Corso di Metodi Formali dell'Informatica

Soluzioni Compitino – 6 Giugno 2003

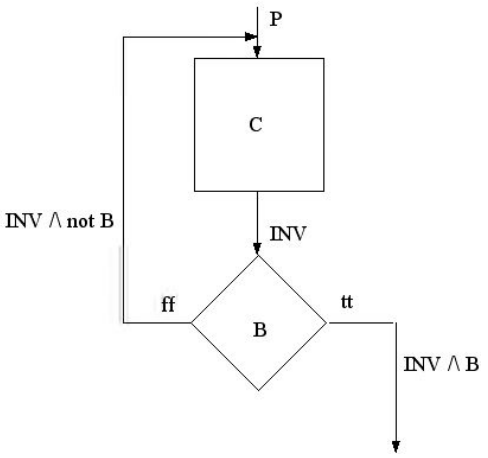
a.a. 2002/2003

1.

(a) (Punti 5)

$$\text{(Do-Until)} \quad \frac{\{P\}\ C\ \{INV\} \quad \{INV \wedge \neg[\![B]\!]\}\ C\ \{INV\}}{\{P\}\ \texttt{do}\ \{C\}\ \texttt{until}\ B\ \{INV \wedge [\![B]\!]\}}$$

(b) (Punti 3) Dobbiamo provare che se vale P prima di eseguire $\texttt{do}\ \{C\}\ \texttt{until}\ B$ e il comando termina, allora vale $INV \wedge [\![B]\!]$. Se il comando termina vuol dire che si è eseguito il body una volta, ed in base alla prima premessa vale quindi INV, e poi ancora il body un numero finito di volte il body, sia $n \geq 0$ questo numero. È semplice allora provare per induzione aritmetica su n che ogni volta all'inizio dell'esecuzione del body vale INV: infatti questo vale la prima volta ed ogni volta continua a valere in base alla premessa (infatti se si riesegue il body vuol dire che vale anche $\neg[\![B]\!]$. All'ultimo passo quindi vale ancora INV, vale B perchè altrimenti si effettuerebbe ancora un passo e quindi non si fa nulla e vale INV alla fine (vedi diagramma).



(c) (Punti 3) Dobbiamo provare che per ogni $s \in States$, se $P(s) = tt$, allora, assumendo che l'esecuzione termini, si ha $(INV \wedge [\![B]\!])([\![\texttt{do}\ \{C\}\ \texttt{until}\ B]\!]\, s) = tt$. Sappiamo che $[\![\texttt{do}\ \{C\}\ \texttt{until}\ B]\!]\, s = [\![C;\texttt{if}\ (!B)\{\texttt{do}\ \{C\}\ \texttt{until}\ B\}]\!]\, s$. Poichè la prima premessa è valida, $INV([\![C]\!]\, s) = tt$. È quindi sufficiente provare che se $INV(s) = tt$, allora $(INV \wedge [\![B]\!])([\![\texttt{if}\ (!B)\{\texttt{do}\ \{C\}\ \texttt{until}\ B\}]\!]\, s) = tt$ (sempre assumendo che l'esecuzione termini).

Proviamo questa tesi per induzione sulla definizione della funzione $[\![\texttt{if}\ (!B)\{\texttt{do}\ \{C\}\ \texttt{until}\ B\}]\!]$.

- Sia $[\![B]\!]\, s = tt$, quindi $[\![\texttt{if}\ (!B)\{\texttt{do}\ \{C\}\ \texttt{until}\ B\}]\!]\, s = s$. Allora la tesi vale banalmente.
- Sia $[\![B]\!]\, s = ff$, quindi $[\![\texttt{if}\ (!B)\{\texttt{do}\ \{C\}\ \texttt{until}\ B\}]\!]\, s = [\![\texttt{do}\ \{C\}\ \texttt{until}\ B]\!]\, s = [\![C;\texttt{if}\ (!B)\{\texttt{do}\ \{C\}\ \texttt{until}\ B\}]\!]\, s = [\![\texttt{if}\ (!B)\{\texttt{do}\ \{C\}\ \texttt{until}\ B\}]\!]\, s''$ per $s'' = [\![C]\!]\, s$; dalla premessa sappiamo che vale $INV(s'')$ e per ipotesi induttiva si ha la tesi.

2.

