Semantica denotazionale dei programmi while

La semantica dei programmi while può essere definita induttivamente utilizzando la definizione induttiva della loro sintassi

Questa definizione farà uso del Teorema del minimo punto fisso

Anche in questo caso perché questo teorema sia applicato le funzioni introdotte devono essere continue e i dominii sottostanti dei cpo

Semantica denotazionale dei programmi while (ctnd.)

- V insieme numerabile di variabili individuali
- $B = (\mathcal{F}, \mathcal{P})$ una base per la logica predicativa
- Sia $\boldsymbol{\mathcal{I}} = (\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}}_0)$ una interpretazione di questa base
- Sia Σ l'insieme degli stati

Per poter trasformare l'insieme Σ in un cpo, quest'ultimo viene esteso con un elemento ω che restituisce la cpo piatta Σ_ω

Questo elemento ω sta per lo 'stato indefinito'.

ω-estensione della funzione significato

La funzione significato $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S) : \Sigma \rightsquigarrow \Sigma$ di un programma while S, viene dedotta da una funzione totale $\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(S) : \Sigma_\omega \to \Sigma_\omega$, detta ω-estensione del significato

Sia S un programma while su una data base ed $\boldsymbol{\mathcal{I}}$ un'interpretazione su questa base. L'ω-estensione della funzione significato di S in $\boldsymbol{\mathcal{I}}$ è la funzione $\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(S) : \Sigma_\omega \to \Sigma_\omega$, definita per induzione strutturale come segue

a. Statement di assegnamento:

$$\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(x := t)(\sigma) = \begin{cases} \sigma[x/\boldsymbol{\mathcal{I}}(t)(\sigma)] & \text{se } \sigma \neq \omega \\ \omega & \text{se } \sigma = \omega \end{cases}$$

b. Statement composto:

$$\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(S_1; S_2) = \boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(S_2) \circ \boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(S_1)$$

dove '$\circ$' denota la composizione

c. Statement condizionale:

$$\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(\texttt{if } e \texttt{ then } S_1 \texttt{ else } S_2 \texttt{ fi})(\sigma) =$$

$$= \begin{cases} \texttt{ifthenelse}(\boldsymbol{\mathcal{I}}(e)(\sigma), \boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(S_1)(\sigma), \boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(S_2)(\sigma)) & \text{se } \sigma \neq \omega \\ \omega & \text{se } \sigma = \omega \end{cases}$$

per tutti i $\sigma \in \Sigma_\omega$

d. Statement while:

$$\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(\texttt{while } e \texttt{ do } S_1 \texttt{ od}) = \mu\Phi$$

dove $\Phi : [\Sigma_\omega \to \Sigma_\omega] \to [\Sigma_\omega \to \Sigma_\omega]$

viene definito da

$$\Phi(f)(\sigma) = \begin{cases} \texttt{ifthenelse}(\boldsymbol{\mathcal{I}}(e)(\sigma), f(\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(S_1)(\sigma)), \sigma) & \text{se } \sigma \neq \omega \\ \omega & \text{se } \sigma = \omega \end{cases}$$

per tutte le $f \in [\Sigma_\omega \to \Sigma_\omega]$ e $\sigma \in \Sigma_\omega$

Consistenza della definizione di ω-estensione del significato

Teorema

Per ogni programma while S

1) $\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}$ è ben definita

2) $\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}$ è continua

Dim.

Per induzione strutturale su S.

- *Statement di assegnamento:* Supponiamo che S ha la forma $x := t$. 1) vale banalmente. 2) vale anche, perché $\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(x := t)$ è stretta e dunque continua
- *Statement composto:* Supponiamo che S ha la forma $S_1; S2$. 1) vale perché per hp induttiva $\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(S_1)$ e $\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(S_2)$ sono ben

definite. 2) vale perché $\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(S_1)$ e $\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(S_2)$ sono continue per hp induttiva, e la composizione preserva la continuità.

