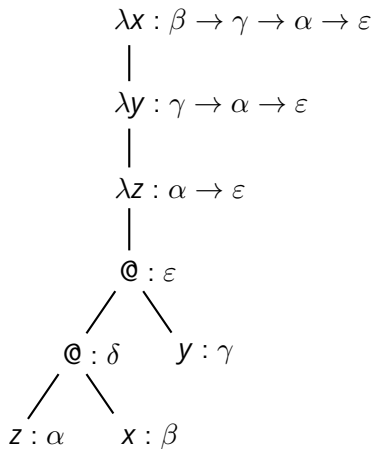
- ▶ $\alpha = \beta \rightarrow \beta$
- ▶ $\alpha = \text{Bool} \rightarrow \gamma$

Nota

- ▶ le due occorrenze di x hanno annotazioni diverse (α e β) perché sono legate a λ diversi

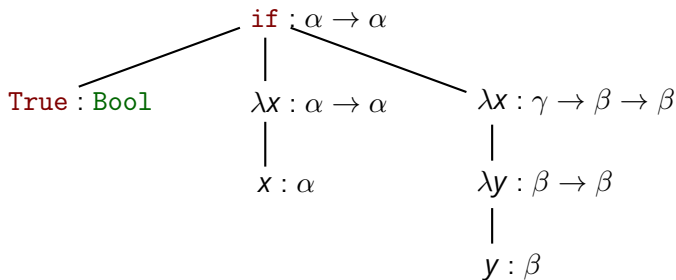
Albero annotato

Vincoli generati

► $\alpha = \beta \rightarrow \delta$ ► $\delta = \gamma \rightarrow \varepsilon$

soluzione

if True ($\lambda x.x$) ($\lambda x.\lambda y.y$)



Vincoli generati

- ▶ $\text{Bool} = \text{Bool}$
- ▶ $\alpha \rightarrow \alpha = \gamma \rightarrow \beta \rightarrow \beta$

Note

- ▶ bisogna introdurre una nuova variabile γ per il λx a destra in quanto quella x non compare altrove



Algoritmo di inferenza dei tipi (fase 3)

Luca Padovani

Linguaggi e Paradigmi di Programmazione

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza. Ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

algoritmo di inferenza del tipo di una λ -espressione

Sommario delle fasi dell'algoritmo

- 1 costruzione dell'**albero sintattico** della λ -espressione
- 2 **annotazione** dell'albero + **generazione** dei vincoli
 - **variabile di tipo** = tipo sconosciuto
 - **espressione di tipo** = tipo (parzialmente) sconosciuto
 - **vincolo** = relazione di uguaglianza tra tipi espressa nelle regole di tipo
- 3 **risoluzione** dei vincoli
 - determinare se il sistema ammette almeno una **soluzione**
 - calcolare la soluzione "più generale", da cui derivare tutte le altre

sistemi di equazioni e soluzioni

- ▶ la fase di generazione dei vincoli produce un sistema

$$\{\tau_i = \sigma_i\}_{1 \leq i \leq n}$$

- ▶ ora bisogna determinare se questo sistema ammette una soluzione
- ▶ se sì, vogliamo determinare qual è la soluzione più generale

Definizione (sostituzione)



Una **sostituzione** θ è una funzione da variabili di tipo a espressioni di tipo. Scriviamo $\theta(\tau)$ per l'espressione ottenuta da τ sostituendo ogni α con $\theta(\alpha)$.

Definizione (soluzione)

Dato un sistema di vincoli $\{\tau_i = \sigma_i\}_{1 \leq i \leq n}$ e una sostituzione θ diciamo che θ è **soluzione** (o **unificatore**) del sistema se $\theta(\tau_i) = \theta(\sigma_i)$ per ogni $1 \leq i \leq n$. Diciamo inoltre che θ è l'**unificatore più generale** del sistema se ogni soluzione del sistema è ottenibile componendo θ con un'altra sostituzione.

algoritmo di risoluzione

Applicare le trasformazioni seguenti fino a quando è possibile:

