Regole per la correttezza parziale di programmi deterministici

Sistema formale PD:

$$\frac{}{\{p\}\ skip\ \{p\}}\ \text{skip} \qquad \frac{}{\{p[u := t]\}\ u := t\ \{p\}}\ :=$$

$$\frac{\{p \wedge B\}\ S_1\ \{q\}\quad \{p \wedge \neg B\}\ S_2\ \{q\}}{\{p\}\ \textbf{if}\ B\ \textbf{then}\ S_1\ \textbf{else}\ S_2\textbf{fi}\ \{q\}}\ \textbf{if} \qquad \frac{\{p \wedge B\}\ S\ \{p\}}{\{p\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}\ \{p \wedge \neg B\}}\ \textbf{while}$$

$$\frac{\{p\}\ S_1\ \{r\}\quad \{r\}\ S_2\ \{q\}}{\{p\}\ S_1; S_2\ \{q\}}\ ; \qquad \frac{p \rightarrow p_1\quad \{p_1\}\ S\ \{q_1\}\quad q_1 \rightarrow q}{\{p\}\ S\ \{q\}}\ \rightarrow$$

L'asserzione p nella regola **while** è chiamata **invariante di ciclo**.

La regola $\rightarrow$ è chiamata regola di **conseguenza logica**.

Composizionalità di PD

Il sistema formale *PD* è **composizionale**, in quanto il comportamento di un programma complesso è spiegato in termini del comportamento dei suoi sottoprogrammi.

Esempio 1

Sia $S \equiv x := x + 1; y := y + 1$. Dimostriamo la correttezza in *PD* della formula

$$\{x = y\}\ S\ \{x = y\}\ , \tag{1}$$

cioè dimostriamo che la formula (1) è derivabile in *PD*.

Applichiamo l'assioma $:=$ due volte:

- per sostituzione all'indietro, $(x = y)[y := y + 1] \equiv x = y + 1$,
  per l'assioma $:=$ $\quad \{x = y + 1\}\ y := y + 1\ \{x = y\}$

- per sostituzione all'indietro, $(x = y + 1)[x := x + 1] \equiv x + 1 = y + 1$,
  per l'assioma $:=$ $\quad \{x + 1 = y + 1\}\ x := x + 1\ \{x = y + 1\}$

- per la regola ; $\quad \{x + 1 = y + 1\}\ x := x + 1; y := y + 1\ \{x = y\}$

- dalla regola di conseguenza $\rightarrow$, poichè $(x = y)\ \rightarrow\ (x+1 = y+1)$, deriviamo la (1).

Esempio 2

Dimostrare che la seguente formula è derivabile in *PD*:

$$\{\textbf{true}\}\ (x := 1; a[1] := 2; a[x] := x)\ \{a[1] = 1\}$$

Esempio 3: iterazione

Sia *DIV* il seguente programma per calcolare quoziente e resto della divisione di due naturali x, y:

$$DIV \equiv quo := 0; rem := x; S_0$$

dove

$$S_0 \equiv \textbf{while } rem \geq y \textbf{ do } rem := rem - y; quo := quo + 1 \textbf{ od}$$

L'obiettivo è dimostrare che: "se x, y sono interi positivi e *DIV* termina, allora *quo* è il quoziente intero e *rem* è il resto della divisione di x per y" .

Dimostriamo che è derivabile in *PD*:

$$\{x \geq 0 \ \wedge \ y \geq 0\} \ DIV \ \{quo \cdot y + rem = x \ \wedge \ 0 \leq rem < y\} \qquad (2)$$

Qual è l'**invariante** del ciclo while?

$$p \equiv quo \cdot y + rem = x \ \ \wedge \ \ rem \geq 0$$

Il programma DIV annotato

$\{x \geq 0 \ \wedge \ y \geq 0\}$

$quo := 0;$
$rem := x;$

$\{\textbf{inv} : p\}$

while $rem \geq y$ **do**

$\{p \ \wedge \ rem \geq y\}$

$rem := rem - y;$
$quo := quo + 1$

$\{p\}$

od

$\{p \ \wedge \ rem < y\}$
$\{quo \cdot y + rem = x \ \wedge \ 0 \leq rem < y\}$

Ma abbiamo dimostrato la **correttezza** (parziale) del programma?