- *Statement condizionale:*

  1) viene provato come nel caso degli assegnamenti composti mentre 2) come nel caso di uno statement di assegnamento

- *Statement while:* Supponiamo che S abbia la forma `while` e `do` S_1 `od`.

  $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S_1)$ è ben definita. Per dimostrare che Φ mappa davvero l'insieme $[\Sigma_\omega \to \Sigma_\omega]$ in se stesso, e che Φ è continua, Φ può essere espressa come un λ-termine:

  $\Phi = [\lambda F.[\lambda\sigma.\texttt{if}\ \sigma = \sigma$
  $\texttt{then if}\ \boldsymbol{\mathcal{I}}(e)(\sigma)\ \texttt{then}\ F(\boldsymbol{\mathcal{M}}^{\omega}(S_1)(\sigma))\ \texttt{else}\ \sigma\ \texttt{fi}$
  $\texttt{else}\ldots\texttt{fi}]]$

Semantica denotazionale di $\mathcal{L}_2$

Sia S un programma while su una data base, $\mathcal{I}$ un'interpretazione di questa base. Il significato di S (in $\mathcal{I}$) è la funzione $\mathcal{M}_{\mathcal{I}}(S) : \Sigma \rightsquigarrow \Sigma$ definita da

$$\mathcal{M}_{\mathcal{I}}(S)(\sigma) = \begin{cases} \mathcal{M}^{\omega}_{\mathcal{I}}(S)(\sigma) & \text{se } \mathcal{M}^{\omega}_{\mathcal{I}}(S)(\sigma) \neq \omega \\ \text{indefinito} & \text{se } \mathcal{M}^{\omega}_{\mathcal{I}}(S)(\sigma) = \omega \end{cases}$$

Esempio

Consideriamo il programma while:

$$\texttt{while}\ x \neq 0\ \text{do}\ x := x - 1\ \text{od}$$

Vogliamo calcolare la ω-estensione della sua funzione significato

$\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(\texttt{while}\ x \neq 0\ \text{do}\ x := x - 1\ \text{od}) = \mu\Phi$

$\Phi : [\Sigma_\omega \to \Sigma_\omega] \to [\Sigma_\omega \to \Sigma_\omega]$

è definito

$$\Phi(f)(\sigma) = \begin{cases} \texttt{ifthenelse}(\boldsymbol{\mathcal{I}}(x \neq 0)(\sigma), f(\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(x := x - 1)(\sigma)), \sigma) & \text{se } \sigma \neq \omega \\ \omega & \text{se } \sigma = \omega \end{cases}$$

Esempio (ctnd.1)

Calcoliamo il minimo punto fisso

$\mu\Phi = \sqcup\{\Phi^i(\bot)|i \in Nat\}$

$\bot_{[\Sigma_\omega \to \Sigma_\omega]}$ è la funzione $f_\bot(\sigma) = \omega$ per ogni $\sigma \in \Sigma_\omega$

Calcoliamo $\Phi(f_\bot)(\sigma)$ al variare di σ distinguendo i seguenti casi

- $\sigma = \omega$, $\Phi(f_\bot)(\omega) = \omega$
- $\sigma \neq \omega$ tale che $\boldsymbol{\mathcal{I}}(x \neq 0)(\sigma) = \mathbf{true}$ $(\sigma(x) \neq 0)$, allora $\Phi(f_\bot)(\sigma) = \omega$
- $\sigma \neq \omega$ tale che $\boldsymbol{\mathcal{I}}(x \neq 0)(\sigma) = \mathbf{false}$ $(\sigma(x) = 0)$, allora $\Phi(f_\bot)(\sigma) = \sigma_0$ (indichiamo con σ_0 la classe di stati che assegna ad x il valore 0)

Esempio (ctnd.2)

Poniamo $f_1 = \Phi(f_0)$

$$f_1(\sigma) = \begin{cases} \sigma_0 & \text{se } \sigma \neq \omega \text{ e } \boldsymbol{\mathcal{I}}(x \neq 0)(\sigma) = \mathbf{false} \\ \omega & \text{se } \sigma = \omega \text{ oppure } \boldsymbol{\mathcal{I}}(x \neq 0)(\sigma) = \mathbf{true} \end{cases}$$

f_1 è continua perchè è monotona

Calcoliamo $\Phi(\Phi(\bot)) = \Phi^2(\bot) = \Phi(f_1)$ al variare di σ distinguendo i seguenti casi

