Logica di Hoare: correttezza

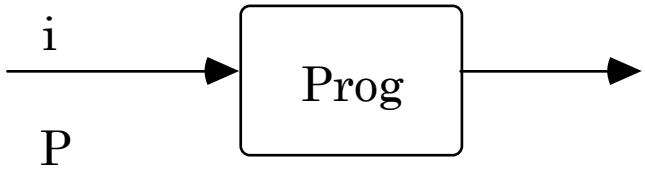
- Associa ad ogni programma di un linguaggio L la ***relazione calcolata dal programma*** (formula logica)

- Programma P che termina sull'ingresso i e produce l'output o:

R_P: $R_P(i,o)$

- R_P viene generalmente espressa nel calcolo dei predicati

- Lega ***pre-*** e ***post-condizione***:



- Se la precondizione P è vera sui dati di ingresso i ed il programma Prog termina, allora la postcondizione è vera sui dati di uscita o.

Metodo assiomatico

- Il metodo assiomatico viene utilizzato per eseguire ***dimostrazioni formali*** della correttezza di un programma

- Per ***ciascuna istruzione*** del linguaggio occorre definire come sono correlate pre- e post-condizioni

- Specificato attraverso ***assiomi*** o ***regole di inferenza***

- Dimostrare che Prog è ***corretto*** rispetto alla specifica:

{P} Prog {Q} ***specifica***

  significa dimostrare che per ogni insieme dei dati di ingresso che soddisfa P, Prog termina ed i dati di uscita soddisfano Q

- Q è ***derivabile*** a partire da P, usando assiomi e regole di inferenza che definiscono la semantica del linguaggio in cui è scritto il programma Prog.

Esempio (mini-C):

$$\{P\}\ \textbf{scanf}(\text{“\%d”},\&n)\ \{P[i/n]\}$$

$$\{P\}\ \textbf{printf}(\text{“\%d”},n)\ \{P\}$$

$$\{P\}\ n=\text{exp}\ \{P[v/n]\}$$

dove v è il valore risultante dalla valutazione di exp

$$\frac{\{P_1\}\ i_1\ \{P_2\} \qquad \{P_2\}\ i_2\ \{P_3\}}{\{P_1\}\ i_1; i_2\ \{P_3\}}$$

$$\frac{\{P_1 \wedge v(bool)=true\}\ i_1\ \{P_2\} \qquad \{P_1 \wedge v(bool)=false\}\ i_2\ \{P_2\}}{\{P_1\}\ \textbf{if}\ (bool)\ i_1\ \textbf{; else}\ i_2\ \{P_2\}}$$

$$\frac{\{P \wedge v(bool)=true\}\ i\ \{P\}}{\{P\}\ \textbf{while}\ (bool)\ \{i\} \qquad \{P \wedge v(bool)=false\}}$$

- Se P è vera prima di eseguire l'istruzione while e questa termina, P è vera anche dopo (ed a questo punto bool ha valore falso):***invariante del ciclo***.

Prove formali di correttezza: esempio

- Calcolo del modulo di due numeri interi non negativi: r=x **mod** y (usando solo +, -, *).
  P: x>=0 **and** y>0
  Q: deve esistere un intero q e un intero r:
  x=y*q+r **and** 0<=r<y

```
void main()
int x,y,q;
{  scanf("%d%d", &x, &y);          {P: x>=0 and y>0}
   q=0;
   r=x;                    {x=y*q+r  and r>=0}
   while (r>=y)
     {     r=r-y;
           q=q+1;        }
   printf("%d\t%d", r,q);        {Q: x=y*q+r  and 0<=r<y}
}
```

```
scanf("%d%d", &x, &y);              {P: x>=0 and y>0}

   q=0;
   r=x;                        {W: x=y*q+r  and r>=0}

   {W}
   while (r>=y)
     {
          r=r-y;
          q=q+1;
     }
   {Q:     x=y*q+r  and  0<=r<y}
```

Dimostriamo che W è l'invariante del ciclo: *{W: x=y*q+r and r>=0}*

while (r>=y) *{W and r>=y}*
{
r=r-y;
q=q+1;
}
{W and r<y}

Dopo una iterazione:
r'=r-y;
q'=q+1;
x'=x;
y'=y
*{W': x'=y'*q'+r' and r'>=0}*

è vera, infatti:
r'>=0, perché si è decrementato r di y e si aveva r>=y
x'=y'*q'+r', perché si è incrementato q e decrementato r di y.

All'uscita dal ciclo **while:** *{W and r<y}* = *{x=y*q+r and 0<=r<y}* = Q