# Métodos Formais em Engenharia de Software

**Ana Paiva**

**apaiva@fe.up.pt**

Universidade do Porto
Faculdade de Engenharia
FEUP



# Exercises

1 - Assumindo que x e y são inteiros, determine a pré-condição mais fraca dos triplos que se seguem.

a) {P} x := z+1; y := x+y; {y > 5}
b) {P} y := x-y; x := x-y; y := y+x {x=A ∧ y=B}
c) {P} if (x>y) y:=x; {y=max(x,y)}
d) {P} if (x>=10) then x := x/2; else x := x+5; {x>=5 ∧ x<=15}

2 – Prove que o triplo seguinte é verdadeiro.

a) {True} b:=2; a:=5+b; b:=a+2 {b>a}



a. {P} x := z+1; y := x+y; {y > 5}

b. {P} y := x-y; x := x-y; y := y+x {x=Y ∧ y=X}

c. {P} if (x>y) y:=x; {y=max(x,y)}

d. {P} if (x>=10) then x := x/2; else x := x+5; {x>=5 ∧ x<=15}



a. {True} b:=2; a:=5+b; b:=a+2 {b>a}

# Verification of cycles (wp – rule R5')

**We want to prove that :**

***wp*( while n ≠ 0 do n:=n-1, *n* = 0) ≡ (*n* ≥ 0)**

**Step 1** - find *Pk* :

*P*0 ≡ ¬ (*n* ≠ 0) /\ (*n* = 0) ≡ (*n* = 0)
*P*1 ≡ (*n* ≠ 0) /\ *wp*( n:=n-1, (*n* = 0)) ≡ (*n* = 1)
*P*2 ≡ (*n* ≠ 0) /\ *wp*( n:=n-1, (*n* = 1)) ≡ (*n* = 2)
. . . . .
*Pk* ≡ (*n* = *k*)

**Step 2** - find weakest precondition :

∃ *k*. (*k* ≥ 0) /\ *Pk* ≡ (*n* = 0) \/ (*n* = 1) \/ (*n* = 2) \/ ...

**Then**, *wp*( while n ≠ 0 do n:=n-1, *n* = 0) ≡ (*n* ≥ 0)



# Verification of cycles (inference - rule R5)

To prove

{P} while C do S {Q}
find the invarant I and variant v, such that
P ⇒ I - the invariant is initially true (R6)
I ∧ ¬C ⇒ Q - the invariant is sufficient (R7)
I ∧ C ⇒ v∈N - variant function takes a non-negative integer (R5)
{I ∧ C ∧ v=V} S {I ∧ v<V} - the invariant is maintained and strictly decreasing function variation(R5)
(by rules 5, 6 and 7)

Schematically :

{P} {I}
while C do
{I ∧ C} {v ∈N}
{I ∧ C ∧ v=V} S {I ∧ v<V}
end
{I ∧¬C} {Q}



# Verification of cycles (inference - rule R5)

```
class LinearSearch {
  // pre A != null && (∃ int i; 0 <= i && i < A.length; A[i] == r);
  // post A[result]==r;

      int linearsearch(int A[], int r) {
      int i = 0;
      while (A[i] != r) i = i + 1;
          return i;
      }
}
```

Cycle invariant :

Variant function: