

# Ficha 1

## Algoritmos e Complexidade

### Análise de correcção de programas

## 1 Especificação de Programas e Triplos de Hoare

1. Descreva por palavras as seguintes especificações:

- (a)  $\left\{ \begin{array}{ll} \text{pré-condição:} & x == x_0 \geq 0 \wedge y == y_0 > 0 \\ \text{pós-condição:} & 0 \leq r < y_0 \wedge (\exists_{q \geq 0}. q * y_0 + r == x_0) \end{array} \right.$   
( $x, y, q, r$  variáveis de tipo inteiro)
- (b)  $\left\{ \begin{array}{ll} \text{pré-condição:} & x == x_0 \wedge y == y_0 \\ \text{pós-condição:} & (x == x_0 \vee x == y_0) \wedge x \geq x_0 \wedge x \geq y_0 \end{array} \right.$
- (c)  $\left\{ \begin{array}{ll} \text{pré-condição:} & x == x_0 \geq 0 \wedge e == e_0 > 0 \\ \text{pós-condição:} & |r * r - x_0| < e_0 \end{array} \right.$   
( $x, e, r$  variáveis de vírgula flutuante)
- (d)  $\left\{ \begin{array}{ll} \text{pré-condição:} & \forall_{0 \leq i < N}. A[i] == a_i \\ \text{pós-condição:} & 0 \leq p < N \wedge (\forall_{0 \leq i < N}. A[i] == a_i \wedge A[p] \leq a_i) \end{array} \right.$   
( $A$  array de tipo numérico;  $N$  constante de tipo inteiro,  $p$  variável de tipo inteiro)

2. Escreva especificações (pré- e pós-condições) para os seguintes problemas:

- (a) Um programa que, coloca na variável  $r$  a soma dos valores (iniciais) das variáveis  $x$  e  $y$ .
- (b) Um programa que, para valores não negativos da variável  $y$ , coloca na variável  $r$  o produto dos valores (iniciais) das variáveis  $x$  e  $y$ , sem alterar os valores dessas variáveis.
- (c) Um programa que coloca na variável  $r$  o mínimo múltiplo comum das variáveis  $x$  e  $y$ .
- (d) Um programa que recebe dois arrays  $A$  e  $B$  com  $N$  elementos como parâmetros, e calcula o comprimento do prefixo mais longo que os dois têm em comum.

3. Pronuncie-se sobre a validade dos seguintes triplos de Hoare. Corrija os triplos inválidos, modificando apenas a *pós-condição* (a pós-condição deverá sempre ser tão informativa quanto possível).

- (a)  $\{j == a\} \ j = j + 1 \ \{a == j + 1\}$
- (b)  $\{i == j\} \ i = j + i \ \{i > j\}$
- (c)  $\{i == i_0\} \ j = i + 1; \ i = j + 1 \ \{i == i_0 + 1\}$
- (d)  $\{i \neq j\} \ \text{if } (i > j) \ m = i - j \ \text{else } m = j - i \ \{m > 0\}$
- (e)  $\{x == b\} \ \text{while } (x > a) \ x = x - 1 \ \{b == a\}$

## 2 Prova de correcção de programas sem ciclos

1. Prove cada um dos seguintes triplos de Hoare, começando por gerar as respectivas condições de verificação.

- (a)  $\{i > j\} \ j = i + 1; \ i = j + 1 \ \{i > j\}$
- (b)  $\{s == 2^i\} \ i = i + 1; \ s = s * 2 \ \{s == 2^i\}$
- (c)  $\{\text{True}\} \ \text{if } (i < j) \ \text{min} = i \ \text{else } \text{min} = j \ \{\text{min} \leq i \wedge \text{min} \leq j\}$
- (d)  $\{i \neq j\} \ \text{if } (i > j) \ m = i - j \ \text{else } m = j - i \ \{m > 0\}$

2. Apresente condições de verificação para os seguintes programas anotados com pré-condições, pós-condições, e asserções intermédias.

