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) $\begin{cases} \textbf{pré-condição:} & x == x_0 \geq 0 \wedge y == y_0 > 0 \\ \textbf{pós-condição:} & 0 \leq r < y_0 \wedge (\exists_{q \geq 0}.\, q * y_0 + r == x_0) \end{cases}$

   (x, y, q, r variáveis de tipo inteiro)

   (b) $\begin{cases} \textbf{pré-condição:} & x == x_0 \wedge y == y_0 \\ \textbf{pós-condição:} & (x == x_0 \vee x == y_0) \wedge x \geq x_0 \wedge x \geq y_0 \end{cases}$

   (c) $\begin{cases} \textbf{pré-condição:} & x == x_0 \geq 0 \wedge e == e_0 > 0 \\ \textbf{pós-condição:} & |r * r - x_0| < e_0 \end{cases}$

   (x, e, r variáveis de vírgula flutuante)

   (d) $\begin{cases} \textbf{pré-condição:} & \forall_{0 \leq i < N}.\, A[i] == a_i \\ \textbf{pós-condição:} & 0 \leq p < N \wedge \forall_{0 \leq i < N}.\, (A[i] == a_i \wedge A[p] \leq a_i) \end{cases}$

   (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) $\{\mathtt{j} == \mathtt{a}\}\ \mathtt{j} = \mathtt{j} + \mathtt{1}\ \{\mathtt{a} == \mathtt{j} + 1\}$

   (b) $\{\mathtt{i} == \mathtt{j}\}\ \mathtt{i} = \mathtt{j} + \mathtt{i}\ \{\mathtt{i} > \mathtt{j}\}$

   (c) $\{\mathtt{i} == \mathtt{i}_0\}\ \mathtt{j} = \mathtt{i} + \mathtt{1};\ \mathtt{i} = \mathtt{j} + \mathtt{1}\ \{\mathtt{i} == \mathtt{i}_0 + 1\}$

   (d) $\{\mathtt{i}\ != \ \mathtt{j}\}$ if $(\mathtt{i} > \mathtt{j})\ \mathtt{m} = \mathtt{i} - \mathtt{j}$ else $\mathtt{m} = \mathtt{j} - \mathtt{i}\ \{\mathtt{m} > 0\}$

   (e) $\{\mathtt{x} == \mathtt{b}\}$ while $(\mathtt{x} > \mathtt{a})\ \mathtt{x} = \mathtt{x} - \mathtt{1}\ \ \{\mathtt{b} == \mathtt{a}\}$

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

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

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

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.

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

   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.

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

   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
}
```

3. Considere o seguinte programa que determina se um vector tem elementos repetidos. Encontre invariantes de ciclo que lhe permitam provar a sua correcção parcial:

$$\begin{cases} \textbf{pré-condição:} & \forall_{0 \leq K < N} \cdot A[k] == a_k \\ \textbf{pós-condição:} & r \Leftrightarrow \exists_{0 \leq k,l < N} \cdot a_k == a_l \end{cases}$$

```
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;
}
```

5 Correcção total

1. Para cada um dos programas apresentados na secção anterior, determine um variante de ciclo que lhe permita provar a correcção total face às especificações apresentadas.

2. Considere o seguinte programa alternativo para determinar se um vector tem elementos repetidos. Anote-o convenientemente e, a partir do programa anotado, determine as condições de verificação necessárias à prova da sua correcção total.

$$\begin{cases} \textbf{pré-condição:} & \forall_{0 \leq K < N} .\, A[k] == a_k \\ \textbf{pós-condição:} & r \Leftrightarrow \exists_{0 \leq k,l < N} .\, a_k == a_l \end{cases}$$

```
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 Correcção parcial

Consequência

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

Skip

$$\frac{}{\{\, P \,\}\; \{\} \;\{\, P \,\}} \;(\text{skip})$$

Atribuição

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

Sequência

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

Condicional

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

Ciclo

$$\frac{\{\, I \wedge c \,\}\; S \;\{\, I \,\}}{\{\, I \,\}\; \mathsf{while}\ (c)\ S \;\; \{\, I \wedge \neg c \,\}} \;(\text{while})$$

A.2 Correcção 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

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

Ciclo

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

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

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

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

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

   (a) $\{\,P\,\}\ x = E\ \{\,Q\,\}$
   (b) $\{\,P\,\}\ \mathsf{while}\ (c)\ S\ \ \{\,Q\,\}$
   (c) $\{\,P\,\}\ x = E_1\ ;\ \mathsf{while}\ (c)\ \{S\ ;\ x = E_2\}\ \ \{\,Q\,\}$.
   Note que esta última construção corresponde ao comando `for` do C.
   (d) $\{\,P\,\}\ \mathsf{if}\ (c)\ S\ \{\,Q\,\}$

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

$$\{\,P\,\}\ \mathsf{while}\ (\mathsf{true})\ \{\}\ \ \{\,Q\,\}$$

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

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

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;
   ```

   $$\begin{cases} \textbf{pré-condição:} & a == a_0 > 0 \wedge b == b_0 > 0 \\ \textbf{pós-condição:} & a == mdc(a_0, b_0) \end{cases}$$

   Use como invariante o predicado

   $$I \doteq mdc(a, b) == mdc(a0, b_0)$$

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

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

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.

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

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 negtivo.

$$\begin{cases} \textbf{pré-condição:} & (x = x_0 \geq 0) \wedge (e = e_0 > 0) \\ \textbf{pós-condição:} & |r - \sqrt{x_0}| < e_0 \end{cases}$$

```
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.