Ossia: dal fatto che la formula (2) è derivabile in *PD* possiamo dedurre

$$\models \{x \geq 0 \ \wedge \ y \geq 0\} \ DIV \ \{quo \cdot y + rem = x \ \wedge \ 0 \leq rem < y\} \ \ ?$$

Cioè

$$\mathcal{M}[\![DIV]\!]([\![x \geq 0 \ \wedge \ y \geq 0]\!]) \subseteq [\![quo \cdot y + rem = x \ \wedge \ 0 \leq rem < y]\!] \ \ ?$$

Un sistema formale G si dice **corretto** rispetto alla semantica operazionale, quando la derivabilità in G di una formula implica la sua verità.

Correttezza di un sistema formale

Sia G un sistema formale per derivare formule $\{p\}\ S\ \{q\}$ per una classe C di programmi.

- G è **corretto** rispetto alla semantica operazionale per la **correttezza parziale** se, per ogni formula $\{p\}\ S\ \{q\}$, con S in C,

$$\vdash_G \{p\}\ S\ \{q\} \quad \text{implica} \quad \models \{p\}\ S\ \{q\}\ .$$

- G è **corretto** rispetto alla semantica operazionale per la **correttezza totale** se, per ogni formula $\{p\}\ S\ \{q\}$, con S in C,

$$\vdash_G \{p\}\ S\ \{q\} \quad \text{implica} \quad \models_{tot} \{p\}\ S\ \{q\}\ .$$

Correttezza di PD

Teorema.
Il sistema formale *PD* è **corretto** rispetto alla semantica operazionale per la **correttezza parziale** di **programmi deterministici**, cioè

$$\vdash_{PD} \{p\}S\{q\} \implies \mathcal{M}[\![S]\!]([\![p]\!]) \subseteq [\![q]\!] \ .$$

dim. Per **induzione** sull'altezza della derivazione in *PD*.
Caso base: gli assiomi sono **veri**.
Passo induttivo: le regole sono **corrette**, cioè se le premesse sono vere, allora anche la conclusione è vera.

Terminazione del programma DIV

Il programma *DIV* **termina**?

La derivabilità della formula (2) in *PD* garantisce la **terminazione** di *DIV*?

Il sistema *PD* permette di dimostrare la **terminazione** di programmi?

Correttezza totale

Il sistema *PD* non permette di dimostrare la **terminazione**. In particolare, il programma *DIV* **diverge** per ogni stato iniziale in cui y vale 0.

L'unica regola che introduce la possibilità di non terminazione è il **while**.

Per garantire la **correttezza totale** sostituiamo la regola **while** con la regola

$$\frac{\begin{array}{l}\{p \wedge B\}\ S\ \{p\} \\ \{p \wedge B \wedge t = z\}\ S\ \{t < z\} \\ p \rightarrow t \geq 0\end{array}}{\{p\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}\ \{p \wedge \neg B\}}\ \textbf{while tot}$$

dove t è un'espressione intera e z una variabile intera che non compare in p, B, t, S.

Chiamiamo *TD* il sistema così ottenuto.

Analisi della regola while tot

L'espressione t è chiamata **funzione di terminazione** (**bound function**) e limita la complessità del ciclo **while**.

Lo scopo della variabile z è contenere il valore iniziale di t, prima di ogni iterazione. Poichè z non compare in S, z non viene modificata da S.

Per la seconda premessa, t **decresce** ad ogni iterazione.

Per la terza premessa, t è **non negativa**.

Quindi la computazione è garantita **terminare**.

