

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: $\{x \geq y - 1\} x := x+1; y := y - 1 \{x > y\}$

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

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

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

- 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) $\{x = N \wedge y = M\} t := E \{R1\}$

2) $\{R1\} x := y \{R2\}$

3) $\{R2\} y := t \{x = M \wedge y = N\}$

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:

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

1) $x \geq 0 \wedge y > 0 \Rightarrow def(x \text{ div } y > 0)$

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:

● $x \geq 0 \wedge y > 0 \wedge x \text{ div } y > 0 \Rightarrow def(x) \wedge (z = \max(x, y))[x/z]$

○ Analogamente, per il 3) occorre mostrare:

● $x \geq 0 \wedge y > 0 \wedge x \text{ div } y \leq 0 \Rightarrow def(y) \wedge (z = \max(x, y))[y/z]$

○ **Esercizio:** completare la dimostrazione

```
{x ≥ 0 ∧ y > 0}
  if x div y > 0
    then z := x
    else z := y
  fi
{z = max(x, y)}
```



ESERCIZIO

Verificare la seguente tripla:

- Per la **Regola per la Sequenza**, dobbiamo trovare una asserzione **R** tale che:
 - 1) $\{ x = 5 \} x := 3 \{ \mathbf{R} \}$
 - 2) $\{ \mathbf{R} \} \text{ if } (x=3) \text{ then } y := 7 \text{ else } y := 5 \text{ fi } \{ x=3 \wedge y=7 \}$
- 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

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



REGOLA PER IL COMANDO ITERATIVO

$$P \Rightarrow Inv \wedge def(E) \quad Inv \wedge \sim E \Rightarrow Q \quad Inv \Rightarrow t \geq 0$$
$$\{Inv \wedge E\} C \{Inv \wedge def(E)\} \quad \{Inv \wedge E \wedge t = V\} C \{t < V\}$$

{P} while E do C 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**

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

e come **funzione di terminazione**

$$t : n - x$$

verificare la tripla nel riquadro a destra.

```
{s = 0 ∧ x = 0 ∧ n ≥ 0}
while x < n do
    x, s := x+1, s+x
endw
{s = (∑ i : i ∈ [0, n). i)}
```

- Per la **Regola per il Comando Iterativo** è sufficiente mostrare:

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

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

$$3) s = (\sum i : i \in [0, x). i) \wedge 0 \leq x \wedge x \leq n \Rightarrow n - x \geq 0$$

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

$$5) \{s = (\sum i : i \in [0, x). i) \wedge 0 \leq x \wedge x \leq n \wedge x < n \wedge n - x = V\}$$
$$x, s := x+1, s+x \{n - x < V\}$$

Esercizio: completare la dimostrazione



COMANDO DI INIZIALIZZAZIONE

- Spesso la preconditione di una tripla con un **while** non è sufficiente per soddisfare la condizione $P \Rightarrow Inv \wedge def(E)$
- 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 preconditione 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 ≥ 0} ??  
while x < n do  
    x, s := x+1, s+x  
endw  
{s = (∑ i: i ∈ [0, n). i)}
```

```
{n ≥ 0}  
x, s := 0, 0 ;  
{s = 0 ∧ x = 0 ∧ n ≥ 0}  
while x < n do  
    x, s := x+1, s+x  
endw  
{s = (∑ i: i ∈ [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 dimostrare:

$$1) s = 0 \wedge x = 0 \wedge n \geq 0 \Rightarrow \text{def}(x < n) \wedge \text{Inv}$$

$$2) \text{Inv} \wedge \sim(x < n) \Rightarrow s = (\sum i: i \in [0, n). i)$$

$$3) \text{Inv} \Rightarrow n - x \geq 0$$

$$4) \{ \text{Inv} \wedge x < n \} x, s := x+1, s+x \\ \{ \text{Inv} \wedge \text{def}(x < n) \}$$

$$5) \{ \text{Inv} \wedge x < n \wedge n - x = V \} x, s := x+1, s+x \\ \{ n - x < V \}$$

```
{ n ≥ 0 }
```

```
x, s := 0, 0 ;
```

```
while x < n do
```

```
    x, s := x+1, s+x
```

```
endw
```

```
{ s = (∑ i: i ∈ [0, n). i) }
```

```
{ n ≥ 0 }
```

```
x, s := 0, 0 ;
```

```
{ s = 0 ∧ x = 0 ∧ n ≥ 0 }
```

```
{ Inv : s = (∑ i : i ∈ [0, x). i) ∧ } \\ 0 ≤ x ∧ x ≤ n } { t: n - x }
```

