Ficha 1

Algoritmos e Complexidade

Análise de correcção de programas

1 Especificação de Programas

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

   (a) Um programa que, coloca na variábel `r` a soma dos valores (iniciais) das variáveis `x` e `y`.

   (b) Um programa que, para valores não negativos da variável `y`, coloca na variábel `r` o produto dos valores (iniciais) das variáveis `x` e `y`.

   (c) 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.

   (d) Um programa que coloca na variável `r` o mínimo múltiplo comum das variáveis `X` e `Y`.

   (e) Um programa que recebe dois arrays `A` e `B` como parâmetros, e verifica se eles têm um elemento em comum.

   (f) Um programa que recebe dois arrays `A` e `B` como parâmetros, e calcula o comprimento do prefixo mais longo que os dois têm em comum.

2. Descreva por palavras (i.e., faça o exercício inverso da alínea anterior) as seguintes especificações:

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

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

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

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

2 Correcção

1. 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) $\{\mathtt{i} > \mathtt{j}\}$ $\mathtt{j} = \mathtt{i} + \mathtt{1};\ \mathtt{i} = \mathtt{j} + \mathtt{1}$ $\{\mathtt{i} > \mathtt{j}\}$

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

   (f) $\{\mathtt{i} = 3 * \mathtt{j}\}$ $\mathtt{if}\ (\mathtt{i} > \mathtt{j})\ \mathtt{then}\ \mathtt{m} = \mathtt{i} - \mathtt{j}\ \mathtt{else}\ \mathtt{m} = \mathtt{j} - \mathtt{i}$ $\{\mathtt{m} - 2 * \mathtt{j} = 0\}$

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

2. Sejam P, Q dois predicados arbitrários, e seja S um programa arbitrário. O que significa a validade dos seguintes triplos (tendo em conta que a notação [ ] representa correcção total):

   (a) $\{P\}$ $\mathtt{S}$ $\{\mathsf{true}\}$

   (b) $[P]$ $\mathtt{S}$ $[\mathsf{true}]$

   (c) $\{P\}$ $\mathtt{S}$ $\{\mathsf{false}\}$

   (d) $[P]$ $\mathtt{S}$ $[\mathsf{false}]$

   (e) $\{\mathsf{false}\}$ $\mathtt{S}$ $\{Q\}$

   (f) $[\mathsf{false}]$ $\mathtt{S}$ $[Q]$

   (g) $\{\mathsf{true}\}$ $\mathtt{S}$ $\{Q\}$

   (h) $[\mathsf{true}]$ $\mathtt{S}$ $[Q]$

3 Lógica de Hoare

Consequência

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

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

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

Ciclo

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

1. Apresente regras derivadas cujas conclusões sejam:
   (a) $\{\,P\,\}\; x := E \;\{\,Q\,\}$
   (b) $\{\,P\,\}\; while\ (c)\ S\ \ \{\,Q\,\}$
   (c) $\{\,P\,\}\; x := E_1\ ;\ while\ (c)\ \{S\ ;\ x := E_2\}\ \ \{\,Q\,\}$.
   Note que esta última construção corresponde ao comando `for` do C.
   (d) $\{\,P\,\}\; if\ (c)\ then\ S \;\{\,Q\,\}$
2. Apresente uma prova que justifique cada um dos seguintes triplos de Hoare:
   (a) $\{\mathtt{i} > \mathtt{j}\}\ \mathtt{j} = \mathtt{i} + \mathtt{1};\ \mathtt{i} = \mathtt{j} + \mathtt{1}\ \{\mathtt{i} > \mathtt{j}\}$
   (b) $\{\mathtt{i}\ ! =\ \mathtt{j}\}\ \mathtt{if}\ (\mathtt{i} > \mathtt{j})\ \mathtt{then}\ \mathtt{m} = \mathtt{i} - \mathtt{j}\ \mathtt{else}\ \mathtt{m} = \mathtt{j} - \mathtt{i}\ \{\mathtt{m} > 0\}$
   (c) $\{\mathtt{a} > \mathtt{b}\}\ \mathtt{m} = \mathtt{1}; \mathtt{n} = \mathtt{a} - \mathtt{b}\ \{\mathtt{m} * \mathtt{n} > 0\}$
   (d) $\{\mathtt{s} = 2^{\mathtt{i}}\}\ \mathtt{i} = \mathtt{i} + \mathtt{1}; \mathtt{s} = \mathtt{s} * 2\ \{\mathtt{s} = 2^{\mathtt{i}}\}$
   (e) $\{\mathtt{True}\}\ \mathtt{if}\ (\mathtt{i} < \mathtt{j})\ \mathtt{then}\ \mathtt{min} = \mathtt{i}\ \mathtt{else}\ \mathtt{min} = \mathtt{j}\ \{\mathtt{min} \leq \mathtt{i} \wedge \mathtt{min} \leq \mathtt{j}\}$
   (f) $\{\mathtt{i} > 0 \wedge \mathtt{j} > 0\}\ \mathtt{if}\ (\mathtt{i} < \mathtt{j})\ \mathtt{then}\ \mathtt{min} = \mathtt{i}\ \mathtt{else}\ \mathtt{min} = \mathtt{j}\ \{\mathtt{min} > 0\}$

4 Correcção parcial de ciclos

1. 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.

```
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

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

2. Considere o seguinte programa para calcular a divisão inteira e o resto da divisão inteira entre dois números inteiros positivos.

```
r = x; q = 0;
while y <= r {
    r = r-y; q = q+1;
}
```

Prove que este programa está correcto face à especificação

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

Use como invariante o predicado

$$I \doteq (r \geq 0) \wedge (q * y_0 + r = x_0) \wedge (y = y_0)$$

3. 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;
}
```

Prove que este programa está correcto face à especificação

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

Use como imvariante o predicado

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

4. Considere a sequinte especificação de um programa que calcula o somatório dos elementos de um vector.

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

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[i]; i = i+1;
}
```

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

5. Considere a sequinte 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;
}
```

6. Considere a sequinte 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)
```
res = 0; i = 0;
while (i<y) {
    res = res + x;
    i = i+1
}
```

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

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

```
        y = y/2
    }
```

7. Considere a sequinte 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<y) {
    i = i+1;
    f = f * i;
}
```

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

5 Correcção total

Consequência

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

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

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

Ciclo

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

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

$$\{ P \} \; while \; (\mathsf{true}) \; \mathsf{Skip} \; \{ Q \}$$

2. 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.

3. 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.

6 Geração de condições de verificação

1. Apresente as condições de verificação necessárias à prova de correcção parcial dos seguintes programas (anotados em comentário).

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

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

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

2. Para cada um dos programas apresentados nas secções anteriores:
   (a) anote convenientemente os programas.
   (b) a partir dos programas anotados, apresente as condições de verificação necessárias às provas de correcção parcial/total desses programas.

3. 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 correcção total.

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

(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;}
}
```