# Il Lambda calcolo tipato

# Lambda calcolo tipato semplice

- Come abbiamo visto a lezione, Il lambda-calcolo nella sua forma "pura" non prevede tipi.
- Per comodità è più semplice avere un insieme di tipi di base, quindi, a rigore, nella letteratura, esistono numerose varianti del lambda calcolo tipato semplice a seconda della scelta dei tipi base.
- Noi consideriamo inizialmente una variante costruita su valori booleani (oltre che su valori di tipo funzione, già visti).

# #1: Sintassi dei tipi

| | |
|---|---|
| **$\tau$ ::=** | **Tipi** |
| **Bool** | **Tipo dei booleani** |
| **$\tau \rightarrow \tau$** | **Tipo delle funzioni** |

Esempi di tipi sintatticamente corretti:

Bool    Bool → Bool    (Bool → Bool) → Bool

(Bool → Bool) → (Bool → Bool)

Attenzione: → associa a destra!! quindi:

Bool → Bool → Bool = Bool → (Bool → Bool)

# #2: Sintassi del linguaggio

**FUN: Lambda calcolo tipato semplice con booleani**

| | |
|---|---|
| **e ::=** | **Espressioni** |
| **x** | **Variabili** |
| **fun x: $\tau$ = e** | **Funzioni** |
| **Apply(e, e)** | **Applicazione** |
| **true** | **Costante true** |
| **false** | **Costante false** |
| **if e then e else e** | **Condizionale** |

# #2: Sintassi del linguaggio (formato tradizionale)

**FUN: Lambda calcolo tipato semplice con booleani**

| | |
|---|---|
| **e ::=** | **Espressioni** |
| **x** | **Variabili** |
| **$\lambda$ x:$\tau$. e** | **Funzioni** |
| **e e** | **Applicazione** |
| **true** | **Costante true** |
| **false** | **Costante false** |
| **if e then e else e** | **Condizionale** |

# #2: Valori

| | |
|---|---|
| **V::=** | **Valori** |
| **fun x:$\tau$ = e** | **Funzioni** |
| **true** | **Valore true** |
| **false** | **Valore false** |

# #3: Semantica (cose già viste… riscritte con la nuova sintassi)

$$Apply\ (fun\ x{:}\ \tau = e_1\ , v) \to e_1\{x \coloneqq v\}$$

$$\frac{e_1 \to e'}{Apply(e_1, e_2) \to Apply(e', e_2)}$$

$$\frac{e_2 \to e'}{Apply(v, e_2) \to Apply(v, e')}$$

$$\frac{e_1 \to e_4}{if\ e_1\ then\ e_2\ else\ e_3 \to if\ e_4\ then\ e_2\ else\ e_3}$$

$$if\ true\ then\ e_2\ else\ e_3 \to e_2$$

$$if\ false\ then\ e_2\ else\ e_3 \to e_3$$

# #3: Semantica (cose già viste… riscritte con la nuova sintassi)

$$Apply\ (fun\ x : \tau = e_1\ , v) \rightarrow e_1\{x := v\}$$

β-riduzione con strategia call-by-value

$$\frac{e_1 \rightarrow e'}{Apply(e_1, e_2) \rightarrow Apply(e', e_2)} \qquad \frac{e_2 \rightarrow e'}{Apply(v, e_2) \rightarrow Apply(v, e')}$$

$$\frac{e_1 \rightarrow e'}{if\ e_1\ then\ e_2\ else\ e_3 \rightarrow if\ e'\ then\ e_2\ else\ e_3}$$

$$if\ true\ then\ e_2\ else\ e_3 \rightarrow e_2$$

$$if\ false\ then\ e_2\ else\ e_3 \rightarrow e_3$$

espressione condizionale

# #4: Composizionalità del type checker

**Definamo le regole del type checker induttivamente sulla struttura sintattica del linguaggio**

**Problema:**

$$\frac{\boldsymbol{x}\colon \boldsymbol{\tau_1} \quad ?????}{\boldsymbol{fun\ x}\colon \boldsymbol{\tau_1} = \boldsymbol{e}\colon \boldsymbol{\tau_1} \rightarrow \boldsymbol{\tau_2}}$$

