Formal Methods

Assignment One

Neam Farroukh 201820049

Problem One:

The main goal of this exercise is to implement and apply liveness and fairness. By liveness we mean that only ONE process of both processes can be in the critical section. By fairness we mean that the process making the request before has the higher priority in entering the critical section before.

So To make the model live and fair I followed the following steps :

I started with the initial state S1 where neither process one nor process two is requesting. If Process one requested then there will be a transition to S2 where process one gets the mutex. Here we could have two scenarios, either process two requested before process one entering into the critical section or directly after process one being granted the mutex it entered the critical section. In the case of the first scenario, process one will still be the one entering the CS due to fairness. In both cases after process one entering the critical section I assumed process two is requesting. After process one leaving the critical section which is S7 it will still have a mutex and process two is still requesting. Then it will release the mutex(S8) and move to state S3 which states that process two has requested.

The same goes here we will have on of both scenarios : either process one didn't make any request or process one made a request directly after process two having the mutex. In both scenarios process two will enter the critical section and during that time I assumed that process one will definitely have made a request (in order to insure the transition from one process to the another in the model). After it has left the critical section it will still be having the mutex which it will then release. Since I assumed process one has already made a request, there will be a transition from S13 to S2 again.

If we entered this formula : (AG(T1 => (AF(C1)))) & (AG(T2 => (AF(C2)))) for global modal check we will find the model is live and fair.

Problem Two:

Let f = O1O2O3…On where for each Oi is either F or G

So in order to simplify first we have to produce different combinations for this path so we can check if it could be reduced.

- IF f = Fp or Gp the it's definitely reduced.
- Else if f = FFFF.. Fp, F(FFF…Fp) is equivalent to f so is FFFF….(Fp) which eventually means p holds.
- IF f= GGGG….Gp, same as the previous one , G(GGG….Gp) is equivalent to f so is GGGGG…(Gp) meaning the p eventually holds again.
- If f=FGFp we could deduce that GFp always holds thus Fp will surely hold and so does p:

  FGFp => GFp holds => Fp holds globally => p holds.

  Thus FGFp can be reduces to GFp.

- Now if f= GFGp this means that FGp is globally holds and Gp eventually holds and thus p globally holds (ppppp..) the same state will be reached if f = FGp which means that f can be reduced to FGp.

Thus any formula wi an alternating FGFG.. or GFGF.. can be reduced to one of the previous reductions.

Example :

FGFGFG….FGFp => GFp (FGFp can be reduced to GFp)

GFGFGF…GFGp => FGp (GFGp can be reduced to FGp)