

Semantica delle espressioni

La sintassi delle espressioni è la seguente:

$\langle \text{Exp} \rangle$
 $E \rightarrow A \mid B$
 $A \rightarrow I \mid N \mid A \text{ op } B$
 $B \rightarrow \text{true} \mid \text{false} \mid \text{not } B \mid B \text{ bop } B$

dove:

- A e B indicano rispettivamente espressioni aritmetiche e booleane;
- L'insieme op indica le operazioni $+, -, *$;
- L'insieme bop indica le operazioni $\text{or}, =$.

Rappresentazione Le metaviariabili sono le seguenti:

- Espressioni ε : e
- Numeri N : n, m, p
- Booleani B : t

Valutazione La valutazione delle espressioni è una funzione $\text{Eval} : \varepsilon \rightarrow N \cup B$ che descrive il comportamento dinamico delle espressioni restituendo il valore in cui esse sono valutate:

$$(\text{Eval}(e) = k) \longleftrightarrow (e \rightarrow^* k)$$

Equivalenza L'equivalenza di espressioni è una relazione $\equiv \subseteq \varepsilon \times \varepsilon$ definita come:

$$(e_1 \equiv e_2) \longrightarrow (\text{Eval}(e_1) = \text{Eval}(e_2))$$

Regole

A questo punto introduciamo le regole di transizione. Facciamo attenzione a distinguere tra il simbolo sintattico di una operazione (bop in grassetto rosso) e il simbolo dell'operazione nella macchina sottostante (bop normale). di questo non ci interessa l'implementazione, ma solo che lo possiamo usare per interpretare il simbolo del linguaggio. Qui eseguiamo astrazione. Partiamo dalle regole necessarie per valutare espressioni aritmetiche.

$$\mathcal{E}_1: m \text{ op } n \rightarrow p \quad \text{se } m \text{ op } n = p, \\ m, n, p \in \mathcal{N}$$

La prima regola è un assioma, che semplicemente valuta l'espressione sintattica contenente un operatore aritmetico, nel valore che esso rappresenta. Ad esempio, $3+5$ è una scrittura sintattica dove il simbolo 3 rappresenta il valore 3 , il simbolo 5 rappresenta il simbolo 5 , e quindi interpretando il simbolo della somma come addizione restituiamo il simbolo 8 che rappresenta il valore 8 .

$$\mathcal{E}_2: \frac{e \rightarrow e'}{e \text{ op } e_0 \rightarrow e' \text{ op } e_0}$$

La seconda regola è invece una regola induttiva di valutazione. Questa regola mi dice che se gli operandi non sono valori primitivi allora dobbiamo valutarli. In particolare, questa regola mi dice che prima valutiamo l'espressione a sinistra dell'operatore. La regola successiva stabilisce invece che, nel momento in cui l'operando a sinistra è un valore allora posso iniziare a valutare l'operando a destra.

Chiaro che quando anche l'operatore a destra è un valore allora ricadiamo nell'assioma e possiamo restituire il valore finale.

$$\mathcal{E}_3: \frac{e \rightarrow e'}{m \text{ op } e \rightarrow m \text{ op } e'}$$

Continuano quindi ad introdurre le regole, ora vediamo quelle delle espressioni booleane molto simili a quelle aritmetiche, con la sola aggiunta dell'operatore unario.

$$\mathcal{E}_4: t_1 \text{ bop } t_2 \rightarrow t \quad \text{se } t_1 \text{ op } t_2 = t, \\ t_1, t_2, t \in \mathcal{B}$$

Anche questa regola è un assioma, che semplicemente valuta l'espressione sintattica contenente un operatore booleano, nel valore che esso rappresenta. Ddizione restituiamo il simbolo 8 che rappresenta il valore 8.

