Logica per la Programmazione

Lezione 13

- Comando Iterativo
- Sequenze (array) ed Aggiornamento Selettivo

Semantica informale del Comando Iterativo

- 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.
- 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 è sempre vera (ciclo infinito), oppure
  - la guardia non è valutabile ($def(E)$ è falso)
  - Per vedere 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

$$P \Rightarrow Inv \wedge def(E) \qquad Inv \ \wedge \neg E \Rightarrow Q$$

$$\{Inv \wedge E \wedge t = V\}\ C\ \{t < V\}$$

$$(WHILE)\quad \frac{\{Inv \wedge E\}\ C\ \{Inv \wedge def(E)\} \quad Inv \Rightarrow t \geq 0}{\{P\}\ \textbf{while}\ E\ \textbf{do}\ C\ \textbf{endw}\ \{Q\}}$$

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

Esempio di Comando Iterativo (1)

Usando come

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

Esempio di Comando Iterativo (2)

Per la Regola per il Comando Iterativo è sufficiente mostrare:

1. $s = 0 \wedge x = 0 \wedge n \geq 0 \;\Rightarrow$
   $def(\, x < n \,) \wedge s = (\Sigma i : i \in [0, x).i) \wedge 0 \leq x \wedge x \leq n$
2. $s = (\Sigma i : i \in [0, x).i) \wedge 0 \leq x \wedge x \leq n$
   $\wedge \neg (x < n) \;\Rightarrow\; s = (\Sigma i : i \in [0, n).i)$
3. $s = (\Sigma i : i \in [0, x).i) \wedge 0 \leq x \wedge x \leq n \;\Rightarrow\;$ n - x ≥ 0
4. $\{ s = (\Sigma i : i \in [0, x).i) \wedge 0 \leq x \wedge x \leq n \wedge x < n \}\; x, s := x + 1, s + x$
   $\{ s = (\Sigma i : i \in [0, x).i) \wedge 0 \leq x \wedge x \leq n \wedge def(\, x < n \,)\}$
5. $\{ s = (\Sigma i : i \in [0, x).i) \wedge 0 \leq x \wedge x \leq n \wedge x < n \wedge$ 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 $P \Rightarrow Inv \wedge def(E)$ (dove P è la precondizione)
- 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 fosse solo $\{n \geq 0\}$, la 1) è falsa (invariante non vale).
- Possiamo renderla vera con un comando che inizializzi x ed 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 ;
while $x < n$ **do**
$x, s := x + 1, s + x$
endw
$\{s = (\Sigma i : i \in [0, n).i)\}$

Programmi Annotati

- Invece di indicare solo pre e postcondizioni di un programma, è utile aggiungere altre **annotazioni** per facilitarne la comprensione e rendere esplicito ciò che si deve dimostrare
- Esempio annotando il programma del lucido precedente con invariante, funzione di terminazione e altre asserzioni:

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

Esercizio: Somma con Incrementi Unitari

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

il programma annotato che calcola in z la somma dei valori di z ed n usando incrementi unitari

Esercizio: Somma con Incrementi Unitari

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

Applicando la regola $(WHILE)$ dobbiamo dimostrare che:

1. $z = A \wedge n = B \wedge B \geq 0 \Rightarrow Inv \wedge def(not(n = 0))$
2. $Inv \wedge n = 0 \Rightarrow z = A + B$
3. **Ipotesi di Invarianza**:
   $\{Inv \wedge \neg(n = 0)\}\ z := z + 1; n := n - 1\ \{Inv \wedge def(not(n = 0))\}$
4. **Ipotesi di Terminazione**: $Inv \Rightarrow n \geq 0$
5. **Ipotesi di Progresso**:
   $\{Inv \wedge \neg(n = 0) \wedge n = V\}\ z := z + 1; n := n - 1\ \{n < V\}$

Esercizio: Calcolo MCD

- Si consideri il seguente programma annotato:

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

- Dimostrare la correttezza facendo uso delle seguenti note proprietà dell'operatore *mcd*:

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

Sequenze (array) e Aggiornamento Selettivo

- Estendiamo il linguaggio di programmazione per manipolare *array* o *sequenze*
- Estendiamo sia la sintassi che la semantica
- Introduciamo una nuova regola di inferenza: aggiornamento selettivo

Sequenze: Sintassi

- Scriviamo $a : \mathit{array}[0, n)\ \mathit{of}\ \mathbf{T}$ per dire che a è una variabile di tipo:

  sequenza di elementi di tipo **T** con dominio $[0, n)$

  dove **T** può essere **int** o **bool**
- Il dominio di a viene indicato come $\mathit{dom}(a)$
- Al solito $a[E]$ denota l'elemento di a nella posizione E. Ad esempio:

$$a[4], a[a[0]], a[a[2] + a[4]] \ldots$$

- La sintassi delle espressioni diventa:

$$\mathit{Exp} ::= \mathit{Const} \mid \mathit{Ide} \mid \mathrm{Ide[Exp]} \mid (\mathit{Exp}) \mid \mathit{Exp}\ \mathit{Op}\ \mathit{Exp} \mid \mathit{not}\ \mathit{Exp}$$

