

# Verifica formale di programmi con la logica di Hoare

A.Gargantini  
Infomatica III  
Unibg 2007

# Scopo

- provare che i programmi sono corretti (sicuri)
  - diverso dal testing (qui proviamo veramente)
  - lo faremo a mano anche se ci sono strumenti
  - **analisi statica**: non richiede esecuzione del programma
- Si utilizza solo per programmi critici !
  - casi di studio: java card
  - vedi <http://www.key-project.org/>
  - in combinazione con JML: ESC/Java2

# materiale

- Questi lucidi
- Altro materiale su web
  - file AxiomaticSemantics.pdf
  - Capitolo 10, sezione 2 del Finkel: finkel10.pdf

# While programs

- I nostri programmi saranno composti dalle seguenti istruzioni
  - Assignments  $y := t$
  - Composition  $S1; S2$
  - If-then-else  $\text{if } e \text{ then } S1 \text{ else } S2 \text{ fi}$
  - While  $\text{while } e \text{ do } S \text{ od}$
- Variabili *intere*

# Cosa vuol dire provare che un programma è corretto

Proveremo

$$\{P\}S\{Q\}$$

- Se  $P$  è vero prima di eseguire il programma  $S$ , allora dopo varrà  $Q$ 
  - $P$ : preconditione – vale prima dell'esecuzione
  - $Q$ : postcondizione – vale alla fine dell'esecuzione
  - $S$ : programma o frammento di programma
- $P$  e  $Q$  vanno pensate e scritte

# Asserzioni

- P e Q sono asserzioni
- In genere metteremo le asserzioni tra { }
- Scritte in logica proposizionale
  - Con l'uso di variabili che saranno anche nel programma
  - $\wedge$  per AND
    - Esempio  $\{ x > 0 \wedge y = 3 \}$
  - $\vee$  per OR
  - $\neg$  per NOT
  - $\rightarrow$  per implica :  $A \rightarrow B$  : A implica B, cioè A è più forte di B, es:  $x = 2 \rightarrow x > 0$