La regola successiva è esattamente la regola \mathcal{E}_3 dove ammettiamo che l'espressione contenga operatori binari booleani. Anche in questo caso fissiamo l'ordine di valutazione da sinistra verso destra.

$$\mathcal{E}_{3'}: \frac{e \rightarrow e'}{e \text{ bop } e_0 \rightarrow e' \text{ bop } e_0}$$

La regola successiva stabilisce invece che, nel momento in cui l'operando a sinistra è un valore booleano allora posso iniziare a valutare l'operando a destra. Chiaro che, nuovamente, quando anche l'operatore a destra è un valore booleano allora ricadiamo nell'assioma \mathcal{E}_4 e possiamo restituire il valore finale.

$$\mathcal{E}_5: \frac{e \rightarrow e'}{t \text{ op } e \rightarrow t \text{ op } e'}$$

Nel caso booleano, dobbiamo aggiungere la regola per l'operatore unario. Per prima cosa l'assioma che restituisce il valore corrispondente all'applicazione dell'operatore sul valore rappresentato, e poi la regola di valutazione, analoga alle precedenti.

$$\mathcal{E}_6: \text{not } t_1 \rightarrow t \quad \text{se } \text{not } t_1 = t, t_1 \in \mathcal{B}$$

$$\mathcal{E}_7: \frac{e \rightarrow e'}{\text{not } e \rightarrow \text{not } e'}$$

Esempio di applicazione delle regole:

$$(2 + 3) * (5 - (1 + 4))$$

Poichè l'operatore * non ha numeri né a destra né a sinistra non è possibile applicare la regola \mathcal{E}_1 . Applico quindi le altre due regole:

Applico la regola \mathcal{E}_2 :

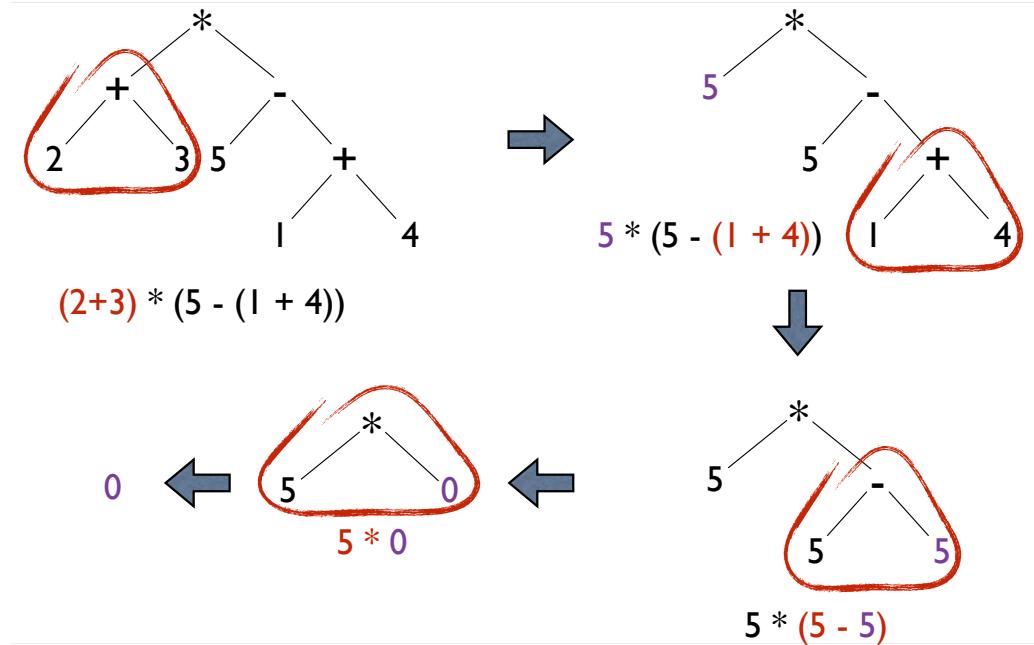
$$\frac{2 + 3 \rightarrow 5}{(2 + 3) * (5 - (1 + 4)) \rightarrow 5 * (5 - (1 + 4))}$$

Applico la regola \mathcal{E}_3 :

$$\frac{\overline{\begin{array}{c} 1+4 \rightarrow 5 \\ 5-(1+4) \rightarrow 5-5 \end{array}}}{5 * (5 - (1 + 4)) \rightarrow 5 * 5 - 5}$$

Applico la regola \mathcal{E}_3 :

$$\frac{5 - 5 \rightarrow 0}{5 * 5 - 5 \rightarrow 5 * 0}$$



In gran parte dei linguaggi le espressioni devono essere risolte da destra verso sinistra. In specifici linguaggi è possibile anche il contrario (non nel nostro caso).

