Il sistema dei tipi in ML

cenni

Inferenza tipi ML

```
- val f = fn x => x+1;
val f = fn : int -> int
- f 2;
val it = 3 : int
-
```

```
- fun f(x) = x+1;
val f = fn : int -> int
- f 2;
val it = 3 : int
-
```

```
- val g = fn (x,y) => x+y;
val g = fn : int * int -> int
```

```
- fn (x,y) => x+y;
val it = fn : int * int -> int
```

```
- fn (x,y) => x+y+3.5;
val it = fn : real * real -> real
```

meccanismo per dare un nome a termini → val g = M;

val f = fn x=> x+1;
fun f(x)= x+1;

fun f x_1 x_2 ... x_n = M;

↕

val f = fn x_1=> (fn x_2 => (... =>(fn x_n => M)...))

i termini di un mini ML sono

parametro formale

corpo della funzione

funzione

argomento

fn x ⇒ M

x

M(N)

variabile

+,-,...,
0,1,...

termini buit-in

I tipi:
$t ::= int \mid t_1 \rightarrow t_2$

indichiamo con M:t il fatto che M ha tipo t

- ⊕: int → (int → int), 0:int, 1:int **...**
- **assumendo che** x:u **abbiamo che** x:u
- **se** M: t → u **e** N:t **allora** M(N):u
- **se** M:u **nell'assunzione che** x:t **allora** (**cancellando l'assunzione** x:t) fn x ⇒ M: t → u

Tipi/Logica/Deduzione naturale

$$x : u$$

$$\frac{}{\oplus : \mathrm{int} \rightarrow (\mathrm{int} \rightarrow \mathrm{int})} \qquad \frac{}{0 : \mathrm{int}} \qquad \frac{}{1 : \mathrm{int}}$$

$$\frac{M : t \rightarrow u \qquad N : t}{M(N) : u} \qquad \frac{\begin{array}{c}[x : t] \\ \vdots \\ M : u\end{array}}{\mathrm{fn}\ x \Rightarrow M : t \rightarrow u}$$

Γ è un insieme $x_1{:}t_1,..,x_n{:}t_n$

Assunzioni/Contesto/Ambiente

$$\frac{}{\Gamma, x{:}u \vdash x{:}u}$$

$$\frac{}{\Gamma \vdash \oplus{:}\ \mathrm{int} \rightarrow (\mathrm{int} \rightarrow \mathrm{int})}$$

$$\frac{}{\Gamma \vdash 0{:}\mathrm{int}} \qquad \frac{}{\Gamma \vdash 1{:}\mathrm{int}} \qquad \ldots \qquad \frac{}{\Gamma \vdash n{:}\mathrm{int}} \qquad \ldots$$

$$\frac{\Gamma \vdash M{:}\ t \rightarrow u \qquad \Gamma \vdash N{:}t}{\Gamma \vdash M(N){:}u}$$

$$\frac{\Gamma, x{:}t \vdash M{:}u}{\Gamma \vdash \mathrm{fn}\ x \Rightarrow M{:}\ t \rightarrow u}$$

```
val f = fn x=> (⊕ x) 1;
val f = fn : int -> int
```

$$
\dfrac{\dfrac{\dfrac{x{:}\mathrm{int} \vdash \oplus : \mathrm{int} \rightarrow (\mathrm{int} \rightarrow \mathrm{int}) \qquad x{:}\mathrm{int} \vdash x : \mathrm{int}}{x{:}\mathrm{int} \vdash \oplus\, x : \mathrm{int} \rightarrow \mathrm{int}} \qquad x{:}\mathrm{int} \vdash 1 : \mathrm{int}}{x{:}\mathrm{int} \vdash (\oplus\, x)1 : \mathrm{int}}}{\vdash \mathrm{fn}\ x \Rightarrow (\oplus\, x)\ 1 : \mathrm{int} \rightarrow \mathrm{int}}
$$

```
val id = fn x=> x;
val id = fn : ???
```

quale è il tipo di id?

per ogni tipo **t** abbiamo la seguente derivazione corretta di tipo

$$\frac{x{:}t \vdash x{:}t}{\vdash \mathrm{fn}\ x \Rightarrow x : t \to t}$$

l'idea è che id abbia tipo $\forall\alpha.\ \alpha \to \alpha$
(**'a->'a** nella sintassi ML)

dove α è una **variabile di tipo** e $\forall$ è un **quantificatore del secondo ordine** (quantifica rispetto alla classe dei tipi) occorre quindi arricchire i tipi con:

1. variabili di tipo (α,β,...)
2. quantificazioni di tipo

la trattazione tecnica rigorosa esula dagli scopi di questo corso :-(

diamo un cenno al sistema di tipi

I tipi

$t ::= int \mid \alpha \mid t_1 \rightarrow t_2$

Schemi di tipo

$\sigma ::= t \mid \forall\alpha.\,\sigma$

α è una variabile di tipo

sostituzione $\sigma' = \sigma[t/\alpha]$

Un termine chiuso M è tipabile se
è possibile derivare per M uno schema di tipo chiuso
(senza variabili di tipo libere)
ovvero o un tipo t senza variabili di tipo oppure uno
schema $\forall\alpha_1 \ldots \alpha_n.t$ dove $\alpha_1 \ldots \alpha_n$ sono tutte e sole le variabili
di tipo in t)

