Esempio 3.

Terminazione di un programma ricorsivo utilizzando la semantica operazionale.

Come con i programmi flowchart, dimostriamo prima un lemma che esprime delle proprietà delle sequenze di computazione.

Sia $t_1, \ldots, t_n$, con $n \geq 2$ una sequenza di computazione finita per un input $m \in Nat$. Se il termine t_n non è una costante (o equivalentemente la sequenza di computazione non è una computazione), allora

1. il termine t_n contiene esattamente una variabile funzionale (vale a dire F_2)
2. per il sottotermine $F_2(c_1, c_2, c_3, c_4)$ del termine t_n,
   - c_1, c_2, c_3 e c_4 sono costanti
   - $\boldsymbol{\mathcal{I}}_0(c_2) > 0$
   - $\boldsymbol{\mathcal{I}}_0(c_3) + 3 > n$
   - $\boldsymbol{\mathcal{I}}_0(c_4) = m$

La dimostrazione viene fatta come sempre per induzione sulla lunghezza della sequenza di computazione

- CASO BASE: Poiché $t_1 = F_1(m)$ e $t_2 = F_2(0, 1, 0, m)$, il lemma vale per $n = 2$.
- PASSO INDUTTIVO: Supponiamo che $n > 2$ e che il lemma valga per $n - 1$. Quindi, il termine t_{n-1} contiene esattamente una variabile per funzione, vale a dire F_2. Poiché il termine t_n è assunto non essere costante, esso deve contenere almeno una variabile funzionale.

  Questa variabile funzionale deve essere F_2 per la forma delle equazioni ricorsive.

  Questo dimostra la prima parte del lemma.

  Sia $F_2(c_1, c_2, c_3, c_4)$ il sottotermine di t_{n-1} fornito dall'ipotesi induttiva.

Applicando la sostituzione si ha $F_2(c_1 + 1, c_2 + 2, c_2 + c_3, c_4)$.

La semplificazione porta a $F_2(c'_1, c'_2, c'_3, c_4)$
dove c'_1, c'_2, c'_3 sono le costanti interpretate

$$\begin{aligned}
\boldsymbol{\mathcal{I}}_0(c'_1) &= \boldsymbol{\mathcal{I}}_0(c_1) + 1 \\
\boldsymbol{\mathcal{I}}_0(c'_2) &= \boldsymbol{\mathcal{I}}_0(c_2) + 2 \\
\boldsymbol{\mathcal{I}}_0(c'_3) &= \boldsymbol{\mathcal{I}}_0(c_2) + \boldsymbol{\mathcal{I}}_0(c_3)
\end{aligned}$$

Da ciò il lemma risulta completamente dimostrato infatti
$\boldsymbol{\mathcal{I}}_0(c'_2) > 0$, e $\boldsymbol{\mathcal{I}}_0(c'_3) + 3 > n$ poiché $\boldsymbol{\mathcal{I}}_0(c_2) > 0$ e
$\boldsymbol{\mathcal{I}}_0(c_3) + 3 > n - 1$

Si supponga per contraddizione che il programma ricorsivo non termina per un input $m \in Nat$. Allora esistono delle sequenze di computazione per m arbitrariamente lunghe.
Sia $t_1, \ldots, t_n$ una di queste sequenze di computazione, con $n > m + 3$. Sia $F_2(c_1, c_2, c_3, c_4)$ il sottotermine occorrente in t_n dato dal lemma dimostrato in precedenza.
Per quel lemma: $\boldsymbol{\mathcal{I}}_0(c_2) + \boldsymbol{\mathcal{I}}_0(c_3) + 3 > n$ e $\boldsymbol{\mathcal{I}}_0(c_4) = m$. Quindi il fatto che $\boldsymbol{\mathcal{I}}_0(c_2) + \boldsymbol{\mathcal{I}}_0(c_3) > \boldsymbol{\mathcal{I}}_0(c_4)$ e la sostituzione applicata a t_n, restituisce un termine senza variabili per funzione.

Come conseguenza esiste una computazione finita per m (fatta da $n + 1$ termini). Ciò contraddice l'assunzione che il programma ricorsivo non termina per m.

Correttezza totale di un programma ricorsivo usando la semantica denotazionale

Strategia:

1. Trovare le funzioni $\Phi(S)^i(\bot)$
2. Determinare $\sqcup\{\Phi(S)^i(\bot)|i \in Nat\}$
3. Derivare la proprietà desiderata per il programma

Esempio 4.