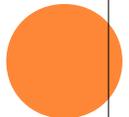
```
while x < n do
```

```
    x, s := x+1, s+x
```

```
endw
```

```
{ Inv ∧ ~(x < n) }
```

```
{ s = (∑ i: i ∈ [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:

$Exp ::= Const \mid Id \mid Ide[Exp] \mid (Exp) \mid Exp \ Op \ Exp \mid not \ Exp$



SEQUENZE: SEMANTICA

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

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

- Estendiamo il concetto di stato: se \mathbf{v} è un array di tipo \mathbf{T} ,

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

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

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

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

- $\text{Ide}[\text{Exp}]$ non è sempre definito: estendiamo la funzione *def*

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

- Nota: le operazioni non possono essere applicate a sequenze, ma solo a singoli elementi di sequenze.

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



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:

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



ESEMPIO DI AGGIORNAMENTO SELETTIVO

- Si verifichi la tripla:

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

$$v[k] := 3$$

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

- Applicando la regola, è sufficiente dimostrare:

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

$$\text{dove } v' = v[3/k]$$

- **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 ∧ 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}
```

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

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



ESERCIZIO: MODIFICA DI SEQUENZA

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

$$\{n \geq 0 \wedge a : \text{array } [0, n) \text{ of int} \wedge (\forall k: k \in [0, n) \Rightarrow a[k] = V[k])\}$$

$x := 0;$

$$\{Inv : x \in [0, n] \wedge (\forall k: k \in [0, x) \Rightarrow a[k] = V[k] + 1) \wedge$$
$$(\forall k. k \in [x, n) \Rightarrow a[k] = V[k])\} \{t : n - x\}$$

while $x < n$ **do**

$$a[x] := a[x] + 1; \quad x := x + 1$$

endw

$$\{Inv \wedge \sim(x < n)\}$$
$$\{(\forall k. k \in [0, n) \Rightarrow a[k] = V[k] + 1)\}$$

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



- Condizione di invarianza

$$\{x \in [0, n] \wedge (\forall k. k \in [0, x) \Rightarrow a[k] = V[k] + 1) \wedge$$
$$(\forall k. k \in [x, n) \Rightarrow a[k] = V[k]) \wedge x < n\}$$

$$\mathbf{a[x] := a[x] + 1; \quad x := x + 1}$$

$$\{x \in [0, n] \wedge (\forall k. k \in [0, x) \Rightarrow a[k] = V[k] + 1) \wedge$$
$$(\forall k. k \in [x, n) \Rightarrow a[k] = V[k]) \wedge \cancel{\text{def}(x < n)} \}$$

- Per la regola della sequenza dobbiamo determinare **R** in modo che

$$\{x \in [0, n) \wedge (\forall k. k \in [0, x) \Rightarrow a[k] = V[k] + 1) \wedge$$
$$(\forall k. k \in [x, n) \Rightarrow a[k] = V[k]) \}$$

$$\mathbf{a[x] := a[x] + 1;$$

$$\{\mathbf{R}\}$$

$$\mathbf{x := x + 1}$$

$$\{x \in [0, n] \wedge (\forall k. k \in [0, x) \Rightarrow a[k] = V[k] + 1) \wedge$$
$$(\forall k. k \in [x, n) \Rightarrow a[k] = V[k]) \}$$



- In questo caso **R** viene determinata dall'assioma per l'assegnamento

$$\{x \in [0, n) \wedge (\forall k. k \in [0, x) \Rightarrow a[k] = V[k] + 1) \wedge$$

$$(\forall k. k \in [x, n) \Rightarrow a[k] = V[k])\}$$

$$\mathbf{a[x] := a[x] + 1;}$$

$$\{\text{def}(x+1) \wedge x+1 \in [0, n] \wedge (\forall k. k \in [0, x+1) \Rightarrow a[k] = V[k] + 1) \wedge$$

$$(\forall k. k \in [x+1, n) \Rightarrow a[k] = V[k]) \}$$

$$\mathbf{x := x + 1}$$

$$\{x \in [0, n] \wedge (\forall k. k \in [0, x) \Rightarrow a[k] = V[k] + 1) \wedge$$

$$(\forall k. k \in [x, n) \Rightarrow a[k] = V[k]) \}$$

- Ci resta da dimostrare la tripla

$$\{x \in [0, n) \wedge (\forall k. k \in [0, x) \Rightarrow a[k] = V[k] + 1) \wedge$$