**Il tipo del corpo e della funzione dipende dal tipo del parametro formale x. Come teniamo conto di questa associazione?**

# #3: tipi delle variabili

**Ambiente dei tipi.**

L'ambiente dei tipi è una funzione (di dominio finito) che associa nomi a tipi. Noi scriveremo:

$$\boldsymbol{\Gamma = x_1 : \tau_1, x_2 : \tau_2 \ \dots xk : \tau_k}$$

per indicare la funzione

$$\boldsymbol{\Gamma(x_i) = \tau_i}$$

che associa il tipo $\tau_i$ al valore $x_i$

La notazione

$$\boldsymbol{\Gamma, x : \tau}$$

Sarà usata per indicare l'estensione della funzione $\Gamma$ con l'associazione $x : \tau$

$$\boldsymbol{(\Gamma, \mathrm{x} : \tau)(\mathrm{x}) = \tau \quad \mathrm{e} \quad (\Gamma, \mathrm{x} : \tau)(\mathrm{y}) = \Gamma(y)\ per\ y\ != x}$$

# Esempio

$$\boldsymbol{\Gamma = x:\tau, y:\tau'}$$

$$\boldsymbol{\Gamma(x) = \tau}$$

$$\boldsymbol{\Gamma(z) = undefined}$$

# Esempio

$$\boldsymbol{\Gamma = x\colon \tau, y\colon \tau'}$$

$$\boldsymbol{\Gamma(x) = \tau}$$

$$\boldsymbol{\Gamma' = \Gamma, \mathrm{z}\colon \tau_2}$$

$$\boldsymbol{\Gamma'(z) = \tau_2}$$

# Esempio

**L'ambiente dei tipi vuoto $\emptyset$ non contiene alcun legame di tipo per le variabili**

$$\emptyset(x) = \boldsymbol{undefined}$$
$$\boldsymbol{per\ tutti\ i\ nomi\ x}$$

# Giudizio di tipo

Supponiamo che $\Gamma$ sia un ambiente di tipo.
La notazione:

$$\boldsymbol{\Gamma \vdash e : \tau}$$

È usata per indicare che l'espressione e ha tipo $\tau$ nell'ambiente di tipo $\Gamma$.

**INTUIZIONE: L'ambiente dei tipi tiene conto dei legami tra i nomi che compaiono nel programma (l'espressione e) e il loro tipo**

# Giudizio di tipo

Supponiamo che $\Gamma$ sia un ambiente di tipo.
La notazione:

$$\boldsymbol{\Gamma \vdash e : \tau}$$

È usata per indicare che l'espressione e ha tipo $\tau$ nell'ambiente di tipo $\Gamma$.

**Le regole sono applicata dal compilatore in fase di analisi statica.**
**L'ambiente dei tipi nel gergo dei compilatori è chiamato**
**Tabella dei Simboli (Symbol Table)**

# Regole di tipo

Definiamo il sistema per il controllo dei tipi (type checker) per il nostro Lambda calcolo tipato semplice

$$\Gamma \vdash true : Bool$$

$$\Gamma \vdash false : Bool$$

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

**Una variabile ha il tipo a lei associato nell'ambiente dei tipi.**

# Condizionale

$$\frac{\boldsymbol{\Gamma \vdash e : Bool \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}}{\boldsymbol{\Gamma \vdash if\ e\ then\ e_1\ else\ e_2 : \tau}}$$

# Tipi per le funzioni

- Quale è il tipo di una funzione?
- Il costruttore di tipo $\tau 1 \to \tau 2$ descrive il tipo della funzioni che prendono in ingresso un argomento di tipo $\tau 1$ e restituiscono un risultato di tipo $\tau 2$