Sia S il programma ricorsivo
$F(x,y) <= \texttt{if } x = y \texttt{ then } 1 \texttt{ else } (y+1) * F(x,y+1) \texttt{ fi}$

Dimostriamo che per tutti gli $m \in Nat$, $\boldsymbol{\mathcal{M}}(S)(m,0)$ è definita e che $\boldsymbol{\mathcal{M}}(S)(m,0) = m!$

PASSO 1: dimostrare che per tutti gli $i \in Nat$ $\Phi(S)^i(\perp) = f_i$
dove $f_i : Nat_\omega^2 \to Nat_\omega$ viene definito da

$$f_i(m,n) = \begin{cases} m!/n! & \text{se } m,n \neq \omega, n \leq m, \text{ e } m < n+i \\ \\ \omega & \text{altrimenti} \end{cases}$$

La dimostrazione viene fatta per induzione su i.

- CASO BASE Per tutti gli $m,n \in Nat_\omega$, $\Phi(S)^0(\perp)(m,n) = \omega$.

Ma anche $f_0(m,n) = \omega$.

- PASSO INDUTTIVO: per tutti gli $m, n \in Nat_\omega$

$$\Phi(S)^{i+1}(\bot)(m,n) = \Phi(S)(f_i)(m,n) =$$

$$= \texttt{if } m = n \texttt{ then } 1 \texttt{ else } (n+1) * f_i(m, n+1) \texttt{ fi}$$

per la definizione di $\Phi(S)$. La dimostrazione procede per casi.

- Se $m = \omega$ oppure $n = \omega$, allora
  $\Phi(S)^{i+1}(\bot)(m,n) = \omega = f_{i+1}(m,n)$;
- Se $m, n \neq \omega$ e $n > m$, allora

$$\begin{aligned} \Phi(S)^{i+1}(\bot)(m,n) &= (n+1) * f_i(m, n+1) \\ &= (n+1) * \omega \\ &= \omega = f_{i+1}(m,n) \end{aligned}$$

– Se $m, n \neq \omega$ e $m \geq n + i + 1$, allora

$$\begin{aligned}\Phi(S)^{i+1}(\bot)(m,n) &= (n+1) * f_i(m, n+1)\\ &= (n+1) * \omega\\ &= \omega = f_{i+1}(m,n)\end{aligned}$$

– Se $m, n \neq \omega$ e $m = n$, allora

$$\begin{aligned}\Phi(S)^{i+1}(\bot)(m,n) &= 1\\ &= f_{i+1}(m,n)\end{aligned}$$

– Se $m, n \neq \omega$ e $n < m$, e $m < n + i + 1$, allora

$$\begin{aligned}\Phi(S)^{i+1}(\bot)(m,n) &= (n+1) * f_i(m, n+1)\\ &= (n+1) * (m!/(n+1)!)\\ &= m!/n!\\ &= f_{i+1}(m,n)\end{aligned}$$

PASSO 2: Mostrare che $\sqcup\{\Phi(S)^i(\bot)|i \in Nat\} = f$
dove $f : Nat_\omega^2 \to Nat_\omega$ è definita da

$$f(m,n) = \begin{cases} m!/n! & \text{se } m,n \neq \omega, n \leq m \\ \\ \omega & \text{altrimenti} \end{cases}$$

Questo viene dimostrato facendo vedere che
$f_i \sqsubseteq f$ per ogni $i \in Nat$
$f \sqsubseteq g$ per ogni maggiorante g di $\{f_i|i \in Nat\}$

Per tutti gli $i \in Nat$ ed $m,n \in Nat_\omega$, $f_i(m,n) \sqsubseteq f(m,n)$ perché tutte le volte che $f_i(m,n) \neq \omega$, $f_i(m,n) = m!/n! = f(m,n)$.

Sia g un maggiorante di $\{f_i|i \in Nat\}$ tale che $f \sqsubseteq g$ non vale.
Ma allora ci devono essere $m,n \in Nat_\omega$ tali che $f(m,n) \neq \omega$ e

$f(m,n) \neq g(m,n)$. Ma allora $m, n \neq \omega$ e $n \leq m$. Si consideri ora la funzione f_i con $i = m - n + 1$. Per definizione di f_i, $f_i(m,n) = m!/n! \neq \omega$.
Poiché f e g sono entrambi maggioranti di $\{f_i | i \in Nat\}$, $f(m,n) = m!/n! = g(m,n)$ che contraddice l'ipotesi.

Per definizione di significato di un programma ricorsivo si ha per tutti gli $m, n \in Nat$

$$\boldsymbol{\mathcal{M}}(S)(m,n) = \begin{cases} f(m,n) & \text{se } f(m,n) \neq \omega \\ \\ \text{indefinita} & \text{altrimenti} \end{cases}$$

In particolare, per tutti gli $m \in Nat$

$$\boldsymbol{\mathcal{M}}(S)(m,0) = f(m,0) = m!$$

Metodo delle asserzioni induttive

Introduciamo un metodo per la dimostrazione della correttezza parziale dei programmi flowchart.

