Fondamenti dell'informatica

Semantica del linguaggio WHILE

Simona Ronchi Della Rocca

Il linguaggio WHILE è generato dalla seguente grammatica:

$$
\begin{array}{rcl}
< \mathit{variabile} > & ::= & \mathbf{X} < \mathit{indice} > \\
< \mathit{indice} > & ::= & < \mathit{digit} > | < \mathit{indice} > < \mathit{digit} > \\
< \mathit{assegnazione} > & ::= & < \mathit{variabile} > := \mathbf{0} \; | < \mathit{variabile} > := \mathbf{succ}(< \mathit{variabile} >) \; | \\
 & & < \mathit{variabile} > := \mathbf{pred}(< \mathit{variabile} >) \\
< \mathit{test} > & ::= & < \mathit{variabile} > \neq < \mathit{variabile} > \\
< \mathit{istruzione} > & ::= & < \mathit{assegnazione} > | \; \mathbf{while} < \mathit{test} > \mathbf{do} < \mathit{istruzione} > | \\
 & & \mathbf{while} < \mathit{test} > \mathbf{do} < \mathit{programma} > \\
< \mathit{sequenza} > & ::= & < \mathit{istruzione} > | < \mathit{istruzione} > ; < \mathit{sequenza} > \\
< \mathit{programma} > & ::= & \mathbf{begin\ end} \; | \; \mathbf{begin} < \mathit{sequenza} > \mathbf{end}
\end{array}
$$

dove le parole tra parentesi acute rappresentano i simboli non terminali, e quelle in grassetto i simboli terminali.

Assumiamo, nel seguito, che in un programma con k variabili queste siano denominate in sequenza $X1, ..., Xk$.

Il significato di un programma (semantica) dipende sia dal testo del programma che dal valore delle sue variabili. I valori delle variabili saranno memorizzati in un vettore di stato, che è una sequenza di numeri naturali $(a_1, ..., a_k)$, dove a_i è il valore della variabile Xi. $\vec{a}$ denoterà un generico vettore di stato, e a_i la sua i-esima componente.

La semantica di un programma WHILE verrà definita in modo S.O.S. (Semantica Operazionale Strutturata), cioè tramite un insieme di regole. Una regola di S.O.S. è della forma:

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

e significa: "una istanza dell'asserzione A è vera, se sono vere le corrispondenti istanze di $A_1, .., A_n$". $A_1, .., A_n$ sono le "premesse" della regola, e può essere che $n = 0$: in questo caso la regola è un'assioma, e ogni istanza di R è sempre vera. Le regole che definiscono la semantica dei programmi WHILE dimostrano asserzioni del tipo:

$$\vec{a}, P \Downarrow \vec{b}$$

dove P è un programma, $\vec{a}$ e $\vec{b}$ sono due vettori di stato la cui lunghezza è maggiore o uguale al numero di variabili di P. $\vec{a}, P \Downarrow \vec{b}$ può essere letto come "il programma P, se inizia l'esecuzione con il vettore di stato $\vec{a}$, termina e alla fine dell'esecuzione il vettore di stato è stato trasformato in $\vec{b}$".

Le regole sono:

$$\frac{}{\vec{a}, \mathbf{begin\ end} \Downarrow \vec{a}}$$

$$\frac{n \geq 1 \quad \vec{a}, \mathbf{begin}\ \delta_1\ \mathbf{end} \Downarrow \vec{a'} \quad \vec{a'}, \mathbf{begin}\ \delta_2; ...; \delta_n\ \mathbf{end} \Downarrow \vec{b}}{\vec{a}, \mathbf{begin}\ \delta_1; ...; \delta_n\ \mathbf{end} \Downarrow \vec{b}}$$

$$\frac{}{(a_1,...,a_i,a_{i+1},...,a_k),\textbf{begin Xi} := \mathbf{0} \textbf{ end} \Downarrow (a_1,...,0,a_{i+1},...,a_k)}$$

$$\frac{}{(a_1,...,a_i,a_{i+1},...,a_k),\textbf{begin Xi} := \textbf{succ(Xj) end} \Downarrow (a_1,...,a_j+1,a_{i+1},...,a_k)}$$

$$\frac{}{(a_1,...,a_i,a_{i+1},...,a_k),\textbf{begin Xi} := \textbf{pred(Xj) end} \Downarrow (a_1,...,a_j-1,a_{i+1},...,a_k)}$$

(dove $-$ denota la sottrazione aritmetica)

$$\frac{a_i \neq a_j \quad \vec{a},\textbf{begin } \vec{\delta} \textbf{ end} \Downarrow \vec{b} \quad \vec{b},\textbf{begin while Xi} \neq \textbf{Xj do } \vec{\delta} \textbf{ end} \Downarrow \vec{c}}{\vec{a},\textbf{begin while Xi} \neq \textbf{Xj do } \vec{\delta} \textbf{ end} \Downarrow \vec{c}}$$

$$\frac{a_i = a_j}{\vec{a},\textbf{begin while Xi} \neq \textbf{Xj do } \vec{\delta} \textbf{ end} \Downarrow \vec{a}}$$

Se, dati $\vec{a}$ e P, non esiste una dimostrazione di $\vec{a}, P \Downarrow \vec{b}$, per qualche $\vec{b}$, il programma P non termina sul vettore di stato $\vec{a}$, e lo denoteremo con $\vec{a}, P \Uparrow$.

