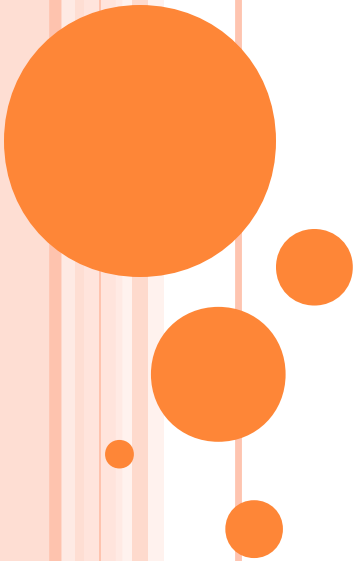

ESEMPI DI TRIPLE DI HOARE

Corso di Logica per la Programmazione

A.A. 2010/11

Andrea Corradini, Paolo Mancarella

ESEMPIO DI SEQUENZA DI COMANDI

Verificare la tripla: $\{x >= y - 1\}\ x := x+1; y := y - 1\ \{x > y\}$

- Per la **Regola per la Sequenza**, dobbiamo trovare una asserzione **R** e verificare le seguenti triple:

1) $\{x \geq y - 1\}\ x := x+1\ \{\mathbf{R}\}$
2) $\{\mathbf{R}\}\ y := y - 1\ \{x > y\}$

- Per determinare **R**, usiamo l'**Assioma dell'Assegnamento** nella seconda tripla. Sappiamo infatti che la seguente è verificata:

$$\{def(y-1) \wedge (x > y)[y-1/y]\}\ y := y - 1\ \{x > y\}$$

- Quindi fissiamo **R** = $def(y-1) \wedge x > y-1$. Resta da verificare:

$$\{x \geq y - 1\}\ x := x+1\ \{def(y-1) \wedge x > y-1\}$$

- Usando la **Regola per l'Assegnamento**, basta dimostrare:

$$x \geq y - 1 \Rightarrow def(x+1) \wedge (def(y-1) \wedge x > y-1)[x+1/x]$$

- **Esercizio:** completare la dimostrazione

SEQUENZA CON **VARIABILI DI SPECIFICA**

Determinare E in modo che la tripla sia verificata:

$$\{x = N \wedge y = M\}\ t := E;\ \ x := y;\ \ y := t\ \ \{x = M \wedge y = N\}$$

- Per la **Regola per la Sequenza**, dobbiamo trovare **R1** e **R2** tali che:

1) $\{x = N \wedge y = M\}\ t := E\ \{\mathbf{R1}\}$
2) $\{\mathbf{R1}\}\ x := y\ \{\mathbf{R2}\}$
3) $\{\mathbf{R2}\}\ y := t\ \ \{x = M \wedge y = N\}$

Nota: M e N sono **variabili di specifica**: non possono essere usate nei comandi

- Per determinare **R2**, usiamo l'**Assioma dell'Assegnamento** in 3):

$$\{def(t) \wedge x = M \wedge y = N)[t/y]\}\ y := t\ \ \{x = M \wedge y = N\}$$

- Fissiamo **R2 =** $(x = M \wedge t = N)$. Per **R1**, usiamo l'assioma in 2):

$$\{def(y) \wedge (x = M \wedge t = N)[y/x]\}\ x := y\ \{x = M \wedge t = N\}$$

- Fissiamo **R1 =** $(y = M \wedge t = N)$. Resta da verificare:

$$\{x = N \wedge y = M\}\ t := E\ \{y = M \wedge t = N\}$$

- Usando la **Regola per l'Assegnamento**, basta trovare un E tale che:

$$x = N \wedge y = M\ \Rightarrow def(E)\ \wedge (y = M \wedge t = N)[E/t]$$

ESEMPIO DI COMANDO CONDIZIONALE

Verificare la seguente tripla:

$\{x \geq 0 \wedge y > 0\}$
if $x \text{ div } y > 0$
then $z := x$
else $z := y$
fi
$\{z = \max(x, y)\}$

- Per la **Regola per il Condizionale**, dobbiamo mostrare:

1) $x \geq 0 \wedge y > 0 \Rightarrow \mathit{def}(x \text{ div } y > 0)$
2) $\{x \geq 0 \wedge y > 0 \wedge x \text{ div } y > 0\}\ z := x\ \{z = \max(x, y)\}$
3) $\{x \geq 0 \wedge y > 0 \wedge x \text{ div } y \leq 0\}\ z := y\ \{z = \max(x, y)\}$