Sia B una base, $\mathcal{I}$ una interpretazione, $p, q \in WFF_B$ ed $S \in \mathrm{L}_1^B$.
Il metodo delle asserzioni induttive è un modo per dimostrare la parziale correttezza del programma S rispetto alle formule p e q

IDEA:
Ragionare sui cammini di un programma piuttosto che sul programma in se

Cammino in un programma flowchart

Siano l, l' due etichette che occorrono in un programma flowchart S. Un *cammino* da l ad l' (in S) è una sequenza $(l_0, l_1, \ldots, l_k)$, $k \geq 0$ di etichette tali che

(1) $l_0 = l$;

(2) per $i = 0, 1, \ldots, k-1$ l'etichetta l_{i+1} ha un'occorrenza applicata nel comando in cui l'etichetta l_i ha un'occorrenza definente;

(3) $l_k = l'$

La lunghezza del cammino $l_0, l_1, \ldots, l_k$ è k.

Cammino in un programma flowchart (ctnd.1)

Un cammino da un'etichetta a se stessa di lunghezza $k \geq 1$ viene detto *loop*

Un cammino da $begin$ a end in un programma flowchart S viene detto un cammino *attraverso* S

Cammino in un programma flowchart (ctnd.2)

Il significato di un cammino α nel programma flowchart S e nell'interpretazione $\mathcal{I}$ è la funzione $\mathcal{M}_{\mathcal{I}}(\alpha) : \Sigma \rightsquigarrow \Sigma$ definita

$$\mathcal{M}_{\mathcal{I}}(\alpha)(\sigma) = \begin{cases} \sigma' & \text{se ci sono } (k+1) \text{ stati } \sigma_0, \sigma_1, \ldots, \sigma_k \\ & \text{tali che } \sigma = \sigma_0,\ \sigma' = \sigma_k \text{ e} \\ & \text{per } i = 0, \ldots, k-1,\ (l_i, \sigma_i) \Rightarrow^S (l_{i+1}, \sigma_{i+1}) \\ \text{indefinita} & \text{altrimenti} \end{cases}$$

NOTE:

- la sequenza $(l_0, \sigma_0), \ldots, (l_k, \sigma_k)$ non è necessariamente una sequenza di computazione del programma in quanto l_0 può essere differente da $begin$

- il significato di un programma è indefinito se lo stato di input σ porta ad un loop infinito. Il significato di un cammino è indefinito se il cammino è 'impossibile' per σ.

Correttezza parziale, totale e terminazione per cammini di programmi flowchart

Sia B una base per la logica predicativa, $\mathcal{I}$ una interpretazione di questa base e Σ il corrispondente insieme degli stati.
Sia α un cammino di un programma flowchart $S \in \mathrm{L}_1^B$, e sia $\mathcal{M}_{\mathcal{I}}(\alpha)$ il significato di α.
Siano inoltre p e q due formule di WFF_B chiamate rispettivamente *precondizione* e *postcondizione*. Il cammino viene detto:

- parzialmente corretto rispetto a p e q nell'interpretazione $\mathcal{I}$, se per tutti gli stati $\sigma \in \Sigma$:

  if $\mathcal{I}(p)(\sigma) = \mathbf{true}$ and $\mathcal{M}_{\mathcal{I}}(\alpha)(\sigma)$ is defined
  then $\mathcal{I}(q)(\mathcal{M}_{\mathcal{I}}(\alpha)(\sigma)) = \mathbf{true}$

- terminare rispetto alla formula p nell'interpretazione $\mathcal{I}$, se per

tutti gli stati $\sigma \in \Sigma$:

if $\boldsymbol{\mathcal{I}}(p)(\sigma) = \mathbf{true}$
then $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma)$ is defined

- essere totalmente corretto rispetto alle formule p e q nell'interpretazione $\boldsymbol{\mathcal{I}}$, se per tutti gli stati $\sigma \in \Sigma$:

  if $\boldsymbol{\mathcal{I}}(p)(\sigma) = \mathbf{true}$
  then $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma)$ is defined and $\boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma)) = \mathbf{true}$

Weakest preconditions and strongest postconditions

Siano dati B, $\boldsymbol{\mathcal{I}}$, Σ, α, S e q.