Γ è un insieme $x_1:t_1,..,x_n:t_n$

Assunzioni/Contesto/Ambiente

$$\frac{}{\Gamma, x:u \vdash x:u}$$

$$\frac{}{\Gamma \vdash \oplus : int \to (int \to int)}$$

$$\frac{}{\Gamma \vdash 0:int} \qquad \frac{}{\Gamma \vdash 1:int} \qquad \ldots \qquad \frac{}{\Gamma \vdash n:int} \qquad \ldots$$

$$\frac{\Gamma \vdash M: t \to u \qquad \Gamma \vdash N:t}{\Gamma \vdash M(N):u}$$

$$\frac{\Gamma \vdash M: \sigma}{\Gamma \vdash M: \forall\alpha.\ \sigma}$$

α non deve occorrere in Γ

$$\frac{\Gamma, x:t \vdash M:u}{\Gamma \vdash fn\ x \Rightarrow M: t \to u}$$

$$\frac{\Gamma \vdash M: \forall\alpha.\ \sigma}{\Gamma \vdash M: \sigma[t/\alpha]}$$

Il caso delle dichiarazioni locali

Γ è un insieme $x_1:\sigma_1,..,x_n:\sigma_n$

(t,u ➪ tipi, σ ➪ schema di tipo)

$$\frac{}{\Gamma, x:\sigma \vdash x:\sigma}$$

$$\frac{}{\Gamma \vdash 0:int}$$

$$\frac{}{\Gamma \vdash 1:int}$$

$$\frac{}{\Gamma \vdash n:int}$$

$$\frac{\Gamma, x:t \vdash M:u}{\Gamma \vdash fn\ x \Rightarrow M: t \to u}$$

$$\frac{}{\Gamma \vdash \oplus: int \to (int \to int)}$$

$$\frac{\Gamma \vdash M: t \to u \quad \Gamma \vdash N:t}{\Gamma \vdash M(N):u}$$

$$\frac{\Gamma \vdash M: \sigma}{\Gamma \vdash M: \forall\alpha.\ \sigma}$$

α non deve occorrere libera in Γ

$$\frac{\Gamma \vdash M: \sigma \quad \Gamma, z:\sigma \vdash N:t}{\Gamma \vdash \texttt{let val}\ z=M\ \texttt{in}\ N\ \texttt{end}:t}$$

$$\frac{\Gamma \vdash M: \forall\alpha.\ \sigma}{\Gamma \vdash M: \sigma[t/\alpha]}$$

$$\dfrac{\dfrac{x:\alpha \vdash x:\alpha}{\vdash \text{fn } x \Rightarrow x : \alpha \to \alpha}}{\vdash \text{fn } x \Rightarrow x : \forall\alpha.\ \alpha \to \alpha}$$

```
val id = fn x=> x;
val id = fn : ∀α. α → α
```

```
val f = fn x=> (⊕ x) 1;
val f = fn : int -> int
```

$$\dfrac{\dfrac{\dfrac{\vdash id : \forall\alpha.\ \alpha \to \alpha}{\vdash id : (int \to int) \to (int \to int)} \qquad \vdash f : int \to int}{\vdash id\ f : int \to int} \qquad \dfrac{\dfrac{\vdash id : \forall\alpha.\ \alpha \to \alpha}{\vdash id : int \to int} \qquad \vdash 2 : int}{\vdash id\ 2 : int}}{\vdash (id\ f)(id\ 2) : int}$$

id è una funzione polimorfa

The 'generalises' relation

We say a type scheme $\sigma = \forall\,\alpha_1, \ldots, \alpha_n\,(\tau)$ *generalises* a type τ', and write $\boxed{\sigma \succ \tau'}$ if τ' can be obtained from the type τ by simultaneously substituting some types τ_i for the type variables α_i ($i = 1, \ldots, n$):

$$\tau' = \tau[\tau_1/\alpha_1, \ldots, \tau_n/\alpha_n].$$

(N.B. The relation is unaffected by the particular choice of names of bound type variables in σ.)

The converse relation is called specialisation: a type τ' is a *specialisation* of a type scheme σ if $\sigma \succ \tau'$.

Typeable closed expressions

We write $\boxed{\vdash_{ML} M : \sigma}$ to indicate that

- M is *closed expression* (i.e. has no free variables)
- σ is a *closed type scheme* (i.e. is of the form $\forall\,A\,(\tau)$ with all the type variables occurring in the type τ contained in the set A)

Principal type schemes

A closed type scheme σ is the *principal* type scheme of a closed ML expression M if

(a) $\vdash_{ML} M : \sigma$

(b) for all closed σ', if $\vdash_{ML} M : \sigma'$ then $\sigma \succ \sigma'$

where by definition $\forall\,A\,(\tau) \succ \forall\,A'\,(\tau')$ holds if $\forall\,A\,(\tau)$ *generalises* τ' (We are assuming the type schemes are closed, so in particular all the type variables of τ are in A.)

esercizi

$\vdash$ fn x =>x : $\forall\alpha.\ \alpha \rightarrow \alpha$

$\vdash$ (fn x =>x)(fn x =>x) : $\forall\alpha.\ \alpha \rightarrow \alpha$

fn x =>x(x): NON è TIPABILE

$\vdash$ let val f = fn x => x in f(f) end: $\forall\alpha.\ \alpha \rightarrow \alpha$