$$(\forall k. k \in [x, n) \Rightarrow a[k] = V[k])\}$$

$$\mathbf{a[x] := a[x] + 1;}$$

$$\{x+1 \in [0, n] \wedge (\forall k. k \in [0, x] \Rightarrow a[k] = V[k] + 1) \wedge$$

$$(\forall k. k \in (x, n) \Rightarrow a[k] = V[k]) \}$$



○ Avendo a che fare con un aggiornamento selettivo, dobbiamo mostrare

$$x \in [0, n) \wedge (\forall k. k \in [0, x) \Rightarrow a[k] = V[k] + 1) \wedge$$

$$(\forall k. k \in [x, n) \Rightarrow a[k] = V[k])$$

\Rightarrow

$$x+1 \in [0, n] \wedge (\forall k. k \in [0, x] \Rightarrow b[k] = V[k] + 1) \wedge$$

$$(\forall k. k \in (x, n) \Rightarrow b[k] = V[k]) \wedge x \in [0, n) \wedge \text{def}(a[x]+1)$$

con $b = a[a[x]+1 / x]$

Alcune osservazioni:

(1) $\text{def}(a[x]+1) \equiv x \in [0, n)$ che quindi possiamo omettere

(2) $(\forall k. k \in [0, x) \vee k \in (x, n) \Rightarrow b[k] = a[k])$

(3) $b[x] = a[x] + 1$



$$x+1 \in [0, n] \wedge (\forall k. k \in [0, x] \Rightarrow b[k] = V[k] + 1) \wedge$$

$$(\forall k. k \in (x, n) \Rightarrow b[k] = V[k]) \wedge x \in [0, n)$$

$$\equiv \{ \mathbf{Ip}: x \in [0, n) \}$$

$$(\forall k. k \in [0, x] \Rightarrow b[k] = V[k] + 1) \wedge$$

$$(\forall k. k \in (x, n) \Rightarrow b[k] = V[k])$$

$$\equiv \{ \text{Intervallo, } x \in [0, x] \}$$

$$(\forall k. k \in [0, x) \Rightarrow b[k] = V[k] + 1) \wedge b[x] = V[x] + 1 \wedge$$

$$(\forall k. k \in (x, n) \Rightarrow b[k] = V[k])$$

$$\equiv \{ \text{Osservazioni (2) e (3) precedenti} \}$$

$$(\forall k. k \in [0, x) \Rightarrow a[k] = V[k] + 1) \wedge a[x] + 1 = V[x] + 1 \wedge$$

$$(\forall k. k \in (x, n) \Rightarrow a[k] = V[k])$$

$$\equiv \{ \mathbf{Ip}: (\forall k. k \in [0, x) \Rightarrow a[k] = V[k] + 1) \}$$

$$a[x] + 1 = V[x] + 1 \wedge (\forall k. k \in (x, n) \Rightarrow a[k] = V[k])$$

$$\equiv \{ \text{calcolo, Intervallo} \}$$

$$(\forall k. k \in [x, n) \Rightarrow a[k] = V[k])$$



ESERCIZIO: Calcolo del fattoriale

- Si consideri la seguente specifica

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

$\{n > 0\}$

$f, x := 1, 1;$

$\{Inv : ?\} \{t : ?\}$

while $x \leq n$ **do**

$f, x := f * x, x + 1;$

endw

$\{Inv \wedge \sim(x \leq n)\}$

$\{f = n!\}$

- Come determinare l'invariante e la funzione di terminazione?



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

$\{n > 0\}$	x	f
f, x := 1, 1;	1	1
$\{Inv : ?\} \{t : ?\}$	2	1
while x <= n do	3	2
f, x := f * x, x + 1;	4	6
endw	5	24
$\{Inv \wedge \sim(x \leq n)\}$	6	120
$\{f = n!\}$		

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

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

- Scegliamo questa formula come invariante candidata: non riusciamo però a dimostrare

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

```
{n>0}
  f, x:= 1, 1;
  {Inv : f = (x - 1)! ∧ x ∈ [0, n+1]} {t : ?}
while x ≤ n do
  f, x := f * x, x + 1;
endw
  {Inv ∧ ¬(x ≤ n)}
  {f = n!}
```

- In questo modo

$$Inv \wedge \sim(x \leq n) \Rightarrow f = n!$$

- Dimostrazione per esercizio



Ipotesi di invarianza

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

$$f * x = x! \wedge x+1 \in [0, n+1]$$

$$\equiv \{ \mathbf{Ip}: f = (x-1)! \}$$

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

$$\equiv \{ \text{def. fattoriale} \}$$

$$x+1 \in [0, n+1]$$

←

$$x \in [0, n+1)$$

- Abbiamo dimostrato

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

che sappiamo essere equivalente a quanto richiesto dalla tripla (perché?)