Le *precondizioni liberali più deboli* del cammino α del programma flowchart S e della formula q sono tutte le formule $p \in WFF_B$ tali che $\boldsymbol{\mathcal{I}}(p)(\sigma) = \mathbf{true}$ se e solo se $\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\sigma)$ è indefinito oppure $\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\sigma)$ è definito e $\boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\sigma)) = \mathbf{true}$

Le *precondizioni più deboli* del cammino α del programma flowchart S e della formula q sono tutte le formule $p \in WFF_B$ tali che $\boldsymbol{\mathcal{I}}(p)(\sigma) = \mathbf{true}$ se e solo se $\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\sigma)$ è definito e $\boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\sigma)) = \mathbf{true}$

Le *postcondizioni più forti* della formula p e del cammino α del programma flowchart S sono tutte le formule $q \in WFF_B$ tali che $\boldsymbol{\mathcal{I}}(q)(\sigma) = \mathbf{true}$ se e solo se esiste uno stato $\sigma' \in \Sigma$ per cui $\boldsymbol{\mathcal{I}}(p)(\sigma') = \mathbf{true}$ e $\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\sigma') = \sigma$

Considerazioni

E' importante notare la differenza fra le nozioni di terminazione di un programma e di un cammino

E' ovvio che per tutti gli stati $\sigma, \sigma' \in \Sigma$,

$\boldsymbol{\mathcal{M}_{\mathcal{I}}}(S)(\sigma) = \sigma'$ se e solo se esiste un cammino α attraverso S tale che $\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\sigma) = \sigma'$

Parziale correttezza di programmi e cammini

LEMMA

Sia B una base per la logica predicativa, $\boldsymbol{\mathcal{I}}$ un'interpretazione di questa base e Σ il corrispondente insieme ddegli stati. Un programma flowchart $S \in \mathrm{L}_1^B$ è parzialmente corretto rispetto alle formule p e q in $\boldsymbol{\mathcal{I}}$ se e solo se ogni cammino attraverso S è parzialmente corretto rispetto a p e a q in $\boldsymbol{\mathcal{I}}$.

Dim:

$\Rightarrow$ Sia S un programma flowchart parzialmente corretto rispetto a p e a q in $\boldsymbol{\mathcal{I}}$, cioé tale che per ogni $\sigma \in \Sigma$

$$(\boldsymbol{\mathcal{I}}(p)(\sigma) \wedge \boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)\downarrow) \supset \boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)).$$

Supponiamo per contraddizione che esista un cammino α in S che non è parzialmente corretto rispetto a $\boldsymbol{\mathcal{I}}$, cioè tale che esiste un

$\bar{\sigma} \in \Sigma$ per cui $\boldsymbol{\mathcal{I}}(p)(\sigma) \wedge \boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\bar{\sigma}) \downarrow = \mathbf{true}$ ma $\boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\bar{\sigma})) = \mathbf{false}$. Poiché $\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\bar{\sigma}) \downarrow$, deve essere $\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\bar{\sigma}) = \sigma'$ per un qualche $\sigma' \in \Sigma$ e, di conseguenza, $\boldsymbol{\mathcal{M}_{\mathcal{I}}}(S)(\bar{\sigma}) = \sigma'$.Ma allora S non è parzialmente corretto rispetto a p e q. Questo porta a una contraddizione.

$\Leftarrow$ Per ipotesi, per ogni α che attraversa S e per ogni $\sigma \in \Sigma$

$$(\boldsymbol{\mathcal{I}}(p)(\sigma) \wedge \boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\sigma) \downarrow) \supset \boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\sigma)).$$

Supponiamo per contraddizione che S non sia parzialmente corretto. Allora esiste un $\bar{\sigma} \in \Sigma$ per cui $\boldsymbol{\mathcal{I}}(p)(\bar{\sigma}) \wedge \boldsymbol{\mathcal{M}_{\mathcal{I}}}(S)(\bar{\sigma}) \downarrow = \mathbf{true}$ e $\boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}_{\mathcal{I}}}(S)(\bar{\sigma})) = \mathbf{false}$. Poiché $\boldsymbol{\mathcal{M}_{\mathcal{I}}}(S)(\bar{\sigma}) \downarrow$, deve essere $\boldsymbol{\mathcal{M}_{\mathcal{I}}}(S)(\bar{\sigma}) = \sigma'$ per un qualche $\sigma' \in \Sigma$ e, di conseguenza, esiste un cammino α' attraverso S tale che $\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha')(\bar{\sigma}) = \sigma'$. Abbiamo trovato un cammino che non e' parzialmente corretto, contraddicendo l'ipotesi.

Considerazioni

- E' possibile sostituire lo studio di un programma flowchart con quello dei cammini che lo attraversano (possibilmente infiniti)
- I cammini sono più semplici dei programmi perché i loop vengono dispiegati e quindi resi espliciti
- Se $\mathcal{M}_{\mathcal{I}}(\alpha)(\sigma)\downarrow$, ogni nodo del cammino viene visitato esattamente una volta
- I cammini (possibilmente un numero infinito) che attraversano il programma, possono essere considerati come costruiti a partire da un numero finito di 'cammini elementari'
- Il ragionamento su infiniti cammini può essere ricondotto al ragionamento su un numero finito di cammini elementari.

Correttezza parziale di un cammino

Sia α un cammino arbitrario. Facciamo vedere come dimostrare la correttezza parziale di α rispetto alle formule p e q.

