

Programozáselmélet - gyakorlatokra javasolt feladatok - 11. alkalom

1. Mutasd meg, hogy az alábbi annotált program holtpontmentes.

$$A = (x:\mathbb{Z})$$

$$B = (x':\mathbb{Z})$$

$$Q = (x = x' \wedge x = 0)$$

$$R = (x = 1)$$

$$\{x = 0\}$$

**parbegin**  $S_1 \parallel S_2$  **parend**

$$\{x = 1\}$$

$S_1$ :

$$\{x = 0 \vee x = 1\}$$

**await**  $x = 1$  **then** SKIP **ta**

$$\{x = 1\}$$

$S_2$ :

$$\{x = 0\}$$

$x := 1$

$$\{x = 1\}$$

2.  $A = (a:\mathbb{Z}^n, b:\mathbb{Z}^n)$

$$B = (a':\mathbb{Z}^n)$$

$$Q = (a = a')$$

$$R = (a = a' \wedge b = a')$$

Mutasd meg, hogy az alábbi annotált program holtpontmentes.

$i:\mathbb{N}$  és  $j:\mathbb{N}$  segédváltozói a programnak.

$$i, j := 1, 1;$$

$$\{a = a' \wedge i = 1 \wedge j = 1\}$$

**parbegin**  $S_1 \parallel S_2$  **parend**

$S_1$ :

$$\{Inv\}$$

**while**  $i \leq n$  **do**

$$\{Inv \wedge i \leq n\}$$

**await**  $i = j$  **then**

$$x, i := a[i], i + 1$$

**ta**

$$\{Inv\}$$

**od**

$$\{Inv \wedge i = n + 1\}$$

$S_2$ :

$$\{Inv\}$$

**while**  $j \leq n$  **do**

$$\{Inv \wedge j \leq n\}$$

**await**  $i > j$  **then**

$$b[j], j := x, j + 1$$

**ta**

$$\{Inv\}$$

**od**

$$\{Inv \wedge j = n + 1\}$$

$$Inv = (a = a' \wedge 0 \leq i-1 \leq j \leq i \leq n+1 \wedge \forall k \in [1..j-1]: b[k] = a[k] \wedge (i > j \rightarrow x = a[i-1]))$$

3. A következő program biztosítja a kölcsönös kizárást: az  $S_1$  és  $S_2$  programok (folyamatok) nem tartózkodhatnak egyszerre a kritikus szakaszukban (mert nem használhatnak egyszerre egy közös erőforrást). Lássuk be hogy nem fordulhat elő holtponthelyzet, azaz hogy a folyamatok egymásra várnak.

```

{TRUE}
turn, flag1, flag2 := 1, false, false;
{¬flag1 ∧ ¬flag2}
parbegin S1||S2 parend
{FALSE}

```

```

{Inv ∧ ¬flag1}
S1:
{Inv ∧ ¬flag1}
while TRUE do
{Inv ∧ ¬flag1}
  non_critical_section_of_S1
  {Inv ∧ ¬flag1}
  flag1, turn := true, 1
  {Inv ∧ flag1}
  await (¬flag2 ∨ turn ≠ 1) then
  {Inv ∧ flag1 ∧ (¬flag2 ∨ turn = 2)}
    critical_section_of_S1
    {Inv ∧ flag1 ∧ (¬flag2 ∨ turn = 2)}
    flag1 := false
  {Inv ∧ ¬flag1}
  ta
{Inv ∧ ¬flag1}
od
{FALSE}

```

```

{Inv ∧ ¬flag2}
S2:
{Inv ∧ ¬flag2}
while TRUE do
{Inv ∧ ¬flag2}
  non_critical_section_of_S2
  {Inv ∧ ¬flag2}
  flag2, turn := true, 2
  {Inv ∧ flag2}
  await (¬flag1 ∨ turn ≠ 2) then
  {Inv ∧ flag2 ∧ (¬flag1 ∨ turn = 1)}
    critical_section_of_S2
    {Inv ∧ flag2 ∧ (¬flag1 ∨ turn = 1)}
    flag2 := false
  {Inv ∧ ¬flag2}
  ta
{Inv ∧ ¬flag2}
od
{FALSE}

```

$Inv = (turn = 1 \vee turn = 2)$   
 $turn: \{1, 2\}$ , míg  $flag_1$  és  $flag_2$  logikai változók.