Correttezza di TD

Teorema.
Il sistema formale *TD* è **corretto** rispetto alla semantica operazionale per la **correttezza totale** di **programmi deterministici**, cioè

$$\vdash_{TD} \{p\}S\{q\} \implies \mathcal{M}_{tot}[\![S]\!]([\![p]\!]) \subseteq [\![q]\!] \ .$$

dim. Per **induzione** sull'altezza della derivazione in *TD*.
Caso base: gli assiomi sono **veri**.
Passo induttivo: le regole sono **corrette**, cioè se le premesse sono vere, allora anche la conclusione è vera.

Esempio 3'

Dimostriamo che: " se x è un intero non negativo e y è positivo, allora *DIV* **termina**, *quo* è il quoziente intero e *rem* è il resto della divisione di x per y."

A tal fine dimostriamo che è derivabile in *TD*:

$$\{x \geq 0 \ \wedge \ y > 0\} \ DIV \ \{quo \cdot y + rem = x \ \wedge \ 0 \leq rem < y\} \qquad (3)$$

N.B. La formula

$$\{x \geq 0 \ \wedge \ y \geq 0\} DIV \{quo \cdot y + rem = x \ \wedge \ 0 \leq rem < y\}$$

non è invece derivabile in *TD*. Infatti questa formula non è corretta rispetto a $\mathcal{M}_{tot}$.

Qual è l'**invariante** del ciclo?

$$p' \equiv quo \cdot y + rem = x \ \wedge \ rem \geq 0 \ \wedge \ y > 0$$

Qual è la **funzione di terminazione**?

$$t \equiv rem$$

Correttezza totale del programma DIV

$\{x \geq 0 \ \wedge \ y > 0\}$

$quo := 0;$
$rem := x;$

$\{\textbf{inv} : p'\}$ $\{\textbf{bd} : t\}$

while $rem \geq y$ **do**

$\{p' \ \wedge \ rem \geq y\}$

$rem := rem - y;$
$quo := quo + 1$

$\{p'\}$

od

$\{p' \ \wedge \ rem < y\}$
$\{quo \cdot y + rem = x \ \wedge \ 0 \leq rem < y\}$

Fattoriale

Sia *FATT* il seguente programma

$x := n_0$;
$fatt := 1$;
while$(x > 0)$**do**
    $fatt := fatt * x$;
    $x := x - 1$;
od

Dimostriamo che: "se n_0 è un intero positivo, allora *FATT* calcola il fattoriale di n_0."

A tal fine basta dimostrare che

$$\{n_0 > 0\}\ FATT\ \{fatt = !n_0\}$$

è derivabile in *TD*.

Qual è l'**invariante** del ciclo **while**?

$$p \equiv fatt = \frac{n_0!}{x!} \ \wedge \ x \geq 0$$

Qual è la **funzione di terminazione**?

$$t \equiv x$$

Algoritmo di Euclide

$$\text{MCD}(x,y) = \begin{cases} x & \text{se x = y} \\ \text{MCD}(x-y,y) & \text{se x > y} \\ \text{MCD}(x,y-x) & \text{se x < y} \end{cases}$$

$x := m;$
$y := n;$
while $(x! = y)$ **do**
    if $(x > y)$ **then** $x := x - y;$
        else $y := y - x;$
od

Per dimostrare che il programma è corretto basta dimostrare che

$$\{m, n > 0\}\ EUCLIDE\ \{x = MCD(m,n)\}$$

è derivabile in *TD*.

Moltiplicazione Russa

$\{\ m, n > 0\ \}$

$x := m;$
$y := n;$
$z := 0;$
while $\neg(x = 0)$ **do**
    if (x pari) **then** $x := x/2;\ y := y * 2;$
                else $x := x - 1;\ z := z + y;$
od

$\{\ z = m * n\ \}$

Dimostrare la correttezza della Moltiplicazione Russa.

Qual è l'invariante del while?

Qual è la funzione di terminazione?