Definiamo

- Una formula $wlp(\alpha, q)$ (weakest liberal precondition 'p' of α and q)
- Una formula $vc(p, \alpha, q)$ detta la 'condizione di verifica'

Correttezza parziale di un cammino - definizione di wlp

Sia B una base per la logica predicativa, $p, q \in WFF_B$ due formule, $S \in \mathtt{L}_1^B$ un programma flowchart ed $\alpha = (l_0, \ldots, l_k)$ un cammino in S.

La formula $wlp(\alpha, q)$ viene definita per induzione sulla lunghezza k del cammino:

- CASO BASE. Per $k = 0$ la formula $wlp(\alpha, q)$ è definita q

- PASSO INDUTTIVO. Per $k > 0$ sia β il cammino $(l_1, \ldots, l_k)$ ed r la formula $wlp(\beta, q)$ (che è già stata definita). In base alla forma del comando all'etichetta l_0, vengono distinti due casi.
  - Se il comando è un assegnamento parallelo della forma

    $$l_0 \ : \ (x_1, \ldots, x_n) := (t_1, \ldots, t_n) \ ; \ \texttt{goto}\ l_1$$

    allora la formula $wlp(\alpha, q)$ è definita $r^{t_1,\ldots,t_n}_{x_1,\ldots,x_n}$.
  - Se il comando è un salto condizionale della forma

    $$l_0 \ : \ \texttt{if}\ e\ \texttt{then goto}\ l\ \texttt{else goto}\ l'\ \texttt{fi}\,,$$

    allora la formula $wlp(\alpha, q)$ è definita

    $e \supset r$ se $l = l_1$

    $(\neg e) \supset r$ se $l' = l_1$

Correttezza parziale di un cammino - definizione di vc

Sia B una base per la logica predicativa, $p, q \in WFF_B$ due formule, $S \in \mathtt{L}_1^B$ un programma flowchart ed $\alpha = (l_0, \ldots, l_k)$ un cammino in S.

La *condizione di verifica* per il cammino α e le formule p e q viene denotata con $vc(p, \alpha, q)$ e viene definita

$$p \supset wlp(\alpha, q)$$

NOTA: Questa definizione, come quella di wlp, è puramente sintattica. Infatti non dipende dall'interpretazione della base B.

Esempio - trovare wlp e vc

Consideriamo il programma flowchart per il calcolo della radice intera di un intero positivo e calcoliamo $wlp(\alpha_1, q_{test})$ e $vc(p_{begin}, \alpha_1, q_{test})$ dove

- $p_{begin} = (x = a)$
- $\alpha_1 = (begin, test)$
- $q_{test} = (x = a \wedge y_1^2 \leq x \wedge y_3 = (y_1 + 1)^2 \wedge y_2 = 2 * y_1 + 1)$

Esempio - trovare wlp e vc (ctnd.1)

$wlp(\alpha_1, q_{test})$ è

$$\begin{aligned} wlp((begin, test), q_{test}) &= (wlp((test), q_{test}))^{0,1,1}_{y_1,y_2,y_3} \\ &= (q_{test})^{0,1,1}_{y_1,y_2,y_3} \\ &= (x = a \wedge 0 \leq x \wedge 1 = (0+1)^2 \wedge 1 = 2 * 0 + 1) \end{aligned}$$

mentre

$$vc(p_{begin}, \alpha_1, q_{test}) = (x = a \supset (x = a \wedge 0 \leq x \wedge 1 = (0+1)^2 \wedge 1 = 2 * 0 + 1))$$

Esempio - trovare wlp e vc (ctnd.2)

Troviamo ora $wlp(\alpha_2, q_{test})$ e $vc(q_{test}, \alpha_2, q_{test})$ dove $\alpha_2 = (test, loop, upd, test)$

$$
\begin{aligned}
wlp((test), q_{test}) &= q_{test} \\
wlp((upd, test), q_{test}) &= (q_{test})^{y_3+y_2}_{y_3} \\
wlp((loop, upd, test), q_{test}) &= ((q_{test})^{y_3+y_2}_{y_3})^{y_1+1,y_2+2}_{y_1,y_2} \\
wlp((test, loop, upd, test), q_{test}) &= (y_3 \leq x \supset ((q_{test})^{y_3+y_2}_{y_3})^{y_1+1,y_2+2}_{y_1,y_2})
\end{aligned}
$$

$$
vc(q_{test}, \alpha_2, q_{test}) = (x = a \wedge y_1^2 \leq x \wedge y_3 = (y_1+1)^2 \wedge y_2 = 2 * y_1 + 1 \wedge y_3 \leq x) \supset ((q_{test})^{y_3+y_2}_{y_3})^{y_1+1,y_2+2}_{y_1,y_2}
$$

con

$((q_{test})_{y_3}^{y_3+y_2})_{y_1,y_2}^{y_1+1,y_2+2} = (x = a \wedge (y_1+1)^2 \leq x \wedge y_3 + y_2 + 2 = (y_1+1+1)^2 \wedge y_2 + 2 = 2 * (y_1+1)+1)$