- $\sigma = \omega$, $\Phi(f_1)(\omega) = \omega$
- sia $\sigma \neq \omega$ tale che $\sigma(x) = 0$, allora $\Phi(f_1)(\sigma) = \sigma_0$
- sia $\sigma \neq \omega$ tale che $\sigma(x) = 1$, allora $\Phi(f_1)(\sigma) = f_1(\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(x := x - 1)(\sigma)) = f_1(\sigma[x/\boldsymbol{\mathcal{I}}(x-1)(\sigma)]) = f_1(\sigma_0) = \sigma_0$

- sia $\sigma \neq \omega$ tale che $\sigma(x) > 1$

  $\Phi(f_1)(\sigma) = f_1(\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(x := x - 1)(\sigma)) =$

  $f_1(\sigma[x/\boldsymbol{\mathcal{I}}(x - 1)(\sigma)]) = f_1(\sigma_i) = \omega$ per qualche $i > 0$

Esempio (ctnd.3)

Dimostriamo per induzione che
$\Phi^i(\bot) = f_i$, $i \in Nat$
dove $f_i \in [\Sigma_\omega \to \Sigma_\omega]$

$$f_i(\sigma) = \begin{cases} \sigma_0 & \text{se } \sigma(x) < i \\ \\ \omega & \text{altrimenti} \end{cases}$$

- CASO BASE: $i = 0$, $\Phi^0(\bot) = f_0$. E $f_0(\sigma) = \omega$ per ogni $\sigma \in \Sigma_\omega$
- PASSO INDUTTIVO: supponiamo che $\Phi^i(\bot) = f_i$. Dimostriamo che questo vale per $i + 1$.
  $\Phi^{i+1}(\bot) = \Phi(\Phi^i(\bot)) = \Phi(f_i)$. Basta far vedere che

$\Phi(f_i) = f_{i+1}$ distinguendo i seguenti casi

- $\sigma = \omega$, $\Phi(f_i)(\omega) = \omega = f_{i+1}(\omega)$.
- Sia σ tale che $\sigma(x) < i + 1$, allora
  $\Phi(f_i)(\sigma) = f_i(\boldsymbol{\mathcal{M}}^{\omega}_{\boldsymbol{\mathcal{I}}}(x := x - 1)(\sigma)) = \sigma_0 = f_{i+1}(\sigma)$
- altrimenti $\Phi(f_i)(\sigma) = \omega = f_{i+1}(\sigma)$

definiamo $\mu\Phi = \sqcup\{\Phi^i(\bot)|i \in Nat\}$ come la funzione
$f \in [\Sigma_\omega \to \Sigma_\omega]$

$$f(\sigma) = \begin{cases} \sigma_0 & \text{se } \sigma \neq \omega \\ \\ \omega & \text{altrimenti} \end{cases}$$

Verifica dei programmi

Studia le relazioni che valgono fra oggetti definiti in linguaggi di descrizione di diverso livello

- Un linguaggio di programmazione $\mathcal{L}$ che può essere compilato ed eseguito
- Un linguaggio di specifica $\mathcal{SP}$ di più alto livello per rappresentare requisiti e specifiche

Data una specifica $SP \in \mathcal{SP}$ ed un programma $S \in \mathcal{L}$ controllare se essi sono compatibili cioé se il significato di S soddisfa SP

Verifica dei programmi (ctnd.)