(a) (Punti 10)

Sia $C = \texttt{if (A[i] == } k\texttt{)\{found = true\};i = i} + 1$. Scegliamo come invariante INV=

$$\texttt{found} = (\exists j \in 1..\texttt{i} - 1].\texttt{A}[j] \texttt{ == } k) \wedge \texttt{i} \leq n + 1$$

e come funzione di terminazione

t: $States \to \mathbb{Z}$
$t(s) = n - s(\texttt{i})$.

Si ha il seguente albero di prova (una parte è riportata separatamente):

$$\dfrac{\dfrac{\dfrac{\vdots}{\{INV \wedge \texttt{i} \leq n\}\ \texttt{if (A[i] == } k\texttt{)\{found = true\}}\ \{\Downarrow INV[\texttt{i} + 1/\texttt{i}]\}}\,(If) \quad \dfrac{}{\{INV[\texttt{i} + 1/\texttt{i}]\}\ \texttt{i = i} + 1\ \{\Downarrow INV\}}\,(A)}{\{INV \wedge \texttt{i} \leq n\}\ C\ \{\Downarrow INV\}}\,(Seq)}{\{\texttt{i} = 1 \wedge \neg\texttt{found}\}\ \texttt{while ((i} \leq n\texttt{))}\ \{\ C\ \}\ \{\Downarrow \texttt{found} = (\exists j \in [1..n].\texttt{A}[j] = k)\}}\,(W{+}C)$$

$$\dfrac{\dfrac{}{\{INV \wedge \texttt{i} \leq n \wedge \texttt{A}[\texttt{i}] = k\}\ \texttt{found = true}\ \{\Downarrow INV[\texttt{i} + 1/\texttt{i}]\}}\,(A{+}C) \quad \dfrac{}{\{INV \wedge \texttt{i} \leq n \wedge \texttt{A}[\texttt{i}] \neq k\}\ \texttt{skip}\ \{\Downarrow INV[\texttt{i} + 1/\texttt{i}]\}}\,(A{+}C}{\{INV \wedge \texttt{i} \leq n\}\ \texttt{if (A[i] == } k\texttt{)\{found = true\}}\ \{\Downarrow INV[\texttt{i} + 1/\texttt{i}]\}}$$

Proviamo che valgono le side conditions della regola (While+Cons).

- $\texttt{i} = 1 \wedge \neg\texttt{found} \Rightarrow INV$
  vale perché l'intervallo $[1..i - 1]$ risulta vuoto per $\texttt{i} = 1$, e $\texttt{i} \leq n + 1$ vale per $\texttt{i} = 1$.
- $INV \wedge \texttt{i} > n \Rightarrow \texttt{found} = (\exists j \in [1..n].\texttt{A}[j] = k)$
  vale perché si ha $\texttt{i} = n + 1$.
- t è una funzione di terminazione per il ciclo while dato e INV, infatti:
  - $t(\{INV \wedge \texttt{i} \leq n\}) \subseteq \mathbb{N}$ (poiché $i \leq n$);
  - si vede facilmente che per ogni s in $\{INV \wedge \texttt{i} \leq n\}$, $t([\![C]\!]\, s) < t(s)$.

Proviamo che valgono le side conditions delle due istanziazioni della metaregola (Assign+Cons).

- $INV \wedge \texttt{i} \leq n \wedge \texttt{A}[\texttt{i}] = k \Rightarrow \texttt{true} = (\exists j \in [1..\texttt{i}].\texttt{A}[j] = k) \wedge \texttt{i} + 1 \leq n + 1$
  vale perché $A[\texttt{i}] = k$ implica $\exists j \in 1..\texttt{i}].\texttt{A}[j] = k$, e $\texttt{i} \leq n$ implica $\texttt{i} + 1 \leq n + 1$.
- $INV \wedge \texttt{i} \leq n \wedge \texttt{A}[\texttt{i}] \neq k \Rightarrow \texttt{found} = (\exists j \in [1..i].\texttt{A}[j] = k) \wedge \texttt{i} + 1 \leq n + 1$
  vale perché essendo $\texttt{A}[\texttt{i}] \neq k$ si ha che $\exists j \in [1..i].\texttt{A}[j] = k$ se e solo se $\exists j \in [1..i - 1].\texttt{A}[j] = k$, e $\texttt{i} \leq n$ implica $\texttt{i} + 1 \leq n + 1$.