# Esemio: Greatest common divisor GCD

$\{x1 > 0 \wedge x2 > 0\}$

$y1 := x1;$

$y2 := x2;$

while  $\neg(y1 = y2)$  do

    if  $y1 > y2$  then  $y1 := y1 - y2$

        else  $y2 := y2 - y1$  fi

od

$\{y1 = \text{gcd}(x1, x2)\}$

# Esempio 2

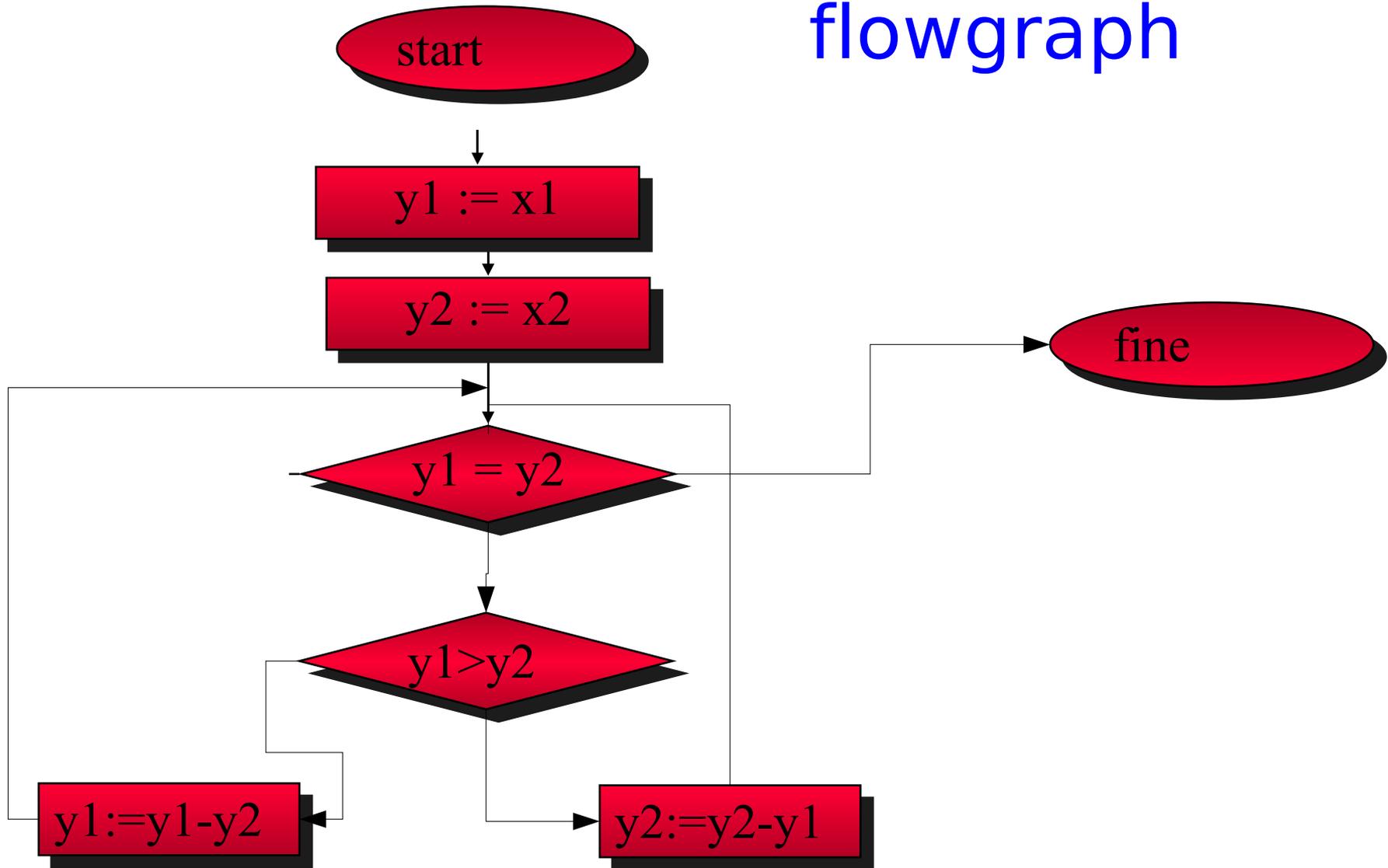
- calcolo del massimo tra due numeri:

{true}

```
if x > y then m := x  
else m := y
```

{m = max(x,y)}

# flowgraph



# Perchè funziona?

Suppose that  $y_1, y_2$  are both positive integers.

- If  $y_1 > y_2$  then  $\gcd(y_1, y_2) = \gcd(y_1 - y_2, y_2)$ 
  - Esempio:  $\gcd(10, 4) = \gcd(6, 4)$
- If  $y_2 > y_1$  then  $\gcd(y_1, y_2) = \gcd(y_1, y_2 - y_1)$
- If  $y_1 = y_2$  then  $\gcd(y_1, y_2) = y_1 = y_2$

# Come fare a dimostrare $\{P\}S\{Q\}$

- Divideremo la dimostrazioni in piccoli passi
- ad ogni passo applichiamo una regola (assioma) che ci permette di ridurre la dimostrazione ad un caso più semplice

# 1. Assignment axiom

$$\{p [y \rightarrow t]\} y := t \{p\}$$

$p[y \rightarrow t]$ : sostituisci  $y$  in  $p$  con  $t$

: al posto di  $y$  ci metto  $t$

Esempi:

$$\{u+1 > 3\} x := u+1 \{x > 3\}$$

$$\{y+y < z\} x := y \{x+y < z\}$$

$$\{y+5=10\} y := y+5 \{y=10\}$$

$$\{2*(y+5) > 20\} y := 2*(y+5) \{y > 20\}$$

$$\{6 > 10\} x := 6 \{x > 10\}$$

Giustificazione: scrivi  $p$  con  $y'$  invece di  $y$  e aggiungi il congiunto  $(\wedge) y'=t$ . Ora elimina  $y'$  sostituendolo con  $t$ .