Esempio di specifica informale:

for all input values $a, b \in Nat$ with $a, b > 0$
the program computes $gcd(a, b)$

Specifica un po' più precisa:

for all $a, b \in Nat$, with $a, b > 0$
the program terminates with output value $gcd(a, b)$

Definizione di specifica di correttezza per programmi ricorsivi

Sia S un programma ricorsivo che computa il massimo comune divisore fra due numeri naturali positivi

for all $a, b \in Nat$
    if $a, b > 0$
    then $\boldsymbol{\mathcal{M}}(S)(a, b)$ is defined and $\boldsymbol{\mathcal{M}}(S)(a, b) = gcd(a, b)$

Una specifica di correttezza come questa, viene detta specifica di correttezza totale

Correttezza parziale e terminazione

Correttezza parziale:
for all input values $a, b \in Nat$ with $a, b > 0$
    if the program terminates, then it terminates with the
    output value $gcd(a, b)$,
and
    for all input values $a, b \in Nat$ with $a, b > 0$
      the program terminates

Correttezza parziale e terminazione per i programmi ricorsivi

for all $a, b \in Nat$

if $a, b > 0$ and $\boldsymbol{\mathcal{M}}(S)(a, b)$ is defined

$\boldsymbol{\mathcal{M}}(S)(a, b) = gcd(a, b)$

and

for all $a, b \in Nat$

if $a, b > 0$

then $\boldsymbol{\mathcal{M}}(S)(a, b)$ is defined

Correttezza parziale e terminazione

In generale spezzare la dimostrazione di correttezza di un programma in dimostrazione di:

- Correttezza parziale
- Terminazione

è conveniente perché più semplice.

Le nozioni di correttezza già date sono sufficienti per lo studio della verifica dei programmi ricorsivi.

Per i programmi flowchart e while dobbiamo invece introdurre delle definizioni più formali

Correttezza parziale e terminazione per programmi flowchart e while

Sia B una base per la logica predicativa, $\mathcal{I}$ una interpretazione di questa base e Σ il corrispondente insieme degli stati.
Sia S un programma flowchart di L_1^B oppure un programma while di L_2^B, e sia $\mathcal{M}_{\mathcal{I}}(S)$ il significato del programma.
Siano inoltre p e q due formule di WFF_B chiamate rispettivamente *precondizione* e *postcondizione*. Il programma 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 $\mathcal{M}_{\mathcal{I}}(S)(\sigma)$ is defined
  then $\mathcal{I}(q)(\mathcal{M}_{\mathcal{I}}(S)(\sigma)) = \mathbf{true}$

- 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 $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)$ 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 $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)$ is defined and $\boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)) = \mathbf{true}$

Weakest preconditions and strongest postconditions

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

Le *precondizioni liberali più deboli* del programma 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 $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)$ è indefinito oppure $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)$ è definito e $\boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)) = \mathbf{true}$

Le *precondizioni più deboli* del programma 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 $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)$ è definito e $\boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)) = \mathbf{true}$

Le *postcondizioni più forti* della formula p e del programma 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 $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma') = \sigma$

Weakest preconditions and strongest postconditions (ctnd.1)

Teorema1

Siano dati B, $\boldsymbol{\mathcal{I}}$, Σ, S e q. Sia $p \in WFF_B$ la precondizione liberale più debole di S e q. Allora

a. S è parzialmente corretto rispetto a p e q

b. per una qualunque formula $r \in WFF_B$, tale che S è parzialmente corretto rispetto ad r e q, si ha che, per tutti gli stati $\sigma \in \Sigma$, $\boldsymbol{\mathcal{I}}(r)(\sigma) \supset \boldsymbol{\mathcal{I}}(p)(\sigma)$

Dim:

p è la precondizione liberale più debole di S e q. Dunque per ogni $\sigma \in \Sigma$:

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

a. E' vero che S è parzialmente corretto rispetto a p e a q?
Diciamo che S è parzialmente corretto rispetto a p e a q se
$(\boldsymbol{\mathcal{I}}(p)(\sigma) \wedge \boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)\downarrow) \supset \boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma))$
questo è banalmente vero.

b. Sia $r \in WFF_B$ tale che S è parzialmente corretto rispetto a r e a q. Allora per ogni $\sigma \in \Sigma$

