Verifica della Correttezza dei Programmi Iterativi

Interpretazione delle terne di Hoare: correttezza parziale

Si dice che il programma p è *parzialmente corretto* rispetto alla precondizione Q e alla postcondizione R, cioè che la terna di Hoare

$$\{ \; Q \; \}$$

$$p$$

$$\{ \; R \; \}$$

è valida, se e solo se vale la seguente implicazione

$$\forall M \in Mem \; . \; \mathcal{I}[[Q]] M \;\wedge\; M \in Dom(\mathcal{C}[[p]]) \quad \Rightarrow \quad \mathcal{I}[[R]] (\mathcal{C}[[p]] M)$$

Intuitivamente, considerata una generica assegnazione di valori alle variabili del programma e ad altri eventuali simboli che compaiono come variabili libere nelle asserzioni ($\forall M \in Mem$), se l'interpretazione dell'asserzione Q è vera per M ($\mathcal{I}[[Q]]M$) e se il comando p è definito e termina se eseguito a partire dalla memoria M ($M \in Dom(\mathcal{C}[[p]])$), allora nella nuova memoria che risulta dalla transizione operata da p ($M' = \mathcal{C}[[p]]M$) deve essere vera l'interpretazione dell'asserzione R ($\mathcal{I}[[R]]M'$). È opportuno osservare che si assume una particolare interpretazione $\mathcal{I}$ delle asserzioni (in genere quella naturale).

Nei casi più semplici M può essere pensata come la memoria nel senso del corrispondente modello operazionale SMC e la condizione $M \in Dom(\mathcal{C}[[p]])$ corrisponde ad assumere che tutte le variabili del programma p appartengano a Dom(M) e che nel modello SMC sia definita una computazione compiuta con la seguente forma:

$$< \lambda, \; M, \; [p] > \quad \xrightarrow{*} \quad < \lambda, \; M', \; \lambda >$$

in base alla quale $M' = \mathcal{C}[[p]]M$ per definizione. Più in generale M potrà assegnare dei valori anche a variabili che compaiono nelle asserzioni ma non nel programma.

Interpretazione delle terne di Hoare: correttezza totale

Si dice che il programma p è *totalmente corretto* rispetto alla precondizione Q e alla postcondizione R, cioè che la terna

$$\{ \; Q \; \}$$

$$p$$

$$\{ \; R \; \}_{tot}$$

è valida, se e solo se vale la seguente implicazione

$$\forall M \in Mem \;.\; \mathcal{I}[[Q]]M \quad \Rightarrow \quad M \in Dom(\mathcal{C}[[p]]) \;\wedge\; \mathcal{I}[[R]](\mathcal{C}[[p]]M)$$

Intuitivamente, considerata una generica assegnazione di valori alle variabili del programma e ad altri eventuali simboli che compaiono come variabili libere nelle asserzioni ($\forall M \in Mem$), se l'interpretazione dell'asserzione Q è vera per M ($\mathcal{I}[[Q]]M$), allora il comando p è definito e termina a partire da M ($M \in Dom(\mathcal{C}[[p]])$) e inoltre nella memoria che risulta dalla transizione operata da p ($M' = \mathcal{C}[[p]]M$) è vera l'interpretazione dell'asserzione R ($\mathcal{I}[[R]]M'$).

La condizione $M \in Dom(\mathcal{C}[[p]])$ corrisponde, nel modello SMC, all'esistenza di una computazione con la seguente forma:

$$< \lambda, \; M, \; [p] > \quad \xrightarrow{*} \quad < \lambda, \; M', \; \lambda >$$

in base alla quale $M' = \mathcal{C}[[p]]M$. Ma questa volta una simile computazione non è presunta, ma "garantita" nell'ipotesi che Q sia vera in M (una condizione, quindi, più forte).

Regole delle dimostrazioni di correttezza: correttezza parziale

Le regole che si applicano per dimostrare la correttezza parziale di un programma rispetto alle relative specifiche (precondizione e postcondizione) corrispondono al calcolo della precondizione più debole nel caso di comandi elementari, cioè "skip" e assegnazione, mentre le dimostrazioni relative ai programmi strutturati sono ricondotte alle rispettive componenti. Inoltre, si possono utilizzare regole di rafforzamento della precondizione o indebolimento della postcondizione.