Troviamo ora $wlp(\alpha_3, q_{end})$ e $vc(q_{test}, \alpha_3, q_{end})$ dove

- $\alpha_3 = (test, end)$
- $q_{end} = (y_1 = \sqrt{a})$

$wlp(\alpha_3, q_{end}) = \neg(y_3 \leq x) \supset (y_1 = \sqrt{a})$

$vc(q_{test}, \alpha_3, q_{end}) = (x = a \wedge y_1^2 \leq x \wedge y_3 = (y_1+1)^2 \wedge y_2 = 2 * y_1 + 1 \wedge \neg(y_3 \leq x)) \supset (y_1 = \sqrt{a})$

Risultato su wlp

TEOREMA:
Sia B una base per la logica predicativa, $S \in \mathrm{L}_1^B$ un programma flowchart, α un cammino in S, e $q \in WFF_B$ una formula. Per ogni interpretazione $\boldsymbol{\mathcal{I}}$ di B $wlp(\alpha, q)$ è la precondizione liberale più debole rispetto al cammino α e a q.

Dim:
Supponiamo che $\alpha = (l_0, l_1, \ldots, l_k)$. La dimostrazione è per induzione sulla lunghezza k di α.

- CASO BASE. Per $k = 0$ $wlp(\alpha, q) = q$ e $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)$ è la funzione identica. Quindi è vero

$$\boldsymbol{\mathcal{I}}(q)(\sigma) \equiv (\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma) \uparrow \vee(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma) \downarrow \wedge \boldsymbol{\mathcal{I}}(q)(\sigma)))$$

- PASSO INDUTTIVO. Per ipotesi induttiva $wlp(\beta, q)$ è la precondizione liberale più debole rispetto al cammino β e a q. Bisogna dimostrare che per ogni stato $\sigma \in \Sigma$

$$\boldsymbol{\mathcal{I}}(wlp(\alpha, q))(\sigma) \equiv \boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma) \uparrow \vee(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma) \downarrow \wedge \boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma)))$$

Distinguiamo due casi:

– **Assegnamento parallelo.** C'è un comando

$$l_0 \ : \ (x_1, \ldots, x_n) := (t_1, \ldots, t_n) \ ; \ \texttt{goto} \ l_1$$

nel programma S. Sia $\sigma \in \Sigma$ uno stato arbitrario. Allora

$\boldsymbol{\mathcal{I}}(wlp(\alpha, q))(\sigma) \equiv \boldsymbol{\mathcal{I}}(wlp(\beta, q))^{t_1,\ldots,t_n}_{x_1,\ldots,x_n}(\sigma) =$
$\boldsymbol{\mathcal{I}}(wlp(\beta, q))(\sigma')$
dove $\sigma' = \sigma[x_1/\boldsymbol{\mathcal{I}}(t_1)\sigma] \ldots [x_n/\boldsymbol{\mathcal{I}}(t_n)\sigma]$

Per ipotesi induttiva
$\boldsymbol{\mathcal{I}}(wlp(\beta, q))(\sigma') \equiv$
$\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\beta)(\sigma') \uparrow \vee(\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\beta)(\sigma') \downarrow \wedge \boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\beta)(\sigma')))$

Poiché $(l_0, \sigma_0) \Rightarrow (l_1, \sigma)$ si ha che
$\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\sigma) \uparrow$ se e solo se $\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\beta)(\sigma') \uparrow$

Se $\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\sigma) \downarrow$ allora $\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\sigma) = \boldsymbol{\mathcal{M}_{\mathcal{I}}}(\beta)(\sigma')$ e quindi

$\boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\alpha)(\sigma))) = \boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}_{\mathcal{I}}}(\beta)(\sigma')))$

– **Salto condizionale.** C'è un comando

$$l_0 \ : \ \texttt{if}\ e\ \texttt{then goto}\ l\ \texttt{else goto}\ l'\ \texttt{fi}\,,$$

Si consideri il caso $l = l_1$. Sia $\sigma \in \Sigma$ uno stato arbitrario. Allora

$$\boldsymbol{\mathcal{I}}(wlp(\alpha, q))(\sigma) = \boldsymbol{\mathcal{I}}(e \supset wlp(\beta, q))(\sigma).$$

Se $\boldsymbol{\mathcal{I}}(e)(\sigma) = \mathbf{true}$ allora

$\boldsymbol{\mathcal{I}}(e \supset wlp(\beta, q))(\sigma) \equiv$

$\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\beta)(\sigma)\uparrow \vee(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\beta)(\sigma)\downarrow \wedge\boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\beta)(\sigma)))$

Poiché $(l_0, \sigma_0) \Rightarrow (l_1, \sigma)$

$\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma)\uparrow$ se e solo se $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\beta)(\sigma)\uparrow$

Se $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma)\downarrow$ allora $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma) = \boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\beta)(\sigma)$

Se invece $\boldsymbol{\mathcal{I}}(e)(\sigma) = \mathbf{false}$, allora
$\boldsymbol{\mathcal{I}}(e \supset wlp(\beta, q))(\sigma) = \mathbf{true}$. Poiché $(l_0, \sigma_0) \Rightarrow (l', \sigma)$,
$\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma)\uparrow$

Altri risultati

COROLLARIO:
Sia B una base per la logica predicativa, $p, q \in WFF_B$ due formule ed α un cammino in un programma flowchart in $\mathtt{L}_1^B$. Per ogni interpretazione $\boldsymbol{\mathcal{I}}$ di B, il cammino α è parzialmente corretto rispetto alle formule p e q, se e solo se la condizione di verifica $vc(p, \alpha, q)$ è valida in $\boldsymbol{\mathcal{I}}$.

Dim:
Sia α parzialmente corretto rispetto a p e a q. Allora per ogni $\sigma \in \Sigma$,

$$(\boldsymbol{\mathcal{I}}(p)(\sigma) \wedge \boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma)\downarrow) \supset \boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma))$$

Vogliamo provare che per ogni $\sigma \in \Sigma$

$$\boldsymbol{\mathcal{I}}(p)(\sigma) \supset \boldsymbol{\mathcal{I}}(wlp(\alpha, q))(\sigma)$$

- Sia $\sigma \in \Sigma$ tale che $\boldsymbol{\mathcal{I}}(p)(\sigma) \wedge \boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma) \downarrow = \mathbf{true}$.
  Dalla definizione di wlp,
  $\boldsymbol{\mathcal{I}}(wlp(\alpha, q))(\sigma) \equiv$
  $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma) \uparrow \vee(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma) \downarrow \wedge \boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma)))$
  si ha la tesi.
- Sia $\sigma \in \Sigma$ tale che $\boldsymbol{\mathcal{I}}(p)(\sigma) \wedge \boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma) \downarrow = \mathbf{false}$.
  Supponiamo che $\boldsymbol{\mathcal{I}}(p)(\sigma) = \mathbf{true}$ (il caso in cui $\boldsymbol{\mathcal{I}}(p)(\sigma) = \mathbf{false}$ è banale), ma allora $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma) \uparrow$, e quindi la tesi segue.

Sia $vc(p, \alpha, q)$ valida nell'interpretazione $\boldsymbol{\mathcal{I}}$. Allora per ogni $\sigma \in \Sigma$, $\boldsymbol{\mathcal{I}}(p)(\sigma) \supset \boldsymbol{\mathcal{I}}(wlp(\alpha, q))(\sigma)$

Vogliamo provare che per ogni $\sigma \in \Sigma$

$$(\boldsymbol{\mathcal{I}}(p)(\sigma) \wedge \boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma) \downarrow) \supset \boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma))$$

- Sia σ tale che $\boldsymbol{\mathcal{I}}(p)(\sigma) = \mathbf{true}$. Allora $\boldsymbol{\mathcal{I}}(wlp(\alpha, q))(\sigma) = \mathbf{true}$ e quindi
  - o $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma) \uparrow$, oppure
  - $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma) \downarrow$ e $\boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(\alpha)(\sigma)) = \mathbf{true}$

  in entrambi i casi la tesi è verificata. Per i σ tali che $\boldsymbol{\mathcal{I}}(p)(\sigma) = \mathbf{false}$ la tesi è verificata.

Altri risultati (ctnd.)

TEOREMA:

Sia B una base per la logica predicativa, $p, q \in WFF_B$ due formule, e $S \in \mathrm{L}_1^B$ un programma flowchart. Per ogni interpretazione $\boldsymbol{\mathcal{I}}$ di B, il programma S è parzialmente corretto rispetto alle formule p e q (nell'interpretazione $\boldsymbol{\mathcal{I}}$) se e solo se per ogni cammino α attraverso S, la condizione di verifica $vc(p, \alpha, q)$ è valida in $\boldsymbol{\mathcal{I}}$.

Dim:

S è parzialmente corretto rispetto alle formule p e q se e solo se ogni cammino attraverso S è parzialmente corretto.
Dal teorema precedente sappiamo che un cammino α è parzialmente corretto rispetto a p e q se e solo se $vc(p, \alpha, q)$ è valida (nell'interpretazione considerata). Da ciò segue la tesi.

