INFO-F-404 : Operating Systems II

1 Exercises

Exercise 1: Let's consider the following protocol of management of critical sections.

First try			
integer turn $\leftarrow$ 1			
p		q	
	loop forever		loop forever
p1:	non-critical section	q1:	non-critical section
p2:	await turn = 1	q2:	await turn = 2
p3:	critical section	q3:	critical section
p4:	turn $\leftarrow$ 2	q4:	turn $\leftarrow$ 1

a) Show that this protocol can not be used in order to manage mutex sections. Use this simplified version of the protocol in order to minimize the size of diagrams.

First try (simplified)			
integer turn $\leftarrow$ 1			
p		q	
	loop forever		loop forever
p1:	await turn = 1	q1:	await turn = 2
p2:	turn $\leftarrow$ 2	q2:	turn $\leftarrow$ 1

b) Use the state diagram in order to show that this protocol guarantees the absence of deadlock.

c) Show that this first protocol could lead to livelock (if the time that one process passes in the critical section is not limited).

Answer : Exercise 1 (a) and (b) : see Figure 1

Exercise 1 (c): If the process *p* quit with an error during its critical section, it will never assign the value 2 to the variable *turn* $\Rightarrow$ *q* is at livelock.

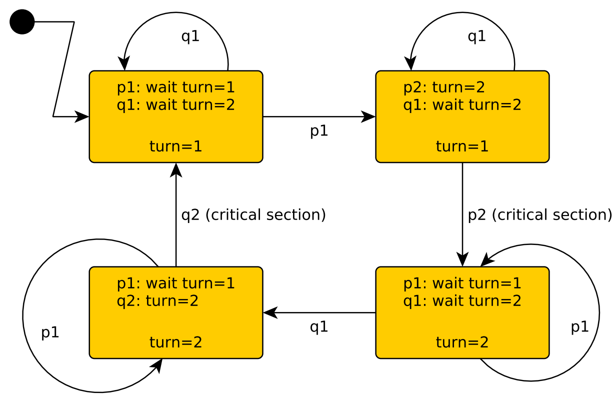


Figure 1: Answer : Exercise 1 (a) and (b)

Exercise 2: Let's consider the following protocol of management of critical sections.

Second try			
boolean wantp ← false, wantq ← false			
p		q	
	loop forever		loop forever
p1:	non-critical section	q1:	non-critical section
p2:	await wantq = false	q2:	await wantp = false
p3:	wantp ← true	q3:	wantq ← true
p4:	critical section	q4:	critical section
p5:	wantp ← false	q5:	wantq ← false

a) Show that this protocol can not guarantee mutual exclusion.

Second try (simplified)			
boolean wantp ← false, wantq ← false			
p		q	
	loop forever		loop forever
p1:	await wantq = false	q1:	await wantp = false
p2:	wantp ← true	q2:	wantq ← true
p3:	wantp ← false	q3:	wantq ← false

b) Could we have a deadlock and/or livelock ?

Answer : Exercise 2 (a) : see Figure 2.
Exercise 2 (b): see Table 1

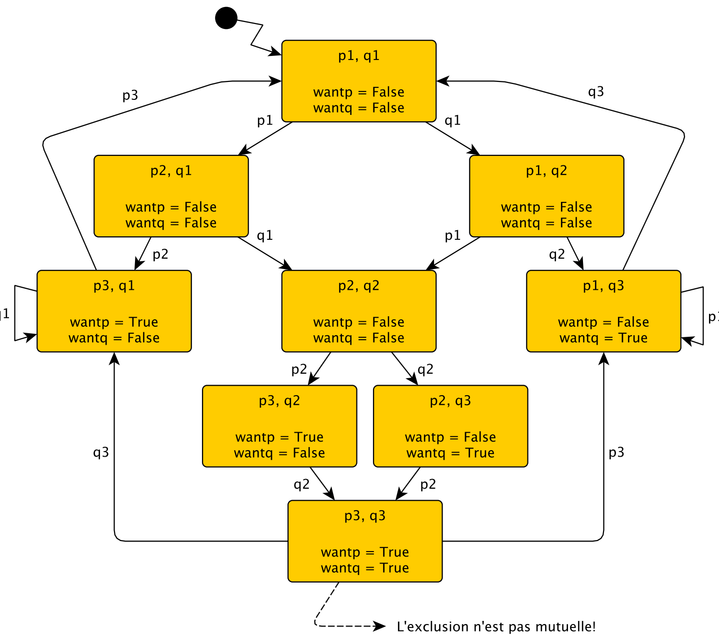


Figure 2: Answer : Exercise 2 (a)