- (a) 

```
// i > j
    j = i+1;
    // j > i
    i = j+1
    // i > j
```
- (b) 

```
// True
    i = j+1;
    // i != j
    if (i>j) m = i-j
    else m = j-i
    // m > 0
```

### 3 Prova de correcção parcial de programas com ciclos

1. Apresente as condições de verificação necessárias à prova de correcção parcial do seguinte programas anotado com pré-condições, pós-condições, e invariantes de ciclo.

```
// x == x0 >= 0  &&  y == y0 > 0
  r = x;
  q = 0;
  while (y <= r) {
    // r >= 0  &&  y == y0  &&  q*y0 + r == x0
    r = r-y;
    q = q+1;
  }
// 0 <= r < y0  &&  q*y0 + r == x0
```

2. A sequência de Fibonacci  $\{F_n\}_{n \geq 0}$  define-se como:

$$F_i = \begin{cases} i & \text{se } i < 2 \\ F_{i-1} + F_{i-2} & \text{se } i \geq 2 \end{cases}$$

Considere o seguinte programa que, dado um número inteiro positivo  $n$ , calcula o  $n$ -ésimo número de Fibonacci.

```
i = 1; r = 1; s = 0;
while (i < n) {
  r = r+s;
  s = r-s;
  i = i+1;
}
```

Prove que este programa está correcto face à especificação

$$\begin{cases} \text{pré-condição:} & n == n_0 > 0 \\ \text{pós-condição:} & r == F_{n_0} \end{cases}$$

Para isso,

- comece por anotar convenientemente os programas,
- gere as condições de verificação correspondentes, e
- mostre a validade dessas condições.

Use como invariante o predicado

$$I \doteq (n == n_0) \wedge (i \leq n) \wedge (r == F_i) \wedge (s == F_{i-1})$$

## 4 Determinação de invariantes de ciclo

1. Considere a seguinte especificação de um programa que calcula o somatório dos elementos de um vector.

$$\begin{cases} \text{pré-condição:} & \forall_{0 \leq i < N}. A[i] = a_i \\ \text{pós-condição:} & s = \sum_{i=0}^{N-1} a_i \end{cases}$$

Encontre invariantes de ciclo que lhe permitam provar a correcção parcial dos seguintes programas (face a essa especificação).

(a) 

```
s = 0; p = 0;
while (p < N) {
    s = s + A[p]; p = p+1;
}
```

(b) 

```
s = 0; p = N;
while (p > 0) {
    p = p-1; s = s + A[p];
}
```

2. Considere a seguinte especificação de um programa que calcula o produto de dois números inteiros.

$$\begin{cases} \text{pré-condição:} & x = x_0 \wedge y = y_0 \geq 0 \\ \text{pós-condição:} & r = x_0 * y_0 \end{cases}$$

Encontre invariantes de ciclo que lhe permitam provar a correcção parcial dos seguintes programas (face a essa especificação):

(a) 

```
r = 0; i = 0;
while (i < y) {
    r = r + x; i = i+1
}
```

(b) 

```
r = 0;
while (y > 0) {
    r = r + x; y = y-1
}
```

(c) 

```
r = 0;
while (y > 0) {
    if (y % 2 != 0) {
        y = y - 1; r = r + x
    }
    x = x*2; y = y/2
}
```

## 5 Correção total

1. Para cada um dos programas apresentados na secção anterior, determine um variante de ciclo que lhe permita provar a correção total face às especificações apresentadas.
2. Considere os seguintes programas que determinam se um vector tem elementos repetidos. Anote-os convenientemente e, a partir do programa anotado, determine as condições de verificação necessárias à prova da sua correção total.

$$\begin{cases} \text{pré-condição:} & \forall_{0 \leq i < N; A[i] = a_i \\ \text{pós-condição:} & r \Leftrightarrow \exists_{0 \leq i, j < N} a_i = a_j \end{cases}$$

(a)

```
r = False; i=0;
while ((i<N-1) && !r) {
    j=i+1;
    while ((j<N) && !r) {
        if (a[i] == a[j]) r = True;
        j = j+1;
    }
    i = i+1;
}
```

(b)

```
r = False; i=0; j = 1;
while ((i<N-1) && !r) {
    if (a[i] == a[j]) r = True;
    j = j+1;
    if (j == N) { i = i+1; j = i+1;}
}
```

