Fondamenti dei linguaggi di programmazione

Aniello Murano
Università degli Studi di Napoli
"Federico II"

Semantica denotazionale del comando while di IMP

Denotazione di Com (1)

- Ricordiamo la sintassi dei comandi Com di IMP

  c ::= skip | X:=a | $c_0;c_1$ | if b then c_0 else c_1 | while b do c
- La semantica denotazionale per $c \in$ Com è una funzione

  $$\mathcal{C}[\![c]\!] : \Sigma \rightharpoonup \Sigma$$
- La denotazione di $c \in$ Com (semantica denotazionale $\mathcal{C}[\![c]\!]$ per c) è una relazione tra stati, definita per induzione sulla struttura dei comandi.
- Si noti che la funzione di valutazione è parziale $\mathcal{C}[\![c]\!] : \Sigma \rightharpoonup \Sigma$ in quanto su alcuni comandi la funzione può non essere definita (per esempio sui loop infiniti)

Denotazione di Com (2)

- $\mathcal{C}[\![skip]\!]$ = $\{(\sigma, \sigma) \mid \sigma \in \Sigma\}$
- $\mathcal{C}[\![X{:=}a]\!]$ = $\{(\sigma, \sigma[n/X]) \mid \sigma \in \Sigma \;\&\; \mathcal{A}[\![a_0]\!]\sigma = n\}$
- $\mathcal{C}[\![c_0;c_1]\!]$ = $\mathcal{C}[\![c_1]\!] \circ \mathcal{C}[\![c_0]\!]$ (si noti l'inversione in accordo alla regola di composizione)
- $\mathcal{C}[\![\text{if } b \text{ then } c_0 \text{ else } c_1]\!] = \{(\sigma, \sigma') \mid \mathcal{B}[\![b]\!]\sigma = true \;\&\; (\sigma, \sigma') \in \mathcal{C}[\![c_0]\!]\} \cup \{(\sigma, \sigma') \mid \mathcal{B}[\![b]\!]\sigma = false \;\&\; (\sigma, \sigma') \in \mathcal{C}[\![c_1]\!]\}$
- Particolarmente difficile è invece la valutazione del comando while per il quale è necessario utilizzare i concetti matematici del punto fisso introdotti nella lezione precedente.

Denotazione del while

- Dalla semantica operazionale (II lezione) abbiamo osservato che esiste la seguente equivalenza:
- Sia w $\equiv$ while b do c allora

  w $\sim$ if b then c;w else skip
- Possiamo allora scriver la semantica del comando while utilizzando le regole precedenti nel seguente modo:
- $\mathcal{C}[\![w]\!] = \mathcal{C}[\![\text{if b then c;w else skip}]\!] =$

  $\{(\sigma,\sigma') \mid \mathcal{B}[\![b]\!]\sigma = \text{true} \ \& \ (\sigma,\sigma') \in \mathcal{C}[\![w]\!] \circ \mathcal{C}[\![c]\!]\} \cup$
  $\{(\sigma,\sigma) \mid \mathcal{B}[\![b]\!]\sigma = \text{false}\} =$

  $\{(\sigma, \sigma') \mid \exists\, \sigma''.\ \mathcal{B}[\![b]\!]\sigma = \text{true} \ \& \ (\sigma, \sigma'') \in \mathcal{C}[\![c]\!] \ \& \ (\sigma'', \sigma') \in \mathcal{C}[\![w]\!]\} \cup$
  $\{(\sigma, \sigma) \mid \mathcal{B}[\![b]\!]\sigma = \text{false}\}$
- Come si vede, il termine w compare in entrambi i lati dell'uguaglianza (**equazione ricorsiva**). Dunque $\mathcal{C}[\![w]\!]$ non ha una soluzione immediata.
- Risolvere questa funzione equivale a calcolare un punto fisso di una funzione $\mathcal{C}[\![w]\!]=f(\mathcal{C}[\![w]\!])$.

Alcune osservazioni su f($\mathcal{C}[\![w]\!]$).

