# Lógica de Floyd-Hoare

## Cálculo de Floyd-Hoare (continuação)

LAÇOS ENQUANTO

```
   (|I∧B|) C (|I|)
___________________________ EnquantoParcial
(|I|) while B {C} (|I∧¬B|)
```

Na regra apresentada `I` deve ser uma invariante de laço. Uma invariante de laço é uma condição que é verdadeira antes e depois de cada iteração do laço, assim como ao final dele, ou seja, sempre é verdadeira. Do ponto de vista da metodologia de programação, a invariante de laço pode ser vista como uma especificação mais abstrata do laço, o qual caracteriza o propósito mais profundo do laço além dos detalhes da implementação.

Exemplo: cálculo do fatorial de x

```
 (| ? |)
  y := 1;
  z := 0;
  while (z != x) {
    z := z + 1;
    y := y * z;
  }
 (| y = x! |)
```

Observe que queremos calcular a condição mais fraca $\phi$ tal que:

(|$\phi$|) while B {C} (|$\varphi$|)

Esta $\phi$ pode ser calculada da seguinte maneira:

a) Descobrir uma fórmula $Inv$ que seja uma invariante de laço.

b) Demonstrar:

$\phi \Rightarrow Inv$, ou seja, o código antes do laço garante a invariante como verdadeira;

$Inv \wedge \neg B \Rightarrow \varphi$, ou seja, ao final do laço a invariante é verdadeira e é utilizada para demonstrar a pós-condição com base no uso da regra da consequência para enfraquecimento da pós-condição.

c) Empurre $Inv$ para cima através de `C`; vamos chamar o resultado de $Inv'$.

d) Demonstrar:

$Inv \wedge B \Rightarrow Inv'$, ou seja, realmente temos uma invariante de laço.

e) Escreva $Inv$ acima do laço de enquanto e escreva $\phi$ acima de $Inv$ denotando a justificativa como uso da regra da consequência de fortalecimento da pré-condição baseado na demonstração no passo b.

<u>Exemplo completo com o uso de Atribuição + Consequência + Enquanto</u>:

Prove:

```
(| T |)
  y := 1;
  z := 0;
  while (z != x) {
    z := z + 1;
    y := y * z;
  }
(| y = x! |)
```

| | | |
|---|---|---|
| | (\|T\|) | |
| | (\|1=0!\|) | **PreForte** |
| | y:=1; | |
| | (\|y=0!\|) | **Atribuição** |
| | z:=0; | |
| | (\|y=z!\|) | **Atribuição** |
| | while (z != x) { | |
| | (\|y=z! ∧ z≠x\|) | **Invariante** |
| | (\|y*(z+1)=(z+1)!\|) | **Implicação** |
| | z:=z+1; | |
| | (\|y*z=z!\|) | **Atribuição** |
| | y:=y*z; | |
| | (\|y=z!\|) | **Atribuição** |
| | } | |
| | (\|y=z! ∧ ¬(z≠x)\|) | **EnquantoParcial** |
| | (\|y=x!\|) | **PosFraco** |

Procure descrever informalmente porquê as implicações indicadas são válidas.

<u>Exercícios:</u>

1) Prove:

(| $a > 0 \wedge b \geq 0$ |)

```
i:=0;
p:=1;
while (i<b) {
  p:=p*a;
  i:=i+1;
}
```

(| $p = a^b$ |)

2) Prove:

(| $x \geq 0$ |)

```
a:=x;
y:=0;
while (a!=0) {
  y:=y+1;
  a:=a-1;
}
```

(| $x = y$ |)

## ARRAYS

Vamos estender a linguagem de programação básica para incluir leitura e escrita de arrays de inteiros.
Podemos usar um array como `a[i]` para ler uma posição indicada pelo índice inteiro `i` entre 0 e tamanho-1.
Também podemos realizar atribuições a posições válidas do array segundo as seguintes regras:

$$\frac{}{(\!|\ Q[a/a\{E\leftarrow t\}]\ |\!)\ a[E]:=t\ (\!|Q|\!)}\ \text{ArrayAtribuição}$$

$$\frac{E = E'}{a\{E\leftarrow t\}[E']=t}\ \text{ArrayIndice=}$$

$$\frac{E \neq E'}{a\{E\leftarrow t\}[E']=a[E']}\ \text{ArrayIndice}\neq$$

A expressão `a{E←t}` representa uma atualização do array original `a`. Este novo array no índice representado pela expressão `E'`, `a{E←t}[E']` tem o valor `t` se o valor de `E` for igual ao valor de `E'`. Do contrário, se `E ≠ E'`, então o valor do array na posição `E'` é o valor original de `a`, isto é, o valor representado pela expressão `a[E']`.

Exemplo:

```
(| ? |) a[i]:=4 (| a[j]=4 |)
(| a{i←4}[j]=4 |) a[i]:=4 (| a[j]=4 |)
```

Exemplo completo com o uso de ArrayAtribuição:

Prove: `(| (i=j) ∧ (a[i]=3) |) a[i]:=4 (| a[j]=4 |)`

| | | |
|---|---|---|
| | (\| (i=j)∧(a[i]=3) \|) | |
| | (\| a{i←4}[j]=4 \|) | **Implicação e ArrayIndice=** |
| | a[i]:=4; | |
| | (\| a[j]=4 \|) | **ArrayAtribuição** |

Procure descrever informalmente porquê a implicação indicada é válida.

<u>Exercícios:</u>

1) Prove: `(|(i=j+1)∧(a[j]=39)|) a[i-1]:=24 (|a[j]=24|)`

2) Prove:

```
(|(i=j-1)∧(a[i]=25)∧(a[j]=12)|)
a[i+1]:=25;
a[j-1]:=12;
(|(a[i]=12)∧(a[j]=25)|)
```