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

Sia $\boldsymbol{\mathcal{I}}(r)(\sigma) = \mathbf{true}$, allora

– o $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)\uparrow$,

– oppure $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)\downarrow$, quindi anche $\boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma))$

quindi $\boldsymbol{\mathcal{I}}(r)(\sigma) \supset \boldsymbol{\mathcal{I}}(p)(\sigma)$

Weakest preconditions and strongest postconditions (ctnd.2)

Teorema2

Siano dati B, $\boldsymbol{\mathcal{I}}$, Σ, S e q. Sia $p \in WFF_B$ la precondizione più debole di S e q. Allora

a. S è totalmente corretto rispetto a p e q

b. per una qualunque formula $r \in WFF_B$, tale che S è totalmente corretto rispetto ad r e q, si ha che, per tutti gli stati $\sigma \in \Sigma$, $\boldsymbol{\mathcal{I}}(r)(\sigma) \supset \boldsymbol{\mathcal{I}}(p)(\sigma)$

Dim:

p è la precondizione più debole del programma S e di q, dunque, per ogni $\sigma \in \Sigma$

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

a. E' vero che S è totalmente corretto rispetto a p e a q? Diciamo che S è totalmente corretto rispetto a p e a q se
$\boldsymbol{\mathcal{I}}(p)(\sigma) \supset (\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma) \downarrow \wedge \boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)))$
questo è banalmente vero.

b. Sia $r \in WFF_B$ tale che S è totalmente corretto rispetto ad r e a q. Allora per ogni $\sigma \in \Sigma$

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

ma allora

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

Weakest preconditions and strongest postconditions (ctnd.3)

Teorema3

Siano dati B, $\boldsymbol{\mathcal{I}}$, Σ, S e q. Sia $q \in WFF_B$ la postcondizione più forte di S e p. Allora

a. S è parzialmente corretto rispetto a p e q

b. per una qualunque formula $r \in WFF_B$, tale che S è parzialmente corretto rispetto ad p ed r, si ha che, per tutti gli stati $\sigma \in \Sigma$, $\boldsymbol{\mathcal{I}}(q)(\sigma) \supset \boldsymbol{\mathcal{I}}(r)(\sigma)$

Dim:

q è la post condizione più forte di S e p.

$$\boldsymbol{\mathcal{I}}(q)(\sigma) \equiv \exists \sigma' \left(\boldsymbol{\mathcal{I}}(p)(\sigma') \wedge \boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma') = \sigma\right)$$

a. E' vero che S è parzialmente corretto rispetto a p e a q?
   Diciamo che S è parzialmente corretto rispetto a p e a q se
   $(\boldsymbol{\mathcal{I}}(p)(\sigma) \wedge \boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma) \downarrow) \supset \boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma))$.
   Sia $\sigma \in \Sigma$ tale che $\boldsymbol{\mathcal{I}}(p)(\sigma) \wedge \boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma) \downarrow$. Allora
   $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma) = \bar{\sigma}$, $\bar{\sigma} \in \Sigma$.
   Applicando la definizione di post condizione più forte si ha che
   $\boldsymbol{\mathcal{I}}(q)(\bar{\sigma})$, ma $\bar{\sigma} = \boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma)$, da cui $\boldsymbol{\mathcal{I}}(q)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma))$.

b. Sia r tale che per ogni $\sigma \in \Sigma$,
   $(\boldsymbol{\mathcal{I}}(p)(\sigma) \wedge \boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma) \downarrow) \supset \boldsymbol{\mathcal{I}}(r)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma))$,
   vogliamo fare vedere che per ogni σ $\boldsymbol{\mathcal{I}}(q)(\sigma) \supset \boldsymbol{\mathcal{I}}(r)(\sigma)$.
   Per hp, dato un σ,

$$\boldsymbol{\mathcal{I}}(q)(\sigma) \supset (\boldsymbol{\mathcal{I}}(p)(\sigma') \wedge \boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma') = \sigma) \supset \boldsymbol{\mathcal{I}}(r)(\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma'))$$