## A Lógica de Hoare

### A.1 Correção parcial

#### Consequência

$$\frac{P \Rightarrow P' \quad \{P'\} S \{Q'\} \quad Q' \Rightarrow Q}{\{P\} S \{Q\}} \quad (\Rightarrow)$$

#### Skip

$$\frac{}{\{P\} \{\} \{P\}} \quad (\text{skip})$$

#### Atribuição

$$\frac{}{\{Q[x \setminus E]\} x = E \{Q\}} \quad (=)$$

#### Sequência

$$\frac{\{P\} S_1 \{R\} \quad \{R\} S_2 \{Q\}}{\{P\} S_1 ; S_2 \{Q\}} \quad (;)$$

#### Condicional

$$\frac{\{P \wedge c\} S_1 \{Q\} \quad \{P \wedge \neg c\} S_2 \{Q\}}{\{P\} \text{if } (c) S_1 \text{ else } S_2 \{Q\}} \quad (;)$$

#### Ciclo

$$\frac{\{I \wedge c\} S \{I\}}{\{I\} \text{while } (c) S \{I \wedge \neg c\}} \quad (\text{while})$$

## A.2 Correccão total

### Consequência

$$\frac{P \Rightarrow P' \quad [P'] S [Q'] \quad Q' \Rightarrow Q}{[P] S [Q]} \quad ([\Rightarrow])$$

### Skip

$$\frac{}{[P] \{\} [P]} \quad ([\text{skip}])$$

### Atribuição

$$\frac{}{[Q[x \setminus E]] \ x = E \ [Q]} \quad ([=])$$

### Sequência

$$\frac{[P] S_1 [R] \quad [R] S_2 [Q]}{[P] S_1 ; S_2 [Q]} \quad ([;])$$

### Condicional

$$\frac{[P \wedge c] S_1 [Q] \quad [P \wedge \neg c] S_2 [Q]}{[P] \text{ if } (c) S_1 \text{ else } S_2 [Q]} \quad ([\text{if}])$$

### Ciclo

$$\frac{I \wedge c \Rightarrow V \geq 0 \quad [I \wedge c \wedge (V = v_0)] S [I \wedge (V < v_0)]}{[I] \text{ while } (c) S [I \wedge \neg c]} \quad ([\text{while}])$$

## B Exercícios Adicionais

1. Escreva uma especificação para um programa que recebe dois arrays  $A$  e  $B$  como parâmetros, e verifica se eles têm um elemento em comum.
2. Sejam  $P$ ,  $Q$  dois predicados e  $S$  um programa (arbitrários). O que significa a validade dos seguintes triplos (tendo em conta que a notação  $[ ]$  representa correcção total):

- (a)  $\{P\} S \{\text{true}\}$
- (b)  $[P] S [\text{true}]$
- (c)  $\{P\} S \{\text{false}\}$
- (d)  $[P] S [\text{false}]$
- (e)  $\{\text{false}\} S \{Q\}$
- (f)  $[\text{false}] S [Q]$
- (g)  $\{\text{true}\} S \{Q\}$
- (h)  $[\text{true}] S [Q]$

3. Pronuncie-se sobre a validade dos seguintes triplos de Hoare:

- (a)  $\{j == a\} j = j + 1 \{a == j + 1\}$
- (b)  $\{i == j\} i = j + i \{i > j\}$
- (c)  $\{j == a + b\} i = b; j = a \{j == 2 * a\}$
- (d)  $\{i == 3 * j\} \text{ if } (i > j) \text{ m} = i - j \text{ else m} = j - i \{m - 2 * j == 0\}$

4. Apresente regras derivadas para a lógica de Hoare, cujas conclusões sejam:

- (a)  $\{P\} x = E \{Q\}$
- (b)  $\{P\} \text{ while } (c) S \{Q\}$
- (c)  $\{P\} x = E_1 ; \text{ while } (c) \{S ; x = E_2\} \{Q\}$ .

Note que esta última construção corresponde ao comando **for** do C.

