Type Checking

↓

processo che verifica se le espressioni sono tipate correttamente

Tipo Es. $\mathbb{B}$, int, ...

- Un insieme di valori (Es. numeri interi)
- Un insieme di operazioni che si possono effettuare su quei valori (Es. $+, -, *, /$)

Il nostro obiettivo è: **data un'espressione, verificare che sia tipata correttamente**

Ciò viene fatto attraverso l'uso di REGOLE D'INFERENZA

Una REGOLA D'INFERENZA è composta da un insieme di premesse $A_1, \dots, A_n$ e da una conclusione A

$$\frac{A_1 \quad \dots \quad A_n}{A}$$

← se sono vere $A_1 \dots A_n$ allora A è vera

Una regola senza premesse è chiamata assioma → $\dfrac{}{A}$

Simile alla definizione che si usa in logica, <u>noi siamo interessati ai tipi</u>

quindi $A_1, \dots, A_n, A$ in realtà ci dicono il tipo di una espressione

e si indicano con $\Gamma \vdash e : T$

dal contesto Gamma — deduciamo — che e ha tipo T

Un esempio di regola è quella dell'AND

$\&\& : bool \times bool \to bool$ $\qquad$ $+ : int \times int \to int$

$$\frac{\Gamma \vdash e_1 : bool \qquad \Gamma \vdash e_2 : bool}{\Gamma \vdash e_1 \,\&\&\, e_2 : bool} \qquad \frac{\Gamma \vdash e_1 : int \qquad \Gamma \vdash e_2 : int}{\Gamma \vdash e_1 + e_2 : int}$$

Come capiamo il tipo delle variabili?

<u>Dipende dal contesto</u> (chiamato Γ) in cui si trovano

hint: gamma nelle regole verrà utilizzato solo per accedere al tipo di una variabile! Possiamo pensarla come una funz che presa in input la var. mi dice il tipo nell'ambiente

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

dove T può essere bool, int, ...

Nota: per rendere vera la conclusione, la premessa deve valere, altrimenti la nostra exp. è tipata male!

$\Gamma : x \to int$ $\qquad$ $\dfrac{\Gamma(x) : int}{\Gamma \vdash x : bool}$ ✗

Introduciamo le regole d'inferenza per un ling. con interi, bool, somma e uguaglianza e variabili

Sono tutti assiomi, non servono premesse
. 7 è un int
. true è un bool
. false è un bool

$$\dfrac{}{\Gamma \vdash num : int}\,[Num] \qquad \dfrac{}{\Gamma \vdash true : bool}\,[true] \qquad \dfrac{}{\Gamma \vdash false : bool}\,[false]$$

Nome della regola

$$\dfrac{\Gamma(x) : T}{\Gamma \vdash x : T}\,[Id]$$

qui si accede a gamma per verificare il tipo di x → Nota: queste regole sono sempre nelle foglie dell'albero, immaginate di scomporre una espressione

\+ deve prendere due interi e tornare un intero

$$\dfrac{\Gamma \vdash e_1 : int \quad \Gamma \vdash e_2 : int \quad + : int \times int \to int}{\Gamma \vdash e_1 + e_2 : int}\,[sum]$$

non T, la somma non è definita nei booleani

$$\dfrac{\Gamma \vdash e_1 : T_1 \quad \Gamma \vdash e_2 : T_2 \quad T_1 = T_2 \quad == : T_1 \times T_2 \to bool}{\Gamma \vdash e_1 == e_2 : bool}$$

è bool, un'uguaglianza può essere vera o falsa, ma vogliamo def. sia su bool che su int!

tipiamo $((x+2)+3) == (x+2)$ assumendo $x \to int$

(non ci interessa il valore!)