# Esempio

- voglio trovare la precondizione di

$a := b + 10 \{a > 0\}$

- $p[a \rightarrow b + 10] = \{b + 10 > 0\} = \{b > -10\}$
- ho provato che

$\{b > -10\} a := b + 10 \{a > 0\}$

# Esercizio

- Dato il seguente programma

$x = y/2 - 1 \{x < 10\}$

quale è la precondizione?

# in avanti

- così andiamo all'indietro (da post calcoliamo la pre) e se volessimo andare in avanti?

$\{p\} y:=t \{?\}$

come postcondizione scrivo  $p$  e il congiunto  $y=t$  e sostituisco  $y$  con  $y'$  sia in  $p$  che in  $t$  e poi elimino  $y'$  ( $y'$ : old  $y$ )

$\{y>5\} y:=2*(y+5) \{?\}$

$\{p\} y:=t \{p[y \rightarrow y'] \wedge t[y \rightarrow y']=y\}$

Q:  $y'>5 \wedge y=2*(y'+5) = y>20$

## 2. Composition rule

regola di composizione

$$\frac{\{p\} S1 \{r\}, \{r\} S2 \{q\}}{\{p\} S1;S2 \{q\}}$$

diamo le regole in questa forma

$$\frac{A1, \dots, An}{A}$$

vuol dire:

(all'indietro) se vuoi dimostrare A dimostra A1, ..An

(in avanti) se hai dimostrato A1 .. An allora A è vera

# diversi modi di usare le regole:

$A_1, \dots, A_n$

$A$

- Voglio dimostrare  $A$ : cosa mi serve?
  - dimostrazione a partire dalla tesi
- So  $A_1, A_n$ : cosa posso dedurre?
  - dimostrazione a partire dalle ipotesi
- In generale:
  - ma se  $A_1, A_n$  sono vere, allora è vero anche  $A$ ?

# Esempio

Per esempio se ho le istruzioni

$x:=x+1; y:=y+2$

$q$  è  $x=y$ , e  $p$  è  $x+1=y+2$

posso applicare la regola, infatti

A1.  $\{x+1=y+2\} x:=x+1 \{x=y+2\}$  per regola  
asseg.

A2.  $\{x=y+2\} y:=y+2 \{x=y\}$  per regola  
assegnamento

la conseguenza è A:

$\{x+1=y+2\} x:=x+1; y:=y+2 \{x=y\}$

# Altro esempio

$$y = 3 * x + 1;$$

$$x = y + 3;$$

$$\{x < 10\}$$

quale è la precondizione?

$$\{x < 2\}$$

$$\{3 * x + 1 + 3 < 10\} \quad y = 3 * x + 1; \quad \{y + 3 < 10\}$$

$$\{y + 3 < 10\} \quad x = y + 3; \quad \{x < 10\}$$

# More examples

$$\frac{\{p\} S1 \{r\}, \{r\} S2 \{q\}}{\{p\} S1;S2 \{q\}}$$

$\{x1 > 0 \wedge x2 > 0\}$   $y1 := x1$

$\{gcd(x1, x2) = gcd(y1, x2) \wedge y1 > 0 \wedge x2 > 0\}$

$\{gcd(y1, x2) = gcd(y1, x2) \wedge y1 > 0 \wedge x2 > 0\}$

$y2 := x2$

$\{gcd(x1, x2) = gcd(y1, y2) \wedge y1 > 0 \wedge y2 > 0\}$

$\{x1 > 0 \wedge x2 > 0\}$   $y1 := x1 ; y2 := x2$

$\{gcd(x1, x2) = gcd(y1, y2) \wedge y1 > 0 \wedge y2 > 0\}$

# Come usare gli assiomi per provare la correttezza

- nel caso voglio provare  $\{P\}S\{Q\}$ , con S assegnamento
  - ho già Q, calcolo P e vedo se è quello specificato
  - ho già P, calcolo Q e controllo che sia quello assegnato
- problemi:
  - se P non è esattamente quello trovato
- esempio: prova