Regola per il comando "skip" – la seguente terna è valida:

$$\{ \quad Q \quad \}$$

skip

$$\{ \quad Q \quad \}$$

Regola per il comando di assegnazione – la seguente terna è valida:

$$\{ \quad Q^e_x \quad \}$$

x := e

$$\{ \quad Q \quad \}$$

(con Q^e_x si intende la sostituzione di tutte le occorrenze di x in Q con e, seguita da eventuale semplificazione consistente con l'interpretazione delle asserzioni).

Regola per la composizione sequenziale di comandi – la terna:

$\{ \quad Q \quad \}$
$p_1;$
p_2
$\{ \quad R \quad \}$

è valida se, introducendo un'opportuna asserzione S, è possibile dimostrare la validità di entrambe le terne:

$\{ \quad Q \quad \}$ $\qquad\qquad$ $\{ \quad S \quad \}$
p_1 $\qquad\qquad$ p_2
$\{ \quad S \quad \}$ $\qquad\qquad$ $\{ \quad R \quad \}$

(per esempio, se p_2 è l'assegnazione $x := e$ converrà scegliere $S \equiv R^e_x$).

Regola per il comando di scelta – la terna:

$\{ \quad Q \quad \}$
if b then p_1
else p_2
endif
$\{ \quad R \quad \}$

è valida se è possibile dimostrare la validità di entrambe le terne:

$\{ \; Q \wedge b \; \}$ $\qquad\qquad$ $\{ \; Q \wedge \neg b \; \}$
p_1 $\qquad\qquad$ p_2
$\{ \quad R \quad \}$ $\qquad\qquad$ $\{ \quad R \quad \}$

Regola per il comando iterativo – Introdotta un'opportuna asserzione Inv, la terna:

$\{ \quad Inv \quad \}$
while b do
p
endwhile
$\{ \; Inv \wedge \neg b \; \}$

(Inv è l'invariante del comando iterativo) è valida se è possibile dimostrare la validità della seguente terna:

$\{ \; Inv \wedge b \; \}$
p
$\{ \quad Inv \quad \}$

Regola di rafforzamento della precondizione – la terna:

$$\{ \quad Q \quad \}$$
$$p$$
$$\{ \quad R \quad \}$$

è valida se, introducendo un'opportuna asserzione Q', è possibile dimostrare la verità della seguente implicazione e la validità della seguente terna:

$$Q \Rightarrow Q' \qquad \{ \quad Q' \quad \}\ p\ \{ \quad R \quad \}$$

(Q è un rafforzamento di Q').

Regola di indebolimento della postcondizione – la terna:

$$\{ \quad Q \quad \}$$
$$p$$
$$\{ \quad R \quad \}$$

è valida se, introducendo un'opportuna asserzione R', è possibile dimostrare la verità della seguente implicazione e la validità della seguente terna:

$$R' \Rightarrow R \qquad \{ \quad Q \quad \}\ p\ \{ \quad R' \quad \}$$

(R è un indebolimento di R'). In questo caso e nel precedente, l'implicazione viene dimostrata nell'interpretazione $\mathtt{I}$ per una generica memoria M.

Regole delle dimostrazioni di correttezza: correttezza totale

Le regole che si applicano per dimostrare la correttezza totale di un programma rispetto alle relative specifiche sono sostanzialmente analoghe al caso della correttezza parziale, ad eccezione del caso del comando iterativo. Contestualmente, si deve infatti dimostrare anche la terminazione.

Regola per la correttezza totale del comando iterativo –
– Introdotta un'opportuna asserzione Inv e un'opportuna espressione term che nell'interpretazione $\mathtt{I}$ abbia valori interi, la terna:

$$\{ \quad Inv \quad \}$$
$$\text{while } b \text{ do}$$
$$p$$
$$\text{endwhile}$$
$$\{ \ Inv \wedge \neg b \ \}_{tot}$$

(Inv è l'invariante del comando iterativo, term l'espressione che definisce la funzione di terminazione) è valida se è possibile dimostrare la verità della seguente implicazione e la validità della seguente terna:

$$\text{Inv} \Rightarrow (\text{term} \geq 0) \qquad \{ \ \text{Inv} \wedge b \ \wedge \ (\text{term} = \tau) \ \} \ p \ \{ \ \text{Inv} \ \wedge \ (\text{term} < \tau) \ \}_{tot}$$

Si presti attenzione al fatto che il simbolo τ non deve comparire nel programma (intuitivamente, è un simbolo nuovo).

Esempio 1

```
{  x ≥ 0  }
    i := 0;  y := 0;  z := 1;
    while  i < x  do
        i := i + 1;  y := y + z;  z := z + 2
    endwhile
{  y = x^2  }
```

Invariante del comando iterativo: $\text{Inv} \equiv (i \leq x) \wedge (y = i^2) \wedge (z = 2i + 1)$

Funzione di terminazione: $\text{term} \equiv x - i$

La dimostrazione che il programma è *parzialmente* corretto si riconduce a tre dimostrazioni più semplici.

1) Innanzitutto si deve verificare che l'invariante vale fin dall'inizio e ciò corrisponde alla dimostrazione di correttezza così impostata:

```
{  x ≥ 0  }
    i := 0;
    y := 0;
    z := 1
{  ( i ≤ x ) ∧ ( y = i^2 ) ∧ ( z = 2i + 1 )  }
```

2) Quindi si deve verificare che l'invariante si conserva, ovvero (la precondizione, oltre all'invariante, assume che la condizione del while sia verificata):

```
{  ( i ≤ x ) ∧ ( y = i^2 ) ∧ ( z = 2i + 1 )  ∧  ( i < x )  }
    i := i + 1;
    y := y + z;
    z := z + 2
{  ( i ≤ x ) ∧ ( y = i^2 ) ∧ ( z = 2i + 1 )  }
```

3) Infine si deve verificare che al termine dell'iterazione, cioè quando vale l'invariante ma non è più verificata la condizione del while, si può concludere che vale l'asserzione finale:

$$(i \le x) \wedge (y = i^2) \wedge (z = 2i + 1) \ \wedge \ \neg (i < x) \quad \Rightarrow \quad (y = x^2)$$

Se i comandi sono semplicemente composizioni sequenziali di assegnazioni, allora è facile calcolare a ritroso la precondizione più debole (sulla base della postcondizione) e quindi verificare che le assunzioni sono sufficienti a implicare la precondizione calcolata. Nel caso (1) il calcolo avviene nel seguente modo, dal basso verso l'alto, sostituendo (nelle asserzioni) di volta in volta il simbolo della variabile che compare al membro sinistro dell'assegnazione con l'espressione al membro destro ed eventualmente semplificando l'espressione ottenuta.

$\{ \ (0 \le x) \wedge (0 = 0) \wedge (1 = 0 + 1) \ \}$

i := 0; $\uparrow$

$\{ \ (i \le x) \wedge (0 = i^2) \wedge (1 = 2i + 1) \ \}$

y := 0; $\uparrow$

$\{ \ (i \le x) \wedge (y = i^2) \wedge (1 = 2i + 1) \ \}$

z := 1 $\uparrow$

$\{ \ (i \le x) \wedge (y = i^2) \wedge (z = 2i + 1) \ \}$

È immediato verificare che

$$(x \ge 0) \quad \Rightarrow \quad (0 \le x) \wedge (0 = 0) \wedge (1 = 0 + 1)$$

poiché $(0 = 0)$ e $(1 = 0 + 1)$ sono identità sempre verificate.