- Il punto 1) è facile.
- Per il 2), usando la **Regola per l'Assegnamento** occorre mostrare:
  - $x \geq 0 \wedge y > 0 \wedge x \text{ div } y > 0 \Rightarrow \mathit{def}(x) \wedge (z = \max(x, y))[x/z]$
- Analogamente, per il 3) occorre mostrare:
  - $x \geq 0 \wedge y > 0 \wedge x \text{ div } y \leq 0 \Rightarrow \mathit{def}(y) \wedge (z = \max(x, y))[y/z]$

- **Esercizio**: completare la dimostrazione

ESERCIZIO
Verificare la seguente tripla:

```
{ x = 5 }
   x := 3 ;
   if  ( x = 3 )
         then      y := 7
         else      y := 5
   fi
{ x = 3  ∧ y = 7 }
```

- Per la **Regola per la Sequenza,** dobbiamo trovare una asserzione **R** tale che:

1) { x = 5 } x := 3 { **R** }
2) {**R**} **if** (x=3) **then** y := 7 **else** y := 5 **fi** {x=3 ∧ y=7}

- A differenza di altri esempi, qui non possiamo usare l'assioma dell'assegnamento per trovare **R**.
- Un candidato naturale per **R** è {x = 3}. Infatti con questa asserzione (e con un po' di attenzione) sia 1) che 2) sono facilmente dimostrabili.
- **Esercizio**: completare la dimostrazione

REGOLA PER IL COMANDO ITERATIVO

$$P \Rightarrow Inv \wedge def(E) \qquad Inv \wedge \sim E \Rightarrow Q \qquad Inv \Rightarrow t \geq 0$$

$$\{Inv \wedge E\}\ C\ \{Inv \wedge def(E)\} \qquad \{Inv \wedge E \wedge t = V\}\ C\ \{t < V\}$$

$$\{P\}\ \textbf{while}\ E\ \textbf{do}\ C\ \textbf{endw}\ \{Q\}$$

- 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

- Come si arriva a questa regola? Si veda il Paragrafo 4.7 della dispensa sulle Triple di Hoare

ESEMPIO DI COMANDO ITERATIVO

- Usando come **invariante**

  $Inv : s = (\Sigma\, i : i \in [0, x).\ i) \ \wedge 0 \le x \wedge x \le n$

e come **funzione di terminazione**

  $t : n - x$

verificare la tripla nel riquadro a destra.

$\{s = 0 \ \wedge x = 0 \wedge n \ge 0\}$
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:

1) $s = 0 \wedge x = 0 \wedge n \ge 0 \Rightarrow def(x<n) \wedge s = (\Sigma\, i : i \in [0, x).\ i) \ \wedge 0 \le x \wedge x \le n$

2) $s = (\Sigma\, i : i \in [0, x).\ i) \ \wedge 0 \le x \wedge x \le n \wedge \sim(x < n) \Rightarrow s = (\Sigma\, i: i \in [0, n).\ i)$

3) $s = (\Sigma\, i : i \in [0, x).\ i) \ \wedge 0 \le x \wedge x \le n \wedge x < n \Rightarrow n \dot{-} x \ge 0$

4) $\{s = (\Sigma\, i : i \in [0, x).\ i) \ \wedge 0 \le x \wedge x \le n \wedge x < n\}$ x, s := x+1, s+x
   $\{s = (\Sigma\, i : i \in [0, x).\ i) \ \wedge 0 \le x \wedge x \le n \wedge def(x < n)\}$

5) $\{s = (\Sigma\, i : i \in [0, x).\ i) \ \wedge 0 \le x \wedge x \le n \wedge x < n \wedge n - x = V\}$
   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 $P \Longrightarrow Inv \wedge def(E)$
- In questo caso si può inserire un **comando di inizializzazione** C_I tale che $\{P\}\ C_I\ \{Inv \wedge def(E)\}$
- **Esempio. N**ella tripla vista, se la precondizione è solo $\{n \geq 0\}$, la 1) 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

- Invece di indicare solo pre- e post-condizioni di un programma, come a destra, è utile aggiungere altre annotazioni per facilitarne la comprensione.
- Per esempio, annotiamo il programma come sotto con invariante, funzione di terminazione e altre asserzioni. Questo rende esplicito cosa bisogna dimostare:

