Homework 2

Saturday, 6 March 2021 09:33

Exercise 1:

(X initializated at 0)

lock() :=
    wait X.c&s(0,1)=0
    return()

unlock() :=
    X.c&s(1,0)
    return()

Is the protocol starvation free?
No, it isn't! This is just deadlock free.

RECALL: Deadlock freedom: "if at least one proc. invokes lock(), at least one of them enters in C.S.

Proof: The first proc. P_i, which invokes X.c&s wins!

Counterexample: It is not starvation freedom.
Let P_1, P_2:

P_1 run is denoted

P_2 run is denoted

X.c&s(0,1)=0 ; C.S. ; X.c&s(0,1)≠0 ; X.c&s(1,0) . . .

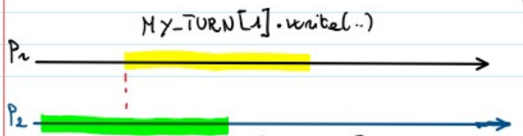
t_{start}

Repeat it forever, P_2 will never wins!

Exercise 2

Lamport's rely on safe registers (2 registers SW-MR)

Assume that each MY_TURN[i] have the following domain: {0 ... 7}



MY_TURN[1].read() → a=7 → read for computing max{MY_TURN[1] ... MY_TURN[N]} (line (2) Fig 2.25)

Then P_2 performs the +1 (line (2) Fig 2.25), hence there isn't enough space to store 8 ...
That's showed that Lamport's algorithm requires unbound registers!

Exercise 3

RECALL: Deadlock freedom: "if at least one proc. invokes lock(), at least one of them enters in C.S.

Let $X = \{i : P_i \text{ invokes lock()}\}$

NOTE :

a) From the initializzation of DATE array :

$\forall i \in X \Longrightarrow$ DATE[i] = i

then the $\min(X) = m$ $(\forall x \in X \,.\, x \neq m \rightarrow m < x)$
has the smallest value of DATE[m ... $|X|$]

Proof: Assume by the way of contradiction, all p_i $(i \in X)$ are blocked in their wait.
Stated differently:

$\forall j \,.\, j \neq i \longrightarrow FLAG[j] = down \vee DATE[i] < DATE[j]$ is FALSE!

$\neg \forall j \,.\, j \neq i \longrightarrow$ is TRUE!

$\exists j \,.\, \underbrace{j \neq i}_{①} \wedge \underbrace{FLAG[j] = up}_{②} \wedge \underbrace{DATE[i] \geq DATE[j]}_{③}$

Now let take P_m, it doesn't exist another P_j, different from it ①
which is in X ② and its DATE value is smaller or equal to
DATE[m] ③

To sum up: we reach a contradiction!

Exercise 4

If the reset of DATE happens at $n+K$ for $K \in \{1, \dots, n-1\}$ the Aravind's bounded algorithm is not anymore starvation freedom.

NOTE: the deadlock freedom, still holds, since it doesn't rely on the reset condition.

Proof: Assume the starvation freedom property: "For every P_i $(i \in N)$ P_i eventually wins"
By the way of contradiction

Let P_1, P_2

$n = 2$
$K = 1$

Assume that both have invoked lock()

P_1 wins (DATE[1] < DATE[2]) and enters in C.S., when invokes its DATE[1] is reset to 1.
Quickly P_1 invokes lock() (FLAG[1] $\leftarrow$ up), go to wait's line an wins, and this is done forever.

P_2 always runs the wait after that P_1 has set to up it FLAG:
P_2 LOSES FOREVER, CONTRADICTION!