Tipi

L'assenza di tipi in un linguaggio di programmazione può portare vari problemi, come ad esempio espressioni **sintatticamente corrette** ma **prive di significato**

λ-Calcolo con booleani

Estendiamo la sintassi aggiungendo le **costanti booleane** e l'**if.**

Espressioni	M, N	$::=$	x	variabile
		$\mid$	$\mathtt{c}$	costante
		$\mid$	$\lambda x.M$	astrazione
		$\mid$	$M\ N$	applicazione
		$\mid$	$\mathtt{if}\ M\ N_1\ N_2$	condizionale

$$\textbf{Costanti} \quad \mathtt{c} \in \{\mathtt{False}, \mathtt{True}\}$$

E estendiamo la semantica con le seguenti riduzioni:

$$\mathtt{if}\ \mathtt{True}\ M\ N \to M$$
$$\mathtt{if}\ \mathtt{False}\ M\ N \to N$$

Tuttavia, possiamo creare delle espressioni che sintatticamente non hanno senso.

Prendendo la nuova espressione **IF M** $N_1\ N_2$**,** se **M** avesse un tipo diverso da quello **booleano**, l'intera espressione non avrebbe senso.

$$\mathtt{if}\ (\lambda x.x)\ M\ N \not\to$$

Espressioni tipate

Per evitare questi errori, possiamo dare dei **tipi** alle espressioni.
Un **tipo** altro non è che una **forma sintattica di classificazione** delle espressioni.

la forma normale di una espressione di tipo `Bool`, se esiste, è una costante booleana (`False` o `True`)

la forma normale di una espressione di tipo $t \rightarrow s$, se esiste, è un'astrazione che, applicata ad una espressione di tipo t, produce una espressione di tipo s

Tipi	t, s	$::=$	`Bool`	booleani
		$\mid$	$t \rightarrow s$	funzioni

Giudizi

Introduciamo una **relazione binaria**,chiamata **Giudizio** della forma
$\vdash M : t$ ovvero
M è ben tipato e ha tipo t

Tuttavia, M potrebbe contenere variabili libere, e quindi bisogna relativizzarle nel suo **contesto.**

Esempio: se ho semplicemente x, non posso sapere che tipo è, ma se so il contesto(ad esempio "attorno" a x ci sono delle astrazioni) allora riesco a capire il tipo

Giudizi in un contesto

$\Gamma \vdash M : t$ ovvero
M è ben tipato e ha tipo t nel contesto Γ

Un contesto è semplicemente una mappa con **dominio** l'insieme delle variabili e con **codominio** il **tipo** di queste variabili.

Definizione (contesto)

*Un **contesto** Γ è una funzione parziale da variabili a tipi.*

Parziale perchè a me interessa soltanto l'insieme finito di variabili presenti in M.

Scriviamo $dom(\Gamma)$ per il dominio di Γ

Scriviamo $x : t$ per il contesto Γ tale che $dom(\Gamma) = \{x\}$ e $\Gamma(x) = t$

Scriviamo Γ, Γ' per l'unione di Γ e Γ' quando $dom(\Gamma) \cap dom(\Gamma') = \emptyset$

Regole di tipo

Forma generale di una regola:

[nome regola]

$$\frac{\textit{premessa}_1 \quad \cdots \quad \textit{premessa}_n}{\textit{conclusione}}$$

- **Se** le premesse sono vere, **allora** la conclusione è vera
- Una regola senza premesse è detta **assioma**

Regole di tipo con costanti

Ogni *conclusione* è un **giudizio.**
Ha sempre un **contesto** Γ (se non c'è significa che è il contesto vuoto), seguito da $\vdash$,seguito da un **termine**(costante, astrazione etc) seguito da **:** e infine il **tipo**

Assiomi

Bool:

[t-bool]

$$\frac{}{\Gamma \vdash \mathtt{c} : \mathtt{Bool}}$$

Una costante booleana c è ben tipata in qualsiasi sia il contesto Γ e ha tipo $Bool$

Var:

[t-var]

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

Nel contesto dove x è presente($\Gamma, x : t$), x è ben tipata e ha tipo t (notare che è lo stesso tipo del contesto)

***N.B.!:** $\Gamma, x : t$ indica il contesto dato dall'unione del contesto Γ e $x : t$*

Altre regole

Lambda astrazioni:

[t-lam]

$$\frac{\Gamma, x : t \vdash M : s}{\Gamma \vdash \lambda x.M : t \to s}$$

- ***Premessa***: *il corpo M è ben tipato nel contesto $\Gamma, x : t$ e ha tipo s. Infatti in M potrebbe comparire x stessa.*
- ***Conclusione***: *la lambda-astrazione è ben tipata e ha tipo $t \to s$ nel contesto Γ.*
  Perchè t? Perchè è il tipo dell'argomento della funzione

If-then-else:

[t-if]

$$\frac{\Gamma \vdash M : \texttt{Bool} \qquad \Gamma \vdash N_1 : t \qquad \Gamma \vdash N_2 : t}{\Gamma \vdash \texttt{if}\ M\ N_1\ N_2 : t}$$

- ***Premessa 1:*** *Il tipo di M, la variabile che uso come test nell'IF, deve essere ben tipata, in qualsiasi contesto, e deve avere tipo $Bool$.*
- ***Premessa 2-3:*** *Il tipo di N_1 e N_2, i termini del $Then$ o dell'$Else$, devono avere lo stesso tipo t nel contesto Γ*
- ***Conclusione:*** *Il tipo dell'espressione è ben tipata e ha tipo t*

Applicazione:

[t-app]

$$\frac{\Gamma \vdash M : t \to s \qquad \Gamma \vdash N : t}{\Gamma \vdash M\ N : s}$$

- ***Premessa 1:*** *M è funzione, dunque è ben tipata e ha tipo $t \to s$*
- ***Premessa 2:*** *N è ben tipata e ha tipo t, ovvero lo stesso tipo del dominio di M*
- ***Conclusione:*** *L'espressione è ben tipata e ha tipo s, ovvero lo stesso tipo del codominio di M*

Proprietà delle espressioni ben tipate

Lemma (subject reduction)

Se $\Gamma \vdash M : t$ e $M \to N$ allora $\Gamma \vdash N : t$.

Se M è ben tipato nel contesto Γ e ha tipo t, allora anche il suo ridotto N è ben tipato nello stesso contesto Γ e ha lo stesso tipo t

Definizione (valore)

*Diciamo che M è un **valore** se M è una costante o un'astrazione.*

Questo ci dice cosa è un valore, ovvero il risultato di una computazione

Esempi:

$(\lambda x.x)$ `True` non è un valore

`True` $(\lambda x.x)$ non è un valore

`if` $(\lambda x.x)$ `True` `False` non è un valore

$\lambda x.(\lambda y.y)$ `True` è un valore

La prima si può ridurre a $\texttt{True}$ che è un valore

Teorema (progresso)

Se $\vdash M : t$ e $M \Rightarrow N \not\to$ allora N è un valore.

Questo ci dice che se M è ben tipato in un contesto vuoto(ovvero M è un termine chiuso), e M si riduce in più passi a N e N è in forma normale, allora N è un valore