Aniello Murano

Semantica denotazionale del comando while di IMP

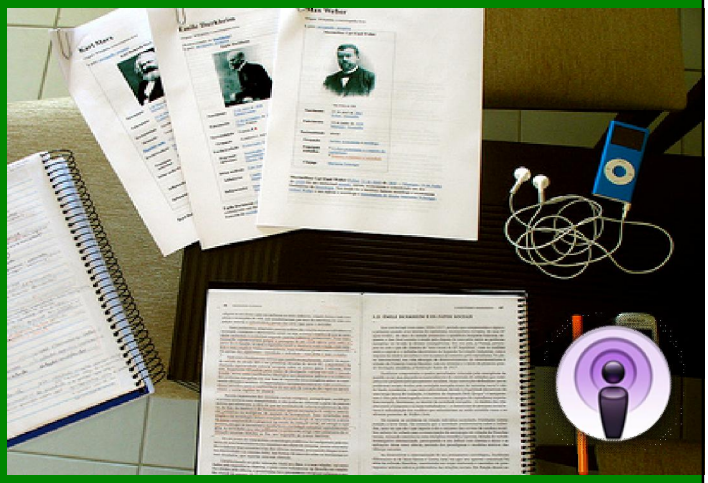
Lezione n. 8
Parole chiave:
while denotazionale

Corso di Laurea:
Informatica

Codice:

Email Docente:
murano@ na.infn.it

A.A. 2008-2009



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 2 Com è una funzione

  C«c¬ : Σ Σ

- La denotazione di c **2** Com (semantica denotazionale 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 C«c¬ : Σ Σ in quanto su alcuni comandi la funzione può non essere definita (per esempio sui loop infiniti)

Denotazione di Com (2)

- $C\llbracket skip \rrbracket = \{(\sigma, \sigma) \mid \sigma \in \Sigma\}$
- $C\llbracket X:=a \rrbracket = \{(\sigma, \sigma[n/X]) \mid \sigma \in \Sigma \ \&\ A\llbracket a_0 \rrbracket\sigma = n\}$
- $C\llbracket c_0;c_1 \rrbracket = C\llbracket c_1 \rrbracket \pm C\llbracket c_0 \rrbracket$ (si noti l'inversione in accordo alla regola di composizione)
- $C\llbracket \text{if } b \text{ then } c_0 \text{ else } c_1 \rrbracket = \{(\sigma, \sigma') \mid B\llbracket b \rrbracket\sigma = true \ \&\ (\sigma, \sigma') \in C\llbracket c_0 \rrbracket\} \cup \{(\sigma, \sigma') \mid B\llbracket b \rrbracket\sigma = false \ \&\ (\sigma, \sigma') \in C\llbracket c_1 \rrbracket\}$
- 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

  $w \simeq$ if b then c;w else skip
- Possiamo allora scrivere la semantica del comando while utilizzando le regole precedenti nel seguente modo:
- $C\llbracket w \rrbracket = C\llbracket \text{if } b \text{ then } c;w \text{ else skip} \rrbracket =$

  $\{(\sigma,\sigma') \mid B\llbracket b \rrbracket\sigma = true \ \&\ (\sigma,\sigma') \in C\llbracket w \rrbracket \pm C\llbracket c \rrbracket\} \cup \{(\sigma,\sigma) \mid B\llbracket b \rrbracket\sigma = false\}$

  $=$

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

Alcune osservazioni su f(C[w])