Sequenze: Semantica delle Espressioni

- Ricordiamo che uno **stato** σ è una funzione $\sigma : \mathit{Ide} \to \mathbb{Z} \cup \mathbb{B}$
- Estendiamo il concetto di stato: se a è un array di tipo **T**,

$$\begin{cases} \sigma(a) : \mathit{dom}(a) \to \mathbb{B} & \text{se } \mathbf{T} = \mathbf{bool} \\ \sigma(a) : \mathit{dom}(a) \to \mathbb{Z} & \text{se } \mathbf{T} = \mathbf{int} \end{cases}$$

- Estendiamo la funzione di interpretazione semantica $\mathcal{E}$ alle espressioni del tipo $\mathit{Ide}[\mathit{Exp}]$:

$$\mathcal{E}(\mathit{Ide}[\mathit{Exp}], \sigma) = \mathcal{E}(\mathit{Ide}, \sigma)(\mathcal{E}(\mathit{Exp}, \sigma)) \text{ se } \mathcal{E}(\mathit{Exp}, \sigma) \in \mathit{dom}(\mathit{Ide})$$

- Esempio: $a : \mathit{array}[0, 4)\ \mathit{of}\ \mathbf{int}$

2	1	10	6

  - $$\sigma(a) : [0, 4) \to \mathbb{Z} = \{(0, 2), (1, 1), (2, 10), (3, 6)\}$$
  - $$\mathcal{E}(a[0], \sigma) = \mathcal{E}(a, \sigma)(\mathcal{E}(0, \sigma)) = \sigma(a)(0) = 2$$

Semantica Sequenze: Esempio

Valutazione di $a[a[0]+a[1]]$ nello stato σ tale che

$$\sigma(a) : [0,4) \to \mathbb{Z} = \{(0,2),(1,1),(2,10),(3,6)\}$$

dove $a : \mathit{array}[0,4)\ \mathit{of}\ \mathbf{int}$

2	1	10	6

$$\begin{aligned}
&\mathcal{E}(a[a[0]+a[1]],\sigma) = \\
&\mathcal{E}(a,\sigma)(\mathcal{E}(a[0]+a[1],\sigma) = \\
&\mathcal{E}(a,\sigma)(\mathcal{E}(a[0],\sigma)+\mathcal{E}(a[1],\sigma)) = \\
&\sigma(a)(\mathcal{E}(a,\sigma)(0)+\mathcal{E}(a,\sigma)(1)) = \\
&\sigma(a)(\sigma(a)(0)+\sigma(a)(1)) = \\
&\sigma(a)(2+1) = \\
&\sigma(a)(3) = \\
&6
\end{aligned}$$

Sequenze: Semantica

- **Attenzione**: Il valore di una espressione $Ide[Exp]$ non è sempre definito !!!!!
- Di conseguenza estendiamo la funzione *def*:

$$def(Ide[Exp]) = def(Exp) \wedge Exp \in dom(Ide)$$

- Riassumendo abbiamo i seguenti casi in cui un'espressione potrebbe non essere definita:
  - $def(E\ mod\ E') = def(E\ div\ E') = def(E) \wedge def(E') \wedge E' \neq 0$
  - $def(Ide[Exp]) = def(Exp) \wedge Exp \in dom(Ide)$
- Nota: le operazioni non possono essere applicate a sequenze, ma solo a singoli elementi di sequenze. Esempi:

$$\mathbf{a}[2] < \mathbf{b}[3], \mathbf{a}[2] * \mathbf{y} + \mathbf{x},\ \text{ma non } \mathbf{a} + \mathbf{b}!$$

Aggiornamento Selettivo: Sintassi

- Ogni elemento di una sequenza è una variabile, quindi può comparire a sinistra di un assegnamento
- Esempi:

$$a[3] := 5$$
$$a[2] := a[0] + a[1]$$

- Estendiamo il comando di assegnamento:

$$Ide\_List := Exp\_List$$

- Formalmente, cambia la definizione di *Ide_List* come segue:

$$Ide\_List ::= Ide \mid |de, Ide\_List \mid Ide[Exp] | Ide[Exp], Ide\_List$$

Aggiornamento Selettivo: Semantica

- Un comando di assegnamento del tipo $v[E] := E'$ è chiamato **aggiornamento selettivo**
- L'effetto è di transire, a partire da uno stato σ, allo stato

$$\sigma[w/v] \quad \text{dove} \quad w = v[d/i]$$

e $\mathcal{E}(E', \sigma) = d$, $\mathcal{E}(E, \sigma) = i$ tale che $i \in dom(v)$. Inoltre $v[d/i]$ indica l'array v modificato ovvero:

$$v[d/i](x) = \begin{cases} d \text{ se } i = x \\ v(x) \text{ altrimenti} \end{cases}$$