- $\mathcal{C}[\![w]\!] = \mathcal{C}[\![\text{if b then c;w else skip}]\!] =$

  $\{(\sigma, \sigma') \mid \exists\, \sigma''.\ \mathcal{B}[\![b]\!]\sigma = \text{true} \ \& \ (\sigma, \sigma'') \in \mathcal{C}[\![c]\!] \ \& \ (\sigma'', \sigma') \in \mathcal{C}[\![w]\!]\}$
  $\cup \{(\sigma, \sigma) \mid \mathcal{B}[\![b]\!]\sigma = \text{false}\}$
- Risolvere questa equazione equivale a calcolare un punto fisso di una funzione $\mathcal{C}[\![w]\!]=f(\mathcal{C}[\![w]\!])$.
- $\mathcal{C}[\![w]\!]$ è una unzione parziale $\mathcal{C}[\![w]\!]: \Sigma \rightharpoonup \Sigma$.
- $\mathcal{C}[\![c]\!]$ è una unzione parziale $\mathcal{C}[\![c]\!]: \Sigma \rightharpoonup \Sigma$.
- Dunque, f è ottenuta eseguendo prima $\mathcal{C}[\![c]\!]$ e poi $\mathcal{C}[\![w]\!]$. Per cui, f è una funzione f: $(\Sigma \rightharpoonup \Sigma) \to (\Sigma \rightharpoonup \Sigma)$. Questa osservazione chiarirà il formalismo utilizzato nelle prossime diapositive.

Idea di valutazione con fixpoint

- Si consideri la concatenazione del comando X:=0 e del comando w = while $X \leq 10$ do X:=X+1
- La denotazione di x:=0; w è $\mathcal{C}[\![x{:=}0;\ w]\!]\sigma = \mathcal{C}[\![w]\!]\sigma \circ \mathcal{C}[\![x{:=}0]\!]\sigma$.
- Sappiamo che $\mathcal{C}[\![x{:=}0]\!]\sigma = \{(\sigma, \sigma[0/X])\}$
- Per $\mathcal{C}[\![w]\!]\sigma$ occorrono 12 valutazioni di $X \leq 10$ e 11 esecuzioni di X:=X+1.
- Dopo la prima valutazione di $X \leq 10$ (ed esecuzione di X:=X+1), la denotazione di w è quella di w valutato solo 10 volte, in combinazione con $\mathcal{C}[\![X{:=}X{+}1]\!]\sigma = \{(\sigma, \sigma[\sigma(X)+1/X])\}$. Questo combinato con $\mathcal{C}[\![x{:=}0]\!]\sigma = \{(\sigma, \sigma[0/X])\}$ restituisce la denotazione di $\mathcal{C}[\![x{:=}0;\ w]\!]\sigma$
- Iterando, $\mathcal{C}[\![x{:=}0;\ w]\!]\sigma$ è anche equivalente alla denotazione del while valutato 9 volte combinato alla funzione $\{(\sigma, \sigma[\sigma(X)+1/X])\}$, combinato a $\{(\sigma, \sigma[\sigma(X)+1/X])\}$ e infine combinato a $\{(\sigma, \sigma[0/X])\}$.
- Questo termina quando non dobbiamo più valutare $X \leq 10$, o meglio, quando ulteriori valutazioni non cambiano il risultato.
- Dunque, $\mathcal{C}[\![x{:=}0;\ w]\!]\sigma = \{(\sigma, \sigma[11/X])\}$. Generalizziamo questa idea.

Valutazione di while con fixpoint(1)

- Nella lezione precedente abbiamo parlato di ordinamento parziale tra funzioni:
- date due funzioni parziali $I, J : \Sigma \rightharpoonup \Sigma$, con $I \leq J$ indichiamo che J raffina I o che J estende I
- Tornando alla funzione parziale di while, si potrebbe pensare che ad ogni iterazione di un while si può raffinare la conoscenza della sua valutazione
- Caso base: prima che b sia valutata, la conoscenza su $\mathcal{C}[\![w]\!]$ è $\mathcal{C}_0[\![w]\!] := \perp : \Sigma \rightharpoonup \Sigma$, che denota la funzione non definita in nessun stato. Quindi $\mathcal{C}_0[\![w]\!]$ corrisponde a nessuna informazione.

