Name: B.Anshuman

Roll No: 200259 (e.g. 170001) **Dept.**: EE (e.g. CSE)

Indian Institute of Technology Kanpur
CS637 Embedded and Cyber-Physical Systems
Homework Assignment 4
Deadline: November 11, 2022

Total: **40 marks**

1. Write the answers **neatly** in the given boxes.
2. You may discuss the solutions with the other students, but you have to write them in your own words.

Problem 1. (10 points) Work out Problem 3 in the Exercises of Chapter 12 in [LS15]:

This problem compares RM and EDF schedules. Consider two tasks with periods $p_1 = 2$ and $p_2 = 3$ and execution times $e_1 = e_2 = 1$. Assume that the deadline for each execution is the end of the period.

1. Give the RM schedule for this task set and find the processor utilization. How does this utilization compare to the Liu and Layland utilization bound of (12.2)?
2. Show that any increase in e_1 or e_2 makes the RM schedule infeasible. If you hold $e_1 = e_2 = 1$ and $p_2 = 3$ constant, is it possible to reduce p_1 below 2 and still get a feasible schedule? By how much? If you hold $e_1 = e_2 = 1$ and $p_1 = 2$ constant, is it possible to reduce p_2 below 3 and still get a feasible schedule? By how much?
3. Increase the execution time of task 2 to be $e_2 = 1.5$, and give an EDF schedule. Is it feasible? What is the processor utilization?

Assuming context-switching takes negligible time.
For the Rate Monotonic Scheduling, $\mu = \sum_{i=1}^{n} \frac{e_i}{p_i}$. Therefore $\mu = \frac{1}{2} + \frac{1}{3} = \frac{5}{6} = 0.833$. Also Utilization bound is given as $\mu \leq n(2^{1/n} - 1)$. Here $n = 2$, so $\mu \leq 0.828$ to guarantee feasibility. But the bound doesn't imply that utilisation above it would lead to non-feasibility, which can be seen from this example and the figure below.

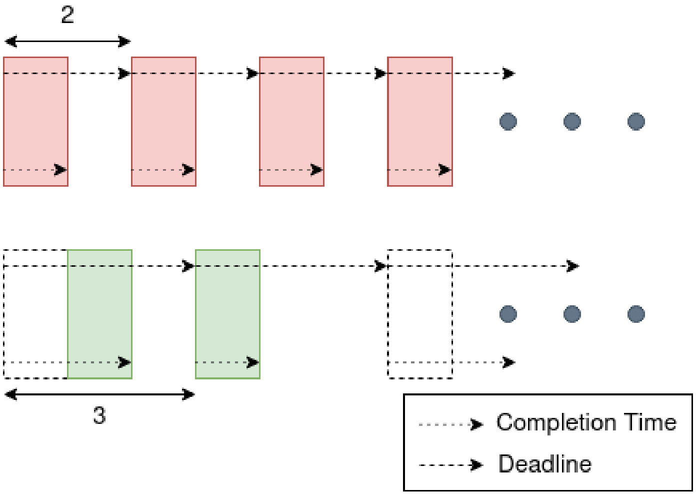


Figure 1: Two task scheduled with $p_1 = 2$, $p_2 = 3$, $e_1 = e_2 = 1$

Suppose e_1 is increased slightly. That would imply task2 not getting completed within deadline. Reasoning, observe the first 3 timestamps. Task2 needs to be completed within this interval. Task1 starts off, gets completed in 1 + Δt time, followed by Task2 executing for 1 - Δt time, after which the next event of task1 arrives. This leads to negative slack for that event of task2.
Suppose e_2 is increased slightly by Δt. This would give rise to the similar case as above, with the first task2 event not getting completed within deadline.

The minimum period p_1 (LEFT) and p_2 (RIGHT) that would still satisfy feasibility is:

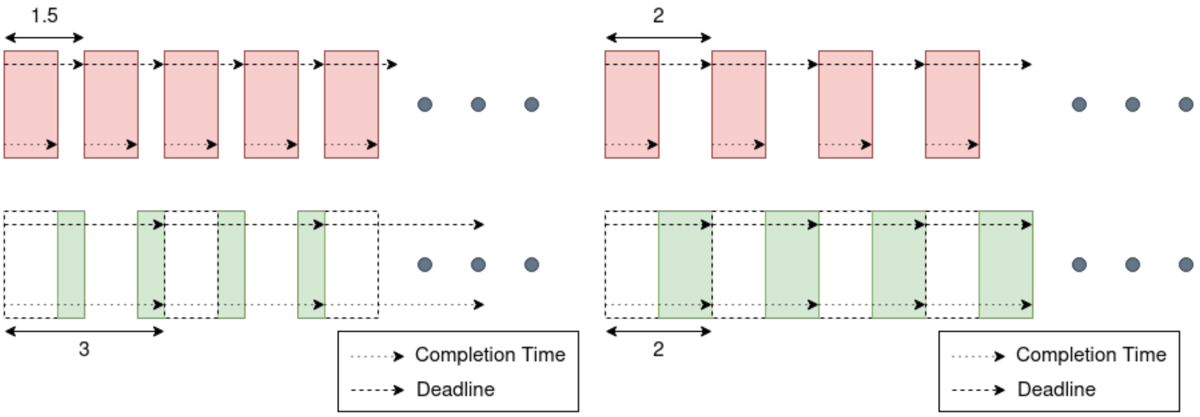


Figure 2:
(LEFT) Two task scheduled with $p_1 = 1.5$, $p_2 = 3$, $e_1 = e_2 = 1$
(RIGHT) Two task scheduled with $p_1 = 2$, $p_2 = 2$, $e_1 = e_2 = 1$

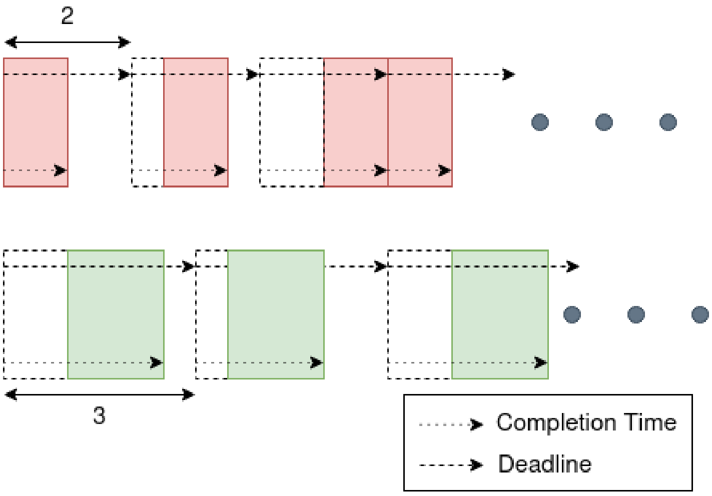
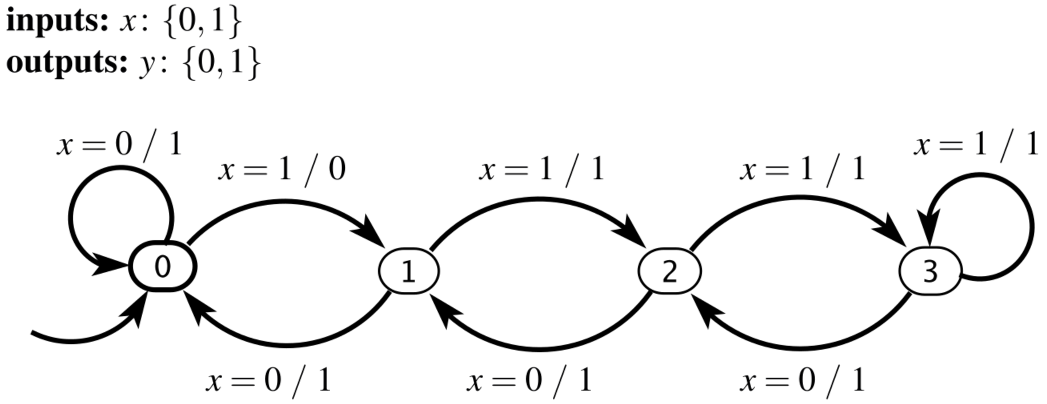


Figure 3: Two task scheduled with $p_1 = 2$, $p_2 = 3$, $e_1 = 1$, $e_2 = 1.5$ using EDF

The fabove is Earliest Deadline First Scheduling for 2 tasks. Clearly, processor utilisation is 1.

Problem 2. (10 points) Work out Problem 3 in the Exercises of Chapter 14 in [LS15]:

The state machine in the figure below has the property that it outputs at least one 1 between any two 0's. Construct a two-state nondeterministic state machine that simulates this one and preserves that property. Give the simulation relation. Are the machines *bisimilar*?

