Corso di Metodi Formali dell'Informatica

Soluzioni Sesto Compitino – 6 Febbraio 2002

a.a. 2001/2002

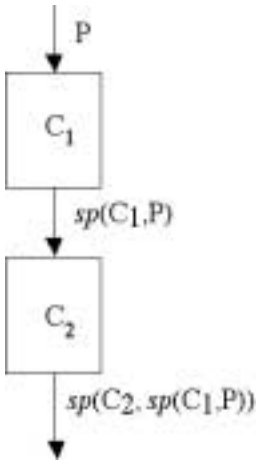
1. (a) (Punti 1) $\{sp(C,P)\} = \{[\![C]\!]\, s \mid s \in \{P\}\}$.

   (b) (Punti 2) Sì. Infatti, se vale $sp(C,P)(s)$, per definizione esiste un s' tale che valga $P(s')$ e $[\![C]\!]\, s' = s$; ma poichè $\{P\}\ C\ \{\Downarrow R\}$ è valida, sappiamo che $P(s') \Rightarrow R([\![C]\!]\, s')$.

   (c) (Punti 2) Dire che $sp(C,P) = \mathit{false}$ signica che per qualunque s deve essere $sp(C,P)(s) = \mathit{ff}$, cioè per qualunque s non può esistere s' tale che $[\![C]\!]\, s' = s$ e $P(s')$. In altri termini, il comando C deve non terminare su tutti gli stati per cui vale la precondizione P.

   Correzione: 1 a chi ha detto che C deve non terminare.

   (d) (Punti 3) Intuitivamente, $sp(C_1\,;C_2,P)$ rappresenta l'insieme degli stati che possiamo ottenere eseguendo $C_1\,;C_2$ a partire da P. Poichè eseguire $C_1\,;C_2$ significa eseguire prima C_1 e poi C_2, sappiamo che al termine dell'esecuzione di C_1 l'insieme di stati che possiamo ottenere è rappresentato da $sp(C_1,P)$, e quindi eseguendo a partire da questa precondizione C_2 otteniamo $sp(C_2, sp(C_1,P))$, come illustrato sotto:



   Formalmente si ha:

   $sp(C_1\,;C_2,P)(s) = tt$ sse esiste s' tale che $[\![C]\!]\, s' = s$ e $P(s')$,
   sse esiste s' tale che $[\![C_2]\!]([\![C_1]\!]\, s') = s$ e $P(s')$,
   sse esistono s', s'' tali che $[\![C_1]\!]\, s' = s''$ e $P(s')$, e $[\![C_2]\!]\, s'' = s$,
   sse esiste s'' tale che $sp(C_1,P)(s'')$ e $[\![C_2]\!]\, s'' = s$,
   sse $sp(C_2, sp(C_1,P))(s)$.

   Correzione: 1 per la caratterizzazione + 2 per una prova informale convincente o formale.

2. (a) (Punti 14) Sia $INV = \forall j \in [1..i-1].\mathsf{A}[j] = 0 \wedge \forall j \in [1..n].\mathsf{A}[j] \geq 0 \wedge i \leq n+1$.

   NB: La prima condizione in *INV* esprime la proprietà su cui si basa il funzionamento dell'algoritmo, cioè il fatto che in un momento intermedio dell'esecuzione tutti gli elementi fino a i (escluso) sono già diventati 0. Ovviamente questo implica che quando si esce dal ciclo while ($i = n+1$) vale la postcondizione desiderata.

   La seconda condizione in *INV* (ovviamente invariante) serve per garantire che nel caso il test nell'if sia falso ($\mathtt{A[i]} \leq 0$) si possa concludere che $\mathtt{A[i]} = 0$ (vedi prova formale per la seconda istanziazione di (Assign+Cons)).

   La terza condizione *non* è necessaria per la prova formale della correttezza parziale, ma la aggiungiamo per chiarezza e perchè serve al punto seguente per provare la terminazione.