Valutazione di while con fixpoint(2)

- Si supponga adesso di valutare b "false" in uno stato σ. Dunque la denotazione del while ritorna $\{(\sigma,\sigma)\}$,
- Questo ci permette di raffinare la nostra conoscenza del while da $\mathcal{C}_0[\![w]\!]$ ad una nuova funzione parziale $\mathcal{C}_1[\![w]\!] : \Sigma \rightharpoonup \Sigma$, che è una funzione identità sugli stati σ in cui b è valutata "false"
- Se invece b è valutato "true" in σ, il comando c viene valutato in σ. Supponendo che la sua valutazione sia $\{(\sigma,\sigma')\}$, il risultato del while è rimandato alla valutazione del while stesso su σ'.
- Se conosciamo anche la valutazione di b in σ' possiamo raffinare la nostra conoscenza sul valore del while in $\mathcal{C}_2[\![w]\!]$. In pratica se b è valutato "false" in σ', il while termina, altrimenti si valuta c in σ' e si itera con la valutazione del while. Chiaramente $\mathcal{C}_1[\![w]\!] \leq \mathcal{C}_2[\![w]\!]$.
- Iterando il processo, abbiamo una ω-catena $\mathcal{C}_0[\![w]\!] \leq \mathcal{C}_1[\![w]\!] \leq \ldots \leq \mathcal{C}_k[\![w]\!] \leq \ldots$ di raffinamenti e un ordine parziale dove $\perp = \mathcal{C}_0[\![w]\!]$

Valutazione di while con fixpoint(3)

- Sia f una funzione totale f: $(\Sigma \rightharpoonup \Sigma) \to (\Sigma \rightharpoonup \Sigma)$ tale che :

$$f(\mathcal{C}[\![w]\!]) = \begin{cases} (\sigma, \sigma) & \text{if } \mathcal{B}[\![b]\!]\sigma = \text{false} \\ (\sigma, \mathcal{C}[\![w]\!](\mathcal{C}[\![c]\!]\sigma)) & \text{if } \mathcal{B}[\![b]\!]\sigma = \text{true} \end{cases}$$

- In pratica, f è definita dalle seguenti regole:

$$\begin{cases} \emptyset \to (\sigma,\sigma) & \text{if } \mathcal{B}[\![b]\!]\sigma = \text{false (assioma)} \\ (\sigma'',\sigma') \to (\sigma,\sigma') & \text{if } \mathcal{B}[\![b]\!]\sigma = \text{true} \;\&\; \mathcal{C}[\![c]\!]\sigma = \sigma'' \end{cases}$$

- Siano $f^0(\perp) = \perp = \mathcal{C}_0[\![w]\!]$ e $f^1(\perp) = f(f^0(\perp)) = \mathcal{C}_1[\![w]\!]$, allora possiamo definire induttivamente
- $f^k(\perp) = f(f^{k-1}(\perp)) = \mathcal{C}_k[\![w]\!]$ come il risultato di k composizioni di f.
- La funzione f è continua e per quanto detto nella lezione precedente, il fixpoint di f (che è anche il fixpoint di $\mathcal{C}[\![w]\!]$) è il "least upper bound" della catena.

