Logica per la Programmazione

Lezione 16

- Sistema di Dimostrazioni per le Triple di Hoare
- Comando Iterativo

Semantica informale del Comando Iterativo

- Come per gli altri comandi (assegnamento, sequenza di comandi, skip, condizionale) introdurremo una regola di inferenza per il comando iterativo sulla base della sua semantica informale:
- *L'esecuzione del comando* **while** E **do** C **endw** *a partire da* σ *porta in* σ *se* $\mathcal{E}(E, sg) = \mathbf{ff}$, *altrimenti porta nello stato* σ' *ottenuto dall'esecuzione di* **while** E **do** C **endw** *a partire dallo stato* σ'' *ottenuto con l'esecuzione di* C *nello stato* σ.
- In altre parole, l'esecuzione inizia valutando la *guardia* E. Se il valore è **ff** l'esecuzione del **while** termina senza modificare lo stato. Altrimenti si esegue il *corpo* C, e nello stato risultante si riesegue l'intero comando **while**.

Semantica informale del Comando Iterativo

- Quindi l'esecuzione del while comporta l'esecuzione del comando C un certo numero di volte, non determinabile a priori. Inoltre l'esecuzione potrebbe non portare ad un stato definito se
  - la guardia non è valutabile ($def(E)$ è falso), oppure
  - la guardia è sempre vera (ciclo infinito)

- **Attenzione:** Per una spiegazione approfondita di come si può arrivare alla regola di inferenza presentata di seguito, si veda il Paragrafo 4.7 della dispensa sulle Triple di Hoare

Regola per il Comando Iterativo

$$\frac{P \Rightarrow Inv \wedge def(E) \qquad \{Inv \wedge E \wedge t = V\}\ C\ \{t < V\} \qquad Inv \Rightarrow t \geq 0 \qquad \{Inv \wedge E\}\ C\ \{Inv \wedge def(E)\} \qquad Inv \wedge \neg E \Rightarrow Q}{\{P\}\ \textbf{while}\ E\ \textbf{do}\ C\ \textbf{endw}\ \{Q\}}\ (\text{WHILE})$$

- t è chiamata **funzione di terminazione**
- Inv è chiamata **invariante**
- V è una **variabile di specifica**: denota un generico valore, non utilizzabile e non modificabile nel programma
- $\{Inv \wedge E \wedge t = V\}\ C\ \{t < V\}$ è l'**ipotesi di progresso**
- $(Inv \Rightarrow t \geq 0)$ è l'**ipotesi di terminazione**
- $\{Inv \wedge E\}\ C\ \{Inv \wedge def(E)\}$ è l'**ipotesi di invarianza**

Regola per il Comando Iterativo: alcuni commenti

$$\frac{\{Inv \wedge E \wedge t = V\}\ C\ \{t < V\} \qquad Inv \Rightarrow t \geq 0 \qquad P \Rightarrow Inv \wedge def(E) \qquad \{Inv \wedge E\}\ C\ \{Inv \wedge def(E)\} \qquad Inv \wedge \neg E \Rightarrow Q}{\{P\}\ \textbf{while}\ E\ \textbf{do}\ C\ \textbf{endw}\ \{Q\}}\ (\text{WHILE})$$

Le condizioni della regola formano due gruppi:

- Le *ipotesi di terminazione* e *di progresso*, insieme, garantiscono che il comando iterativo termini. Infatti
  - ad ogni iterazione il valore di t decresce di almeno un'unità, visto che t assume valori interi
  - t non può assumere valori negativi durante l'esecuzione del ciclo, perché l'invariante è sempre vera.
- Le altre tre condizioni consentono di dimostrare la postcondizione Q sfruttando ipotesi presenti nell'invariante (e indirettemente nella precondizione P), e la negazione delle guardia

Esempio di Comando Iterativo

Usando come **invariante** $Inv : s = (\Sigma i : i \in [0, x) \,.\, i) \ \wedge \ 0 \leq x \ \wedge \ x \leq n$

e come **funzione di terminazione** $t : n - x$ verificare la tripla

$$\{s = 0 \wedge x = 0 \wedge n \geq 0\}$$
while $x < n$ **do**
$$x, s := x + 1, s + x$$
endw
$$\{s = (\Sigma i : i \in [0, n) \,.\, i)\}$$

Per la Regola per il Comando Iterativo è sufficiente mostrare:

[Pre] $s = 0 \ \wedge \ x = 0 \ \wedge \ n \geq 0 \Rightarrow def(x < n) \wedge Inv$

[Post] $Inv \wedge \neg\, x < n \Rightarrow s = (\Sigma i : i \in [0, n) \,.\, i)$