Prime dimostrazioni di correttezza

- Daremo alcune dimostrazioni di correttezza che illustreranno l'uso delle nozioni che abbiamo introdotto
- Queste dimostrazioni di correttezza sono 'ingenue' poiché sono basate su argomentazioni matematiche piuttosto che su uno dei metodi di verifica che discuteremo dopo
- Queste dimostrazioni a 'stile libero' partono dalla semantica dei linguaggi di programmazione e usano, come strumento principale, l'induzione.
- Generalmente l'induzione viene fatta sulla lunghezza delle sequenze di computazione (nel caso della semantica operazionale) e delle catene (nel caso della semantica denotazionale)

Prime dimostrazioni di correttezza (ctnd.1)

- Le dimostrazioni di questo tipo sono tediose e, soprattutto, ripetitive in quanto la stessa argomentazione appare più volte
- Questo giustifica lo sviluppo di metodi per la verifica dei programmi che utilizzano una notazione più appropriata
- Vedremo alcune dimostrazioni basate sulla semantica operazionale
- seguite da una dimostrazione basata sulla semantica denotazionale

Prime dimostrazioni di correttezza (ctnd.2)

- Se il programma viene costruito a partire da un'idea 'iterativa', una dimostrazione che utilizza la semantica operazionale è più appropriata
- Se il programma è costruito attorno ad un'idea 'ricorsiva', una dimostrazione che utilizza la semantica denotazionale è più immediata

Esempio 1.

Proviamo la correttezza parziale di un programma flowchart con l'aiuto della semantica operazionale. Tipicamente queste dimostrazioni funzionano in questo modo.

1. Prima si dimostra un lemma riguardante le sequenze di computazione usando l'induzione
2. La correttezza parziale viene derivata da questo lemma.

PROVIAMO CHE:

Il programma che calcola la radice quadrata è parzialmente corretto rispetto alle formule $x = c$ e $y_1 = \sqrt{c}$, dove c è qualche variabile differente da x, y_1, y_2, y_3.

Esempio 1.

Prima si dimostra un lemma che esprime una relazione fra i valori delle variabili ogni volta che l'etichetta `test` viene raggiunta (ad ogni ciclo)

In particolare il lemma dice: Sia $(l_1, \sigma_1), \ldots, (l_n, \sigma_n)$ una sequenza di computazione (quindi in cui $l_1 = begin$) con $n \geq 1$ ed $l_n =$ `test`. Allora

$$\begin{array}{ll} \sigma_n(x) = & \sigma_1(x) \\ (\sigma_n(y_1))^2 \leq & \sigma_n(x) \\ \sigma_n(y_2) = & 2\sigma_n(y_1) + 1 \\ \sigma_n(y_3) = & (\sigma_n(y_1) + 1)^2 \end{array}$$

Esempio 1.

La dimostrazione del lemma è per induzione sulla lunghezza della sequenza di computazione

- CASO BASE: $n = 1$, $l_n = l_1 = begin$, $l_1 = \texttt{test}$ non è possibile e quindi la proposizione vale banalmente.
- PASSO INDUTTIVO: Supponiamo che la proposizione valga per $j = 1, \ldots, n$ e che adesso debba essere provata per $n + 1$.

  Assumiamo che $l_{n+1} = \texttt{test}$. La dimostrazione si divide in due casi concordemente al valore di l_n.

  Se $l_n = begin$, quindi $n = 1$, allora

$\sigma_{n+1}(x) = \sigma_n(x) = \sigma_1(x)$
$\sigma_{n+1}(y_1) = 0$
$\sigma_{n+1}(y_2) = 1$
$\sigma_{n+1}(y_3) = 1$
E' ovvio che la proposizione vale in questo caso

Supponiamo che $l_n = \texttt{upd}$, allora ricaviamo tutte le informazioni che abbiamo. Poiché upd è un assegnamento, lo stato dalla configurazione (l_n, σ_n) alla configurazione (l_{n+1}, σ_{n+1}) cambia.