# Funzioni

$$\frac{\Gamma, x{:}\,\tau_1 \vdash e{:}\,\tau_2}{\Gamma \vdash fun\; x{:}\,\tau_1 = e{:}\,\tau_1 \rightarrow \tau_2}$$

**Dato che il tipo del parametro formale x ($\tau_1$) è noto:**
**(i) il tipo delle occorrenze del parametro x nel corpo della funzione saranno associate a $\tau_1$**
**(ii) il tipo del risultato della funzione sarà il tipo del corpo della funzione.**

**Intuizione**
**La funzione richiede un parametro di tipo $\tau_1$ e restituisce come risultato un valore di tipo $\tau_2$.**
**Notazione: $\tau_1 \rightarrow \tau_2$**

# Funzioni: regole di visibilità

$$\frac{\Gamma, x: \tau_1 \vdash e: \tau_2}{\Gamma \vdash fun\ x: \tau_1 = e: \tau_1 \to \tau_2}$$

**Intuizione (informatica)**
**La definizione del parametro formale x e del suo tipo è una specie di dichiarazione dinamica. La dichiarazione del parametro x sovrascrive e annulla (nell'ambiente esteso $\Gamma, x: \tau_1$ ) le precedenti dichiarazioni per x (che sono in $\Gamma$). La portata (scope) della dichiarazione del parametro x è il corpo della funzione. Infatti $\Gamma, x: \tau_1$ viene usato solo per tipare il corpo della funzione.**

# Esempio

$$\frac{\emptyset, x{:}\,Bool \vdash x{:}\,Bool}{\emptyset \vdash fun\; x{:}\,Bool = x{:}\,Bool \to\; Bool}$$

# Chiamata di funzione

$$\frac{\boldsymbol{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}}{\boldsymbol{\Gamma \vdash Apply(e_1, e_2) : \tau_2}}$$

# Esempio

$$\cfrac{\cfrac{\Gamma(f) = Bool \to Bool}{\Gamma \vdash f : Bool \to Bool} \quad \cfrac{\Gamma \vdash false\colon Bool \quad \Gamma \vdash true\colon Bool \quad \Gamma \vdash false\colon Bool}{\Gamma \vdash if\ false\ then\ true\ else\ false\colon Bool}}{\cfrac{\Gamma = f\colon Bool \to Bool \vdash Apply(f, if\ false\ then\ true\ else\ false)\colon Bool}{\emptyset \vdash fun\ f\colon Bool \to Bool =\ Apply(f, if\ false\ then\ true\ else\ false)\colon (Bool \to Bool) \to Bool}}$$

# Type Safety

La correttezza (type safety) del sistema di tipo del lambda calcolo Tipato è espressa formalmente da queste due proprietà

**Progresso**: Se $\emptyset \vdash \boldsymbol{e} : \boldsymbol{\tau}$ allora **e** è un valore oppure $\boldsymbol{e} \rightarrow \boldsymbol{e}'$ per una qualche espressione **e'**

**Progresso: Una espressione senza variabili libere ben tipata non si blocca a run-time**

**Conservazione**: Se $\boldsymbol{\Gamma} \vdash \boldsymbol{e} : \boldsymbol{\tau}$ e $\boldsymbol{e} \rightarrow e'$ allora $\boldsymbol{\Gamma} \vdash \boldsymbol{e}' : \boldsymbol{\tau}$

**Conservazione: I tipi sono preservati dalle regole di esecuzione**

# Teorema del progresso

**Progresso**: Se $\emptyset \vdash \boldsymbol{e} : \boldsymbol{\tau}$ allora **e** è un valore oppure $\boldsymbol{e} \rightarrow \boldsymbol{e}'$ per una qualche espressione **e’**

**Come si dimostra?**
**Per induzione sulle derivazioni di tipo.**

**I casi di base (costanti booleane) sono identici ai casi di base del sistema per le espressioni aritmetiche.**
**Il caso delle variabili è banale.**
**Il caso dell'astrazione funzionale è immediato, poiché le funzioni sono valori.**

