Homework 1

Tuesday, 2 March 2021 15:51

Exercise 1:

Let $n = 3$

P_1 run is denoted with

P_2 run is denoted with

P_3 run is denoted with

For simplicity, let group together the following atomic instructions:

fast(y) := FLAG_LEVEL[y] ← 1; AFTER_YOU[1] ← y

fast(2); FLAG_LEVEL[1] ← 1; AFTER_YOU[1] ← 1; AFTER_YOU[1] = 1; AFTER_YOU[1] ≠ 2; ...

t start

P_2 moves to level 2 and enters in C.S.; fast(3); fast(2); P_3 moves to level 2 and enters in C.S. ...

P_2 moves to level 2 and enters in C.S. fast(3); fast(2); P_3 moves to level 2 and enters in C.S

t end

This scenario is repeated until P_1 wakes up and it can progress to level 2:

BUT THE TIMES IN WICH P_1 "SLEEP" IS FINITE BUT NOT BOUNDED.

Exercise 2

P_1 run is denoted with

P_2 run is denoted with

X ← 1; Y = ⊥; Y ← 1; X = 1; X ← 2; Y ≠ ⊥; return(commit); return(abort)

t start t end

Repeat this run forever, P_2 will NEVER RECEIVE COMMIT!

Exercise 3:

P_1 run is denoted with

P_2 run is denoted with

For every run either P_1 or P_2 receive commit is FALSE!

There exist a run which contradicts it:

X ← 1; Y = ⊥; Y ← 1; X ← 2; Y ≠ ⊥; return($abort_2$); X ≠ 1; return($abort_2$)

Exercise 4:

In order to show "Lamport's Fast MUTEX" algorithm is NOT deadlock freedom, it's sufficient to find a run with deadlock:

P_1 run is denoted with

P_2 run is denoted with