Valutazione delle espressioni La valutazione delle espressioni è una funzione $Eval : \varepsilon \rightarrow N \cup B$ (numeri unito a booleani) che descrive il comportamento dinamico delle espressioni restituendo il valore in cui esse sono valutate:

$$(Eval(e) = k) \longleftrightarrow (e \rightarrow^* k)$$

Equivalenza di espressioni L'equivalenza di espressioni è una relazione $\equiv \subseteq \varepsilon \times \varepsilon$ definita come segue:

$$(e_0 \equiv e_1) \longleftrightarrow (Eval(e_0) = Eval(e_1))$$

Quindi due espressioni sintatticamente diverse sono uguali quando rappresentano lo stesso valore (hanno lo stesso significato).

Dichiarazioni

Le dichiarazioni sono la categoria sintattica che crea e modifica gli ambienti. Le dichiarazioni devono essere elaborate affinché possano operare.

Le modifiche effettuate sono reversibili, in quanto valide solo all'interno dello scope dell'identificatore. Quindi è possibile avere più identificatori con lo stesso nome. Possono avere side-effects.

Identificatori

Un identificatore è una stringa di caratteri usata per denotare "altro". Un identificatore può denotare più elementi, mentre un elemento può essere denotato da più identificatori diversi (aliasing).

A seconda del linguaggio, la scelta del nome dell'identificatore può essere vincolato a delle regole:

- Utilizzo del case-sensitive;
- Presenza di parole riservate;
- Possibili vincoli di lunghezza;
- Vincolo nella presenza di caratteri speciali.

Rappresentazione Identificatori *Id*: *id, x*.

Bindings

Il concetto di bindings viene ereditato dalla matematica. Per prima cosa si dichiarano l'identificatore e il suo significato, ovvero si crea il legame nelle occorrenze di bindings (tipo "sia x qualcosa" in matematica):

$$\left(\sum_{i=1}^n \left(\sum_{j=1}^m a_{ij} \dots b_{jk} \right) \right)$$

Binding occurrences (Definizione = creazione binding): Viene definito il **nome** (identificatore) e il suo **significato** (denotazione)

Se il nome è associato ad un significato, allora ogni sua occorrenza è un'occorrenza applicata:

$$\left(\sum_{i=1}^n \left(\sum_{j=1}^m a_{ij} \dots b_{jk} \right) \right)$$

Applied occurrences (Uso = applicazione binding): Viene usato il **nome** (identificatore) per accedere al suo **significato** (denotazione)

Se non viene definito il significato dell'identificatore, allora ogni sua occorrenza è un'occorrenza libera (può rappresentare qualunque cosa):

$$\left(\sum_{i=1}^n \left(\sum_{j=1}^m a_{ij} \dots b_{jk} \right) \right)$$

Free occurrences (Libere = non legate): Viene usato un **nome** (identificatore) il cui **significato** (denotazione) non è stato definito

La differenza tra occorrenze libere e applicate consiste nel raggio di azione dell'occorrenze

$$\left(\sum_{i=1}^n \left(\sum_{j=1}^m a_{ij} \dots b_{jk} \right) \right)$$

Scope per j

Scope per i

Scope di un binding
(*Raggio di azione di una definizione*): Definisce lo spazio in cui si può usare un nome per rappresentare il significato associato da un binding

Esistono quindi tre possibili tipo di occorrenza di un identificatore:

- Occorrenza di definizione (binding occurrence), che si ha quando si definisce (o ridefinisce) il significato dell'identificatore;
- Occorrenza applicata, che si ha quando si utilizza l'identificatore per riferirsi al suo significato;
- Occorrenza libera, che si ha quando l'identificatore non si trova nello scope della sua definizione.

Tipi di bindings Esistono due tipi di bindings:

- Binding statico -> il legame non può variare durante l'esecuzione (es: tipo di una variabile);
- Binding dinamico -> il legame può variare durante l'esecuzione