\*\*  $\{x > 0\} x := x + 1 \{x > 0\}$

Se applico gli assiomi trovo

o  $P = \{x > -1\}$  o  $Q = \{x > 1\}$

intuitivamente \*\* è corretto

# 3 Consequence rules

- in questi casi posso applicare queste due regole
- rafforzare una preconditione

$$\frac{r \rightarrow p, \{p\} S \{q\}}{\{r\} S \{q\}}$$

riesco a provare con una preconditione **p** più debole di **r** allora posso rafforzare la preconditione **r**

- indebolire una postcondizione

$$\frac{\{p\} S \{q\}, q \rightarrow r}{\{p\} S \{r\}}$$

riesco a provare con una postcondizione **q** più forte, allora posso indebolire e ottenere **r**

# Esempio

$\{x > 0\} \ x := x + 1 \ \{x > 0\}$

## 4 If-then-else rule

$\{p \wedge e\} S1 \{q\}, \{p \wedge \neg e\} S2 \{q\}$   
 $\{p\}$  if  $e$  then  $S1$  else  $S2$  fi  $\{q\}$

prova:

$\{y > 1\}$  if  $(x > 0)$  then  $y := y - 1;$   
    else  $y := y + 1;$   $\{y > 0\}$

cioè ( $p = y > 1$ ):

$\{p \wedge x > 0\} y := y - 1 \{y > 0\}, \{p \wedge x \leq 0\} y := y - 1 \{y > 0\}$   
[assi.]

$\{y > -1\} y := y - 1 \{y > 0\}, \{y > 1\} y := y - 1 \{y > 0\}$

ok,  $\{p \wedge x > 0\} \rightarrow \{y > -1\}$  e  $\{p \wedge x \leq 0\} \rightarrow \{y > 1\}$

# If-then-else rule

$\{p \wedge e\} S1 \{q\}, \{p \wedge \neg e\} S2 \{q\}$   
 $\{p\}$  if  $e$  then  $S1$  else  $S2$  fi  $\{q\}$

For example:

$p$  is  $\text{gcd}(y1, y2) = \text{gcd}(x1, x2)$

$\wedge y1 > 0 \wedge y2 > 0 \wedge \neg(y1 = y2)$

$e$  is  $y1 > y2$

$S1$  is  $y1 := y1 - y2$

$S2$  is  $y2 := y2 - y1$

$q$  is  $\text{gcd}(y1, y2) = \text{gcd}(x1, x2) \wedge y1 > 0 \wedge y2 > 0$

# While rule

$\{p \wedge e\} S \{p\}$

$\{p\}$  while  $e$  do  $S$  od  $\{p \wedge \neg e\}$

$\overset{p}{\{x \geq 0\}}$  while  $\overset{e}{x \neq 0}$  do  $x := x - 1$  od  $\overset{q}{\{x = 0\}}$

1  $\{p \wedge \neg e\} \rightarrow \{q\} : \{x \geq 0 \wedge \neg x \neq 0\} = \{x = 0\}$  ok

2  $\{p \wedge e\} S \{p\} : \{x \geq 0 \wedge x \neq 0\} x := x - 1 \{x \geq 0\}$   
 $\{x > 0\} x := x - 1 \{x \geq 0\}$

[assegnamento  $x \rightarrow x - 1$ , pre :  $x - 1 \geq 0$ ]

$\{x - 1 \geq 0\} \rightarrow \{x > 0\}$  ok

# invarianti

Cosa succede se però non ho  $p$ , ma ho:

$\{?\}$  while  $e$  do  $S$  od  $\{q\}$

soluzione: devo trovare  $\{p\}$  tale che

$\{p \wedge e\} S \{p\}$

$\{p\}$  : INVARIANTE

trovare un invariante può essere difficile.

Quando si deve dimostrare,

$\{P\}$  while B do S end  $\{Q\}$

cerco  $I$  tale che

1  $\{I \wedge \text{not } B\} \rightarrow \{Q\}$ :

l'invariante all'uscita del ciclo implica la postcondizione

2  $\{I \wedge B\} S \{I\}$   $I$  è effettivamente invariante

3  $P \rightarrow I$  :  $I$  è valido all'inizio del ciclo

posso allora applicare rafforzamento della pre  
(3), indebolire la post(1), e la regola del  
while (2)

# Esempio: divisione intera

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

$a := 0;$

$b := x;$

while  $b \geq y$  do

$b := b - y;$

$a := a + 1$

od.