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma(x):int}{\Gamma \vdash x : int} \quad \dfrac{}{\Gamma \vdash 2 : int} \quad + \dots}{\Gamma \vdash x+2 : int}\,[sum] \quad \dfrac{}{\Gamma \vdash 3 : int} \quad + : int \times int \to int}{\Gamma \vdash (x+2)+3 : T_1^{\,int}}\,[sum] \qquad \dfrac{\dfrac{\Gamma(x):int}{\Gamma \vdash x : int} \quad \dfrac{}{\Gamma \vdash 2 : int} \quad + \dots}{\Gamma \vdash x+2 : T_2^{\,int}} \quad \overset{int}{T_1} = \overset{int}{T_2} \quad == : \overset{int}{T_1} \times \overset{int}{T_2} \to bool}{\Gamma \vdash ((x+2)+3) == (x+2) : bool}\,[eq]$$

(le scelte in rosso si chiamano unifications)

Complichiamo

Regole per if then else e dichiaraz. di funz.

$$\dfrac{\Gamma \vdash e_1 : T_1 \quad \Gamma \vdash e_2 : T_2 \quad \Gamma \vdash e_3 : T_3 \quad T_1 = bool \quad T_2 = T_3}{\Gamma \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : T_2}\,[if]$$

formali e attuali

$$\dfrac{\Gamma \vdash f : T_1 \times \dots \times T_n \to T \quad (\Gamma \vdash e_i : T_i')_{i \in 1..n} \quad (T_i = T_i')_{i \in 1..n}}{\Gamma \vdash f(x_1, \dots, x_n) : T}$$

dichiarazioni

Se dichiaro una var.
↓
Aggiorno l'ambiente

quindi le dichiaraz. tornano un ambiente aggiornato

$$\frac{\Gamma \vdash e : T' \quad x \notin dom(top(\Gamma)) \quad T = T'}{\Gamma \vdash T\ x = e; \ : \Gamma'}\ [VarD]$$

$\Gamma' \rightarrow \Gamma[x \to T]$

$$\frac{\Gamma \vdash d : \Gamma' \quad \Gamma' \vdash D : \Gamma''}{\Gamma \vdash d\,D : \Gamma''}\ [SeqD]$$

simile per le dichiarazioni di funzioni

$$\frac{\Gamma[x_1 \to T_1, \dots, x_n \to T_n] \vdash e : T' \quad T' = T \quad f \notin dom(top(\Gamma))}{\Gamma \vdash T\ f(T_1 x_1, \dots, T_n x_n) = e}\ [fun]$$

← non ammette ricorsive
h piano e senza avere f nel contesto

Slide 25 pacco 6

Aggiungiamo la regola per il let, che ci introduce la creazione dei contesti

nuovo scope

nuovo scope con le dichiaraz.

entrando in uno scope

uscendo

$$\frac{\Gamma \cdot [\,] \vdash D : \Gamma' \quad \Gamma' \vdash e : T}{\Gamma \vdash let\ D\ in\ e : T}\ [let]$$

→ Spoiler: in realtà per come abbiamo definito la sintassi, non serve creare nuovi contesti perché i nostri progr. saranno sempre

dec D [

prog e { Qui non ci sono dichiarazioni!

qui finisce →

regole es.

$$\frac{\Gamma \vdash D : \Gamma' \quad \Gamma' \vdash e : T}{\Gamma \vdash let\ D\ in\ e : T}$$

Nota:
(La regola prog (quella principale) sarà uguale alla let)

- Problema della mutua ricorsione (credo solo teorico)

```
int x() {
   :
   y()
}
int y() {
   :
   x()
}
```

← Questa y non è ancora definita nel mio contesto! Si trova dopo

Risolviamo facendo una visita precedente per aggiungere tutte le dichiarazioni al contesto

Le regole sono uguali ma si usa $\Vdash$ anziché $\vdash$

Nel caso l'es. chiedesse le regole per un ling. con mutua ric. allora si usano queste

Es.

$$\emptyset \Vdash D : \Gamma'$$

$$\frac{\Gamma \cdot \Gamma' \cdot [\,] \vdash D : \Gamma'' \quad \Gamma'' \vdash e : T}{\Gamma \vdash let\ D\ in\ e : T}\ [let]$$

- Subtyping

tipiamo let $T\ x = e$ in $e' : T'$

$$\dfrac{\dfrac{\Gamma\cdot[\,] \vdash e : T'' \quad T = T'' \quad x \notin dom(top(\Gamma\cdot\Gamma))}{\Gamma\cdot[\,] \vdash T x = e : \Gamma\cdot[x \to T]} \qquad \Gamma\cdot[x \to T] \vdash e' : T'}{\Gamma \vdash let\ T\ x = e\ in\ e' : T'}$$

nelle regole precedenti abbiamo sempre questo $T = T''$

può essere troppo restrittivo

↓

introduciamo i sottotipi:

se $T <: T'$ allora posso usare un oggetto di tipo T invece di uno di tipo T'

