Iowa State University, Computer Science Department

COM S 612
Assignment 2

Ibne Farabi Shihab

October 20, 2020

1 Problem 1

We can use n processor Peterson algorithm(Lecture 10 and 11) where n=4. The whole process will be same except that now we have the specific value for n. For this, the formulation has not been explained as it is already been explained in lecture 10 and 11. For 4 processors, the tournament will take place 3 times. Lets a single node for instance. In the n processor algorithm from lecture 10. we can notice that there are one variable(W_v^{side}) that a node has to access. For 4 processors/ nodes, we need to use 4 variables to the least which proves the lower bound.

The processor could access the different variable for different execution from a execution perspective.

2 Problem 2

```
1  Two shared arrays:
2  NUM(arrays of n integers):$i_{th}$ entry contains waiting no for $p_i$
3  CHOOSE(Array of n boolean values):$i_{th}$ entry is 1 if $p_i$ is choosing number
4  Code for processor $p_i$:
5  Initially: NUM[i]=0; CHOOSE[i]=0
6  <ENTRY >: CHOOSE[i]:=1
7        NUM[i]= max(NUM[1], NUM[2],.....,NUM[n])+1
8        CHOOSE[i]=0
9        val=0
10       for $j \leftarrow 1$ to $n$ do
11           begin
12               wait until CHOOSE[j]=0
13               wait until (NUM[i]=0 or val>=2)
14               val+=1
15           end
16
17 <CS >
18 <EXIT >: NUM[i]:=0,val-=1
19 <REM >
```

Here, the lamport bakery algorithm will be modified to to accommodate 2 processor in critical section. in the original algorithm there was (NUM[j] , ID_j) ≥ (NUM[i] , ID_i) line in line 12 which has been removed here . For this any arbitrary number of process can go into critical section. But to make sure that only 2 processors can go into CS a variable named val has been introduced. Every processor when it enter the critical section will increase it by 1 which mean that there is a processor in the CS. If the value of val is 2 or more than this which mean there are already 2 processor in the CS. Hence it needs to wait. That is how at most 2 algorithm can be in the CS at most.

3 Problem 3

```
Two shared arrays:
NUM(arrays of n integers):$i_{th}$ entry contains waiting no for $p_i$
Code for processor $p_i$:
Initially: NUM[i]=0;
<ENTRY >:
    NUM[i]= max(NUM[1], NUM[2],.....,NUM[n])+1

    for $j \leftarrow 1$ to $n$ do
        begin

            wait until NUM[i]=0 or (NUM[j] , $ID_j$) $\geq$ (NUM[i] , $ID_i$)
        end

<CS >
<EXIT >: NUM[i]:=0
<REM >
```

Above is the algorithm without the choose variable. Say, we have 3 processors. We know line 6 is parallel which mean every processor will be executing this line simultaneously. Say two of our processors executed this line 6. In that case two of them will have the same number and it is 1. After going inside the for loop, it will find num[j]=0. In line 11 both the processors will get the access to get into <CS> as they have the same ticket number and there is not priority here. This will violate the mutual exclusion.

4 Problem 4

Answer. Lets say ρ=α_1,α_2,α_3 and ρ contains p_1,p_2.... p_i-only set (From lecture notes). From the lecture note, there are two configuration named C_i and C_j. It is also stated there that C_i and C_j have same memory state. From the diagram , we can notice that p_1,p_2.... p_i have not take any steps between C_i and C_j. This implies that $\sigma(C_i) \overset{p_l}{\sim} \sigma(C_j)$ where 1 ≤ l ≤i. From an earlier lemma we know that $p(C_i) \overset{p_l}{\sim} p(C_j)$ where 1 ≤ l ≤i. As the configurations are equivalent ,we can claim that $E \overset{p_l}{\sim} E'$ which respect to the set of processors.

From the diagram of lecture notes, we know that in exec(E,α_3) p_1 enters <CS> k+1 times. In the same way, p_1 also enters <CS> k+1 times where α_3 is p_1 only.However, This violates k-bounded waiting since p_{i+1}.... p_i are in <ES> in E' unlike E which is a contradiction. Here this E and E' is being used to create the situation of processors in different section which helps to establish the contradiction.

5 Problem 5

If a processor p_j stays in Entry section , it means that configuration is not quiescent configuration. If a configuration is not quiescent configuration it violate the definition of K-bounded waiting. If it is not quiescent configuration, there is not guarantee of no-deadlock.