$\{x = a * y + b \wedge 0 \leq b \wedge b < y\}$

Invariante

$x = a * y + b \wedge b \geq 0$

calcola  $a = x/y$  e  $b$  il resto ( $x \bmod y$ )

# Dimostrazione

divido in due  
la dimostrazione

prendo P del ciclo  
while esattamente  
uguale a I  
(cond 3 su I è OK)

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

1  $a := 0;$   
 $b := x;$

$\{I\}$

while  $b \geq y$  do

2  $b := b - y;$

$a := a + 1$

od.

$\{x = a * y + b \wedge 0 \leq b \wedge b < y\}$

1  $\{x \geq 0 \wedge y \geq 0\} a := 0; b := x; \{x = a * y + b \wedge b \geq 0\}$

# Dimostrazione parte 1

$\{p[y \rightarrow t]\} y := t \{p\}$

$\frac{\{p\} S1 \{r\}, \{r\} S2 \{q\}}{\{p\} S1; S2 \{q\}}$

(1)  $\{x = a*y + x \wedge x >= 0\} b := x \{x = a*y + b \wedge b >= 0\}$   
(Assignment)

(2)  $\{x = 0*y + x \wedge x >= 0\} a := 0 \{x = a*y + x \wedge x >= 0\}$   
(Assignment)

(3)  $\{x = 0*y + x \wedge x >= 0\} a := 0; b := x \{x = a*y + b \wedge x >= 0\}$   
(Composition)

(2), (1))

# Proof (cont.)

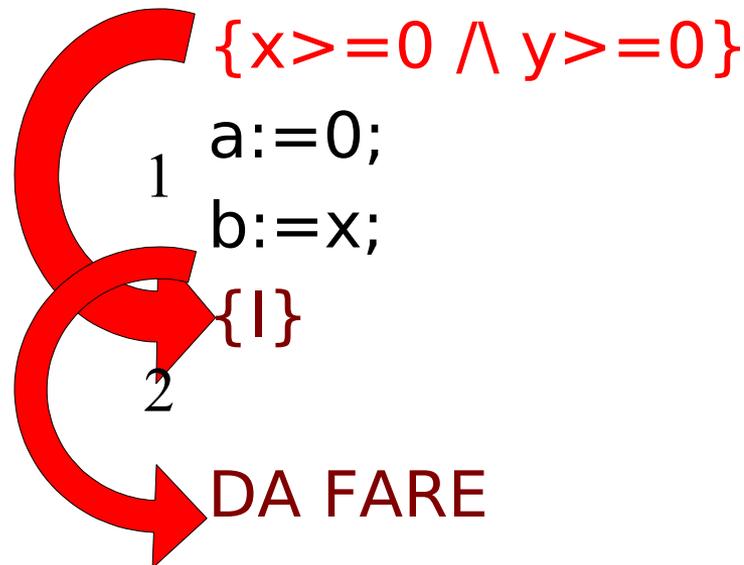
(4)  $x \geq 0 \wedge y \geq 0 \rightarrow x = 0 * y + x \wedge x \geq 0$  (Logic)

(5)  $\{x \geq 0 \wedge y \geq 0\} \quad a := 0; b := x$

$\{x = a * y + b \wedge b \geq 0 \wedge b < y\}$  (Consequence 3 e

4)

ok quindi



# parte 2 : ciclo while

- I ha le proprietà degli invarianti?