# Teorema del progresso

**Progresso**: Se $\emptyset \vdash e: \tau$ allora **e** è un valore oppure $e \rightarrow e'$ per una qualche espressione **e'**

**Casi induttivi (consideriamo solo il caso dell'applicazione)**
**Applicazione: e = Apply($e_1$, $e_2$), $\emptyset \vdash e_1: \tau_1 \rightarrow \tau_2$ $\emptyset \vdash e_2: \tau_1$ .**
**Per l'ipotesi induttiva possiamo affermare che,**
**$e_1$ è un valore o può fare un passo di valutazione, e così pure $e_2$.**
**Se le espressioni possono fare un passo applichiamo le regole di riduzione dell'applicazione e terminiamo.**
**Se invece sono entrambi valori, allora abbiamo che $e_1$ deve essere della forma**

$$fun\, x: \tau_1 = e': \tau_1 \rightarrow \tau_2$$

**Pertanto applichiamo la regola di beta riduzione.**

# Teorema della conservazione

**Conservazione**: Se $\boldsymbol{\Gamma} \vdash \boldsymbol{e}: \boldsymbol{\tau}$ e $\boldsymbol{e} \rightarrow e'$ allora $\boldsymbol{\Gamma} \vdash \boldsymbol{e}': \boldsymbol{\tau}$

Si dimostra sempre per induzione strutturale sulle regole di tipo.
Quale è il caso difficile?

# Teorema della conservazione

**Conservazione**: Se $\boldsymbol{\Gamma \vdash e : \tau}$ e $\boldsymbol{e \to e'}$ allora $\boldsymbol{\Gamma \vdash e' : \tau}$

Si dimostra sempre per induzione strutturale sulle regole di tipo.
Quale è il caso difficile?

Applicazione: e = Apply($e_1$, $e_2$). Quindi vale che

$$\boldsymbol{\Gamma \vdash e : \tau}$$
$$\boldsymbol{\Gamma \vdash e_1 : \tau_1 \to \tau_2}$$
$$\boldsymbol{\Gamma \vdash e_2 : \tau_2}$$
$$\boldsymbol{\tau = \tau_2}$$
$$\boldsymbol{e \to e'}$$

**Si deve pertanto dimostrare che**

$$\boldsymbol{\Gamma \vdash e' : \tau_2}$$

# Teorema della conservazione

Applicazione: e = Apply(e1, e2). Quindi vale che

$$\Gamma \vdash e: \tau$$
$$\Gamma \vdash e_1: \tau_1 \rightarrow \tau_2$$
$$\Gamma \vdash e_2: \tau_2$$
$$\tau = \tau_2$$
$$e \rightarrow e'$$

**Si deve pertanto dimostrare che**

$$\Gamma \vdash e': \tau_2$$

$\Gamma \vdash e_1: \tau_1 \rightarrow \tau_2$ comporta che:

$$e_1 = fun\ x: \tau_1 = e_3 \qquad \Gamma, x: \tau_1 \vdash e_3: \tau_2$$

$$e' = e_3\ \{x := e_2\}$$

**Problema: abbiamo dimostrato $e_3: \tau_2$ e invece la semantica dell'applicazione ci porta in $e_3\ \{x = e_2\}$, non in $e_3$**
**… si deve gestire la sostituzione**

# Substitution lemma

Lemma: I tipi sono preservati dall'operazione di sostituzione.

$$\Gamma, x{:}\,\tau_1 \vdash e{:}\ \tau$$
$$\Gamma \vdash e_1{:}\,\tau_1$$
$$\Gamma \vdash e\{x = e_1\}{:}\,\tau$$

Come si dimostra?
Per induzione sulla derivazione di

$$\Gamma, x{:}\,\tau_1 \vdash e{:}\ \tau$$

I casi più interessanti sono quelli per le variabili e le astrazioni funzionali.
Trovate la dimostrazione completa sulle note didattiche