Nel caso (2), sempre calcolando le precondizioni più deboli dal basso verso l'alto analogamente a quanto appena fatto:

$\{ \ (i+1 \le x) \wedge (y+z = (i+1)^2) \wedge (z+2 = 2(i+1) + 1) \ \}$

i := i + 1; $\uparrow$

$\{ \ (i \le x) \wedge (y+z = i^2) \wedge (z+2 = 2i + 1) \ \}$

y := y + z; $\uparrow$

$\{ \ (i \le x) \wedge (y = i^2) \wedge (z+2 = 2i + 1) \ \}$

z := z + 2 $\uparrow$

$\{ \ (i \le x) \wedge (y = i^2) \wedge (z = 2i + 1) \ \}$

L'ultima precondizione calcolata (quella più in alto) può essere semplificata come segue:

$$(i < x) \wedge (y + z = i^2 + 2i + 1) \wedge (z = 2i + 1)$$

Ora $i < x$ è implicato dalla condizione del while (guardia), $z = 2i+1$ è contenuta nell'invariante, come del resto $y = i^2$, e quindi $y + z = i^2 + (2i + 1)$ è una conseguenza immediata dell'invariante. Se ne deduce che

$$(i \le x) \wedge (y = i^2) \wedge (z = 2i + 1) \ \wedge \ (i < x) \ \Rightarrow \ (i < x) \wedge (y + z = i^2 + 2i + 1) \wedge (z = 2i + 1)$$

che è quanto si voleva dimostrare.

L'implicazione (3) è una diretta conseguenza del fatto che $i \le x$ per l'invariante e inoltre $i \ge x$ perché non vale più la guardia, cioè $i = x$ e quindi $y = x^2$, sempre per l'invariante.

La dimostrazione che *inoltre* il programma termina comporta due ulteriori verifiche.

4) La funzione di terminazione ha valori interi (ovvio nell'intesa interpretazione delle variabili del programma) ed è limitata inferiormente da zero se vale l'invariante:

$$(i \le x) \wedge (y = i^2) \wedge (z = 2i + 1) \quad \Rightarrow \quad (x - i \ge 0)$$

5) La funzione di terminazione decresce strettamente in conseguenza dell'esecuzione di un passo iterativo:

$\{ \; (i \le x) \wedge (y = i^2) \wedge (z = 2i + 1) \; \wedge \; (i < x) \; \wedge \; (x - i = \tau) \; \}$

```
    i := i + 1;
    y := y + z;
    z := z + 2
```

$\{ \; x - i < \tau \; \}$

Per quanto riguarda il punto (4), basta osservare che $i \le x$ in base all'invariante. Il calcolo della precondizione più debole relativa al passo iterativo e all'asserzione $x - i < \tau$ è immediato:

$\{ \; x - (i + 1) < \tau \; \}$

```
    i := i + 1;
```

↑

$\{ \; x - i < \tau \; \}$

```
    y := y + z;
```

↑

$\{ \; x - i < \tau \; \}$

```
    z := z + 2
```

↑

$\{ \; x - i < \tau \; \}$

Ora è ovvio che

$$(x - i = \tau) \; \Rightarrow \; (x - i - 1 < \tau)$$

che completa la dimostrazione.

Esempio 2

$\{ \; (x = b > 0) \wedge (y = e \ge 0) \; \}$

```
    z := 1;
    while  y > 0  do
        if  odd( y )  then
            z := z * x;  y := y – 1
        else
            x := x * x;  y := y div 2
        endif
    endwhile
```

$\{ \; z = b^e \; \}$

Invariante del comando iterativo: $Inv \equiv (y \geq 0) \wedge (z \cdot x^y = b^e)$

Funzione di terminazione: $term \equiv y$

1) Invariante all'inizio del ciclo:

$\{ \; (x = b > 0) \wedge (y = e \geq 0) \; \}$

z := 1;

$\{ \; (y \geq 0) \wedge (z \cdot x^y = b^e) \; \}$

La verifica è immediata:

$\{ \; (y \geq 0) \wedge (x^y = b^e) \; \}$

z := 1 $\uparrow$

$\{ \; (y \geq 0) \wedge (z \cdot x^y = b^e) \; \}$

e inoltre:

$$(x = b) \wedge (y = e) \quad \Rightarrow \quad (x^y = b^e)$$

2) Conservazione dell'invariante (semplificando la precondizione in quanto la guardia del while rafforza una delle condizioni espresse dall'invariante):