[Term] $Inv \Rightarrow n - x \geq 0$

[Inv] $\{Inv \wedge x < n\} x, s := x + 1, s + x \{Inv \wedge def(x < n)\}$

[Prog] $\{Inv \wedge x < n \wedge n - x = V\} \ x, s := x + 1, s + x \ \{n - x < V\}$

Esercizio: completare la dimostrazione

Comando di Inizializzazione

- Spesso la precondizione di una tripla con un while non è sufficiente per soddisfare la condizione [Pre] $P \Rightarrow Inv \wedge def(E)$
- In questo caso si può inserire un **comando di inizializzazione** C_I tale che $\{P\}\ C_I\ \{Inv \wedge def(E)\}$
- **Esempio**. Nella tripla vista, se la precondizione è solo $\{n \geq 0\}$, la [Pre] è falsa (invariante e $def(x < n)$ non valgono).
- Possiamo renderla vera con un comando che inizializzi x e s.

$\{n \geq 0\}??$
while $x < n$ **do**
$x, s := x + 1, s + x$
endw
$\{s = (\Sigma i : i \in [0, n)\ .\ i)\}$

$\{n \geq 0\}$
x, s := 0, 0 ;
$\{s = 0 \wedge x = 0 \wedge n \geq 0\}$
while $x < n$ **do**
$x, s := x + 1, s + x$
endw
$\{s = (\Sigma i : i \in [0, n)\ .\ i)\}$

Programmi Annotati

- Si possono aggiungere annotazioni al programma per rendere esplicito ciò che si deve dimostrare, p.es. *invariante* e *funzione di terminazione* di un ciclo, o *asserzione* in un punto arbitrario del programma.
- Per esempio, annotiamo il programma a sinistra come mostrato a destra:

$\{n \geq 0\}$
$x, s := 0, 0;$
while $x < n$ **do**
$x, s := x + 1, s + x$
endw
$\{s = (\Sigma i : i \in [0, n) \,.\, i)\}$

$\{n \geq 0\}$
$x, s := 0, 0;$
$\{P : s = 0 \wedge x = 0 \wedge n \geq 0\}$
$\{Inv : s = (\Sigma i : i \in [0, x) \,.\, i) \wedge 0 \leq x \wedge x \leq n\}$
$\{t : n - x\}$
while $x < n$ **do**
$x, s := x + 1, s + x$
endw
$\{s = (\Sigma i : i \in [0, n) \,.\, i)\}$

- Le annotazioni forniscono, in questo caso, le info necessarie per applicare le regole della sequenza e del comando iterativo.

Esempio di Programma Annotato: formule da dimostrare

$\{n \geq 0\}$
$x, s := 0, 0;$
$\{P : s = 0 \wedge x = 0 \wedge n \geq 0\}$
$\{Inv : s = (\Sigma i : i \in [0, x) \,.\, i) \wedge 0 \leq x \wedge x \leq n\}$
$\{t : n - x\}$
while $x < n$ **do**
$x, s := x + 1, s + x$
endw
$\{s = (\Sigma i : i \in [0, n) \,.\, i)\}$

[ASS] $\{n \geq 0\}\ x, s := 0, 0\ \{P : s = 0 \wedge x = 0 \wedge n \geq 0\}$

[Pre] $s = 0 \wedge x = 0 \wedge n \geq 0 \Rightarrow def(x < n) \wedge Inv$

[Post] $Inv \wedge \neg(x < n) \Rightarrow s = (\Sigma i : i \in [0, n) \,.\, i)$

[Term] $Inv \Rightarrow n - x \geq 0$

[Inv] $\{Inv \wedge x < n\}\ x, s := x + 1, s + x\ \{Inv \wedge def(x < n)\}$

[Prog] $\{Inv \wedge x < n \wedge n - x = V\}\ x, s := x + 1, s + x\ \{n - x < V\}$

Esercizio: Somma con Incrementi

$\{z = A \ \wedge \ n = B \ \wedge \ B \geq 0\}$
$\{Inv : z + n = A + B \wedge n \geq 0\}\{t : n\}$
while *not* $(n = 0)$ **do**
$z := z + 1; n := n - 1$
endw
$\{Inv \wedge n = 0\}$
$\{z = A + B\}$

Il programma calcola
in **z** la somma dei valori iniziali di **z** ed **n**
usando incrementi unitari

Per verificare la tripla, per la Regola per il Comando Iterativo è sufficiente dimostrare:

$$\begin{aligned}
\text{[Pre]} \quad & z = A \wedge n = B \wedge B \geq 0 \ \Rightarrow \ Inv \wedge def(not(n = 0)) \\
\text{[Post]} \quad & Inv \wedge n = 0 \ \Rightarrow \ z = A + B \\
\text{[Inv]} \quad & \{Inv \wedge not(n = 0)\} \ z := z + 1; \\
& n := n - 1 \ \{Inv \wedge def(not(n = 0))\} \\
\text{[Term]} \quad & Inv \Rightarrow n \geq 0 \\
\text{[Prog]} \quad & \{Inv \wedge not(n = 0) \wedge n = V\} \ z := z + 1; n := n - 1 \ \{n < V\}
\end{aligned}$$