p	q	wantp	wantq
p1	q1	False	False
p2	**q1**	False	False
p2	q2	False	False
p3	**q2**	True	False
p3	q3	True	True

Table 1: Exercise 2 (b).

Exercise 3: Let's consider the following protocol of management of critical sections.

Third try	
boolean wantp ← false, wantq ← false	
p	q
loop forever	loop forever
p1: non-critical section	q1: non-critical section
p2: wantp ← true	q2: wantq ← true
p3: await wantq = false	q3: await wantp = false
p4: critical section	q4: critical section
p5: wantp ← false	q5: wantq ← false

a) Write a simplified version of this protocol.

b) Create a state diagram of this protocol and verify that we have mutual exlusion.

c) Show that this protocol could lead to a deadlock.

d) Describe a scenario that can lead to a deadlock.

Answer : Exercise 3 (a): see Figure 3.
Exercise 3 (b) and (c): see Figure 4.
Exercise 3 (d): see Table 2.

Third try : simplified			
boolean wantp ← false, wantq ← false			
p		q	
	loop forever		loop forever
p1:	wantp ← true	q1:	wantq ← true
p2:	await wantq = false	q2:	await wantp = false
p3:	wantp ← false	q3:	wantq ← false

Figure 3: Exercise 3 (a).

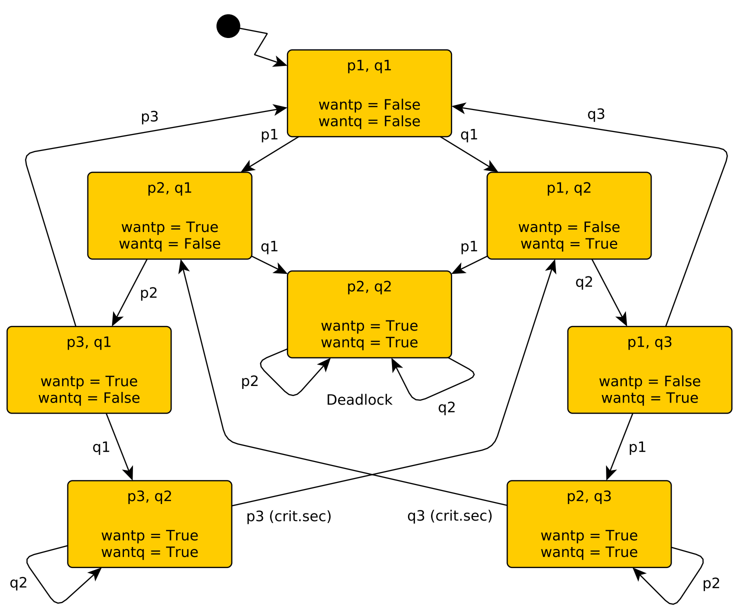


Figure 4: Answer : Exercise 3 (b) and (c)

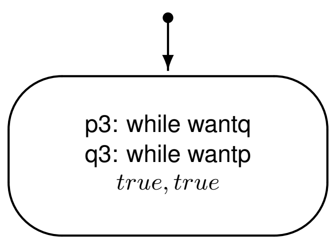
p	q	wantp	wantq
p1	q1	False	False
p2	**q1**	True	False
p2	q2	True	True

Table 2: Exercise 3 (d).

Exercise 4: Let's consider the following protocol of management of critical sections. This protocol guarantees the absence of deadlocks and mutual exclusion.

