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 $S_1 \| S_2$ **parend**
   $\{x = 1\}$

$S_1:$	$S_2:$
$\{x = 0 \vee x = 1\}$ await $x = 1$ then SKIP ta $\{x = 1\}$	$\{x = 0\}$ $x := 1$ $\{x = 1\}$

2. Mutasd meg, hogy az alábbi annotált program holtpontmentes.
   $A = (a{:}\mathbb{Z}^n, b{:}\mathbb{Z}^n)$
   $B = (a'{:}\mathbb{Z}^n)$
   $Q = (a = a')$
   $R = (a = a' \wedge b = a')$

   $i, j := 1, 1;$
   $\{a = a' \wedge i = 1 \wedge j = 1\}$
   parbegin $S_1 \| S_2$ **parend**

$S_1:$	$S_2:$

$\{Inv\}$
while $i \le n$ do
$\{Inv \wedge i \le n\}$
  await $i = j$ then
    $x, i := a[i], i + 1$
  ta
$\{Inv\}$
od
$\{Inv \wedge i = n + 1\}$
	$\{Inv\}$
	while $j \le n$ do
	$\{Inv \wedge j \le n\}$
	  await $i > j$ then
	    $b[j], j := x, j + 1$
	  ta
	$\{Inv\}$
	od
	$\{Inv \wedge j = n + 1\}$

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

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.

$\{TRUE\}$
$turn, flag_1, flag_2 := 1, false, false;$
$\{\neg flag_1 \wedge \neg flag_2\}$
parbegin $S_1 \| S_2$ **parend**
$\{FALSE\}$

$\{Inv \wedge \neg flag_1\}$ $S_1:$ $\{Inv \wedge \neg flag_1\}$ while TRUE do $\{Inv \wedge \neg flag_1\}$ non_critical_section_of_S1 $\{Inv \wedge \neg flag_1\}$ $flag_1, turn := true, 1$ $\{Inv \wedge flag_1\}$ await $(\neg flag_2 \vee turn \neq 1)$ then $\{Inv \wedge flag_1 \wedge (\neg flag_2 \vee turn = 2)\}$ critical_section_of_S1 $\{Inv \wedge flag_1 \wedge (\neg flag_2 \vee turn = 2)\}$ $flag_1 := false$ $\{Inv \wedge \neg flag_1\}$ ta $\{Inv \wedge \neg flag_1\}$ od $\{FALSE\}$	$\{Inv \wedge \neg flag_2\}$ $S_2:$ $\{Inv \wedge \neg flag_2\}$ while TRUE do $\{Inv \wedge \neg flag_2\}$ non_critical_section_of_S2 $\{Inv \wedge \neg flag_2\}$ $flag_2, turn := true, 2$ $\{Inv \wedge flag_2\}$ await $(\neg flag_1 \vee turn \neq 2)$ then $\{Inv \wedge flag_2 \wedge (\neg flag_1 \vee turn = 1)\}$ critical_section_of_S2 $\{Inv \wedge flag_2 \wedge (\neg flag_1 \vee turn = 1)\}$ $flag_2 := false$ $\{Inv \wedge \neg flag_2\}$ ta $\{Inv \wedge \neg flag_2\}$ 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.

$w, r := 0, 0;$
$\{w = 0 \wedge r = 0\}$
parbegin $W_1 \| \ldots \| W_n \| R_1 \| \ldots \| R_m$ **parend**
$\{FALSE\}$

R_j :

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

W_i :

```
while TRUE do
{I ∧ w_i = 0}
   work;
   await w = 0 ∧ r = 0 then
      w := 1; w_i := 1
   ta;
   {I ∧ w_i = 1}
   write;
   {I ∧ w_i = 1}
   [w := 0; w_i := 0]
{I ∧ w_i = 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 holtpont.

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

   (+) 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 $DP_0 \| \ldots \| DP_4$ **parend**

$\{Inv(forks) \wedge \forall i \in [0..4]\colon eating[i] = 0\}$

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

$\{Inv(forks) \wedge eating[i] = 0\}$

/* the program of the ith philosopher */

DP_i:

$j_i := 1$;

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