Esempio 1. Il programma $\mathbf{P_1} = \textbf{begin while X1} \neq \textbf{X2 do X1} := \textbf{succ(X1) end}$, sul vettore di stato $(1,2)$, termina e trasforma il vettore di stato in $(2,2)$. Infatti:

$$\frac{1 \neq 2 \quad \mathbf{R} \quad \dfrac{2=2}{(2,2),\textbf{begin while X1} \neq \textbf{X2 do X1} := \textbf{succ(X1) end} \Downarrow (2,2)}}{(1,2),\textbf{begin while X1} \neq \textbf{X2 do X1} := \textbf{succ(X1) end} \Downarrow (2,2)}$$

dove $\mathbf{R}$ denota la derivazione:

$$\frac{}{(1,2),\textbf{begin X1} := \textbf{succ(X1) end} \Downarrow (2,2).}$$

Esempio 2.

Cerchiamo di trovare una derivazione che dimostri:
$(3,2),\textbf{begin while X1} \neq \textbf{X2 do X1} := \textbf{succ(X1) end} \Downarrow \vec{b}$, per qualche $\vec{b}$.

$$\frac{3 \neq 2 \quad \mathbf{R'} \quad \dfrac{4 \neq 2 \quad R'' \quad \dfrac{...}{R'''}}{(4,2),\textbf{begin while X1} \neq \textbf{X2 do X1} := \textbf{succ(X1) end} \Downarrow \vec{b}}}{(3,2),\textbf{begin while X1} \neq \textbf{X2 do X1} := \textbf{succ(X1) end} \Downarrow \vec{b}}$$

dove $\mathbf{R}'$ è:

$(3,2)$, **begin X1** $:=$ **succ(X1) end** $\Downarrow (4,2)$ E' evidente che la ricerca non finisce

mai, perchè ogni volta la prima componente del vettore cresce, e non diventerà mai uguale alla seconda. Quindi possiamo affermare
$(3,2)$, **while X1** $\neq$ **X2 do X1** $:=$ **succ(X1) end** $\Uparrow$.

Teorema Sia **P** un programma con k variabili, e sia $\vec{a}$ un vettore di naturali di lunghezza k. Per ogni vettore di naturali $\vec{b}$, se $\vec{ab}$ è la concatenazione dei due vettori, $\vec{a} \Downarrow \vec{c}$ sse $\vec{ab} \Downarrow \vec{c}$, $\vec{a} \Uparrow$ sse $\vec{ab} \Uparrow$.

Ogni programma calcola infinite funzioni, una per ogni arità, secondo la definizione seguente.

Dato un programma WHILE P (con k variabili) e un numero naturale $n \geq 1$, la funzione ψ, di arita' n, calcolata da P, è definita nel modo seguente:

- Per ogni sequenza di naturali $(a_1, ..., a_n)$:

$n \leq k$ Se $(a_1, ..., a_n, \underbrace{0, ..., 0}_{k-n})P \Downarrow \vec{b}$, allora $\psi(a_1, ..., a_n) = b_1$, dove b_1 è il primo elemento della sequenza $\vec{b}$.
Se $(a_1, ..., a_n, \underbrace{0, ..., 0}_{k-n})P \Uparrow$, allora $\psi(a_1, ..., a_n)$ è indefinita;

$n > k$ Se $(a_1, ..., a_k, a_{k+1}, .., a_n)P \Downarrow \vec{b}$, allora $\psi(a_1, ..., a_k, a_{k+1}, ..., a_n) = b_1$, dove b_1 è il primo elemento della sequenza $\vec{b}$. Si noti che, per il teorema precedente, il risultato non dipende dai valori $a_{k+1}, .., a_n$, quindi anche la funzione ψ non dipendera' dai valori dei parametri dal $k+1$-esimo in poi.
Se $(a_1, ..., a_k, a_{k+1}, .., a_n)P \Uparrow$, allora $\psi(a_1, ..., a_k, a_{k+1}, ..., a_n)$, per ogni $a_{k+1}, ..., a_n$, è indefinita.

Esempio 3
Sia P_1 il programma dell'esempio 1, che ha 2 variabili. E' facile vedere che:

1. Per ogni s, $(0, 0, \underbrace{0, ..., 0}_{s}), P_1 \Downarrow (0, 0, \underbrace{0, ..., 0}_{s})$;
2. Per ogni s, $(n, m, \underbrace{0, ..., 0}_{s}), P_1 \Downarrow (m, m, \underbrace{0, ..., 0}_{s})$, se $n \leq m$;
3. Per ogni s, $(n, m, \underbrace{0, ..., 0}_{s}), P_1 \Uparrow$, se $n > m$;

Quindi le funzioni calcolate sono:

$$\psi^{(1)}(n) = \begin{cases} 0 & \text{se } n = 0 \\ \bot & \text{altrimenti} \end{cases}$$

$$\psi^{(p+1)}(n_1, ..., x_{p+1}) = \begin{cases} n_2 & \text{se } n_1 \leq n_2 \\ \perp & \text{altrimenti} \end{cases}$$