(b) (Punti 6=3+3) Si consideri ora la versione ottimizzata, sia da qui in poi

$$C = \texttt{if (A[i] == } k\texttt{)\{found = true\} else \{i = i} + 1\}.$$

Anzitutto osserviamo che la precedente scelta dell'invariante non va più bene. Infatti, nel caso venga trovato un (primo) indice $\texttt{i}$ tale che $\texttt{A}[\texttt{i}] = k$, si ha che $\texttt{found} = \exists j \in [1..i - 1].\texttt{A}[j] = k$ vale prima di eseguire C (`found` falso, e $\exists j \in [1..i - 1].\texttt{A}[j] = k$ falso), ma non vale più dopo l'assegnazione `found = true`.

Inoltre, anche la precedente scelta della funzione di terminazione non va più bene; infatti sempre nel caso descritto sopra si ha che $n - \texttt{i}$ non decresce.

Possiamo modificare l'invariante ad esempio nel modo seguente:

$$INV = ((\neg\texttt{found} \wedge \nexists j \in [1..\texttt{i} - 1].\texttt{A}[j] = k) \vee (\texttt{found} \wedge \exists j \in [1..n].\texttt{A}[j] = k)) \wedge \texttt{i} \leq n + 1$$

e la funzione di terminazione ad esempio nel modo seguente:

t: $States \to \mathbb{Z} \times \{tt, ff\}$
$t(s) = (n - s(\texttt{i}), s(\texttt{found}))$.

dove l'ordinamento sulle coppie in $\mathbb{Z} \times \{tt, ff\}$ è definito da

$(tt, z) < (ff, z)$ per ogni $z \in \mathbb{Z}$,
$(ff, z) < (ff, z')$ sse $z < z'$.

Possiamo provare che questa scelta di INV va bene nel modo seguente .

- Vale all'inizio, infatti la precondizione $i = 1 \wedge \neg\texttt{found}$ implica la prima alternativa nell'or, e $\texttt{i} \leq n + 1$ come prima.
- Alla fine, INV congiuntamente alla condizione di uscita del ciclo ($i > n \vee \texttt{found}$) garantisce la postcondizione $\texttt{found} = \exists j \in [1..n].\texttt{A}[j] = k$. Infatti, se vale la prima alternativa nell'or $\texttt{found}$ è falso, si ha anche $i > n$, quindi $i = n + 1$ e $\exists j \in [1..n].\texttt{A}[j] = k$ falso. Se invece vale la seconda alternativa $\texttt{found}$ è vero e $\exists j \in [1..n].\texttt{A}[j] = k$ è vero.
- INV è effettivamente un'invariante per il ciclo. Informalmente, assumendo che prima di eseguire C valga INV congiuntamente alla condizione del ciclo ($i \leq n \wedge \neg\texttt{found}$), si hanno i due casi seguenti:
  - $A[i] = k$ (con $\texttt{i} \leq n$)e $\texttt{found}$ diventa vero, in questo caso possiamo concludere che vale la seconda alternativa nell'or.
  - $A[i] \neq k$ e i diventa $i + 1$; in questo caso $\texttt{found}$ resta falso e possiamo concludere che vale la prima alternativa nell'or.

Inoltre, è facile vedere che t è una funzione di terminazione per il ciclo dato e INV. Infatti:

- analogamente a prima, $t(\{INV \wedge \texttt{i} \leq n \wedge \neg\texttt{found}\}) \subseteq \mathbb{N} \times \{tt, ff\}$ (che è ovviamente noetheriano essendo $\mathbb{N}$ noetheriano e $\{tt, ff\}$ finito);
- per ogni s in $\{INV \wedge \texttt{i} \leq n\}$, $t([\![C]\!]\, s) < t(s)$ (infatti o $\texttt{found}$ resta falso e decresce $n - \texttt{i}$, oppure $\texttt{found}$ diventa vero.

3. (Punti 6 = 4 + 2) Un comando che rende vere entrambe le condizioni è ad esempio

$$C = \texttt{while (x != y) \{ x = x} - 1 \texttt{ \}}.$$

Infatti è facile vedere che $[\![C]\!]\, s = s[s(\texttt{y})/\texttt{x}]$ se $s(\texttt{x}) \geq s(\texttt{y})$, indefinito altrimenti.

Quindi

$$wp(C, \texttt{x} = \texttt{y}) = \{s \mid [\![C]\!]\, s \in \{\texttt{x} = y\}\} = \{\texttt{x} \geq \texttt{y}\},$$
$$wlp(C, false) = \{s \mid [\![C]\!]\, s = s' \Rightarrow s' \in \{\texttt{false}\}\} = \{s \mid [\![C]\!]\, s \text{ indefinito}\} = \{\texttt{x} < \texttt{y}\}.$$