Semantica (pt. 2)

Poichè il significato di un programma può essere guardato da punti di vista diversi, si sono sviluppate diverse tipologie di analisi semantica:

- Semantica denotazionale: descrive le funzionalità studiando gli effetti dell'esecuzione e cercando le proprietà del programma a partire da quelle della funzione calcolata;
- Semantica assiomatica: descrive le proprietà, in particolare serve per dare deduzione da assiomi dati;
- Semantica operazionale: descrive le trasformazioni di stato occupandosi di come i risultati finali vengono prodotti (permette l'implementazione di un interprete).

In ogni caso, tutte e tre le semantiche sono equivalenti.

Semantica denotazionale

La semantica denotazionale consiste in un modello matematico dei programmi basato sulla ricorsione. Si tratta della semantica più astratta con cui descrivere i programmi.

La sua costruzione consiste nel definire un oggetto matematico per ogni entità del linguaggio e nel definire poi una funzione che mappa istanze delle entità del linguaggio in istanze dei corrispondenti oggetti matematici.

Generalmente un programma consiste in una funzione del tipo

$$E : Prog \rightarrow [(VarxVal) \rightarrow (VarxVal)]$$

L'equivalenza dei programmi si dimostra mediante l'equivalenza tra funzioni.
La semantica denotazionale, quindi, descrive gli effetti operando su oggetti matematici.

Bisogna definire la funzione di valutazione $\mathcal{E}[\![P]\!]\sigma$, che valuta il programma P sulla memoria σ, restituendo σ', la memoria σ modificata da P

```
P:
    z := 2;
    y := z;
    y := y+1;
    z := y;
```

$$\mathcal{E}[\![P]\!][z=\bot, y=\bot] =$$
$$(\mathcal{E}[\![z:=y]\!] \circ \mathcal{E}[\![y:=y+1]\!] \circ \mathcal{E}[\![y:=z]\!] \circ \mathcal{E}[\![z:=2]\!])[z=\bot, y=\bot] =$$
$$(\mathcal{E}[\![z:=y]\!](\mathcal{E}[\![y:=y+1]\!](\mathcal{E}[\![y:=z]\!](\mathcal{E}[\![z:=2]\!])[z=\bot, y=\bot])))$$

$[z=2, y=\bot]$

$[z=2, y=2]$

$[z=2, y=3]$

$[z=3, y=3]$

Semantica assiomatica

La semantica assiomatica consiste in un modello matematico dei programmi basato sulla logica formale (calcolo dei predicati).
Si basa su assiomi e regole di inferenza, fornite per ogni tipo di costrutto del linguaggio.
Le espressioni logiche della semantica sono chiamate asserzioni:

- L'asserzione prima di un comando (precondizione) dichiara le relazioni e i vincoli validi prima dell'esecuzione del comando;
- L'asserzione che segue un comando è una post-condizione;
- Una weakest precondition è la precondizione meno restrittiva che garantisce la post-condizione.

Attraverso queste asserzioni, la semantica assiomatica permette di dimostrare proprietà parziali di correttezza: quando lo stato iniziale rispetta la precondizione e il programma termina, allora lo stato finale soddisfa la postcondizione.

Poichè questo tipo di semantica considera solo gli aspetti descritti dalle pre e post condizioni, possono esistere ininiti programmi che le soddisfano, avendo però comportamenti potenzialmente diversi.

Notazione: {P} statement {Q}. Questa semantica si calcola mediante un sistema di prova: abbiamo una tripla da verificare e cerco di dimostrarla applicando le regole.

$$\frac{\{P\}\,S\,\{Q\},\ P' \Rightarrow P,\ Q \Rightarrow Q'}{\{P'\}\,S\,\{Q'\}}$$

$$\frac{\{P1\}\ S1\ \{P2\},\ \{P2\}\ S2\ \{P3\}}{\{P1\}\ S1;\ S2\ \{P3\}}$$

$$\frac{\{B \text{ and } P\}\ S1\ \{Q\},\ \{(\text{not } B) \text{ and } P\}\ S2\ \{Q\}}{\{P\}\ \texttt{if}\ B\ \texttt{then}\ S1\ \texttt{else}\ S2\ \{Q\}}$$

$$\frac{(I \text{ and } B)\ S\ \{I\}}{\{I\}\ \text{while } B \text{ do } S\ \{I \text{ and } (\text{not } B)\}}$$

```
P:
    z := 2;
    y := z;
    y := y+1;
    z := y;
```

precondizione { z=y }
z:=2;
y:=z;
precondizione { z=y } postcondizione
y:=y+1;
z:=y;
{ z=y } postcondizione

Semantica operazionale

La semantica operazionale consiste in un modello matematico dei programmi basato sui sistemi di transizione.

Descrive il significato del programma eseguendo i suoi comandi su una macchina, simulata o reale, dove il cambiamento di stato definisce il significato del contenuto.

Per il calcolo dei risultati finali, utilizza il modello matematico dei sistemi di transizione.

Esempio iniziale:

```
P:
    z := 2;
    y := z;
    y := y+1;
    z := y;
```

<z:=2;y:=z;y:=y+1;z:=y,[z=⊥,y=⊥]>

<y:=z;y:=y+1;z:=y,[z=2,y=⊥]>

<y:=y+1;z:=y,[z=2,y=2]>

<z:=y,[z=2,y=3]>

<ε,[z=3,y=3]>

Composizionalità

La composizionalità è una proprietà della semantica necessaria per caratterizzare i comportamenti e significati di sistemi che possono avere infiniti elementi. Indica che il significato di ogni programma deve essere funzione del significato dei costituenti immediati. Queso principio è rispettato, di natura, dalla semantica denotazionale e assiomatica.

La composizione è molto importante per l'analisi di software di grandi dimensioni. Infatti, se la semantica su cui si basa l'analisi è composizionale, è possibili scomporre il software in moduli più piccoli, studiarli singolarmente e poi ricomporre i risultati per ottenere quello globale.