$$\sigma_{n+1} = \sigma_n[y_3/\boldsymbol{\mathcal{I}}(y_2 + y_3)\sigma_n]$$

Inoltre $l_{n-1} = \texttt{loop}$ e

$$\sigma_n = \sigma_{n-1}[y_1/\boldsymbol{\mathcal{I}}(y_1+1)\sigma_{n-1}][y_2/\boldsymbol{\mathcal{I}}(y_2+2)(\sigma_{n-1})]$$

Infine $l_{n-2} = \texttt{test}$, $\sigma_{n-1} = \sigma_{n-2}$ e $\boldsymbol{\mathcal{I}}(y_3 \leq x)(\sigma_{n-2}) = \mathbf{true}$

Vediamo che

$$\begin{array}{llll} \sigma_{n+1}(x) = & \sigma_n(x) = & \sigma_{n-1}(x) = & \sigma_{n-2}(x) \\ \sigma_{n+1}(y_1) = & \sigma_n(y_1) = & \sigma_{n-1}(y_1)+1 = & \sigma_{n-2}(y_1)+1 \\ \sigma_{n+1}(y_2) = & \sigma_n(y_2) = & \sigma_{n-1}(y_2)+2 = & \sigma_{n-2}(y_2)+2 \end{array}$$

$$\sigma_{n+1}(y_3) = \sigma_n(y_2) + \sigma_n(y_3) =$$
$$\sigma_{n-1}(y_2) + 2 + \sigma_{n-1}(y_3) = \sigma_{n-2}(y_2) + 2 + \sigma_{n-2}(y_3)$$

L'ipotesi induttiva, con $j = n-2$, può adesso venire utilizzata per

ottenere il risultato richiesto.

$$\begin{aligned}\sigma_{n+1}(x) &= \sigma_{n-2}(x)\\ &= \sigma_1(x)\end{aligned}$$

$$\begin{aligned}(\sigma_{n+1}(y_1))^2 &= (\sigma_{n-2}(y_1)+1)^2\\ &= \sigma_{n-2}(y_3)\\ &\leq \sigma_{n-2}(x)\\ &= \sigma_{n+1}(x)\end{aligned}$$

$$\begin{aligned}\sigma_{n+1}(y_2) &= \sigma_{n-2}(y_2)+2\\ &= 2\sigma_{n-2}(y_1)+3\\ &= 2\sigma_{n+1}(y_1)+1\end{aligned}$$

$$\begin{aligned}\sigma_{n+1}(y_3) &= \sigma_{n-2}(y_2) + \sigma_{n-2}(y_3) + 2 \\ &= 2\sigma_{n-2}(y_1) + 1 + (\sigma_{n-2}(y_1) + 1)^2 + 2 \\ &= (\sigma_{n-2}(y_1) + 2)^2 \\ &= (\sigma_{n+1}(y_1) + 1)^2\end{aligned}$$

Questo completa la dimostrazione del lemma. Per provare la correttezza parziale, si consideri una computazione finita

$$(l_1, \sigma_1) \Rightarrow \ldots \Rightarrow (l_k, \sigma_k)\,,$$

dove $k \geq 2$, $l_1 = \mathit{begin}$, $l_k = \mathit{end}$ e $\sigma_1(x) = \sigma_1(c)$.

Bisogna dimostrare che $\sigma_k(y_1) = \sqrt{\sigma_1(c)}$ oppure, equivalentemente, che

- $(\sigma_k(y_1))^2 \leq \sigma_1(c)$ e,
- $\sigma_1(c) < (\sigma_k(y_1) + 1)^2$.