Fourth try			
boolean wantp ← false, wantq ← false			
p		q	
	loop forever		loop forever
p1:	non-critical section	q1:	non-critical section
p2:	wantp ← true	q2:	wantq ← true
p3:	while wantq	q3:	while wantp
p4:	wantp ← false	q4:	wantq ← false
p5:	wantp ← true	q5:	wantq ← true
p6:	critical section	q6:	critical section
p7:	wantp ← false	q7:	wantq ← false

a) Use state diagram and show that this protocol could lead to livelock.



b) Why is it a livelock (and not a deadlock) ?

Answer : Exercise 4 (a): see Figure 5.
Exercise 4 (b): Processes could be scheduled in a bad order.

Exercise 5: Consider Lamport's algorithm for mutex sections.

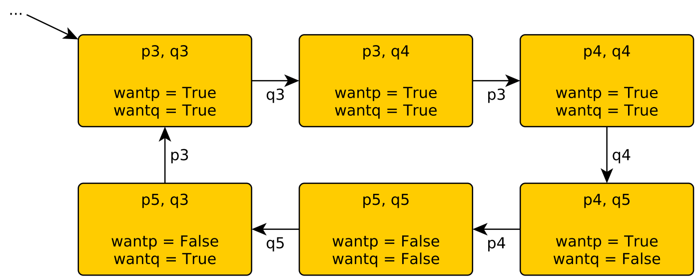


Figure 5: Answer : Exercise 4 (a).

Bakery algorithm (2 proces)			
integer np $\leftarrow$ 0, nq $\leftarrow$ 0			
p		q	
	loop forever		loop forever
p1:	non-critical section	q1:	non-critical section
p2:	np $\leftarrow$ nq $+$ 1	q2:	nq $\leftarrow$ np $+$ 1
p3:	await nq = 0 or np $\leq$ nq	q3:	await np = 0 or nq $<$ np
p4:	critical section	q4:	critical section
p5:	np $\leftarrow$ 0	q5:	nq $\leftarrow$ 0

a) Use a state diagram in order to show that this algorithm is correct.

Answer : Exercise 5 (a): see Figure 6.

Exercise 6: Let's consider the system represented by Table 3. These are all *periodic*, *asynchronous* tasks with *constrained deadline*.

Task index	Release time	WCET	Deadline	Period
Tâche τ_1	0	40	60	150
Tâche τ_2	50	70	100	150
Tâche τ_3	100	30	150	150

Table 3: System of 3 periodic, asynchronous tasks with constrained deadline.

a) Use a distributed algorithm in order to verify that this system can be scheduled on a single processor platform.

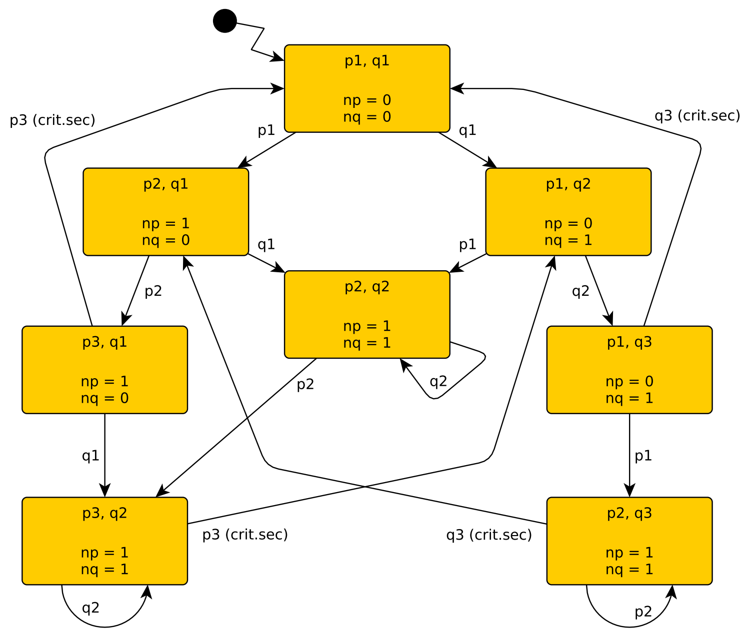


Figure 6: Answer : Exercise 5 (a).