- La funzione di terminazione è lasciata per esercizio.

ESERCIZIO: Calcolo MCD

- Si consideri il seguente programma annotato

```
{x = A ∧ y = B ∧ A > 0 ∧ B > 0}
```

```
{Inv : x > 0 ∧ y > 0 ∧ 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) }
```

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

$$mcd(v,w) = v \quad \text{se } v=w$$

$$mcd(v,w) = mcd(v-w,w) \quad \text{se } v>w$$

$$mcd(v,w) = mcd(v,w-v) \quad \text{se } v<w$$



ESERCIZIO: FUNZIONE DI TERMINAZIONE

- Si consideri il seguente programma annotato

```
{x > 0 ∧ y > 0}
```

```
{Inv: T} {t : ?}
```

```
while x <> y do
```

```
    if x > y then x := x - 1; else x := x + 1; fi
```

```
endw
```

```
{x = y}
```

- Quale può essere una buona funzione di terminazione?
- Osserviamo che il corpo del ciclo fa sì che, ad ogni iterazione, diminuisca la *distanza* tra i valori delle variabili x e y
- Dunque, una buona t è

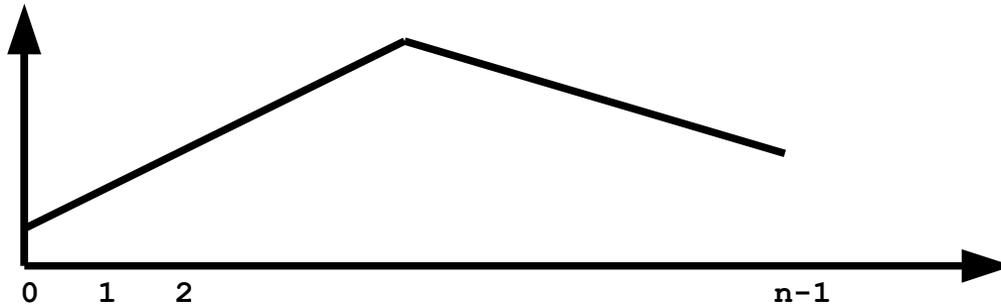
$$|x - y|$$

- Dimostrare progresso e terminazione

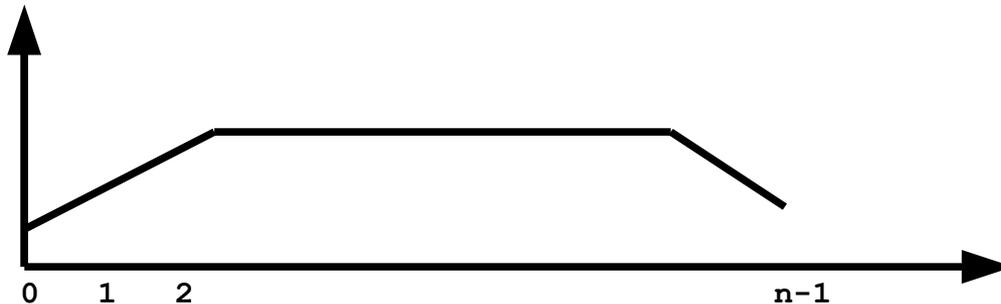


ESERCIZI SU FORMALIZZAZIONE DI SEQUENZE

- Siano \mathbf{a}, \mathbf{b} : array $[0, n)$ of \mathbf{nat} . Si formalizzino i seguenti enunciati, usando l'interpretazione standard su naturali e sequenze di naturali.
- La sequenza \mathbf{b} è l'inverso della sequenza \mathbf{a}
- La sequenza \mathbf{a} è strettamente crescente fino alla posizione $k \in [0, n)$
- \mathbf{a} è una sequenza che ha il seguente andamento:



- \mathbf{b} è una sequenza che ha il seguente andamento:



ALTRI ESERCIZI DI FORMALIZZAZIONE

1. l'array **b** è palindromo
2. l'array **a** ha un solo minimo locale
3. **a** ha tutti elementi distinti
4. **b** è l'array **a** ordinato in senso crescente

