



UNIVERSITÀ
DEGLI STUDI
FIRENZE

SCUOLA DI INGEGNERIA
Corso di Laurea Magistrale in Ingegneria
Informatica

Clustering di immagini basato su Normalized Cuts per l'identificazione di camera

Elaborazione e Protezione delle Immagini

Lorenzo Cioni e Saverio Meucci

ANNO ACCADEMICO 2014/2015

Variabili di tipo

Scegliamo un sistema di tipi contenente un insieme di tipi primitivi ed i tipi freccia

$$T = T_p \cup T_f$$

con

$$T_p = \{Bool, Nat, \dots\} \quad T_f = \{X_1 \rightarrow X_2 : X_1, X_2 \in T_p \cup T_f\}$$

A partire da questo sistema di tipi così definito definiamo le **variabili di tipo** come *placeholders*, ovvero variabili di cui non conosciamo il tipo specifico, ma che utilizziamo all'interno dei termini.

Polimorfismo parametrico

Le variabili di tipo introducono il concetto di **polimorfismo parametrico**, dove le stesse variabili possono essere usate in contesti diversi con differenti tipi concreti.

Durante la fase di *type checking* non sarà dunque necessario effettuare la sostituzione della variabile con un tipo concreto, ma verificheremo che il termine sia *ben tipabile* a prescindere dalla sostituzione stessa.

Type reconstruction

Scopo della *type reconstruction* è inferire istanze valide per le variabili di tipo di un termine, lasciando al programmatore la libertà di specificare o meno i tipi.

Siamo interessati ad inferire il **tipo principale** di un termine, ovvero:

1. Che contiene variabili (*schema di tipo*)
2. Il più generale che rappresenta tutti i possibili tipi (*polimorfismo parametrico*)

Sostituzioni di tipo

Le variabili di tipo possono essere *sostituite*:

Definizione (Sostituzione di tipo)

*Una sostituzione di tipo è una **funzione** σ che associa variabili di tipo a tipi. L'insieme delle variabili che compaiono a sinistra delle coppie in σ è detto **dom**(σ); l'insieme delle variabili che compaiono a destra delle coppie in σ è detto **range**(σ).*

L'applicazione di una sostituzione di tipo avviene applicando ciascuna clausola contemporaneamente.

Esempio: $\sigma = [X \mapsto \text{Nat}, Y \mapsto X]$ sostituisce la variabile di tipo X con il tipo primitivo Nat e la variabile Y con la variabile X .

Applicazione di sostituzione di tipo

L'**applicazione** di una sostituzione a un tipo è definita come:

$$\sigma(X) = \begin{cases} T & \text{se } (X \mapsto T) \in \sigma \\ X & \text{se } X \notin \text{dom}(\sigma) \end{cases}$$

$$\sigma(\text{Nat}) = \text{Nat}$$

$$\sigma(\text{Bool}) = \text{Bool}$$

$$\sigma(T_1 \rightarrow T_2) = \sigma(T_1) \rightarrow \sigma(T_2)$$

La sostituzione di tipo è estesa al **contesto** definendo:

$$\sigma(x_1 : T_1, \dots, x_n : T_n) = (x_1 : \sigma(T_1), \dots, x_n : \sigma(T_n))$$

Preservazione di tipo

Una proprietà importante delle sostituzioni di tipo è che mantengono la **tipabilità**: se al termine assegno un tipo con variabili, allora tutte le istanze delle sue sostituzioni di tipo sono tipi corretti per il termine.

Teorema (Preservazione di tipo con sostituzioni)

Se σ è una sostituzione di tipo e $\Gamma \vdash t : T$, allora $\sigma\Gamma \vdash \sigma t : \sigma T$

Algoritmo di type reconstruction

L'algoritmo di *type reconstruction*, dunque, a partire da un termine t in un contesto Γ , permette di generare la *sostituzione più generale* che tipa correttamente il termine.

Il procedimento è suddiviso in due fasi consecutive:

1. **Constraint-based typing**: in questa prima fase, prima vengono assegnate delle variabili di tipo ai termini, poi vengono generati dei vincoli logici dipendenti dal contesto su tali variabili;
2. **Unification**: a partire dai vincoli generati al passo precedente, viene cercata la sostituzione più generale in grado di unificarli tutti.