- (d)  $\{P\} \text{ if } (c) S \{Q\}$

5. Mostre que, para predicados arbitrários  $P$  e  $Q$ , é válido o triplo

$$\{P\} \text{ while } (\text{true}) \{\} \{Q\}$$

6. Prove cada um dos seguintes triplos de Hoare, começando por gerar as condições de verificação necessárias.

- (a)  $\{a > b\} m = 1; n = a - b \{m * n > 0\}$
- (b)  $\{i > 0 \wedge j > 0\} \text{ if } (i < j) \text{ min} = i \text{ else min} = j \{\text{min} > 0\}$



7. Apresente as condições de verificação necessárias à prova de correcção parcial dos seguintes programas anotados:

```
(a) // exists (i in [0..n-1]) v[i] == x
    k = 0;
    while (v[k] != x)
        // exists (i in [k..n-1]) v[i] == x
        k=k+1
    // v[k] == x

(b) // True
    v = 0;
    i = 0;
    while (i<=N) {
        // v == sum (k in [0..i-1]) b[k] * 2^(i-1-k) && i <= N+1
        v = v*2 + b[i];
        i = i+1
    }
    // v == sum (k in [0..N]) b[k] * 2^(N-k)
```

8. O algoritmo de Euclides para o cálculo do máximo divisor comum (mdc) entre dois inteiros positivos baseia-se em duas propriedades fundamentais:

- $\forall_x. mdc(x, x) = x$
- $\forall_{x,y}. mdc(x, y) = mdc(x + y, y) = mdc(x, x + y)$

Use estas propriedades para provar a correcção do seguinte programa para calcular o mdc de dois inteiros positivos. Comece por anotar convenientemente o programa, gere as condições de verificação correspondentes, e mostre a validade dessas condições.

```
while (a != b)
    if (a > b) a = a-b;
    else b = b-a;
```

$\left\{ \begin{array}{ll} \text{pré-condição:} & a == a_0 > 0 \wedge b == b_0 > 0 \\ \text{pós-condição:} & a == mdc(a_0, b_0) \end{array} \right.$

Use como invariante o predicado

$$I \doteq mdc(a, b) == mdc(a_0, b_0)$$

9. Considere a seguinte especificação de um programa que calcula o quadrado de um número inteiro.

$$\begin{cases} \text{pré-condição:} & x = x_0 \geq 0 \\ \text{pós-condição:} & r = x_0^2 \end{cases}$$

Encontre invariantes de ciclo que lhe permitam provar a correcção parcial dos seguintes programas (face a essa especificação):

- (a) 

```
r = 0; i = 0;
while (i < x) {
    i = i+1; r = r+x;
}
```
- (b) 

```
r = 0; i = 0; p = 1;
while (i < x) {
    i = i+1; r = r+p; p = p+2;
}
```

10. Considere a seguinte especificação de um programa que calcula o factorial de um número inteiro não negativo.

$$\begin{cases} \text{pré-condição:} & x = x_0 \geq 0 \\ \text{pós-condição:} & f = x_0! = \prod_{i=1}^{x_0} i \end{cases}$$

Encontre invariantes de ciclo que lhe permitam provar a correcção parcial dos seguintes programas (face a essa especificação):

- (a) 

```
f = 1; i = 0;
while (i < x) {
    i = i+1;
    f = f * i;
}
```
- (b) 

```
f = 1;
while (x > 0) {
    f = f * x;
    x = x-1
}
```

11. Considere o seguinte programa para cálculo de uma aproximação à raiz quadrada de um número (float) não negativo.

$$\left\{ \begin{array}{ll} \text{pré-condição:} & (x = x_0 \geq 0) \wedge (e = e_0 > 0) \\ \text{pós-condição:} & |r - \sqrt{x_0}| < e_0 \end{array} \right.$$

```
a = 0; b = x; r = x/2;
while ((b-a) >= e) {
    if (r*r > x) b = r;
    else a = r;
    r = (a+b)/2;
}
```

Determine um variante que, juntamente com o invariante

$$I \doteq (a \leq r \leq b) \wedge (a \leq \sqrt{x_0} \leq b) \wedge (e = e_0)$$

lhe permita provar a correcção total deste programa.