Se c'è un vincolo	e	allora
$\tau = \tau$	—	eliminare il vincolo
$\tau = \alpha$	τ non è una variabile	rimpiazzare il vincolo con $\alpha = \tau$
$\tau \rightarrow \tau' = \sigma \rightarrow \sigma'$	—	rimpiazzare il vincolo con $\tau = \sigma$ e $\tau' = \sigma'$
$\tau \rightarrow \sigma = \text{Bool}$ o $\text{Bool} = \tau \rightarrow \sigma$	—	 l'algoritmo fallisce (type error)
$\alpha = \tau$	$\alpha \neq \tau$ ma α compare in τ	 l'algoritmo fallisce (occur check)
$\alpha = \tau$	α non compare in τ α compare altrove	sostituire α con τ in tutti gli altri vincoli ($\alpha = \tau$ rimane)

- ▶ l'ordine in cui si applicano le trasformazioni non è importante
- ▶ nessuna trasformazione applicabile \Rightarrow algoritmo ha **successo**

proprietà dell'algoritmo

Applicando l'algoritmo di risoluzione a un sistema iniziale di vincoli

$$\{\tau_i = \sigma_i\}_{1 \leq i \leq n}$$

si ha che:

- 1 Prima o poi l'algoritmo fallisce o ha successo.
- 2 Se l'algoritmo fallisce, allora il sistema iniziale è insoddisfacibile.
- 3 Se l'algoritmo ha successo, allora:
 - il sistema finale ha la forma $\{\alpha_i = \rho_i\}_{1 \leq i \leq m}$ in cui ciascuna α_i compare una sola volta nel sistema
 - la sostituzione $\theta \stackrel{\text{def}}{=} \{\alpha_i \mapsto \rho_i\}_{1 \leq i \leq m}$ è l'unificatore più generale del sistema iniziale, in particolare $\theta(\tau_i) = \theta(\sigma_i)$ per ogni $1 \leq i \leq n$

esempi

$$\begin{aligned} \blacktriangleright \quad & \alpha = \beta \rightarrow \gamma \\ & \alpha = \gamma \rightarrow \delta \end{aligned}$$

$$\begin{aligned} & \alpha = \beta \rightarrow \gamma \\ \blacktriangleright \quad & \beta \rightarrow \gamma = \gamma \rightarrow \delta \end{aligned}$$

$$\begin{aligned} \blacktriangleright \quad & \alpha = \beta \rightarrow \gamma \\ & \beta = \gamma \\ & \gamma = \delta \end{aligned}$$

$$\begin{aligned} & \alpha = \gamma \rightarrow \gamma \\ & \beta = \gamma \\ \blacktriangleright \quad & \gamma = \delta \end{aligned}$$

$$\begin{aligned} & \alpha = \delta \rightarrow \delta \\ & \beta = \delta \\ & \gamma = \delta \end{aligned}$$

$$\begin{aligned} \blacktriangleright \quad & \alpha = \beta \rightarrow \beta \\ & \alpha = \text{Bool} \rightarrow \gamma \end{aligned}$$

$$\begin{aligned} & \alpha = \beta \rightarrow \beta \\ \blacktriangleright \quad & \beta \rightarrow \beta = \text{Bool} \rightarrow \gamma \end{aligned}$$

$$\begin{aligned} \blacktriangleright \quad & \alpha = \beta \rightarrow \beta \\ & \beta = \text{Bool} \\ & \beta = \gamma \end{aligned}$$

$$\begin{aligned} & \alpha = \text{Bool} \rightarrow \text{Bool} \\ & \beta = \text{Bool} \\ \blacktriangleright \quad & \text{Bool} = \gamma \end{aligned}$$

$$\begin{aligned} & \alpha = \text{Bool} \rightarrow \text{Bool} \\ & \beta = \text{Bool} \\ & \gamma = \text{Bool} \end{aligned}$$

Il simbolo \blacktriangleright indica il vincolo che innesca la trasformazione

esercizi

Applicare l'algoritmo di inferenza dei tipi ai seguenti termini e determinarne, se possibile, il tipo più generale:

- 1 $(\lambda x. (\lambda y. x y) y) z$
- 2 $(\lambda x. \lambda y. x y y) (\lambda a. a) b$
- 3 $\lambda x. x z (\lambda y. x y)$
- 4 $\lambda x. \lambda y. \lambda z. (x z) (y z)$
- 5 $(\lambda x. \lambda y. y x) (\lambda x. \lambda y. x y)$