   Il corrispondente albero di prova è il seguente (abbiamo indicato con C il ciclo while).

$$\dfrac{\dfrac{\dfrac{}{\{\mathit{INV} \wedge \mathtt{i} \neq n+1 \wedge \mathtt{A[i]} > 0\}\ \mathtt{A[i]=A[i]-1}\ \{\mathit{INV}\}}(A+C) \quad \dfrac{}{\{\mathit{INV} \wedge \mathtt{i} \neq n+1 \wedge \mathtt{A[i]} \leq 0\}\ \mathtt{i=i+1}\ \{\mathit{INV}\}}(A+C)}{\{\mathit{INV} \wedge \mathtt{i} \neq n+1\}\ \mathtt{if\ (A[i]>0)\{A[i]=A[i]-1\}\ else}\ \{\mathtt{i=i+1}\}\ \{\mathit{INV}\}}(If)}{\{\mathtt{i} = 1 \wedge \forall j \in [1..n].\mathsf{A}[j] \geq 0\}\ C\ \{\forall j \in [1..n].\mathsf{A}[j] = 0\}}(W+C)$$

Proviamo che le side conditions della regola (While+Cons) sono verificate:

- $\mathtt{i} = 1 \wedge \forall j \in [1..n].\mathsf{A}[j] \geq 0$ implica banalmente *INV* (l'intervallo $[1..i-1]$ risulta vuoto, e $\mathtt{i} \leq n+1$ poichè n positivo);
- $\mathit{INV} \wedge \mathtt{i} = n+1$ implica $\forall j \in [1..n].\mathsf{A}[j] = 0$.

Proviamo che le side conditions di (Assign+Cons) sono verificate per le due istanziazioni:

- $\mathit{INV} \wedge \mathtt{i} \neq n+1 \wedge \mathtt{A[i]} > 0$ implica banalmente $\mathit{INV}[\mathtt{A[i]} - 1/\mathtt{A[i]}] = \forall j \in [1..i-1].\mathsf{A}[j] = 0 \wedge \forall j \in [1..n], j \neq i.\mathsf{A}[j] \geq 0 \wedge \mathsf{A}[i] + 1 \geq 0 \wedge i \leq n+1$;
- $\mathit{INV} \wedge \mathtt{i} \neq n+1 \wedge \mathtt{A[i]} \leq 0$ implica $\mathit{INV}[\mathtt{i}+1/\mathtt{i}] = \forall j \in [1..i].\mathsf{A}[j] = 0 \wedge \forall j \in [1..n].\mathsf{A}[j] \geq 0 \wedge \mathtt{i} \leq n$, infatti la prima condizione segue dal fatto che vale *INV* (quindi gli elementi fino a $i-1$ sono 0 e l'elemento di posto $\mathtt{i}$ è ≥ 0), ed inoltre vale $\mathtt{A[i]} \leq 0$, quindi $\mathtt{A[i]} = 0$. La terza condizione vale perchè $\mathtt{i} \leq n+1$ (da *INV*) e $\mathtt{i} \neq n+1$.

Correzione: 10 a chi non ha messo la seconda condizione nell'invariante; 12 a chi ha messo $\mathtt{i} \leq n$ invece di $\mathtt{i} \leq n+1$ o varianti.

(b) (Punti 5) Una funzione di terminazione è ad esempio la seguente:

$t\colon \mathit{States} \to \mathbb{Z} \times \mathbb{Z}$,
$t(s) = (n - s(\mathtt{i}), \sum_{j=1}^{n} s(\mathsf{A}[j]))$

dove consideriamo su $\mathbb{Z} \times \mathbb{Z}$ l'ordinamento prodotto, cioè $(x, y) \leq (x', y')$ sse $x \leq x', y \leq y'$.

Infatti è facile vedere che per s in $\{\mathit{INV} \wedge \mathtt{i} \neq n+1\}$ si ha che $n - \mathtt{i}$ è limitato inferiormente da zero (qui serve l'ipotesi $i \leq n+1$), e così pure $\sum_{j=1}^{n} s(\mathsf{A}[j])$ perchè gli elementi sono tutti positivi.

Inoltre, se si esegue il body a partire da un s in $\{\mathit{INV} \wedge i \neq n+1\}$, il valore di *t* decresce (strettamente): infatti se si esegue il ramo "then" la prima componente resta uguale e la seconda diminuisce di uno, e viceversa se si esegue il ramo "else".

