Giorno 4
Dati

Sommario

1 Funzioni come dati

1.1 Type Inference

Supponendo di eliminare l'annotazione di tipo dalla sintassi e delle regole, può essere possibile ricostruire il tipo di un'espressione facendo *inferenza di tipo*.

1.1.1 Variabili di tipo

Il primo passo verso l'inferenza di tipo è l'introduzione di *variabili di tipo* non interpretate, che servono da placeholder per tipi specifici su cui non abbiamo (ancora) informazioni.

Le variabili di tipo possono essere **sostituite** o **istanziate** con altri tipi; L'*idea* è di definire un **sistema di vincoli** sulle variabili di tipo, e risolverlo per scoprire se esistono sostituzioni capaci di soddisfare tutti i vincoli.

Esempio:

$$fun(x) = x + 1$$

Il tipo ha la forma $X \to X$, e dato che utilizziamo x nell'operazione di somma tra interi, esiste il vincolo $X = Int$. In questo caso la risoluzione del sistema è banale (è formato di una sola equazione con una singola incognita, e si ha perciò che il tipo dell'espressione è $Int \to Int$)

1.1.2 Notazione

- Estendiamo la notazione usata per la sostituzione nel λ-calcolo alla sostituzione dei tipi:

  $\tau\{X := \tau_1\}$ è il tipo ottenuto sostituendo tutte le occorrenze della var. di tipo X in τ con il tipo τ_1.

- Chiamiamo C il sistema di equazioni:

  $$\rho_i = \{\kappa_i \mid i = 1, \cdots, n\}$$

  Dove ρ_i e κ_i sono tipi contenenti variabili di tipo.

- C è soddisfatto se esiste una sostituzione σ tale che $\forall i : \sigma(\rho_i) = \sigma(\kappa_i)$

- *Esempio*:

  $$C = \{X \to Int = Y, \quad X = Int\}$$

  La sostituzione tale che $\sigma(\tau) = \tau\{X := Int\}$ risolve il sistema (sempre banale per la seconda equazione).

1.1.3 Osservazione

Quando vado a processare una funzione ed ottengo espressione di tipo e vincoli, possono succedere due cose:

- Esistono infinite σ che rendono soddisfacibile un tipo (o anche: per ogni σ il tipo è soddisfacibile) $\Longrightarrow$ la funzione è **polimorfa**

- Solo una qualche σ rende il tipo soddisfacibile.

1.2 Type inference

1.2.1 Costruzione dei vincoli

Costruiamo i vincoli insieme ai tipi:

- Le costanti (in questo esempio solo naturali) sono semplicemente tipate con il loro tipo:

  $$\frac{}{\Gamma \vdash n : Nat \,|\, \varnothing}$$

  $\varnothing$ indica i vincoli aggiunti da questa espressione (nessuno)

- Lo stesso vale per le variabili, non aggiungono nessun vincolo:

  $$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau \,|\, \varnothing}$$

- Astrazione (fun)

  $$\frac{\Gamma, x : X \vdash e : \tau \,|\, C}{\Gamma fun\ x = e : X \to \tau \,|\, C}, \quad X \text{ fresh}$$

  Qua si applica l'esempio di prima: $fun(x) = x + 1$, l'espressione $x + 1$ genera il vincolo $X = Int$, che ci portiamo dietro nell'astrazione.

  In realtà l'unica operazione che posso fare in questa versione del λ-calcolo tipato sono le applicazioni:

- Applicazione:

  $$\frac{\Gamma \vdash e_1 : \tau \,|\, C_1 \quad \Gamma \vdash e_2 : \tau_1 \,|\, C_2}{\Gamma \vdash Apply(e_1, e_2) : X \,|\, C_1 \cup C_2 \cup \{\tau = \tau_1 \to X\}}, \quad X \text{ fresh}$$

- La ricorsione non viene approfondita, perché non aggiunge niente di interessante, ma la riporto:

  $$\frac{\Gamma \vdash e : \tau \,|\, C}{\Gamma \vdash fix\ e : X \,|\, C_1 \cup \{\tau_X \to X\}}, \quad X \text{ fresh}$$

Idea: Sommarizzando, l'idea è di portarsi dietro i vincoli generati tramite l'applicazione funzionale (e la fix) lungo l'albero di sintassi astratta, in modo da avere poi tutti i vincoli alla fine.

[Le regole possono essere tradotte in codice del type-checker, vedi pseudocodice nelle slide]

1.2.2 Risoluzione del sistema di vincoli[1]

```
1  function aux(T0, C)
2     while C is not empty
3         pop a constraint S=T from C
4         if S = T
5             do nothing
6         else if S = X and X not in  FV(T)
7             T0 = T0{X:= T}
8             C = C{X:=S}
9         else if T = X and X not in FV(S)
10            T0 = T0{X:= S}
11            C = C{X:=T}
12        else if S = S1 -> S2 and T = T1->
   T2
13            C = C  U  {S1=T1, S2=T2}
14        else
15            fail
16    return T0
```

- Prendo un vincolo $S = T$ dalla lista dei vincoli
- Se $S = T$, e.g. $S = Int$, $T = Int$, allora rimuovo il vincolo (non mi dice niente)
- Se $S = X$ ed X non è una variabile libera di T, allora posso sostituire la lista dei tipi e quella dei vincoli.
- Caso $T = X$ simmetrico
- Se $S = S1 \to S_2$, $T = T_1 \to T_2$, allora aggiungiamo ai vincoli le singole uguaglianze tra "componenti".
- Se il tipo è decidibile, l'espressione termina, se no fallisce il "patternmatching" e fallisce tutto

(Con U ho indicato l'unione, alla riga 13.) L'operazione svolta dal programma è detta "di unificazione".

[1] Paolomil si è confuso nel confondersi