Università degli Studi di Verona

DIPARTIMENTO DI INFORMATICA

Linguaggi di programmazione

Riassunto dei principali argomenti

Autore:
Davide Bianchi

Anno Accademico 2017/2018

Indice

1 Introduzione

Un linguaggio di programmazione è composto da:

- *Sintassi:* insieme di regole di scrittura del linguaggio;
- *Semantica:* descrizione del comportamento del programma a tempo di esecuzione;
- *Pragmatica:* descrizione delle caratteristiche del linguaggio, delle sue funzionalità ecc.

Gli stili per dare la semantica di un linguaggio sono 3:

- *Operazionale*: la semantica è data con sistemi di transizione, fornendo i passi della computazione passo passo;
- *Denotazionale*: il significato di un programma è dato dalla struttura di un insieme;
- *Assiomatica*: il significato è dato attraverso regole assiomatiche o qualche tipo di logica.

2 Esempio di linguaggio basilare

La semantica operazionale di un linguaggio è data attraverso un sistema di regole di inferenza, date come segue:

$$(Assioma)\ \frac{-}{(Conclusione)} \qquad (Regola)\ \frac{(Hyp_1)\ (Hyp_2)\ ...\ (Hyp_n)}{(Conclusione)}$$

Introduciamo la sintassi di un linguaggio basato solo su espressioni aritmetiche:

$$E := \ n \mid E \mid E + E \mid E * E \ ...$$

La valutazione di programmi generati con questa sintassi può procedere in due modi:

- *Small step:* la semantica fornisce il sistema per procedere nell'esecuzione, passo dopo passo;
- *Big step:* la semantica va subito al risultato finale.

2.1 Semantica big-step

Forniamo la semantica big-step per il linguaggio dato sopra:

$$\text{B-Num}\frac{-}{n \Downarrow n} \qquad \text{B-Add}\frac{E_1 \Downarrow n_1 \ E_2 \Downarrow n_2}{E_1 + E_2 \Downarrow n_3}\ n_3 = add(n_1, n_2)$$

2.1.1 Esempio

$$\text{B-Add}\frac{\text{B-Num}\ \dfrac{-}{3 \Downarrow 3} \qquad \text{B-Add}\ \dfrac{\text{B-Num}\ \dfrac{-}{2 \Downarrow 2} \qquad \text{B-Num}\ \dfrac{-}{1 \Downarrow 1}}{2 + 1 \Downarrow 3}}{3 + (2 + 1) \Downarrow 6}$$

2.2 Semantica small-step

Indichiamo con $E_1 \twoheadrightarrow E_2$ lo svolgimento di un solo passo di semantica.

$$\text{S-Left}\frac{E_1 \twoheadrightarrow E_1'}{E_1 + E_2 \twoheadrightarrow E_1' + E_2}$$

$$\text{S-N.Right}\frac{E_2 \twoheadrightarrow E_2'}{n_1 + E_2 \twoheadrightarrow n_1 + E_2'}$$

$$\text{S-Add}\frac{-}{n_1 + n_2 \twoheadrightarrow n_3} n_3 = add(n_1, n_2)$$

Con queste regole l'ordine di valutazione degli statement è fisso, procede sempre da sinistra verso destra. Diamo un'alternativa:

$$\text{S-Left}\frac{E_1 \twoheadrightarrow_{ch} E_2}{E_1 + E_2 \twoheadrightarrow_{ch} E_1' + E_2}$$

$$\text{S-Right}\frac{E_2 \twoheadrightarrow_{ch} E_2'}{E_1 + E_2 \twoheadrightarrow_{ch} E_1 + E_2'}$$

$$\text{S-Add}\frac{-}{n_1 + n_2 \twoheadrightarrow_{ch} n_3} n_3 = add(n_1, n_2)$$

In questo caso l'ordine di valutazione è arbitrario. La notazione utilizzata in generale è la seguente:

- la relazione $\twoheadrightarrow^k$, con $k \in \mathbb{N}$, indica una sequenza di n passi applicando la semantica small-step;
- la relazione $\twoheadrightarrow^*$, indica una sequenza di derivazione lunga un certo numero di passi. Questa relazione è riflessiva ed è la chiusura transitiva di $\twoheadrightarrow$.

3 Linguaggio imperativo

Definiamo la sintassi di un semplice linguaggio imperativo:

$$\begin{aligned} b &:= true \mid false \\ n &:= \{... -1, 0, 1, 2, ...\} \\ l &:= \{l_0, l_1, ...\} \\ op &:= + \mid \geq \\ e &:= n \mid b \mid e\ op\ e \mid \text{if } e \text{ then } e \text{ else } e \mid l := e \mid !l \mid skip \mid e; e \mid \text{while } e \text{ do } e \end{aligned}$$

Nota: lo statement $!l$ indica l'intero memorizzato al momento alla locazione l. Inoltre il linguaggio non è tipato, quindi sono ammesse le sintassi come $2 \geq true$.

3.1 Memoria

La memoria è necessaria per poter valutare gli statement di lettura da una locazione. In particolare definiamo

$$\begin{aligned} dom(f) &= \{a \in A \mid \exists b \in B \ s.t.\ f(a) = b \\ ran(f) &= \{b \in B \mid \exists a \in A \ s.t.\ f(a) = b \end{aligned}$$

Lo store del linguaggio imperativo in questione è un insieme di funzioni parziali che vanno dalle locazioni di memoria nei numeri interi:

$$s : \mathbb{L} \to \mathbb{Z}$$

L'aggiornamento della memoria funziona come segue:

$$s[l \to n](l') = \begin{cases} n & \text{if } l = l' \\ s(l') & \text{altrimenti} \end{cases}$$