Tipaggio basato su vincoli

Definizione (Soluzione)

*Sia Γ un contesto e t un termine. Una **soluzione** per (Γ, t) è una coppia (σ, T) tale che $\sigma T \vdash \sigma t : T$*

Definizione (Insieme di vincoli)

*Un **insieme di vincoli** C è un insieme di equazioni $\{S_i = T_i^{i \in 1, \dots, n}\}$. Una sostituzione σ **unifica** (o **soddisfa**) un'equazione $S = T$ se $\sigma S = \sigma T$. La sostituzione unifica C se unifica tutte le equazioni in C*

Regole - I

$$\frac{\Gamma, x : T_1 \vdash t_2 : T_2 \quad |_{\mathcal{X}} \quad C}{\Gamma \vdash \lambda x : T_1. t_2 : T_1 \rightarrow T_2 \quad |_{\mathcal{X}} \quad C} \quad (\text{CT-ABS})$$

$$\frac{\begin{array}{c} \Gamma \vdash t_1 : T_1 \quad |_{\mathcal{X}_1} \quad C_1 \quad \Gamma \vdash t_2 : T_2 \quad |_{\mathcal{X}_2} \quad C_2 \\ \mathcal{X}_1 \cap \mathcal{X}_2 = \mathcal{X}_1 \cap FV(T_2) = \mathcal{X}_2 \cap FV(T_1) = \emptyset \\ X \notin \mathcal{X}_1, \mathcal{X}_2, T_1, T_2, C_1, C_2, \Gamma, t_1, t_2 \\ C' = C_1 \cup C_2 \cup \{T_1 = T_2 \rightarrow X\} \end{array}}{\Gamma \vdash t_1 t_2 : X \quad |_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup X} \quad C'} \quad (\text{CT-APP})$$

$$\frac{\begin{array}{c} \Gamma \vdash t_1 : T \quad |_{\mathcal{X}} \quad C \\ C' = C \cup \{T = \text{Nat}\} \end{array}}{\Gamma \vdash \text{succ } t_1 : \text{Nat} \quad |_{\mathcal{X}} \quad C'} \quad (\text{CT-SUCC})$$

Regole - I

$$\frac{\Gamma, x : T_1 \vdash t_2 : T_2 \quad |_{\mathcal{X}} \quad C}{\Gamma \vdash \lambda x : T_1. t_2 : T_1 \rightarrow T_2 \quad |_{\mathcal{X}} \quad C} \quad (\text{CT-ABS})$$

$$\frac{\begin{array}{c} \Gamma \vdash t_1 : T_1 \quad |_{\mathcal{X}_1} \quad C_1 \quad \Gamma \vdash t_2 : T_2 \quad |_{\mathcal{X}_2} \quad C_2 \\ \mathcal{X}_1 \cap \mathcal{X}_2 = \mathcal{X}_1 \cap FV(T_2) = \mathcal{X}_2 \cap FV(T_1) = \emptyset \\ X \notin \mathcal{X}_1, \mathcal{X}_2, T_1, T_2, C_1, C_2, \Gamma, t_1, t_2 \\ C' = C_1 \cup C_2 \cup \{T_1 = T_2 \rightarrow X\} \end{array}}{\Gamma \vdash t_1 t_2 : X \quad |_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup X} \quad C'} \quad (\text{CT-APP})$$

$$\frac{\begin{array}{c} \Gamma \vdash t_1 : T \quad |_{\mathcal{X}} \quad C \\ C' = C \cup \{T = \text{Nat}\} \end{array}}{\Gamma \vdash \text{succ } t_1 : \text{Nat} \quad |_{\mathcal{X}} \quad C'} \quad (\text{CT-SUCC})$$

Regole - I

$$\frac{\Gamma, x : T_1 \vdash t_2 : T_2 \quad |_{\mathcal{X}} \quad C}{\Gamma \vdash \lambda x : T_1. t_2 : T_1 \rightarrow T_2 \quad |_{\mathcal{X}} \quad C} \quad (\text{CT-ABS})$$

$$\frac{\begin{array}{c} \Gamma \vdash t_1 : T_1 \quad |_{\mathcal{X}_1} \quad C_1 \quad \Gamma \vdash t_2 : T_2 \quad |_{\mathcal{X}_2} \quad C_2 \\ \mathcal{X}_1 \cap \mathcal{X}_2 = \mathcal{X}_1 \cap FV(T_2) = \mathcal{X}_2 \cap FV(T_1) = \emptyset \\ X \notin \mathcal{X}_1, \mathcal{X}_2, T_1, T_2, C_1, C_2, \Gamma, t_1, t_2 \\ C' = C_1 \cup C_2 \cup \{T_1 = T_2 \rightarrow X\} \end{array}}{\Gamma \vdash t_1 t_2 : X \quad |_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup X} \quad C'} \quad (\text{CT-APP})$$

$$\frac{\begin{array}{c} \Gamma \vdash t_1 : T \quad |_{\mathcal{X}} \quad C \\ C' = C \cup \{T = \text{Nat}\} \end{array}}{\Gamma \vdash \text{succ } t_1 : \text{Nat} \quad |_{\mathcal{X}} \quad C'} \quad (\text{CT-SUCC})$$

