Università di Pisa

Dipartimento di Informatica
Corso di Laurea Triennale in Informatica

Corso 2° anno - 9 CFU

Paradigmi di programmazione

Professore:
Prof. Paolo Milazzo

Autore:
Filippo Ghirardini

Anno Accademico 2023/2024

Contents

1 Lambda calcolo

2 Sistemi di tipi

2.1 Perché?

Dato che nel lambda calcolo i programmi e i valori sono funzioni possiamo facilmente scrivere programmi che non sono corretti rispetto all'uso inteso dei valori. Ad esempio:

Esempio 2.1 (Errore tipi)**.** Nella seguente espressione si può applicare 0 a *False*, ottenendo quindi un risultato che però non ha alcun senso:

$$False\, 0 = (\lambda t.\lambda f.f)(\lambda z.\lambda s.z) \to \lambda f.f \tag{1}$$

Analogamente per una macchina tutto è un bit: istruzioni, dati e operazioni. Un esempio più pratico è il seguente:

Esempio 2.2. L'istruzione nel corpo dell'*if* contiene un errore di tipo (stringa divisa per un intero). Se non avessimo il controllo dei tipi l'unico modo per scoprire l'errore sarebbe eseguire numerosi test per riuscire a coprire tutte le possibilità, fino ad entrare nel corpo dell'*if*. Richiederebbe tempo e risorse e non ci garantisce neanche la certezza di aver provato tutti i casi possibili.

```
if (condizione_complicata) {
  return "hello"/10;
}
```

Se in certi linguaggi di programmazione ci troveremmo davanti ad errori di esecuzione, in altri (come ad esempio JavaScript) otterremmo un errore nel risultato in quanto l'interprete proverebbe a fare un cast manuale.
Concludendo, la mancanza di **type safety** aumenta il numero di bug, rendendo così un software meno funzionale e più vulnerabile.

2.2 Cosa sono i tipi?

I **sistemi di tipo** sono meccanismi che permettono di rilevare in anticipo errori di programmazione.

Definizione 2.1 (Tipo)**.** *Il tipo è un* ***attributo*** *di un dato che descrive come il linguaggio di programmazione permetta di usare quel particolare dato.*

Un tipo serve quindi a limitare i valori che un'espressione può assumere, che operazioni possono essere effettuate sui dati e in che modo questi ultimi possono essere salvati.

Definizione 2.2 (Sistema dei tipi)**.** *Un sistema dei tipi è un metodo* ***sintattico***, ***effettivo*** *per dimostrare l'assenza di comportamenti anomali del programma* ***strutturando*** *le operazioni del programma in base ai tipi di valori che calcolano.*

Analizziamo i tre aspetti:

- *Sintattico*: l'analisi viene effettuata dal punto di vista sintattico
- *Effettivo*: si può definire un algoritmo che effettui questa analisi
- *Strutturale*: i tipi assegnati si ottengono in maniera **composizionale** dalle sottoespressioni.

2.3 Come funziona?

Un sistema dei tipi associa dei tipi ai valori calcolati. Esaminando il flusso dei valori calcolati prova a dimostrare che non avvengano errori (di tipo, non in generale)facendo un controllo, che può avvenire in due modi:

- *Statico*: avviene in fase di compilazione, non degradando le prestazioni
- *Dinamico*: avviene in fase di esecuzione e aumenta il tempo di esecuzione

2.4 Come si progetta?

2.4.1 Specifiche del linguaggio

Prendiamo come esempio il seguente **linguaggio**:

Espressioni	**Valori**	**Valori numerici**	**Tipi**
E::=	V::=	NV::=	T::=
true	true	$0\|1\|2\|\ldots$	Bool
false	false		Nat
NV	NV		
if E then E else E			
succ E			
pred E			
isZero E			

2.4.2 Regole di valutazione

Avremo le seguenti **regole di valutazione**:

$$if\ true\ then\ E1\ else\ E2 \to E1 \tag{2}$$

$$if\ false\ then\ E1\ else\ E2 \to E2 \tag{3}$$

$$\frac{E \to E'}{if\ E\ then\ E1\ else\ E2 \to if\ E'\ then\ E1\ else\ E2 \to E1} \tag{4}$$

$$\frac{E \to E'}{succ\ E \to succ\ E'} \qquad \frac{m = n + 1}{succ\ E \to succ\ E'} \tag{5}$$

$$\frac{E \to E'}{pred\ E \to pred\ E'} \qquad \frac{n > 0,\ m = n - 1}{pred\ n \to m} \qquad pred\ 0 \to 0 \tag{6}$$

$$\frac{E \to E'}{isZero\ E \to isZero\ E'} \qquad isZero\ 0 \to true \qquad \frac{n > 0}{isZero\ n \to false} \tag{7}$$