Osservazioni sulla definizione di f

- Osserviamo ancora come la definizione di f tramite le regole

$$\emptyset \rightarrow (\sigma,\sigma) \qquad \text{if } \mathcal{B}[\![b]\!]\sigma = \text{false}$$
$$(\sigma'',\sigma') \rightarrow (\sigma,\sigma') \qquad \text{if } \mathcal{B}[\![b]\!]\sigma = \text{true} \;\&\; \mathcal{C}[\![c]\!]\sigma = \sigma''$$

  permette di valutare ricorsivamente il comando iterativo w in σ.
- Consideriamo i seguenti scenari di denotazione di b:
- b denotata false in σ, allora la denotazione di w è (σ,σ)
- b vale true in σ, e false in σ''. Siccome al secondo ciclo si utilizza la prima regola di f, risulta $(\sigma'',\sigma') = (\mathcal{C}[\![c]\!]\sigma, \mathcal{C}[\![c]\!]\sigma)$. Dunque la denotazione di w è $(\sigma, \mathcal{C}[\![c]\!]\sigma)$, e corrisponde ad f applicata ad f al passo ricorsivo precedente cioè su: b valutata false in $\mathcal{C}[\![c]\!]\sigma''$.
- b vale k volte "true" partendo da σ, e false la k+1-ma volta. Allora si applica k volte la seconda regola di f e poi una sola volta la prima. Dunque la valutazione di w in questo caso è data dalla funzione f applicata su f al passo ricorsivo precedente cioè su: b valutata k-1 true partendo da σ, e poi false dopo k-1 volte.

Ultima osservazione su f

Nel libro di testo, la funzione f è riferita come "operatore Γ". In seguito anche noi utilizzeremo indistintamente f e Γ, riferendoci allo stesso operatore definito in questa lezione.

Denotazione del comando while

Sia $\mathcal{C}[\![w]\!]$ =

$\{(\sigma,\sigma') \mid \mathcal{B}[\![b]\!]\sigma = true \ \& \ (\sigma,\sigma') \in \mathcal{C}[\![w]\!] \circ \mathcal{C}[\![c]\!]\} \cup$
$\{(\sigma,\sigma) \mid \mathcal{B}[\![b]\!]\sigma = false\}$ =

$\{(\sigma,\sigma') \mid \mathcal{B}[\![b]\!]\sigma = true \ \& \ (\sigma,\sigma'') \in \mathcal{C}[\![c]\!] \ \& \ (\sigma'',\sigma') \in \mathcal{C}[\![w]\!]\} \cup$
$\{(\sigma,\sigma) \mid \mathcal{B}[\![b]\!]\sigma = false\}$

$\mathcal{C}[\![w]\!]$ è una funzione ricorsiva e la sua soluzione è data dal punto fisso di una funzione $f(\mathcal{C}[\![w]\!]) = \mathcal{C}[\![w]\!]$. Tale funzione è definita dal seguente schema di regole

$$\frac{\emptyset}{\sigma\ ,\ \sigma} \quad \text{se } B[\![b]\!]\sigma = false \quad \textit{(assioma)}$$

$$\frac{\sigma''\ ,\ \sigma'}{\sigma\ ,\ \sigma'} \quad \text{se } B[\![b]\!]\sigma = true \ \& \ (\sigma,\sigma'') \in C[\![c]\!]$$

Fixpoint per $\mathcal{C}[\![w]\!]$

- La denotazione di w è data dalla soluzione della funzione ricorsiva $f(\mathcal{C}[\![w]\!]) = \mathcal{C}[\![w]\!]$. Questo equivale a calcolare il suo fixpoint
- Siano $f^0(\bot) = \bot = \mathcal{C}_0[\![w]\!]$ e $f^1(\bot) = f(f^0(\bot)) = \mathcal{C}_1[\![w]\!]$, allora possiamo definire induttivamente
- $f^k(\bot) = f(f^{k-1}(\bot)) = \mathcal{C}_k[\![w]\!]$ come il risultato di k composizioni di f.
- La funzione f è continua e dunque, esiste il fixpoint di f è corrisponde al "least upper bound" della catena.
- Un modo alternativo per legare la valutazione del while "least upper bound" della catena $f^i(\bot)$ è dato dal theorema di Tarski

Teorema di Tarski

- Dato un insieme definito da un insieme di regole, se
  1. Le premesse delle regole sono positive. Cioè non hanno forma $\frac{\neg\alpha}{\beta}$
  2. Le premesse delle regole sono in numero finito

  allora l'operatore f è continuo.
- Se le condizioni del Teorema di Tarski sono soddisfatte allora

$$C[\![w]\!] = \mathrm{fix}(f) = \bigsqcup_{n \in \omega} f^{n}(\perp)$$

- È possibile provare che la regola 1 da sola garantisce univocamente la monotonicità di f.
- È utile notare che il risultato ottenuto sfrutta il fatto che f è una funzione tra potenze di insiemi (insiemi di possibili coppie di stati) e l'insieme potenza con la relazione di inclusione è un c.p.o.