Regole - II

$$\frac{\Gamma \vdash t_1 : T \quad |_{\mathcal{X}} \quad C}{\Gamma \vdash \text{iszero } t_1 : \text{Bool} \quad |_{\mathcal{X}} \quad C'} \quad (\text{CT-ISZERO})$$

$$\frac{\begin{array}{c} \Gamma \vdash t_1 : T_1 \quad |_{\mathcal{X}_1} \quad C_1 \\ \Gamma \vdash t_2 : T_2 \quad |_{\mathcal{X}_2} \quad C_2 \quad \Gamma \vdash t_3 : T_3 \quad |_{\mathcal{X}_3} \quad C_3 \\ \mathcal{X}_1 \cap \mathcal{X}_2 \cap \mathcal{X}_3 = \emptyset \\ C' = C_1 \cup C_2 \cup C_3 \cup \{T_1 = \text{Bool}, T_2 = T_3\} \end{array}}{\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T_2 \quad |_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3} \quad C'} \quad (\text{CT-IF})$$

Regole - II

$$\frac{\Gamma \vdash t_1 : T \quad |_{\mathcal{X}} \quad C}{\Gamma \vdash \text{iszero } t_1 : \text{Bool} \quad |_{\mathcal{X}} \quad C'} \quad (\text{CT-ISZERO})$$

$$\frac{\begin{array}{c} \Gamma \vdash t_1 : T_1 \quad |_{\mathcal{X}_1} \quad C_1 \\ \Gamma \vdash t_2 : T_2 \quad |_{\mathcal{X}_2} \quad C_2 \quad \Gamma \vdash t_3 : T_3 \quad |_{\mathcal{X}_3} \quad C_3 \\ \mathcal{X}_1 \cap \mathcal{X}_2 \cap \mathcal{X}_3 = \emptyset \\ C' = C_1 \cup C_2 \cup C_3 \cup \{T_1 = \text{Bool}, T_2 = T_3\} \end{array}}{\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T_2 \quad |_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3} \quad C'} \quad (\text{CT-IF})$$

Osservazioni sulle regole

- Quando viene introdotta una nuova variabile di tipo, questa deve essere **fresca**, *fresh*, cioè diversa da qualsiasi altra variabile utilizzata.
- Quando una regola coinvolge due o più sottoderivazioni, i due insiemi di variabili devono essere *disgiunti*.

Le regole così definite permettono **sempre** di costruire una derivazione per un termine (con variabili di tipo): dato un termine t ed un contesto Γ è *sempre* possibile determinare T ed un insieme di vincoli C tali che:

$$\Gamma \vdash t : T \quad | \quad C$$

Il termine sarà però **tipabile** solo se \exists una sostituzione σ che unifica l'insieme di vincoli.

Esempio - I

Esempio di applicazione a $\lambda x. \lambda y. x y$

Esempio - I

Esempio di applicazione a $\lambda x. \lambda y. x y \Rightarrow \lambda x : X. \lambda y : Y. x y$

Esempio - I

Esempio di applicazione a $\lambda x. \lambda y. x y \Rightarrow \lambda x : X. \lambda y : Y. x y$

$$\frac{\frac{\frac{x : X \quad y : Y \quad \{X = Y \rightarrow Z\}}{y : Y \vdash x y : Z \mid \{X = Y \rightarrow Z\}}}{x : X \vdash \lambda y : Y. x y : W \mid \{W = Y \rightarrow Z\} \cup \{X = Y \rightarrow Z\}}}{\lambda x : X. \lambda y : Y. x y : X \rightarrow W \mid \{W = Y \rightarrow Z, X = Y \rightarrow Z\}}$$

Esempio - I

Esempio di applicazione a $\lambda x. \lambda y. x y \Rightarrow \lambda x : X. \lambda y : Y. x y$

$$\frac{\frac{\frac{x : X \quad y : Y \quad \{X = Y \rightarrow Z\}}{y : Y \vdash x y : Z \mid \{X = Y \rightarrow Z\}}}{x : X \vdash \lambda y : Y. x y : W \mid \{W = Y \rightarrow Z\} \cup \{X = Y \rightarrow Z\}}}{\lambda x : X. \lambda y : Y. x y : X \rightarrow W \mid \{W = Y \rightarrow Z, X = Y \rightarrow Z\}}$$

Soluzione: $\sigma = [W \mapsto (Y \rightarrow Z), X \mapsto (Y \rightarrow Z)]$

Tipo principale

Definizione

Sia $\Gamma \vdash t : S \mid C$. Una soluzione per (Γ, t, S, C) è una coppia (σ, T) tale che σ unifica C e $\sigma S = T$

Tra tutte le soluzioni al problema (Γ, t, S, C) vogliamo quella più generale possibile, ovvero vogliamo ricavare il *tipo principale*:

Definizione (Tipo principale)

*La **soluzione principale** di (Γ, t, S, C) è la soluzione (σ, T) tale che per ogni (σ', T') soluzione di (Γ, t, S, C) abbiamo che $\sigma \sqsubseteq \sigma'$.
Definiamo dunque T il **tipo principale** di t in Γ*

Correttezza e completezza

Correttezza: ogni soluzione di (Γ, t, S, C) è anche soluzione di (Γ, T) .