- **Nota**: il comando viene eseguito a patto che entrambe le espressioni E ed E' siano definite in σ, e che il valore di E in σ appartenga dominio dell' array v (ovvero $i \in dom(v)$)

Aggiornamento Selettivo: Assioma e Regola di Inferenza

- Adattando l'assioma (ASS) otteniamo il seguente assioma per aggiornamento selettivo $(AGG - SEL)$:

$$\{def(E) \wedge def(E') \wedge E \in dom(v) \wedge P[w/v]\}\ v[E] := E'\ \{P\}$$

$$\text{dove } w = v[E'/E]$$

- $w = v[E'/E]$ rappresenta l'array v modificato in modo tale che nella posizione E abbia il valore E'.
- le condizioni $def(E) \wedge def(E') \wedge E \in dom(v)$ garantiscono che le due espressioni E ed E' siano definite e che il valore di E appartenga al dominio dell' array v, in ogni stato σ che soddisfa la precondizione

Aggiornamento Selettivo: Regola di Inferenza

- Analogamente abbiamo anche la seguente regola derivata da $(AGG - SEL)$ e (PRE)

$$\frac{R \Rightarrow def(E) \wedge def(E') \wedge E \in dom(v) \wedge P[w/v]}{\{R\}\ v[E] := E'\ \{P\}}$$

dove $w = v[E'/E]$)

Esercizio: Aggiornamento selettivo

- Supponendo che $a : array\ [0, n)\ of\ int$ si verifichi la tripla

$$\{k \in dom(a) \wedge (\forall i.i \in dom(a) \wedge i \neq k \ \Rightarrow\ a[i] \geq 0)\}$$
$$a[k] := 5$$
$$\{(\forall i.i \in dom(a) \ \Rightarrow\ a[i] \geq 0)\}$$

- Applicando la regola $(AGG - SEL)$ otteniamo per $b = a[5/k]$

$$(\forall i.i \in dom(a) \ \Rightarrow\ a[i] \geq 0)[b/a] \wedge k \in dom(a) \wedge def(k) \wedge def(5)$$

- Dobbiamo dimostrare la seguente implicazione (usando $b = a[5/k]$):

$$k \in dom(a) \wedge (\forall i.i \in dom(a) \wedge i \neq k \ \Rightarrow\ a[i] \geq 0) \ \Rightarrow$$

$$(\forall i.i \in dom(b) \ \Rightarrow\ b[i] \geq 0) \wedge k \in dom(a)$$

Esercizio: Scansione di Sequenza

Si consideri il seguente programma annotato che conta il numero di elementi maggiori di zero che compaiono in un array elementi di un array $a : array[0, n)\ of\ int$.

$\{a : array\ [0, n)\ of\ int\}$
$x, c := 0, 0;$
$\{Inv : x \in [0, n] \wedge c = \#\{j : j \in [0, x) \mid a[j] > 0\}\}\{t : n - x\}$
while $x < n$ **do**
if $a[x] > 0$ **then** $c := c + 1$ **else skip fi**;
$x := x + 1$
endw
$\{Inv \wedge \neg(x < n)\}$
$\{c = \#\{j : j \in [0, n) \mid a[j] > 0\}\}$

- Scrivere e dimostrare la Condizione di Invarianza
- Scrivere e dimostrare la Condizione di Terminazione
- Scrivere e dimostrare la Condizione di Progresso

Esercizio: Incremento di Sequenza

Si consideri il seguente programma annotato che incrementa tutti gli elementi di un $a : array[0, n)\ of\ int$

$\{n \geq 0 \wedge (a : array[0, n)\ of\ int) \wedge (\forall k : k \in [0, n) \Rightarrow a[k] = V[k])\}$
$x := 0;$
$\{Inv : x \in [0, n] \wedge (\forall k : k \in [0, x) \ \Rightarrow\ a[k] = V[k] + 1) \wedge$
$\quad (\forall k.k \in [x, n) \Rightarrow a[k] = V[k])\}$
$\{t : n - x\}$
while $x < n$ **do**
$a[x] := a[x] + 1;\ x := x + 1$
endw
$\{Inv \wedge \neg(x < n)\}$
$\{(\forall k.k \in [0, n) \ \Rightarrow\ a[k] = V[k] + 1)\}$

- Scrivere e dimostrare la Condizione di Invarianza
- Scrivere e dimostrare la Condizione di Terminazione
- Scrivere e dimostrare la Condizione di Progresso