Definizione del metodo delle asserzioni induttive

Sia B una base per la logica predicativa, $p, q \in WFF_B$ due formule, $S \in \mathtt{L}_1^B$ un programma flowchart, e $\boldsymbol{\mathcal{I}}$ una interpretazione di B. Il metodo delle asserzioni induttive, che dimostra la correttezza parziale di S rispetto alle formule p e q nell'interpretazione $\boldsymbol{\mathcal{I}}$ consiste dei seguenti tre passi:

1. Si scelga un sottoinsieme C dell'insieme di tutte le etichette che occorrono in S, tale che $begin$ e end sono contenute in C, ed ogni loop in S contiene almeno un elemento di C. Gli elementi di C vengono chiamati *cutpoints*.
2. Si associ una formula $q_l \in WFF_B$ ad ogni cutpoint $l \in C$. Precisamente: la formula p con $begin$, la formula q con end, e una formula 'appropriata' con ogni altro cutpoint. La scelta di quest'ultima formula è libera nella misura in cui permette di

eseguire il passo 3. Se un cutpoint l è contenuto in un loop, la formula corrispondente viene chiamata *invariante*.

3. Per ogni cammino $\alpha = (l_0, \ldots, l_k)$ in S, $k \geq 1$, tale che $l_0, l_k \in C$ e $l_1, \ldots, l_{k-1} \notin C$, si dimostri che la condizione di verifica $vc(q_{l_0}, \alpha, q_{l_k})$ è valida in $\boldsymbol{\mathcal{I}}$.

Esempio

Usiamo il metodo delle asserzioni induttive per dimostrare che il programma che calcola la radice quadrata intera è parzialmente corretto rispetto alle formule $x = a$ e $y_1 = \sqrt{a}$

1. C è l'insieme $\{begin, test, end\}$
2. $q_{begin} = (x = a)$, $q_{end} = (y_1 = \sqrt{a})$,
   $q_{test} = (x = a \wedge y_1^2 \leq x \wedge y_3 = (y_1 + 1)^2 \wedge y_2 = 2 * y_1 + 1)$
3. Basta esaminare i cammini $(begin, test)$, $(test, loop, upd, test)$ e $(test, end)$. Le condizioni di verifica sono banalmente valide

Correttezza del metodo delle asserzioni induttive

TEOREMA:
Se i passi 1, 2 e 3 del metodo delle asserzioni induttive possono essere eseguiti con successo, allora il programma S è parzialmente corretto rispetto alle formule p e q nell'interpretazione $\mathcal{I}$.

Dim:
Mostriamo che un qualunque cammino attraverso S consiste dei cammini esaminati nel passo 3. Quindi la correttezza parziale del cammino deriva dalla correttezza parziale di questi cammini 'elementari'.
In particolare, si consideri la sequenza di computazione

$$(l_0, \sigma_0), (l_1, \sigma_1), \ldots, (l_k, \sigma_k)$$

con $k \geq 0$ tale che $\mathcal{I}(p)(\sigma_0) = \mathbf{true}$ ed l_k è un cutpoint, allora

$\boldsymbol{\mathcal{I}}(q_{l_k})(\sigma_k) = \mathbf{true}$.
La dimostrazione di questo asserto è per induzione sul numero m di occorrenze di cutpoint nel cammino $l_0, \ldots, l_k$.

- CASO BASE. Se $m = 1$ allora $k = 0$, perché $l_0 = begin$ è un cutpoint. Quindi $\boldsymbol{\mathcal{I}}(q_{l_k})(\sigma_k) = \boldsymbol{\mathcal{I}}(p)(\sigma_0) = \mathbf{true}$.
- PASSO INDUTTIVO. Supponiamo che $m > 1$. Sia l_j l'ultimo cutpoint prima di l_k o, più precisamente, $j < k$ è tale che $l_j \in C$ e $l_{j+1}, \ldots, l_{k-1} \notin C$. Allora $\boldsymbol{\mathcal{I}}(q_{l_j})(\sigma_j) = \mathbf{true}$ per ipotesi induttiva.

  Inoltre il cammino $(l_j, \ldots, l_k)$ è fra quelli esaminati al passo 3. Questo cammino è parzialmente corretto rispetto a q_{l_j} e a q_{l_k} il che porta a $\boldsymbol{\mathcal{I}}(q_{l_k})(\sigma_k) = \mathbf{true}$. Ciò conclude la dimostrazione dell'asserto.

Adesso, per una computazione, o in altre parole, per una sequenza di computazione $(l_0, \sigma_0), \ldots, (l_k, \sigma_k)$ con $l_k = end$, ogni cammino attraverso il programma S è parzialmente corretto rispetto a p e a q. Da cui la tesi.