- C«w¬ = C«if b then c;w else skip¬ =

  {(σ, σ') | 9 σ". B«b¬σ = true & (σ, σ") 2 C«c¬ & (σ", σ') 2 C«w¬}

  [{(σ, σ) | B«b¬σ = false}
- Risolvere questa equazione equivale a calcolare un punto fisso di una funzione C«w¬=f(C«w¬).
- C«w¬ è una unzione parziale C«w¬: Σ Σ.
- C«c¬ è una unzione parziale C«c¬: Σ Σ.
- Dunque, f è ottenuta eseguendo prima C«c¬ e poi C«w¬. Per cui, f è una funzione f: (Σ Σ) ! (Σ Σ). 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 · 10 do X:=X+1
- La denotazione di x:=0; w è C«x:=0; w¬σ = C«w¬σ ± C«x:=0¬σ.
- Sappiamo che C«x:=0¬σ = {(σ, σ[0/X])}
- Per C«w¬σ occorrono 12 valutazioni di X · 10 e 11 esecuzioni di X:=X+1.
- Dopo la prima valutazione di X · 10 (ed esecuzione di X:=X+1), la denotazione di w è quella di w valutato solo 10 volte, in combinazione con C«X:=X+1¬σ = {(σ, σ[σ(X)+1/X])}. Questo combinato con C«x:=0¬σ = {(σ, σ[0/X])} restituisce la denotazione di C«x:=0; w¬σ
- Iterando, C«x:=0; w¬σ è anche equivalente alla denotazione del while valutato 9 volte combinato alla funzione {(σ, σ[σ(X)+1/X])}, combinato a {(σ, σ[σ(X)+1/X])} e infine combinato a {(σ, σ[0/X])}.
- Questo termina quando non dobbiamo più valutare X · 10, o meglio, quando ulteriori valutazioni non cambiano il risultato.
- Dunque, C«x:=0; w¬σ = {(σ, σ[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 $I, J : \Sigma \quad \Sigma$, con $I \cdot J$ 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 C«w¬ è C_0«w¬ := ? : $\Sigma \quad \Sigma$, che denota la funzione non definita in nessun stato. Quindi 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 {(σ,σ)},
- Questo ci permette di raffinare la nostra conoscenza del while da C_0«w¬ ad una nuova funzione parziale C_1«w¬ : $\Sigma \quad \Sigma$, 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 {(σ,σ')}, 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 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 C_1«w¬ · C_2«w¬.
- Iterando il processo, abbiamo una ω-catena C_0«w¬ · C_1«w¬ · … · C_k«w¬ · … di raffinamenti e un ordine parziale dove ? = C_0«w¬

Valutazione di while con fixpoint(3)

- Sia f una funzione totale f: (Σ Σ) ! (Σ Σ) tale che :

$$f(C\text{«}w\neg) = \begin{cases} (\sigma, \sigma) & \text{if } B\text{«}b\neg\sigma = \text{false} \\ (\sigma, C\text{«}w\neg(C\text{«}c\neg\sigma)) & \text{if } B\text{«}b\neg\sigma = \text{true} \end{cases}$$

- In pratica, f è definita dalle seguenti regole:

$$\begin{cases} ; \; ! \; (\sigma,\sigma) & \text{if } B\text{«}b\neg\sigma = \text{false (assioma)} \\ (\sigma'',\sigma') \; ! \; (\sigma,\sigma') & \text{if } B\text{«}b\neg\sigma = \text{true } \& \; C\text{«}c\neg\sigma = \sigma'' \end{cases}$$

- Siano $f^0(?) = ? = C_0\text{«}w\neg$ e $f^1(?) = f(f^0(?)) = C_1\text{«}w\neg$, allora possiamo definire induttivamente
- $f^k(?) = f(f^{k-1}(?)) = C_k\text{«}w\neg$ 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 C«w¬) è il "least upper bound" della catena.

Osservazioni sulla definizione di f

- Osserviamo ancora come la definizione di f tramite le regole

$$\begin{cases} ; \; ! \; (\sigma,\sigma) & \text{if } B\text{«}b\neg\sigma = \text{false} \\ (\sigma'',\sigma') \; ! \; (\sigma,\sigma') & \text{if } B\text{«}b\neg\sigma = \text{true } \& \; C\text{«}c\neg\sigma = \sigma'' \end{cases}$$

  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 (σ",σ') = (C«c¬σ, C«c¬σ). Dunque la denotazione di w è (σ, C«c¬σ), e corrisponde ad f applicata ad f al passo ricorsivo precedente cioè su: b valutata false in C«c¬σ".
- 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 (e.g., Winskel), 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 C«w¬ =

{(σ,σ') | B«b¬σ = true & (σ,σ') 2 C«w¬ ± C«c¬} [
{(σ,σ) | B«b¬σ = false} =

{(σ,σ') | B«b¬σ = true & (σ,σ") 2 C«c¬ & (σ",σ') 2 C«w¬} [
{(σ,σ) | B«b¬σ = false}

C«w¬ è una funzione ricorsiva e la sua soluzione è data dal punto fisso di una funzione f(C«w¬) = C«w¬. Tale funzione è definita dal seguente schema di regole

$$\frac{;}{\sigma\ ,\ \sigma}$$ se B«b¬σ = false *(assioma)*

$$\frac{\sigma",\ \sigma'}{\sigma\ ,\ \sigma'}$$ se B«b¬σ = true & (σ,σ") 2 C«c¬

Fixpoint per C[w]

- La denotazione di w è data dalla soluzione della funzione ricorsiva f(C«w¬) = C«w¬. Questo equivale a calcolare il suo fixpoint
- Siano $f^0(?) = ? = C_0«w¬$ e $f^1(?) = f(f^0(?)) = C_1«w¬$, allora possiamo definire induttivamente
- $f^k(?) = f(f^{k-1}(?)) = C_k«w¬$ 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 $f^i(?)$ 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

$$C«w¬ = fix(f) = \bigsqcup_{n\in\omega} f^n(\bot)$$

- 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

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

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

- 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 $f^0(?)$ μ $f^1(?)$ μ $f^2(?)$ …

Esercizio 1 – seconda parte

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

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

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

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 ¸ 0.