4. Az alábbi program  $n$  darab adatbázisba író, és  $m$  darab abból olvasó folyamat ütemezését végzi. Az adatbázist egyszerre több olvasó folyamat is használhatja, de ha egy író folyamat használja az adatbázist, aközben más folyamatoknak sem írni, sem olvasni nem lehet. Mutasd meg a rendszer holtpontmentességét.

```

w, r := 0, 0;
{w = 0 ∧ r = 0}
parbegin W1||...||Wn||R1||...||Rm parend
{FALSE}

```

```

Rj:

while TRUE do
  {I ∧ rj = 0}
  await w=0 then
    r := r + 1; rj := 1
  ta ;
  {I ∧ rj = 1}
  read ;
  {I ∧ rj = 1}
  [r := r - 1; rj := 0]
  work ;
  {I ∧ rj = 0}
od
{FALSE}

```

```

Wi:

while TRUE do
  {I ∧ wi = 0}
  work ;
  await w = 0 ∧ r = 0 then
    w := 1; wi := 1
  ta ;
  {I ∧ wi = 1}
  write ;
  {I ∧ wi = 1}
  [w := 0; wi := 0]
  {I ∧ wi = 0}
od
{FALSE}

```

$$I = ((r = 0 \vee w = 0) \wedge w = \sum_{i=1}^n w_i \wedge r = \sum_{j=1}^m r_j)$$

$w_i: \{0, 1\}$ . Értéke 1, ha az adatbázist a  $W_i$  író folyamat használja, különben 0.

$r_i: \{0, 1\}$ . Értéke 1, ha az adatbázist az  $R_i$  olvasó folyamat használja, különben 0.

5. Egy kolostorban öt filozófus él. Minden idejüket egy asztal körül töltik: gondolkodnak és spagettit esznek. Mindegyikük előtt egy tányér van, amelyből sohasem fogy ki a tészta. A spagetti annyira össze van ragadva, hogy mindkét kezükbe villát kell fogniuk, csak így tudnak enni. Egy filozófus csak a tányérja melletti bal- illetve jobboldali villát veheti fel. Az asztalon azonban csak öt villa van, így két egymás mellett ülő filozófus nem tud egyszerre enni.

Hoare megoldásában az  $af[0..4]$  tömb  $af[i]$  eleme az  $i$ -edik filozófus számára szabad villák számát adja meg.

$eating[i]$  értéke 1 ha az  $i$ -edik filozófus eszik, különben 0.

Bizonyítsd be hogy nem fordulhat elő holtpon.

$$Inv(forks) = (\forall i \in [0..4]: 0 \leq eating[i] \leq 1 \wedge (eating[i] = 1 \Rightarrow af[i] = 2) \wedge af[i] = 2 - (eating[i(-)1] + eating[i(+ )1]))$$

(+) és (-) modulo 5 összeadást illetve kivonást jelölnek. Például,  $4(+ )1 = 0$  és  $4(-)1 = 3$ , továbbá  $0(-)1 = 4$ .

```

{TRUE}
for all i in [0..4] af[i] := 2;
parbegin DP0 || ... || DP4 parend
{Inv(forks) ∧ ∀i ∈ [0..4]: eating[i] = 0}

```

$$\begin{aligned}
Inv(forks) = (\forall i \in [0..4]: 0 \leq eating[i] \leq 1 \wedge (eating[i] = 1 \Rightarrow af[i] = 2) \wedge \\
af[i] = 2 - (eating[i(-)1] + eating[i(+ )1]))
\end{aligned}$$

```

{Inv(forks) ∧ eating[i] = 0}
/* the program of the ith philosopher */
DPi:

ji := 1;
while ji ≤ Ni do
{eating[i] = 0 ∧ Inv(forks) ∧ ji ≤ Ni}
  getforks_i:
  await af[i] = 2 then
  {eating[i] = 0 ∧ Inv(forks) ∧ ji ≤ Ni ∧ af[i] = 2}
  af[i(-)1] := af[i(-)1] - 1;
  af[i(+ )1] := af[i(+ )1] - 1;
  eating[i] := 1
  ta;
  {eating[i] = 1 ∧ Inv(forks) ∧ ji ≤ Ni}
  eat_i; /* programcode of the ith philosopher's eating */
  {eating[i] = 1 ∧ Inv(forks) ∧ ji ≤ Ni}
  releaseforks_i:
  await TRUE then
  {eating[i] = 1 ∧ Inv(forks) ∧ ji ≤ Ni}
  af[i(-)1] := af[i(-)1] + 1;
  af[i(+ )1] := af[i(+ )1] + 1;
  eating[i] := 0
  ta;
  {eating[i] = 0 ∧ Inv(forks) ∧ ji ≤ Ni}
  think_i; /* programcode of the ith philosopher's thinking */
  {eating[i] = 0 ∧ Inv(forks) ∧ ji ≤ Ni}
  ji := ji + 1
  {eating[i] = 0 ∧ Inv(forks)}
od
{eating[i] = 0 ∧ Inv(forks) ∧ ji > Ni}

```