$\{ \; (y > 0) \wedge (z \cdot x^y = b^e) \; \}$

if odd(y) then

z := z * x; y := y – 1

else

x := x * x; y := y div 2

endif

$\{ \; (y \geq 0) \wedge (z \cdot x^y = b^e) \; \}$

che si riduce a due dimostrazioni. La prima:

$\{ \; (y > 0) \wedge (z \cdot x^y = b^e) \wedge odd(y) \; \}$

z := z * x; y := y – 1

$\{ \; (y \geq 0) \wedge (z \cdot x^y = b^e) \; \}$

è presto verificata in quanto

$\{ \; (y-1 \geq 0) \wedge ((zx) \cdot x^{y-1} = b^e) \; \}$

z := z * x; $\uparrow$

$\{ \; (y-1 \geq 0) \wedge (z \cdot x^{y-1} = b^e) \; \}$

y := y – 1 $\uparrow$

$\{ \; (y \geq 0) \wedge (z \cdot x^y = b^e) \; \}$

e inoltre

$$(y > 0) \wedge (z \cdot x^y = b^e) \Rightarrow (y-1 \geq 0) \wedge (z \cdot x \cdot x^{y-1} = b^e)$$

La seconda:

$$\{ \quad (y > 0) \wedge (z \cdot x^y = b^e) \quad \wedge \quad \neg \, \mathrm{odd}(y) \quad \}$$

$$x := x * x; \quad y := y \text{ div } 2$$

$$\{ \quad (y \geq 0) \wedge (z \cdot x^y = b^e) \quad \}$$

è anche verificata poiché

$$\{ \quad \left(\left\lfloor \frac{y}{2} \right\rfloor \geq 0 \right) \wedge (z \cdot x^{2\lfloor y/2 \rfloor} = b^e) \quad \}$$

$$x := x * x; \qquad \uparrow$$

$$\{ \quad \left(\left\lfloor \frac{y}{2} \right\rfloor \geq 0 \right) \wedge (z \cdot x^{\lfloor y/2 \rfloor} = b^e) \quad \}$$

$$y := y \text{ div } 2 \qquad \uparrow$$

$$\{ \quad (y \geq 0) \wedge (z \cdot x^y = b^e) \quad \}$$

e una volta che si sia osservato che

$$\neg \, \mathrm{odd}(y) \Rightarrow \left\lfloor \frac{y}{2} \right\rfloor = \frac{y}{2}$$

segue

$$(y > 0) \wedge (z \cdot x^y = b^e) \wedge \neg \mathrm{odd}(y) \Rightarrow \left(\frac{y}{2} \geq 0 \right) \wedge (z \cdot x^{2(y/2)} = b^e)$$

3) Asserzione finale:

$$(y \geq 0) \wedge (z \cdot x^y = b^e) \quad \wedge \quad (y \leq 0) \quad \Rightarrow \quad (z = b^e)$$

Immediato perché dalla premessa $y = 0$ e quindi $x^y = 1$.

4) La funzione di terminazione ha valori naturali in quanto la proprietà $y \geq 0$ è compresa nell'invariante.

5) La funzione di terminazione decresce strettamente in conseguenza dell'esecuzione di un passo iterativo:

$$\{ \quad (y > 0) \wedge (z \cdot x^y = b^e) \quad \wedge \quad (y = \tau) \quad \}$$

```
if odd( y ) then
    z := z * x;   y := y – 1
else
    x := x * x;   y := y div 2
endif
```

$$\{ \quad y < \tau \quad \}$$

dove occorre ancora distinguere i due rami del comando di scelta. Il primo:

$\{\ y - 1 < \tau\ \}$
z := z * x; ↑
$\{\ y - 1 < \tau\ \}$
y := y – 1 ↑
$\{\ y < \tau\ \}$

Il secondo:

$\{\ \lfloor \frac{y}{2} \rfloor < \tau\ \}$
x := x * x; ↑
$\{\ \lfloor \frac{y}{2} \rfloor < \tau\ \}$
y := y div 2 ↑
$\{\ y < \tau\ \}$