Es. int e float

int <: float

$$\dfrac{\Gamma \vdash e : T \quad \Gamma[x \to T'] \vdash e' : T'' \quad T <: T'}{\Gamma \vdash let\ T'\ x = e\ in\ e' : T''}$$

- subtyping con funzioni

$$\dfrac{\Gamma \vdash f : T_1 \times \ldots \times T_n \to T \quad (\Gamma \vdash e_i : T_i')_{i \in 1..n} \quad (T_i' <: T_i)_{i \in 1..n}}{\Gamma \vdash f(e_1, \ldots, e_n) : T} \xrightarrow{\text{non abbiamo incluso } T} \dfrac{(T_i' <: T_i)_{\forall i} \qquad T' <: T}{T_1' \times \ldots \times T_n' \to T' <: T_1 \times \ldots \times T_n \to T}$$

- subtyping nell'if

$$\dfrac{\begin{matrix}\Gamma \vdash e : T \\ T <: bool\end{matrix} \qquad \begin{matrix}\Gamma \vdash e_1 : T_1 \\ T_1 <: T'\end{matrix} \qquad \begin{matrix}\Gamma \vdash e_2 : T_2 \\ T_2 <: T'\end{matrix}}{\Gamma \vdash if\ e\ then\ e_1\ else\ e_2}$$

Statements → principalmente assegnamenti

$$\dfrac{\Gamma \vdash x : T \quad \Gamma \vdash e : T' \quad T = T'}{\Gamma \vdash x = e;\ : void}$$ ← omettiamo

$$\dfrac{\Gamma \vdash e : bool \quad \Gamma \vdash e_1 : T' \quad \Gamma \vdash e_2 : T'' \quad T' = void = T''}{\Gamma \vdash if(e)\ then\ e_1\ else\ e_2}$$

$$\dfrac{\Gamma \vdash s : T \qquad \Gamma \vdash S : T' \quad T = T'}{\Gamma \vdash s S}$$

↖ è lo stesso contesto! non modificano mai l'ambiente!

Succ.
- SubTyping per statements
- Classi e metodi

19 Giugno 2019

1\. $X := E$ [Asg]

$$\frac{\Gamma \vdash x : T_1 \quad \Gamma \vdash E : T_2 \quad T_2 <: T_1}{\Gamma \vdash x = E}$$

```
checkStat(){
 Node l = l.typecheck()
 Node r = r.typecheck()
 if( isSubtype(l,r)){
   return voidTypenode )
 else
   new exception()
```

if(equals(l,r)

2\. $[x \to C_x ,\ y \to C_y ,\ z \to C_z]$

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma(x)=C_x}{\Gamma \vdash x : C_x} \quad C_y <: C_x \quad \dfrac{\Gamma(x)=C_y}{\Gamma \vdash y : C_y}}{\Gamma \vdash x := y} \qquad \dfrac{C_z <: C_y \quad \Gamma \vdash y : C_y \quad \Gamma \vdash z : C_z}{\Gamma \vdash y := z}}{\Gamma \vdash x = y ;\ y := z ;} \qquad \dfrac{\dfrac{\Gamma(z) = C_z}{\Gamma \vdash z : C_z} \quad C <: C_z}{\Gamma \vdash z = new\ C();}}{\Gamma \vdash x := y ;\ y := z ;\ z = new\ C();}$$

$C <: C_z <: C_y <: C_x$

19 Feb. 20

Es. 2

1\. errore!

$$\frac{\Gamma \vdash e : T' \quad \Gamma \vdash e' : T'' \quad T' <: T}{\Gamma \vdash let\ T\ x = e\ in\ e' : T''}$$

x non esiste

let int x = 1 in { x = x+1 }

2\. errato

$$\frac{\Gamma \vdash e : T' \quad \Gamma[x \to T] \vdash e' : T'' \quad T <: T'}{\Gamma \vdash let\ T\ x = e\ in\ e' . T''}$$

let A x = new B() con A <: B

3\. errato

$$\frac{\Gamma \vdash e : T' \quad \Gamma[x : T'] \vdash e' : T'' \quad T' <: T}{\Gamma \vdash let\ T\ x = e\ in\ e' : T''}$$

let A x = new B() in {x metodoA()}

giusta:
$$\frac{\Gamma \vdash e : T' \quad \Gamma[x \to T] \vdash e' : T'' \quad T' <: T}{\Gamma \vdash let\ T\ x = e\ in\ e' : T''}$$