Ricapitolando

- Se ho un comando iterativo la cui denotazione è data una funzione ricorsiva di cui non so a priori il numero delle iterazioni, definisco un operatore di punto fisso, cioè una funzione (ricorsiva) il cui punto fisso è proprio la denotazione del comando.
- Provo dunque a definire l'operatore tramite un insieme di regole
- Se queste regole hanno premesse finite e positive, allora posso applicare il teorema di Tarski e concludere che la semantica del comando è il l.u.b. della catena ottenuta applicando iterativamente le regole che definiscono l'operatore.

Esercizio 1 – prima parte

- Valutare la semantica denotazionale del comando

  while ¬(x = 0) do x := x-x

- Anche in questo caso la denotazione del comando è una equazione ricorsiva (scrittura lasciata per esercizio), di cui non sappiamo a priori il numero delle iterazioni (dipende dal valore di x).
- Risolvere l'equazione di cui sopra significa calcolare il punto fisso di un operatore f che ad ogni sua applicazione raffina la valutazione del while
- Definiamo l'operatore (di punto fisso) tramite un insieme di regole

$$\frac{}{(\sigma,\sigma)}\ \textit{se}\ \sigma(x)=0$$

$$\frac{\sigma'',\sigma'}{(\sigma,\sigma')}\ \textit{se}\ \sigma(x)\neq 0\ \textit{and}\ \sigma''=\sigma[0/x]$$

- Visto che le premesse delle regole sono finite e positive, f ha un punto fisso e fix(f)= $\bigsqcup_{n\in\omega} f^n(\perp)$ cioè il lub della catena $f^0(\perp) \subseteq f^1(\perp) \subseteq f^2(\perp)$ …

Esercizio 1 – seconda parte

- Dobbiamo dunque costruire la catena per l'operatore definito da

$$\frac{}{(\sigma,\sigma)}\ \textit{se}\ \sigma(x)=0 \qquad \frac{\sigma'',\sigma'}{(\sigma,\sigma')}\ \textit{se}\ \sigma(x)\neq 0\ \textit{and}\ \sigma''=\sigma[0/x]$$

- $f^0(\perp) = \perp$
- $f^1(\perp) = \{(\sigma, \sigma) \mid \sigma(x)=0\}$
- $f^2(\perp) = f^1(\perp) \cup \{(\sigma, \sigma') \mid \sigma(x) \neq 0$ e $\sigma' = \sigma[0/x]\}$
- $f^3(\perp) = f(f^2(\perp))$. Se dimostro che questo è uguale a $f^2(\perp)$, ho trovato il punto fisso
- Per provarlo, ricordo che $f^2(\perp) \subseteq f^3(\perp)$. Dunque basta provare che $f^3(\perp) \subseteq f^2(\perp)$. Sia $(\sigma, \sigma') \in f^3(\perp)$ allora ho due casi
  - (σ, σ') è introdotto dall'assioma $\Rightarrow (\sigma, \sigma') \in f^1(\perp) \subseteq f^2(\perp)$
  - (σ, σ') è introdotto dalla regola ricorsiva, quindi $\sigma(x) \neq 0$ e $\sigma''= \sigma[0/x]\}$, dunque $\sigma''(x)= 0$ e $(\sigma'', \sigma') \in f^1(\perp) \Rightarrow (\sigma, \sigma') \in f^2(\perp)$.

Esercizio per casa

- Definire la semantica denotazionale di

  while x > 0 do y = y*2; x:=x-1

  con la semantica intuitiva di calcolare $y*2^x$ per $x \geq 0$.