Enrambe le dimostrazioni hanno successo perché

$$(y = \tau) \Rightarrow (y - 1 < \tau)$$

e inoltre

$$(y > 0) \wedge (y = \tau) \Rightarrow (\lfloor \frac{y}{2} \rfloor < \tau)$$

Esempio 3

$\{\ n > 0\ \}$
p := 2;
while $n \geq p$ do
p := 2 * p
endwhile;
J := 2*n – p + 1
$\{\ \exists k > 0\ .\ ((2^{k-1} \leq n < 2^k) \wedge (J = 2n - 2^k + 1))\ \}$

Invariante del comando iterativo: $Inv \equiv \exists k > 0\ .\ ((p = 2^k) \wedge (n \geq 2^{k-1}))$
Funzione di terminazione: $term \equiv 2n - p$

1) Invariante all'inizio del ciclo:

$\{\ \exists k > 0\ .\ ((2 = 2^k) \wedge (n \geq 2^{k-1}))\ \}$
p := 2 ↑
$\{\ \exists k > 0\ .\ ((p = 2^k) \wedge (n \geq 2^{k-1}))\ \}$

La condizione $2 = 2^k$ impone la scelta ($\exists$) $k = 1$ e di conseguenza la precondizione calcolata risulta equivalente a $n \geq 1$, che segue a sua volta dall'asserzione iniziale.

2) Conservazione dell'invariante:

$$\{ \quad \exists\, h > 0 \;.\; ((2p = 2^h) \wedge (n \geq 2^{h-1})) \quad \}$$

$$p := 2 * p \qquad \uparrow$$

$$\{ \quad \exists\, h > 0 \;.\; ((p = 2^h) \wedge (n \geq 2^{h-1})) \quad \}$$

(essendo irrilevante il nome della variabile vincolata dal quantificatore esistenziale, si è scelto il simbolo *h* per comodità in modo da distinguerlo dal simbolo *k* che compare nell'invariante come precondizione del passo iterativo). Ora si deve dimostrare che invariante e guardia del while implicano la precondizione appena calcolata, cioè:

$$(\exists k > 0 \;.\; ((p = 2^k) \wedge (n \geq 2^{k-1}))) \wedge (n \geq p) \;\Rightarrow\; ((\exists h > 0 \;.\; ((2p = 2^h) \wedge (n \geq 2^{h-1})))$$

Come si vede, basta scegliere $h = k+1$ per avere $(p = 2^k) \Rightarrow (2p = 2^h)$ e inoltre:

$$(p = 2^k) \wedge (n \geq p) \;\Rightarrow\; (n \geq 2^k = 2^{h-1})$$

3) Asserzione finale. Questa volta occorre introdurre un'opportuna asserzione che sia valida alla fine del comando iterativo; la si può calcolare come precondizione più debole dell'ultimo comando di assegnazione

$$\{ \quad \exists\, k > 0 \;.\; ((2^{k-1} \leq n < 2^k) \wedge (2n - p + 1 = 2n - 2^k + 1)) \quad \}$$

$$J := 2*n - p + 1 \qquad \uparrow$$

$$\{ \quad \exists\, k > 0 \;.\; ((2^{k-1} \leq n < 2^k) \wedge (J = 2n - 2^k + 1)) \quad \}$$

La precondizione calcolata si può semplificare così:

$$\exists\, k > 0 \;.\; ((2^{k-1} \leq n < 2^k) \wedge (p = 2^k)$$

Come si vede facilmente, risulta quindi dimostrata l'implicazione:

$$(\exists k > 0 \;.\; ((p = 2^k) \wedge (n \geq 2^{k-1}))) \wedge (n < p) \;\Rightarrow\; (\exists k > 0 \;.\; ((2^{k-1} \leq n < 2^k) \wedge (p = 2^k)))$$

4) La funzione di terminazione ha valori naturali in quanto $2n \geq 2^k = p$ in base all'invariante.