28 Maggio 2021

- dich. e var. + seq
- asg
- comandi + seq

(assum. solo int)

$$\frac{}{\Gamma, \overset{\text{val offset}}{n} \vdash num : int}$$

$$\frac{x \notin dom(top(\Gamma))}{\Gamma, n \vdash int\ x; : \Gamma[x \to n, int], n+1}\ [dec] \qquad \frac{\Gamma(x) : int}{\Gamma, n \vdash x : int}\ [var] \qquad \frac{\Gamma, n \vdash d : \Gamma', n' \quad \Gamma', n' \vdash D : \Gamma'', n''}{\Gamma, n \vdash dD : \Gamma'', n''}\ [SeqD]$$

$$\frac{\Gamma \vdash x : int \quad \Gamma, n \vdash e : int}{\Gamma, n \vdash x = e}\ [Asg] \qquad \frac{\Gamma, n \vdash s \quad \Gamma, n \vdash S}{\Gamma, n \vdash sS}\ [SeqS]$$

$$\frac{\overset{= : int \times int \to int}{\Gamma, n \vdash e_1 : int \quad \Gamma, n \vdash e_2 : int}}{\Gamma, n \vdash e_1 + e_2 : int}\ [Sum]$$

$$\frac{\emptyset \cdot [\,], 0 \vdash D : \Gamma, n \quad \Gamma, n \vdash S}{\emptyset, 0 \vdash D\ S}\ [Prog]$$

17 Sett. 2021

1.

$$\frac{\emptyset \cdot [\,] \vdash S}{\emptyset : S}\ [prog]$$

$$\frac{\overset{T_1 = T_2}{\Gamma \vdash x : T_1 \quad \Gamma \vdash e : T_2}}{\Gamma \vdash x = e : void}\ [Asg]$$

$$\frac{\Gamma \cdot [\,] \vdash dec : \Gamma' \quad \Gamma' \vdash stm}{\Gamma \vdash \{dec\ stm\}}\ (stm) \qquad \frac{\Gamma \vdash S \quad \Gamma \vdash S'}{\Gamma \vdash SS}$$

$$\frac{x \notin dom(top(\Gamma))}{\Gamma \vdash T\ x; : \Gamma[x \to T]} \qquad \frac{\Gamma \vdash d : \Gamma' \quad \Gamma' \vdash D : \Gamma''}{\Gamma \vdash dD : \Gamma''}$$

$$\frac{}{\Gamma \vdash num : int} \qquad \frac{}{\Gamma \vdash \underset{false\ :\ bool}{true : bool}}$$

$$\frac{\Gamma(x) : T}{\Gamma \vdash x : T}$$

```
prg : stm ;
stm : (Id '=' exp ';' | '{' dec stm '}')+ ;
dec : (type Id ';')+ ;
type: 'int' | 'bool' ;
exp : Int | Bool | Id | exp '+' exp | exp '-' exp | exp '>' exp | exp '==' exp
    | exp '||' exp | exp '&&' exp ;
```

$$\dfrac{\overset{+:\, int \times int \to int}{\Gamma \vdash e_1 : int \quad \Gamma \vdash e_2 : int}}{\Gamma \vdash e_1 + e_2 : int}$$
$-$

D [

S [

S [[

$$\dfrac{\overset{\&\&:\, bool \times bool \to bool}{\Gamma \vdash e_1 : bool \quad \Gamma \vdash e_2 : bool}}{\Gamma \vdash e_1 \,\&\&\, e_2 : bool}$$
$||$

$$\dfrac{\Gamma \vdash e_1 : T_1 \quad \Gamma \vdash e_2 : T_2 \quad T_1 = T_2 \quad == : T_1 \times T_2 \to bool}{\Gamma \vdash e_1 == e_2 : bool}$$