Teorema (Correttezza del tipaggio basato su vincoli)

Sia $\Gamma \vdash t : S \mid C$. Se (σ, T) è una soluzione per (Γ, t, S, C) , cioè σ soddisfa C e $\sigma S = T$, allora lo è anche per (Γ, t)

Completezza: ogni soluzione (Γ, T) può essere estesa ad una soluzione per (Γ, t, S, C) .

Teorema (Completezza del tipaggio basato su vincoli)

Supponiamo $\Gamma \vdash t : S \mid_{\mathcal{X}} C$. Se (σ, T) è una soluzione per (Γ, t) e $\text{dom}(\sigma) \cap \mathcal{X} = \emptyset$, allora esiste una soluzione (σ', T) per (Γ, t, S, C) tale che $\sigma' / \mathcal{X} = \sigma$, con σ' / \mathcal{X} indicata la sostituzione non definita per le variabili in \mathcal{X}

Unificazione

Per il calcolo della soluzione, dato l'insieme di vincoli sulle variabili di tipo, viene utilizzato l'**algoritmo di unificazione** (*Robinson, 1971*). L'algoritmo verifica se l'insieme delle soluzioni è *non vuoto* e restituisce la soluzione più *generale*.

L'algoritmo, per come è definito, previene la generazione di soluzioni che includono *sostituzioni cicliche* (ad esempio $X \mapsto X \rightarrow X$), che non hanno senso in espressioni finite.

Unificazione

Algoritmo Unificazione di Robinson

```
function UNIFY( $C$ )  
  if  $C = \emptyset$  then []  
  else  
    Let  $\{S = T\} \cup C' = C$   
    if  $S = T$  then  
       $unify(C')$   
    else if  $S = X$  and  $X \notin FV(T)$  then  
       $unify([X \mapsto T]C') \circ [X \mapsto T]$   
    else if  $T = X$  and  $X \notin FV(S)$  then  
       $unify([X \mapsto S]C') \circ [X \mapsto S]$   
    else if  $S = S_1 \rightarrow S_2$  and  $T = T_1 \rightarrow T_2$  then  
       $unify(C' \cup \{S_1 = T_1, S_2 = T_2\})$   
    else  
      fail
```

Correttezza e completezza

L'algoritmo **unify** termina sempre, fallisce quando l'insieme di vincoli dato non è unificabile mentre ritorna l'**unificatore principale** nel caso in cui esista almeno un unificatore.

Teorema (Completezza dell'algoritmo di unificazione)

1. *$\text{unify}(C)$ termina sempre, fallendo oppure restituendo una sostituzione che unifica l'insieme di vincoli C ;*
2. *Se $\text{unify}(C) = \sigma$, allora σ unifica l'insieme di vincoli C ;*
3. *Se δ unifica C , allora $\text{unify}(C) = \sigma$ con $\sigma \sqsubseteq \delta$.*

Esempio - II

Esempio di applicazione a $\lambda x. x x$

Esempio - II

Esempio di applicazione a $\lambda x. x x \Rightarrow \lambda x : X. x x$

Esempio - II

Esempio di applicazione a $\lambda x. x x \Rightarrow \lambda x : X. x x$

$$\frac{\frac{x : X \quad x : X \quad \{X = X \rightarrow W\}}{x x : W \mid \{X = X \rightarrow W\}}}{\lambda x : X. x x : X \rightarrow W \mid \{X = X \rightarrow W\}}$$

Nessuna sostituzione finita possibile \Rightarrow **fallimento**

Considerazioni

L'esempio precedente non è tipabile con tipi **finiti** a causa della ricorsione di tipo: l'algoritmo di unificazione *fallisce*.

Questo a causa dei due controlli effettuati, $X \notin FV(T)$ e $X \notin FV(S)$, chiamati *occur check* che impediscono la generazione di espressioni di tipo finite. Questi due controlli vengono rimossi in condizioni particolari quali i **tipi ricorsivi**, ad esempio della forma $X \rightarrow (X \rightarrow X)$.

Altri casi di fallimento:

- Vincoli su tipi primitivi irrisolvibili. *Esempio:* $Bool = Nat$
- Presenza di vincoli di uguaglianza tra tipi primitivi e tipi freccia. *Esempio:* $Nat = X \rightarrow Bool$