5) La funzione di terminazione decresce strettamente ad ogni passo iterativo:

$$\{ \quad 2n - 2p < \tau \quad \}$$

$$p := 2 * p \qquad \uparrow$$

$$\{ \quad 2n - p < \tau \quad \}$$

Per verificare, infine, l'implicazione

$$(\exists k > 0 \;.\; ((p = 2^k) \wedge (n \geq 2^{k-1}))) \;\wedge\; (2n - p = \tau) \;\Rightarrow\; (2n - 2p < \tau)$$

basta osservare che $\exists k > 0 \;.\; (p = 2^k) \;\Rightarrow\; p \geq 2 \;\Rightarrow\; 2p > p$.

Esempio 4

$\{ \; x \geq 0 \; \}$

```
k := 0;  i := 0;  y := 0;  z := 0;
while  i < x  do   { Inv1 }
    z := z + i;  i := i + 1;  z := z + i;
    while  k < z  do   { Inv2 }
        y := y + k;  k := k + 1;  y := y + k
    endwhile
endwhile
```

$\{ \; y = x^4 \; \}$

Invarianti dei comandi iterativi:

$$Inv_1 \equiv (y = k^2) \wedge (z = i^2) \wedge (k = z \geq 0) \wedge (0 \leq i \leq x)$$

$$Inv_2 \equiv (y = k^2) \wedge (z = i^2) \wedge (0 \leq k \leq z) \wedge (0 \leq i \leq x)$$

Funzioni di terminazione: $term_1 \equiv x - i$; $term_2 \equiv z - k$

1) Invariante all'inizio del ciclo esterno (Inv_1):

$\{ \; (0 = 0) \wedge (0 = 0) \wedge (0 = 0) \wedge (0 \leq 0 \leq x) \; \}$
k := 0; ↑
$\{ \; (0 = k^2) \wedge (0 = 0) \wedge (k = 0) \wedge (0 \leq 0 \leq x) \; \}$
i := 0; ↑
$\{ \; (0 = k^2) \wedge (0 = i) \wedge (k = 0) \wedge (0 \leq i \leq x) \; \}$
y := 0; ↑
$\{ \; (y = k^2) \wedge (0 = i^2) \wedge (k = 0 \geq 0) \wedge (0 \leq i \leq x) \; \}$
z := 0 ↑
$\{ \; (y = k^2) \wedge (z = i^2) \wedge (k = z \geq 0) \wedge (0 \leq i \leq x) \; \}$

La precondizione calcolata vale ovviamente se $x \geq 0$.

2) Conservazione dell'invariante Inv_1 :

$\{ \; (y = k^2) \wedge (z = i^2) \wedge (k = z \geq 0) \wedge (0 \leq i \leq x) \; \wedge \; (i < x) \; \}$

```
z := z + i;  i := i + 1;  z := z + i;
while  k < z  do   { Inv2 }
    y := y + k;  k := k + 1;  y := y + k
endwhile
```

$\{ \; (y = k^2) \wedge (z = i^2) \wedge (k = z \geq 0) \wedge (0 \leq i \leq x) \; \}$

A sua volta questa verifica si riduce a tre dimostrazioni (correttezza parziale) relative all'invariante del ciclo interno, mentre le asserzioni impostate ragionando sul ciclo esterno svolgono ora il ruolo di precondizione e postcondizione. Inv_2 deve valere all'inizio:

$\{ \ (y = k^2) \wedge (z+2i+1 = i^2+2i+1) \wedge (0 \le k \le z+2i+1) \wedge (0 \le i+1 \le x) \ \}$

z := z + i; ↑

$\{ \ (y = k^2) \wedge (z+i+1 = (i+1)^2) \wedge (0 \le k \le z+i+1) \wedge (0 \le i+1 \le x) \ \}$

i := i + 1; ↑

$\{ \ (y = k^2) \wedge (z+i = i^2) \wedge (0 \le k \le z+i) \wedge (0 \le i \le x) \ \}$

z := z + i ↑

$\{ \ (y = k^2) \wedge (z = i^2) \wedge (0 \le k \le z) \wedge (0 \le i \le x) \ \}$

