Assignment 01 - Formal Methods

Raphaelle Akhras

February 2018

1 Question 1

There are many assumptions that could be taken while completing this assignment. There could be the assumption where process 1 goes before process 2 all the time. Thus we are forcing process 1 to go before 2. Then, both processes can enter into the critical section in order. On the other hand, process 2 could enter before process 1 and then there is liveness as long as process 2 completes its critical section, exits and process 1 can then enter the critical section. These assumptions force the two processes two follow an order that should not be imposed by the user. Thus there has to be a loop that allows process 1 to complete the critical section and process 2 to complete the critical section in any order that they choose. There are also the states that should be allowed that include when both processes do not wish to enter the critical section, or when 1 process wants to enter the critical section whereas process 2 does not want to enter it. Or the opposite, where process 2 wants to enter the critical section and process 1 does not.

There are the states when process 1 gets the mutex for its process that allows it to enter the critical section and process 2 is either not requesting the mutex or requesting it. In both cases, process 2 will not be able to enter the critical section problem. After process 1 will enter the critical section and process 2 can either be in either of the aforementioned states. After that, process 1 will exit the critical section and process 2 is again in the aforementioned states. Finally, process 1 will release the mutex and again process 1 will be in afore mentioned states. After this state, process 2 may enter the critical section if it has requested it, and may just remain in a state where it does not want to enter the critical section. Either process 1 or process 2 may enter the critical section at this point.

The issue of solving the critical section problem has been solved very long ago but in this first assignment, I used CTL to get liveness in our state. The formula in CTL logic for obtaining liveness in process 1 and 2 is AG((T1 => AF(C1) (T2 => AF(C2))). This means that when T1 requests the critical section, if it is requested, then at in all paths there will finally be a state that contains c1 to be true. This is also the case in terms of T2 for process 2 where if it requests the critical section, it will always have finally a state in which C2 is

true. To conclude, for all paths there it is globally true that when T1 and T2 occur the states can be reached. In my model, as in all models, N is for a non-requesting state. T is for a requesting state. C is for a state where the process is in the critical section. Finally, F is for when a process obtains the mutex. This always happens before the process enters the critical section as I will show below.

The first state of the model is when the two process, 1 and 2, are in a non-requesting state (N1,N2). Neither request to enter the critical section. Then either of two things can occur. Either process 1 can enter the critical section or process 2. This is done first, by requesting to enter the critical section. As mentioned previously, the two processes cannot enter at the same time which is the safety property. Now either process 1 can request to enter the critical section or process 2 either (T1,N2,F1) or (T1,T2,F2). To take the path of process 1 requesting the critical section, and without loss of generality, there are two options that may occur. Either process 2 requests to enter the critical section (T1,T2,F1) or process 1 enters the critical section (C1,N2,F1). Both these states will however end up in the same state which is (C1,T2,F1) since process 1 is still in the critical section but now we added the fact that process 2 is requesting the mutex. After this state, process 1 exits the critical section and becomes a non-requesting state (N1,T2,F1) and then the mutex is released (N1,T2) meaning that process 2 is now the only process requesting to enter the critical section and will thus go to the state where (N1,T2,F2) taking the mutex for itself so it can enter the critical section and thus repeating the cycle all other again.
After a global model check it completed, using the lives formula mentioned before, we will get that all states are reachable and live shown by all the dashed states (A denoting all).

2 Question 2

There are many different expressions that can be simplified into smaller expressions. However, let us first discuss the expressions that cannot. First we can discuss FGF and state that it is in fact not equivalent to GF. First of all, FGFf means that finally there be a point in time where Gf is true. And GFf means that globally there will be a Ff that is true. What Ff means that finally there will be an f that is true. To conclude we are saying that finally, at a certain point in time we will have globally Ff which means that after this point we will have f to be true. On the other hand, GFf means that from the first moment in time and till the last moment in time there will be an Ff that is true. Ff means that finally from that point, there will be a point where f is true. These two expressions are not equivalent. By the same logic, FG is not equivalent to GFG.

Now to discuss expressions that may be reduced we begin with the simplest of formulas which is for instance GGGGGGGf which is the same as Gf since the first time you say globally true it is implied that no matter what there is

after it, it is globally true even though there could be an F after it. But here the idea that there are all consecutive gs it could be reduced to Gf. This also works for FFFFFFFFFf which means that eventually/.finally there will be an FFFFFFFFFf that s true. And furthermore, if we analyze this expression the same is true. This will be repeated till we reach FFf and then f will finally be true (thus it holds under this expression). The main idea is that Ff is not equivalent to FFf. So it is true only when FFf and not when Ff since these two are not equivalent. To conclude, FFFFFFFFFf is equivalent to FFf since it is not equivalent to Ff meaning it can be simplified for an a number of fs greater than 2.