Formal Methods
Assignment 1

Ali Sabeh, ID : 201822768

February 27, 2018

1 Exercise 1

Problem : fix mutex to be live using eshmun
Objective : We must prevent deadlock (prevent a certain process to loop infinitely in the critical section).
Solution :
We will use eshmun to represent the processes and their states and try to achieve the goal. Moreover, we will use the initial diagram given in class for the states of the processes and add/delete states to make it live.
The property of liveness can be represented by the following CTL formula :
$AG(T1 \rightarrow AF(C1))$ AND $AG(T2 \rightarrow AF(C2))$
The following steps are done :
1- Remove state (C1,C2) because of course we need to satisfy mutual exclusion.
2- Add Flags F1 and F2 for processes P1 and P2 to prevent deadlock.
3-In each transition we will make sure that we only change 1 state , for example if we are in state S1 = (N1,N2) then we can only move to (T1,N2) or (N1,T2) this is because if we are working in a shared memory we have to prevent 2 processes from accessing the same file.
Now for the transitions : S0(N1,N2) we have 2 transitions to S1(T1,N1,F1) and S2(N1,T2,F2) were in S1 process P1 will be requesting the critical section, and in S2 process P2 will be in requesting the critical section.
Then from S1, process P1 will enter the critical section in state S3, or both process P1 and P2 will be requesting the critical section but the mutex is with P1.
Now with P1 in the critical section in state S3, process p2 will be requesting the critical section in state S6 but it has no mutex F2, so in order to prevent deadlock and release the mutex F1 we move from S6(C1,T2,F1) to (C1,T2) then to S2(N1,T2) and after that we give the mutex F2 to process P2 in state S5 (N1,T2,F2) . Now, process S2 can enter the critical section S7 (N1,C2,F2).
Same thing is applied if we move from the initial state S0 to S5, were in this case process p2 will enter the critical section first and then P1 and so on. In the assumptions made we garantee liveness since no process can enter a deadlock

and prevent the other process from entering the critcal section. Hence, liveness property is satisfied in the built graph.

2 Exercise 2

Problem : let path formula f = O1 O2 O3 ..... ON P ,for each Oi is either F or G, can we simplify f ?
Solution :
We try to produce different combinations of path f and see if this path can be reduced.
1- If f = Fp (reduced)

2- If f = Gp (reduced)

3- If f = FFFF....Fp . If we try to decompose this formula :
F(F...Fp) = eventually F....Fp will hold, #F = i , then decomposing the internal formula we get :
eventually FF....Fp will hold, #F = i-1 , and so on, until we reach Fp = eventually p will hold.
Then we can reduce FF...Fp to Fp which says that eventually p will hold.

4- If f = GGG....Gp , #G = i , G....Gp will always hold , #G = i-1 , if we follow the same steps as for part 3, we reach Gp which means that p will always be true at each state. Therefore GG....Gp can be reduced to Gp.

5- If f = FGFp this formula means that GFp will eventaully holds, which in turns means that eventually and forever Fp will always hold, and hence p wil eventually holds. From the semantics of the meaning of this formula we can see that it's equivalent to GFp which means that Fp will always hold, and hence p will eventually holds.
We can then deduce that FGFp can be reduced to GFp.
6- Similarly, If f = GFGp = FGp is always true → eventually Gp will hold → eventually p will always be true.
Then the formula GFGp means that we will reach a state where p is true at this state and for all the states after it.
On the other hand, we have FGp means that eventually Gp will be true, and therefore, eventually we will reach a state where p is true at this state and for all states after it. So, GFGp can be reduced to FGp .
By 5 and 6, we can deduce that for any alternating FGFFGF....FGFp or GFGFG....GFGp, we can reduce this formula to the last 2 ... , for example, FGFGFG...FGFGFGFp and GFGFG....FGFGp can be reduced to GFp and can b FGp respectively.
In conclusion, if f = O1 O2 O3 .... ON p , then we can combine all consecutive

F's and G's to single F and G, and for

$$f = O1O2O3....ONp = \begin{cases} FGp & if O(N-1) = F and G = O(N), for N >= 2 \\ GFp & if O(N-1) = G and F = O(N), for N >= 2 \\ Fp & if O1 = O2 = ...ON = F, for N >= 1 \\ Gp & if O1 = O2 = ...ON = G, for N >= 1 \end{cases}$$