2.4.3 Type checking

Il **controllo di tipo** definisce una relazione binaria (E, T) che associa il tipo T all'espressione E. Questo ha due caratteristiche principali:

- Utilizza il *metodo sintattico*
- Le regole sono definite per *induzione strutturale* sul programma

Le regole sono le seguenti:

$$true : Bool \qquad false : Bool \qquad n : Nat \tag{8}$$

$$\frac{E : Nat}{succ\ E : Nat} \qquad \frac{E : Nat}{pred\ E : Nat} \qquad \frac{E : Nat}{isZero\ E : Bool} \tag{9}$$

$$\frac{E : Bool, E1 : T, E2 : T}{if\ E\ then\ E1\ else\ E2} \tag{10}$$

2.4.4 Composizionalità

I sistemi di tipo sono **imprecisi**: non definiscono esattamente quale tipo di valore sarà restituito da ogni programma, ma solo un'**approssimazione conservativa**.

Esempio 2.3. La seguente espressione:

$$if\ E\ then\ 0\ else\ false \tag{11}$$

potrebbe restituire come risultato sia un *Bool* che un *Nat* a seconda del valore di *E*. Il controllo dei tipi quindi non permetterà che possano esserci due risultati diversi, riducendo la precisione ma mantenendo la sicurezza.

Questo avviene proprio per garantire la **composizionalità**, infatti ad esempio la regola dell'equazione 10 necessita che $E1$ ed $E2$ abbiano lo stesso tipo.

2.5 Dimostrazione

La **correttezza** del sistema di tipo è espressa da due proprietà:

- Progresso
- Conservazione

2.5.1 Progresso

Definizione 2.3 (Progresso)**.** *Se $E : T$ allora E è un valore oppure $E \to E'$ per una qualche espressione E'.*

In pratica, un'espressione ben tipata non si blocca a run-time. Può fare sempre un passo a meno che non sia un valore.

Proof. Utilizziamo l'induzione sulla struttura di derivazione di $E : T$.
I *casi base* sono i seguenti:

- $true : Bool$
- $false : Bool$
- $0|1|1| \ldots : Nat$

I *casi induttivi* sono tutti molto simili, vediamo quello per la formula 10.
Per *ipotesi induttiva* abbiamo due casi:

- $E1$ è un valore. In questo caso deve essere $true$ o $false$ e le regole della semantica fanno fare un **passo** del tipo $E \to E1$ o $E \to E2$
- Esiste $E4$ tale che $E1 \to E4$. In questo caso si applica la regola 4 e si esegue un **passo**.

□

2.5.2 Conservazione

Definizione 2.4 (Conservazione)**.** *Se $E : T$ e $E \to E'$ allora $E' : T$.*

In pratica i tipi sono preservati dalle regole di esecuzione.

Proof. Utilizziamo l'induzione come nella precedente dimostrazione.
I *casi base* sono immediati: $true$, $false$ e $0|1|2| \ldots$ sono valori e di conseguenza non fanno nessun passo.
Anche qui per i *casi induttivi* vediamo quello per la formula 4. Per l'ipotesi induttiva abbiamo due casi:

- $E1$ è un valore:
  - $true$: in questo caso $E \to E2$ e sappiamo già per ipotesi induttiva che $E2 : T$ (sappiamo che il passo ha successo)
  - $false$: in questo caso $E \to E3$ e $E3 : T$
- Esiste $E4$ tale che $E1 \to E4$. Questo implica:

  $$E = if\ E1\ then\ E2\ else\ E3 \to if\ E4\ then\ E2\ else\ E3$$

  Dato che per ipotesi induttiva $E1 : Bool$ abbiamo che $E4 : Bool$ e, grazie alle derivazioni che valgono per ipotesi $E2 : T$ e $E3 : T$, possiamo derivare applicando la regola 10.

□