1  $\{I \wedge \text{not } B\} \rightarrow \{Q\}$ :

l'invariante all'uscita del ciclo implica la postcondizione

2  $\{I \wedge B\} S \{I\}$  I è effettivamente invariante

3  $P \rightarrow I$  : I è valido all'inizio del ciclo OK

Dimostro proprietà 1

$\{I \wedge \text{not } B\} \rightarrow \{x=a*y+b \wedge 0 \leq b \wedge b < y\}$

$\{x = a*y+b \wedge b \geq 0 \wedge \text{not } b \geq y\} \rightarrow \{x=a*y+b \wedge 0 \leq b \wedge b < y\}$

ok perchè  $\text{not } b \geq y \rightarrow b < y$

manca 2:  $\{I \wedge B\} S \{I\}$

$\{I \wedge B\} b:=b-y; a:=a+1 \{I\}$

# Proof (cont.)

$$\{p[t/y]\} y:=t \{p\} \quad \frac{\{p\} S1 \{r\}, \{r\} S2 \{q\}}{\{p\} S1;S2 \{q\}}$$

$$(6) \{x=(a+1)*y+b \wedge b \geq 0\} a:=a+1 \{x=a*y+b \wedge b \geq 0\}$$

(Assignment)

$$(7) \{x=(a+1)*y+b-y \wedge b-y \geq 0\} b:=b-y \{x=(a+1)*y+b \wedge b \geq 0\}$$

(Assignment)

$$(8) \{x=(a+1)*y+b-y \wedge b-y \geq 0\} b:=b-y; a:=a+1 \{x=a*y+b \wedge b \geq 0\}$$

(Composition (6), (7))

# Proof (cont.)

(9)  $x = a * y + b \wedge b \geq 0 \wedge b \geq y \rightarrow$   
 $x = (a + 1) * y + b - y \wedge b - y \geq 0$  (Logic)

(10)  $\{x = a * y + b \wedge b \geq 0 \wedge b \geq y\}$   
 $b := b - y; a := a + 1 \{x = a * y + b \wedge b \geq 0\}$   
(Consequence (8), (9))

(9)  $\{x = a * y + b \wedge b \geq 0\}$  while  $b \geq y$  do  $b := b - y;$   
 $a := a + 1$  od  $\{x = a * y + b \wedge b \geq 0 \wedge b < y\}$   
(while (10))

# Come “intuire” l'invariante

- L'invariante deve implicare all'uscita la postcondizione
  - Domandati perchè alla fine la post condizione vale?
- Prova a percorrere il ciclo con qualche caso di test, con un “giro” del ciclo, con due e così via
-

# Esercizi

```
{y >= 0}
i = 0;
{I}
quad = 0;
while ( i != y) {
    quad = quad + y;
    i = i + 1;
}
{quad = y^2}
```

```
I = q = i * y
```

anche in questo caso divido  
la dimostrazione in due parti

nella seconda parte devo dimostrare che  
1) I and not B  $\rightarrow$  Q  
cioè  $q = i * y$  AND not  $(i != y)$   
 $q = i * y$  AND  $i = y \rightarrow q = y * y$

OK

2)  $\{I \text{ and } B\} S \{I\}$   
 $\{q = i * y \text{ and } i != y\} q = q + y; i = i + 1 \{...\}$

da fare

questo I è assegnato

# Altri esempi

- Esempio 1:somma

```
{n>0}
count = 0;
sum = 0;
while count < n do
    count = count + 1;
    sum = sum + count;
end
{sum = 1 +2 + ... + n}
```

count cresce fino a diventare  
n, e sum accumula la  
somma da 1 a count:

