

Logica per la Programmazione

Lezione 14

- ▶ Comando Iterativo
- ▶ Sequenze (array) ed Aggiornamento Selettivo

Semantica informale del Comando Iterativo

- ▶ L'esecuzione del comando **while** E **do** C **endw** a partire da uno stato σ porta in σ se $\mathcal{E}(E, \sigma) = \mathbf{ff}$, 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 **comando iterativo** 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 E è sempre vera (ciclo infinito), oppure
 - ▶ la guardia non è valutabile ($\mathit{def}(E)$ 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

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

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

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

- ▶ 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:** $Inv : s = (\sum i : i \in [0, x].i) \wedge 0 \leq x \wedge x \leq n$
- ▶ **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
{ $s = (\sum i : i \in [0, n].i)$ }
  
```

Esempio di Comando Iterativo (2)

Per la Regola per il Comando Iterativo è sufficiente mostrare:

- $s = 0 \wedge x = 0 \wedge n \geq 0 \Rightarrow$
 $def(x < n) \wedge s = (\sum i : i \in [0, x]. i) \wedge 0 \leq x \wedge x \leq n$
- $s = (\sum i : i \in [0, x]. i) \wedge 0 \leq x \wedge x \leq n$
 $\wedge \neg(x < n) \Rightarrow s = (\sum i : i \in [0, n]. i)$
- $s = (\sum i : i \in [0, x]. i) \wedge 0 \leq x \wedge x \leq n \Rightarrow n - x \geq 0$
- $\{ s = (\sum i : i \in [0, x]. i) \wedge 0 \leq x \wedge x \leq n \wedge x < n \} x, s := x + 1, s + x$
 $\{ s = (\sum i : i \in [0, x]. i) \wedge 0 \leq x \wedge x \leq n \wedge def(x < n) \}$
- $\{ s = (\sum i : i \in [0, x]. i) \wedge 0 \leq x \wedge x \leq n \wedge x < n \wedge n - x = V \}$
 $x, s := x + 1, s + x \{ n - x < V \}$

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 inicializzi x ed s**

```

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

```

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

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** (in particolare nel caso di un comando iterativo)
- ▶ Esempio annotando il programma del lucido precedente con **invariante**, **funzione di terminazione** e altre asserzioni:

```

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

```

Esercizio: Somma con Incrementi Unitari

```

{z = A ∧ n = B ∧ B ≥ 0}
{Inv : z + n = A + B ∧ n ≥ 0} {t : n}
while not (n = 0) do
z := z + 1; n := n - 1
endw
{Inv ∧ 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 ∧ n = B ∧ B ≥ 0}
{Inv : z + n = A + B ∧ n ≥ 0}{t : n}
while not (n = 0) do
z := z + 1; n := n - 1
endw
{Inv ∧ 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:
 $\{Inv \wedge \neg(n = 0) \wedge n = V\} z := z + 1; n := n - 1 s \{n < V\}$

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 l'ipotesi di invarianza e di progresso 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** (sia le espressioni che i **comandi**) per manipolare *array* o *sequenze*
- ▶ Estendiamo sia la **sintassi** che la **semantica**
- ▶ Introduciamo una nuova regola di inferenza: **aggiornamento selettivo**

Sequenze: Sintassi

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

sequenza di elementi di tipo \mathbf{T} con dominio $[0, n)$

dove \mathbf{T} può essere **int** o **bool**

- ▶ Il dominio di a viene indicato come $\text{dom}(a)$
- ▶ Al solito $a[E]$ denota l'elemento di a nella posizione E . Ad esempio:

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

- ▶ La sintassi delle espressioni diventa:

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

Sequenze: Semantica delle Espressioni

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

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

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

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

- ▶ Esempio: $a : \mathbf{array}[0, 4] \text{ of } \mathbf{int}$

2	1	10	6
---	---	----	---



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



$$\mathcal{E}(a[0], \sigma) = \mathcal{E}(a, \sigma)(\mathcal{E}(0, \sigma)) = \sigma(a)(0) = 2$$

Semantica Sequenze: Esempio

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

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

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

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:
 - ▶ $def(E \text{ mod } E') = def(E \text{ div } E') = def(E) \wedge def(E') \wedge E' \neq 0$
 - ▶ $def(Ide[Exp]) = def(Exp) \wedge Exp \in dom(Ide)$
- ▶ Nota: le operazioni non possono essere applicate a sequenze, ma solo a singoli elementi di sequenze. Esempi:

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

Aggiornamento Selettivo: Sintassi

- ▶ Ogni elemento di una sequenza è una variabile e 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:

$$Ide_List ::= Ide \mid Ide, Ide_List \mid Ide[Exp] \mid Ide[Exp], Ide_List$$

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 σ' dove

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

- ▶ dove $\mathcal{E}(E', \sigma) = d$, $\mathcal{E}(E, \sigma) = i$ tale che $i \in \text{dom}(v)$
- ▶ dove $v[d/i]$ 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 $i \in \text{dom}(v)$)

Aggiornamento Selettivo: Assioma e Regola di Inferenza

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

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

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

- ▶ qui l'array $w = v[E'/E]$ rappresenta l'array v modificato in modo tale che nella posizione E abbia il valore E'
- ▶ le condizioni $\text{def}(E) \wedge \text{def}(E') \wedge E \in \text{dom}(v)$ 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)*

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

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

Esercizio: Aggiornamento selettivo

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

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

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

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

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

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

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

Esercizio: Scansione di Sequenza

Si consideri il seguente programma annotato che conta il numero di elementi maggiori di zero che compaiono in un array

$a : \text{array}[0, n) \text{ of } \text{int}$.

```

{a : array [0, n) of int}
x, c := 0, 0;
{x = 0 ∧ c = 0 ∧ n ≥ 0}
{Inv : x ∈ [0, n] ∧ c = #{j : j ∈ [0, x) | 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 ∧ ¬(x < n)}
{c = #{j : j ∈ [0, n) | 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 $a : \text{array}[0, n) \text{ of } \text{int}$

```

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

```

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