Correzione: non ho contato l'errore di indicare come codominio $\mathbb{N}$ invece di $\mathbb{Z}$ perchè lo hanno fatto (quasi) tutti; 3,5 a chi non ha detto che l'ordine era lessicografico quando era indispensabile dirlo, e a chi non ha detto che occorreva aggiungere la terza condizione nell'invariante; 2 a chi ha dato una funzione di terminazione che funzionava in un caso solo.

3. (Punti 5) Un'invariante adatta è ad esempio $\mathit{INV} = (\mathtt{found} \Rightarrow \mathtt{x} = \mathtt{y}) \wedge \mathtt{x} \leq x_0$.

Questa invariante corrisponde alla seguente intuizione: se ad una certa iterazione $\mathtt{found}$ diventa vero, sappiamo che $\mathtt{x} = \mathtt{y}$, e poichè banalmente $\mathtt{x}$ è sempre $\leq x_0$ (infatti all'inizio sono uguali e $\mathtt{x}$ decresce sempre), possiamo concludere che $\mathtt{y} \leq x_0$.

Più precisamente si ha che:

(a) *INV* vale all'inizio, poichè $\neg\mathtt{found} \wedge \mathtt{x} = x_0$ la implica; si noti che essendo $\mathtt{found}$ falso la prima condizione (l'implicazione) in *INV* è vera;

(b) $\mathit{INV} \wedge \mathtt{found}$ implica la postcondizione (è il ragionamento fatto sopra);

(c) se vale $\mathit{INV} \wedge \neg\mathtt{found}$ prima di eseguire il body, allora:

i. se si esegue il ramo "then", $\mathtt{found}$ diventa vero, e sappiamo che $\mathtt{x} = \mathtt{y}$, quindi la prima condizione (l'implicazione) in *INV* è vera e l'altra resta vera,

ii. se si esegue il ramo "else", $\mathtt{found}$ resta falso quindi l'implicazione resta vera, e l'altra condizione resta vera perchè $\mathtt{x}$ decresce.

Il corrispondente albero di prova (non richiesto) è il seguente, dove indichiamo con *C* il ciclo while.

$$\dfrac{\dfrac{\dfrac{}{\{\mathit{INV} \wedge \neg\mathtt{found} \wedge \mathtt{x} = \mathtt{y}\}\ \mathtt{found=true}\ \{\mathit{INV}\}}(A+C) \quad \dfrac{}{\{\mathit{INV} \wedge \neg\mathtt{found} \wedge \mathtt{x} \neq \mathtt{y}\}\ \mathtt{x=x-1}\ \{\mathit{INV}\}}(A+C)}{\{\mathit{INV} \wedge \neg\mathtt{found}\}\ \mathtt{if\ (x==y)\{found=true\}\ else}\ \{\mathtt{x=x-1}\}\ \{\mathit{INV}\}}(If)}{\{\neg\mathtt{found} \wedge \mathtt{x} = x_0\}\ C\ \{\mathtt{y} \leq x_0\}}(W+C)$$

La prova che le due side conditions di (While+Cons) sono verificate è data in (a)+(b) sopra; la prova che le side conditions di (Assign+Cons) sono verificate per le due istanziazioni è la formalizzazione di (c) sopra, ed è la seguente:

- $\mathit{INV} \wedge \neg\mathtt{found} \wedge \mathtt{x} = \mathtt{y}$ implica banalmente $\mathtt{x} = \mathtt{y} \wedge \mathtt{x} \leq x_0 = \mathit{INV}[\mathit{true}/\mathtt{found}]$,
- $\mathit{INV} \wedge \neg\mathtt{found} \wedge \mathtt{x} \neq \mathtt{y}$ implica $(\neg\mathtt{found} \Rightarrow \mathtt{x} - 1 = \mathtt{y}) \wedge \mathtt{x} - 1 \leq x_0 = \mathit{INV}[\mathtt{x} - 1/\mathtt{x}]$. Si noti che essendo $\mathtt{found}$ falso la prima condizione (l'implicazione) è vera.

Correzione: 3,5 a chi ha dato un'invariante incompleta ma sensata.