I: sum = 1 + ... + count

- Esempio 2: assegna

```
{x>=0}
Y:= 0;
while (y < x) {
    y:= y+1;
}
{y=x}
```

## Esempio 3

```
int old_x = x;
int dop = x;
while (x != 0) {
    dop := dop +1 ; x := x-1;
}
{dop = 2 *old_x}
```

## gcd -skip

while  $\neg(y1=y2)$  do

  if  $y1 > y2$  then  $y1 := y1 - y2$

    else  $y2 := y2 - y1$  fi

**p** is  $\{gcd(y1, y2) = gcd(x1, x2) \wedge y1 > 0 \wedge y2 > 0\}$

**e** is  $(y1 = y2)$

**S** is if  $y1 > y2$  then  $y1 := y1 - y2$  else  $y2 := y2 - y1$   
fi

# Combining program

$\{x1 > 0 \wedge x2 > 0\}$

$y1 := x1; y2 := x1;$

$\{gcd(x1, x2) = gcd(y1, y2) \wedge y1 > 0 \wedge y2 > 0\}$

  while S do

    if e then S1 else S2 fi od

$\{gcd(x1, x2) = gcd(y1, y2) \wedge y1 > 0 \wedge y2 > 0\}$

Combine the above using **concatenation rule!**

# Use of first consequence rule

Want to prove

$\{x1 > 0 \wedge x2 > 0\} \text{ } y1 := x1$

$\{gcd(x1, x2) = gcd(y1, x2) \wedge y1 > 0 \wedge x2 > 0\}$

By assignment rule:

$\{gcd(x1, x2) = gcd(x1, x2) \wedge x1 > 0 \wedge x2 > 0\}$

$y1 := x1$

$\{gcd(x1, x2) = gcd(y1, x2) \wedge y1 > 0 \wedge x2 > 0\}$

$x1 > 0 \wedge x2 > 0 \rightarrow$

$gcd(x1, x2) = gcd(x1, x2) \wedge x1 > 0 \wedge x2 > 0$

# Not completely finished

$\{x1 > 0 \wedge x2 > 0\}$

$y1 := x1; y2 := x1;$

  while  $\sim(y1 = y2)$  do

    if  $e$  then  $S1$  else  $S2$  fi od

$\{gcd(x1, x2) = gcd(y1, y2) \wedge y1 > 0 \wedge y2 > 0 \wedge y1 = y2\}$

But we wanted to prove:

$\{x1 > 0 \wedge x1 > 0\}$  Prog  $\{y1 = gcd(x1, x2)\}$

# Use of second consequence rule

$\{x1 > 0 \wedge x2 > 0\}$  Prog

$\{gcd(x1, x2) = gcd(y1, y2) \wedge y1 > 0 \wedge y2 > 0 \wedge y1 = y2\}$

And the implication

$\{gcd(x1, x2) = gcd(y1, y2) \wedge y1 > 0 \wedge y2 > 0 \wedge y1 = y2\}$

$\rightarrow y1 = gcd(x1, x2)$

Thus,

$\{x1 > 0 \wedge x2 > 0\}$  Prog  $\{y1 = gcd(x1, x2)\}$

# Annotating a while program

## fine skip

```
{x1>0∧x2>0}  
y1:=x1;  
{gcd(x1,x2)=gcd(y1,x2  
  )  
  ∧y1>0∧x2>0}  
y2:=x2;  
{gcd(x1,x2)=gcd(y1,y2  
  )  
  ∧y1>0∧y2>0}
```

```
while ¬(y1=y2) do  
  {gcd(x1,x2)=gcd(y1,y2)∧  
    y1>0∧y2>0∧¬(y1=y2)  
  }  
  if y1>y2 then y1:=y1-y2  
    else y2:=y2-y1 fi  
od  
{y1=gcd(x1,x2)}
```

# Soundness - Consistenza

la logica di Hoare è consistente (**sound**) nel senso che ogni cosa che può essere provata è corretta (cioè è vera)

Dim.: ogni assioma conserva la verità delle asserzioni (consistente).

# Completezza

Un sistema di dimostrazioni è detto completo se ogni asserzioni vera può essere dimostrata

- la logica proposizionale è completa
- Per l'aritmetica invece non c'è sistema di regole che sia completo (Godel).
- puoi leggere “Gödel, Escher, Bach: An Eternal Golden Braid”, in italiano: Godel, Escher, Bach un'Eterna Ghirlanda Brillante di Douglas R. Hofstadter  
[http://en.wikipedia.org/wiki/G%C3%B6del,\\_Escher,\\_Bach](http://en.wikipedia.org/wiki/G%C3%B6del,_Escher,_Bach)

# E la logica di Hoare?

Due problemi:

1) incompletezza del calcolo aritmetico

2) problema della terminazione:

$\{p\} S \{false\}$  significa che  $S$  non termina a partire dalle precondizioni  $p$ . Questo è indecidibile, cioè le regole non riescono a provarlo sempre:

il sistema però è relativamente completo