Esercizio: Calcolo MCD

$\{x = A \ \wedge \ y = B \ \wedge \ A > 0 \ \wedge \ B > 0\}$
$\{Inv : x > 0 \ \wedge \ y > 0 \ \wedge \ mcd(A, B) = mcd(x, y)\}\{t : x + y\}$
while $(x <> y)$ **do**
if $x > y$ **then** $x := x - y$ **else** $y := y - x$**fi**
endw
$\{x = mcd(A, B)\}$

Dimostrare la correttezza del programma annotato, facendo uso delle seguenti note proprietà dell'operatore *mcd*:

$$mcd(v, w) = \begin{cases} v & \text{se } v = w \\ mcd(v - w, w) & \text{se } v > w \\ mcd(v, w - v) & \text{se } v < w \end{cases}$$

Esercizio: Calcolo del Fattoriale

- Si consideri la seguente specifica:

$$\{n > 0\}\ C\ \{f = n!\}$$

- Si chiede quindi di scrivere un comando C che calcoli il fattoriale di un numero n maggiore di zero
- Proponiamo come soluzione il seguente programma:

$\{n > 0\}$
$f, x := 1, 1;$
while $(x <= n)$ **do**
$f, x := f * x, x + 1;$
endw
$\{f = n!\}$

- Dobbiamo verificare che la tripla sia soddisfatta
- Ma come possiamo determinare l'invariante e la funzione di terminazione?

Esercizio: Calcolo del Fattoriale (2)

- Eseguiamo manualmente il programma (es: per n $= 5$):

$\{n > 0\}$
$\quad f, x := 1, 1;$
$\{Inv :?\}\{t :?\}$
while $(x <= n)$ **do**
$\quad f, x := f * x, x + 1;$
endw
$\{Inv \wedge \neg(x <= n)\}$
$\{f = n!\}$

x	f
1	1
2	1
3	2
4	6
5	24
6	**120**

- Osserviamo che, ad ogni iterazione, i valori di x e f sono legati dalla seguente relazione

$$f = (x - 1)!$$

- Tuttavia, scegliendo $Inv = f = (x - 1)!$ non riusciamo a dimostrare

$$\text{[Post]} \qquad Inv \wedge \neg E \Rightarrow Q \quad \equiv$$

$$f = (x - 1)! \wedge \neg(x \leq n) \;\Rightarrow\; f = n!$$

Esercizio: Calcolo del Fattoriale (3)

- Aggiungendo $x \in [0, n+1]$ in *Inv* siamo in grado di dimostrare:

$$[\text{Post}] \quad Inv \ \wedge \neg(x <= n) \Rightarrow f = n!$$

- Aggiungiamo l'invariante alla tripla:

$\{n > 0\}$
$f, x := 1, 1;$
$\{Inv : f = (x-1)! \wedge x \in [0, n+1]\}\{t :?\}$
while $(x <= n)$ **do**
$f, x := f * x, x + 1;$
endw
$\{Inv \wedge \neg(x <= n)\}$
$\{f = n!\}$

- Possiamo ora verificare la tripla...

Esercizio: Calcolo del Fattoriale (4)

- Ipotesi di invarianza

  $\{f = (x-1)! \wedge x \in [0, n+1)\}\ f, x := f*x, x+1\ \{f = (x-1)! \wedge x \in [0, n+1]\}$

- Per la regola dell'assegnamento multiplo, basta dimostrare:

  $f = (x-1)! \ \wedge\ x \in [0, n+1) \Rightarrow def(f*x) \ \wedge\ def(x+1) \ \wedge\ f*x = x! \ \wedge\ x+1 \in [0, n+1]$

- Partiamo dalla conseguenza (eliminando i def(...) che sono **T**):

  $$\begin{array}{ll} & f*x = x! \wedge x+1 \in [0, n+1] \\ \equiv & \quad \{\textbf{Ip}:\ f = (x-1)!\} \\ & (x-1)!*x = x! \wedge x+1 \in [0, n+1] \\ \equiv & \quad \{\text{def. fattoriale}\} \\ & x+1 \in [0, n+1] \\ \equiv & \quad \{\textbf{Ip}:\ x \in [0, n+1),\ \text{calcolo}\} \\ & \textbf{T} \end{array}$$

- La funzione di terminazione (che è molto semplice) è lasciata per esercizio.