Homework 3

Monday, 15 March 2021 10:35

Exercise 1

We want that: $\#op_i \leq \#op_{i+1} + 1$

In our case, since we 3 processes, P_1, P_2 and P_3:

$$\#op_1 \leq \#op_2 + 1 \leq \#op_3 + 2$$

Let sems an array of binary semaphores.

Initialized as follows: sems[1,0,0]

```
op_i():=
  sems[i].down()
  ⋮
  sems[(i+1) mod 3].up()
  return()
```

Exercise 2

Philosopher i:

```
Repeat forever
    think;
    if (i mod 2 = 0) then
        fork[i].down()        ①
        fork[(i+1) mod 5].down()  ⓪
    else
        fork[(i+1) mod 5].down()  ②
        fork[i].down()            ⓔ
        ⋮
```

Deadlock freedom "if at least one phi. want to eat, at least one of them will eat.

Let $X = \{i : P_i \text{ wants to eat}\}$

Proof: Assume by the way of contradition all $p_i (i \in X)$ can't eat:

- For All $P_i (i+1)$ by A, they are blocked at either ⓪ or ②, otherwise at least one P_i eats.

A

1. P_i is blocked at ① (i is odd) and it means that $P_{(i-1) \bmod 5}$ has passed ⓪ $\Rightarrow$ $P_{(i-1) \bmod 5}$ eats.
2. P_i is blocked at ② (i is even) and it means that $P_{(i+1) \bmod 5}$ has passed ② $\Rightarrow$ $P_{(i+1) \bmod 5}$ eats.

- Let P_j $(j \in X)$ $(j$ odd$)$ the first who pass ② $\Rightarrow$ $P_{(j-1) \bmod 5}$ will get fork[j] = 0 and it'll be blocked at ②
  hence we have at least one process which is blocked at ②, by A2 P_j eats!
- Let P_j $(j \in X)$ $(j$ even$)$ the first who pass ② $\Rightarrow$ $P_{(j+1) \bmod 5}$ will get fork[$(j+1) \bmod 5$] = 0 and it'll be blocked at ①
  hence we have at least one process which is blocked at ①, by (A) P_j eats!

- For P_1
  - if it is blocked at ①, P_5 eats!
  - it cannot be blocked at ⓪, since P_5 must be blocked $\Rightarrow$ P_1 eats!

P_0 runs denoted as
P_2 runs denoted as

Counter example:

FREE.down(); SP.down(); i ← N; i ← N; N ← (N+2) mod K; SP.up(); FREE.down(); SP.down(); i ← N; N ← (N+2) mod K; SP.up();

t_{start} BUFF[i] ← v; BUFF[i] ← w

P_2 immediatelly overwrite P_1

THE VALUE PRODUCED BY P_2 IS LOST!

Exercise 4)

monitor solution
- cnt ∈ {0,1} init 1
- condition C

```
operation A():=
  if cnt>0
    then C.wait()
    ⋮
  return()

operation B():=
  if cnt>0
    then C.wait()
    ⋮
  return()

operation C():=
  ⋮
  if cnt>0:
    c ← c-1
    C.signal()
  return()
```