Adesso: $l_{k-1} = \texttt{test}, \sigma_{k-1} = \sigma_k$, e $\boldsymbol{\mathcal{I}}(y_3 \leq x)(\sigma_{k-1}) = \mathbf{false}$.
Quindi

$$\sigma_{k-1}(y_3) > \sigma_{k-1}(x).$$

Questo, insieme al lemma restituisce

$$\begin{aligned}(\sigma_k(y_1))^2 &= (\sigma_{k-1}(y_1))^2 \\ &\leq \sigma_{k-1}(x) \\ &= \sigma_1(x) \\ &= \sigma_1(c)\end{aligned}$$

e inoltre

$$\begin{aligned}(\sigma_k(y_1)+1)^2 &= (\sigma_{k-1}(y_1)+1)^2 \\ &= \sigma_{k-1}(y_3) \\ &> \sigma_{k-1}(x) \\ &= \sigma_1(x) \\ &= \sigma_1(c)\end{aligned}$$

Esempio 2.

Ora dimostriamo la correttezza totale per lo stesso programma flowchart.

La correttezza totale viene anch'essa dimostrata rispetto alle due formule:

- $x = c$
- $y_1 = \sqrt{c}$

LEMMA PRELIMINARE

per ogni stato $\sigma_1 \in \Sigma$, ed ogni numero $k \in Nat$, tale che $0 \leq k \leq \sqrt{\sigma_1(x)}$, c'è una sequenza di computazione

$$(l_1, \sigma_1) \Rightarrow \ldots \Rightarrow (l_{3k+2}, \sigma_{3k+2})$$

per questo stato σ_1, tale che

$$\begin{aligned} l_{3k+2} &= \texttt{test} \\ \sigma_{3k+2}(x) &= \sigma_1(x) \\ \sigma_{3k+2}(y_1) &= k \\ \sigma_{3k+2}(y_2) &= 2k+1 \\ \sigma_{3k+2}(y_3) &= (k+1)^2 \end{aligned}$$

La dimostrazione del lemma viene fatta per induzione su k

- CASO BASE. La proprietà vale per $k = 0$. In particolare, $l_2 = \texttt{test}$, $\sigma_2(x) = \sigma_1(x)$...

- PASSO INDUTTIVO. Supponiamo che $k+1 \leq \sqrt{\sigma_1(x)}$.
  Poiché $\sigma_{3k+2}(y_3) = (k+1)^2$ e $\sigma_{3k+2}(x) = \sigma_1(x)$ si ha che

$$\sigma_{3k+2}(y_3) \leq \sigma_{3k+2}(x).$$

  Quindi abbiamo che $l_{3k+3} = \texttt{loop}$, $l_{3k+4} = \texttt{upd}$, e $l_{3(k+1)+2} = \texttt{test}$. Inoltre segue che

$$\begin{aligned}
\sigma_{3(k+1)+2}(x) &= \sigma_{3k+2}(x) = \sigma_1(x) \\
\sigma_{3(k+1)+2}(y_1) &= \sigma_{3k+2}(y_1) + 1 = k+1 \\
\sigma_{3(k+1)+2}(y_2) &= \sigma_{3k+2}(y_2) + 2 = 2(k+1)+1 \\
\sigma_{3(k+1)+2}(y_3) &= \sigma_{3k+2}(y_3) + \sigma_{3k+2}(y_2) + 2 \\
&= (k+1)^2 + 2k + 1 + 2 = (k+2)^2
\end{aligned}$$

Adesso si consideri la sequenza di computazione per lo stato $\sigma_1 \in \Sigma$:

$$(l_1, \sigma_1) \Rightarrow \ldots \Rightarrow (l_{3k+2}, \sigma_{3k+2})$$

Con $k = \sqrt{\sigma_1(x)}$. Per il lemma preliminare,

$$\begin{aligned} \sigma_{3k+2}(y_3) &= (k+1)^2 \\ &= (\sqrt{\sigma_1(x)} + 1)^2 \\ &> \sigma_1(x) \\ &= \sigma_{2k+2}(x) \end{aligned}$$

e l'etichetta $l_{3k+2} = \texttt{test}$. In questo modo abbiamo che $l_{3k+3} = end$, e la sequenza di computazione considerata è una computazione. Inoltre

$$\begin{aligned}\sigma_{3k+3}(y_1) &= \sigma_{3k+2}(y_1) \\ &= k \\ &= \sqrt{\sigma_1(x)}\end{aligned}$$