1) $s = 0 \wedge x = 0 \wedge n \geq 0 \Rightarrow \mathit{def}(x<n) \wedge \mathit{Inv}$
2) $\mathit{Inv} \wedge \sim(x < n) \Rightarrow s = (\Sigma\ i: i \in [0, n).\ i)$
3) $\mathit{Inv} \wedge x < n \Rightarrow n - x \geq 0$
4) $\{\mathit{Inv} \wedge x<n\}$ x,s := x+1,s+x $\{\mathit{Inv} \wedge \mathrm{def}(x<n)\}$
5) $\{\mathit{Inv} \wedge x < n \wedge n-x = V\}$ x,s := x+1,s+x $\{n-x < V\}$

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

SEQUENZE: SINTASSI

- Estendiamo il linguaggio per usare *array* o *sequenze*
- Scriviamo **v : array [a,b) of T** per dire che **v** è una variabile di tipo:
  "sequenza di elementi di tipo **T** con dominio **[a,b)**",
  dove **T** può essere **int** o **bool**
- Il dominio di **v** viene indicato come *dom*(**v**)
- Scriviamo **v[i]** per denotare l'**i**-esimo elemento di **v**
- La **sintassi** delle espressioni diventa:

Exp ::= Const | Id | Ide[Exp] | (Exp) | Exp Op Exp | not Exp

SEQUENZE: SEMANTICA

- Ricordiamo che uno **stato** σ è una funzione

$$\sigma : \text{Ide} \rightarrow B \cup Z$$

- Estendiamo il concetto di stato: se **v** è un array di tipo **T**,

$\sigma(\mathbf{v})$: $dom(\text{v}) \rightarrow \mathbf{B}$ se **T = bool**

$\sigma(\mathbf{v})$: $dom(\text{v}) \rightarrow \mathbf{Z}$ se **T = int**

- Estendiamo la funzione di **interpretazione semantica**:

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

- Ide[Exp] non è sempre definito: estendiamo la funzione *def*

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

- Nota: le operazioni non possono essere applicate a sequenze, ma solo a singoli elementi di sequenze.
  Es: **a[2] < b[3]**, **a[2] * y + x** , ma non **a + b** !

AGGIORNAMENTO SELETTIVO

- Ogni elemento di una sequenza è una variabile, quindi può comparire a sinistra di un assegnamento. Es: **a[3] := 5**
- La semantica di questo comando (**assegnamento selettivo**) è data dal seguente assioma *(ass-sel):*

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

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

- Si può usare anche la seguente regola derivata:

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

ESEMPIO DI AGGIORNAMENTO SELETTIVO

- Si verifichi la tripla:

  $\{k \in dom(v) \wedge (\forall\ i : i \in dom(v) \wedge i \neq k\ .\ v[i] > 0)\}$

  $v[k] := 3$

  $\{(\forall\ i : i \in dom(v)\ .\ v[i] > 0)\}$

- Applicando la regola, è sufficiente dimostrare:

  $k \in dom(v) \wedge (\forall\ i : i \in dom(v) \wedge i \neq k\ .\ v[i] > 0) \Rightarrow$
  $def(k) \wedge def(3) \wedge k \in dom(v) \wedge (\forall\ i : i \in dom(v)\ .\ v'[i] > 0)$

  dove $v' = v[3/k]$

- **Esercizio:** si completi la dimostrazione

ESERCIZIO: SOMMA CON INCREMENTI UNITARI

- Si consideri il programma annotato che calcola in **z** la somma dei valori di **z** ed **n** usando incrementi unitari. Si noti l'uso di variabili di specifica.
- Dimostrarne la correttezza.
- Per la **Regola per il Comando Iterativo**, usando le annotazioni occorre dimostrare:

$\{z = A \wedge n = B \wedge B \geq 0\}$

$\{Inv : z+n = A+B \wedge n \geq 0\}\ \{t : n\}$

while not (n = 0) **do**

z := z+1; n:= n-1

endw

$\{Inv \wedge n = 0\}$

$\{z = A+B\}$

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

ESERCIZIO: SCANSIONE DI SEQUENZA

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

{a : array [0, n) of int }
x, c := 0, 0;
{*Inv* : c = #{j | j ∈ [0, x) ∧ a[j] > 0 } ∧ x∈ [0, n] } {*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