$$\dfrac{\dfrac{x \notin dom(top(\Gamma))}{\emptyset \cdot [\,] \vdash int\ x : \Gamma'_{x \to int}} \quad \dfrac{y \notin dom(top(\Gamma))}{\Gamma' \vdash int\ y : \Gamma''_{x \to int,\, y \to int}}}{\emptyset \cdot [\,] \vdash int\ x;\ int\ y : \Gamma''}$$

$$\dfrac{\Gamma'' \vdash x : int \quad \Gamma'' \vdash 5 : int}{\Gamma'' \vdash x = 5 : \Gamma''}$$

$$\dfrac{\emptyset \cdot [\,] \vdash int\ x;\ int\ y : \Gamma'' \quad \Gamma'' \vdash x = 5 : \Gamma''}{\emptyset \cdot [\,] \vdash int\ x;\ int\ y;\ x = 5 : \Gamma''}$$

$$\dfrac{z \notin dom(top(\Gamma))}{\Gamma'' \cdot [\,] \vdash bool\ z : \Gamma'''_{z \to bool}}$$

$$\dfrac{\dfrac{\Gamma(x) : int}{\Gamma''' \vdash x : int} \quad \Gamma''' \vdash 5 : int \ \ldots}{\Gamma''' \vdash x > 5 : bool}$$

$$\dfrac{\dfrac{\Gamma(z) : bool}{\Gamma''' \vdash z : bool} \quad \dfrac{\Gamma''' \vdash x > 5 : bool \quad \Gamma''' \vdash false : bool \quad || : bool \ldots}{\Gamma''' \vdash x > 5 \,||\, false : bool}}{\Gamma''' \vdash z = (x > 5) \,||\, false : \Gamma'''}$$

$$\dfrac{\Gamma'' \cdot [\,] \vdash bool\ z : \Gamma''' \quad \Gamma''' \vdash z = (x > 5) \,||\, false : \Gamma'''}{\Gamma'' \cdot [\,] \vdash bool\ z;\ z = (x > 5) \,||\, false; : \Gamma'''}$$

$$\dfrac{z \notin dom(top(\Gamma))}{\Gamma''' \cdot [\,] \vdash int\ z : \Gamma^{IV}_{z \to int}}$$

$$\dfrac{\dfrac{\Gamma(y) : int}{\Gamma^{IV} \vdash y : int} \quad \dfrac{\Gamma^{IV} \vdash 6 : int \quad \dfrac{\Gamma(x) : int}{\Gamma^{IV} : x : int}}{\Gamma^{IV} \vdash 6 + x : int} \ + \ldots}{\Gamma^{IV} \vdash y = 6 + x : \Gamma^{IV}}$$

$$\dfrac{\Gamma''' \cdot [\,] \vdash int\ z : \Gamma^{IV} \quad \Gamma^{IV} \vdash y = 6 + x : \Gamma^{IV}}{\Gamma''' \cdot [\,] \vdash int\ z;\ y = 6 + x : \Gamma}$$

$$\dfrac{\dfrac{\Gamma(z) : int}{\Gamma^{IV} \vdash z : int} \quad \dfrac{\Gamma^{IV} \vdash 3 : int \quad \dfrac{\Gamma(y) : int}{\Gamma^{IV} \vdash y : int}}{\Gamma^{IV} \vdash 3 + y}}{\Gamma^{IV} \vdash z = 3 + y :}$$

$$\dfrac{\Gamma''' \cdot [\,] \vdash int\ z;\ y = 6 + x : \Gamma \quad \Gamma^{IV} \vdash z = 3 + y :}{\Gamma''' \vdash \{int\ z;\ y = 6 + x;\ z = 3 + y\}}$$

$$\dfrac{\Gamma'' \cdot [\,] \vdash bool\ z;\ z = (x > 5) \,||\, false; : \Gamma''' \quad \Gamma''' \vdash \{int\ z;\ y = 6 + x;\ z = 3 + y\}}{\Gamma'' \vdash \{bool\ z;\ z = (x > 5) \,||\, false;\ z\}}$$

$$\dfrac{\emptyset \cdot [\,] \vdash int\ x;\ int\ y;\ x = 5 : \Gamma'' \quad \Gamma'' \vdash \{bool\ z;\ z = (x > 5) \,||\, false;\ z\}}{\emptyset \vdash \{int\ x;\ int\ y;\ x = 5;\ s\}}$$