Verifica che la precondizione calcolata discende dalle assunzioni (invariante e guardia del ciclo esterno):

$$(y = k^2) \wedge (z = i^2) \ \Rightarrow \ (y = k^2) \wedge (z+2i+1 = i^2+2i+1)$$

$$(k = z \ge 0) \ \Rightarrow \ (0 \le k \le z+2i+1)$$

$$(0 \le i < x) \ \Rightarrow \ (0 \le i+1 \le x)$$

Inv_2 deve conservarsi:

$\{ \ (y+2k+1 = k^2+2k+1) \wedge (z = i^2) \wedge (0 \le k+1 \le z) \wedge (0 \le i \le x) \ \}$

y := y + k; ↑

$\{ \ (y+k+1 = (k+1)^2) \wedge (z = i^2) \wedge (0 \le k+1 \le z) \wedge (0 \le i \le x) \ \}$

k := k + 1; ↑

$\{ \ (y+k = k^2) \wedge (z = i^2) \wedge (0 \le k \le z) \wedge (0 \le i \le x) \ \}$

y := y + k ↑

$\{ \ (y = k^2) \wedge (z = i^2) \wedge (0 \le k \le z) \wedge (0 \le i \le x) \ \}$

Verifica che la precondizione calcolata discende da invariante e guardia del ciclo interno:

$$(z = i^2) \wedge (0 \le i \le x) \ \Rightarrow \ (z = i^2) \wedge (0 \le i \le x)$$

$$(y = k^2) \ \Rightarrow \ (y+2k+1 = k^2+2k+1)$$

$$(0 \le k < z) \ \Rightarrow \ (0 \le k+1 \le z)$$

Al termine del ciclo interno deve essere ripristinato l'invariante del ciclo esterno (verifica immediata):

$$(y = k^2) \wedge (z = i^2) \wedge (0 \le k \le z) \wedge (0 \le i \le x) \ \wedge \ (k \ge z)$$
$$\Rightarrow \ (y = k^2) \wedge (z = i^2) \wedge (0 \le k = z) \wedge (0 \le i \le x)$$

3) Asserzione finale:

$$(y = k^2) \wedge (z = i^2) \wedge (k = z \ge 0) \wedge (0 \le i \le x) \ \wedge \ (i \ge x)$$
$$\Rightarrow \ (y = k^2 = z^2 = i^4) \wedge (i = x) \ \Rightarrow \ (y \ = \ x^4)$$

4–5) Quanto alla terminazione . . .

Le funzioni di terminazione hanno valori naturali in quanto $i \le x$ e $k \le z$ in base ai rispettivi invarianti.

Il comando iterativo interno termina (corrispondente funzione decrescente in senso stretto ad ogni passo iterativo):

$\{ \; z - k - 1 < \tau \; \}$
y := y + k; ↑
$\{ \; z - k - 1 < \tau \; \}$
k := k + 1; ↑
$\{ \; z - k < \tau \; \}$
y := y + k ↑
$\{ \; z - k < \tau \; \}$

e inoltre:

$$(z - k = \tau) \Rightarrow (z - k - 1 < \tau)$$

Il comando iterativo esterno termina:

$\{ \; (y = k^2) \wedge (z = i^2) \wedge (k = z \ge 0) \wedge (0 \le i \le x) \wedge (i < x) \wedge (x - i = \tau) \; \}$

z := z + i; i := i + 1; z := z + i;
while k < z do $\{ \; Inv_2 \; \}$
y := y + k; k := k + 1; y := y + k
endwhile

$\{ \; x - i < \tau \; \}$

in quanto la condizione $x - i < \tau$ è un invariante del comando iterativo interno (verifica banale) e inoltre:

$\{ \; x - i - 1 < \tau \; \}$
z := z + i; ↑
$\{ \; x - i - 1 < \tau \; \}$
i := i + 1; ↑
$\{ \; x - i < \tau \; \}$
z := z + i ↑
$\{ \; x - i < \tau \; \}$

dove si vede immediatamente che:

$$(x - i = \tau) \Rightarrow (x - i - 1 < \tau)$$