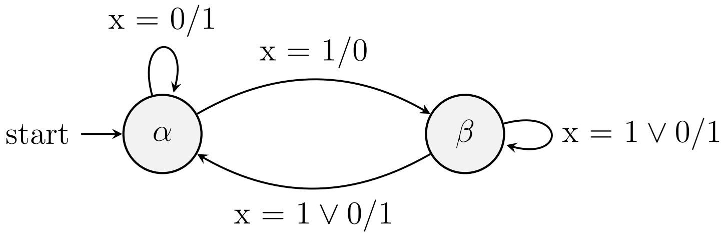


FSM 1: Machine that outputs at least one 1 between any two 0's

The above Machine M1 is defined by:

$$
\begin{aligned}
M_1 &= (S_1, I, O, U_1, init_1)\\
S_1 &= 0, 1, 2, 3\\
I &= 0, 1\\
O &= 0, 1\\
init_1 &= 0\\
U_1(s_i, x, s_o, y) &=\\
&\{(0,0,0,1),(0,1,1,0),(1,0,0,1),(1,1,2,1),(2,0,1,1),(2,1,3,1),(3,0,2,1),(3,1,3,1)\}
\end{aligned}
\tag{1}
$$

input: x: {0, 1}
output: y: {0, 1}



FSM 2: Non deterministic Finite Automaton simulating the behaviour of Figure 4

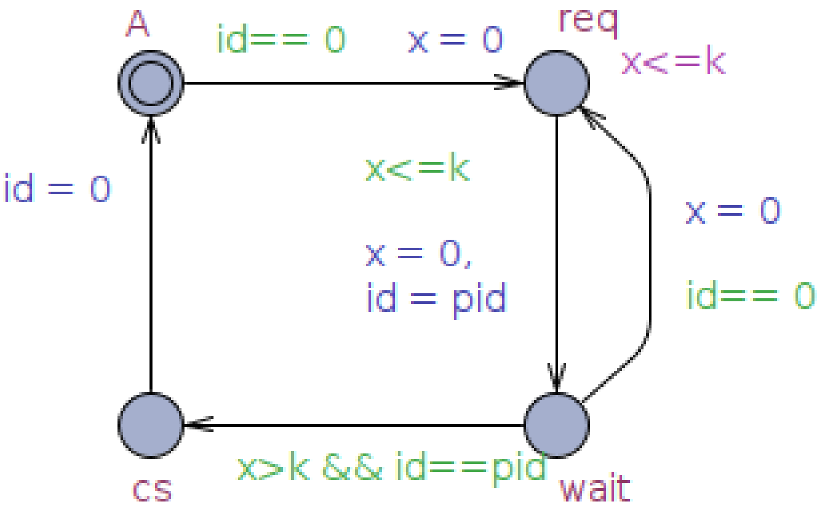
The simulation relation of FSM 1 and FSM 2, where 2 simulates 1, is given as

$$S = \{(0, \alpha), (123, \beta)\}$$

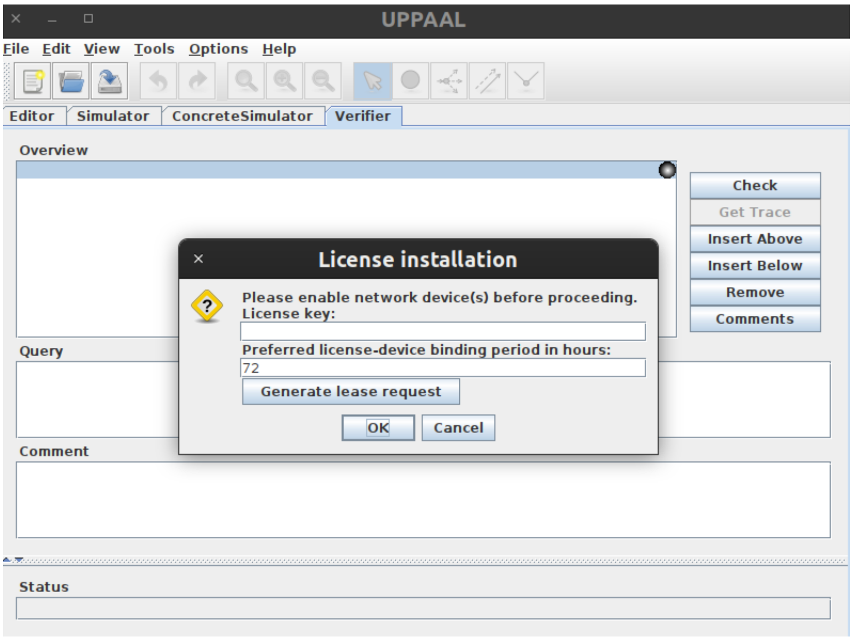
M_1 and M_2 are bi-similar if there exists a bi-simulation relation B. Now, we will try to find B by playing the modified matching game where either machine can move first. Both start at (0, α)
M_2 reacts as 1/0. M_1 matches by 1/0 to 1 :→ (1,β).
M_2 reacts as 0/1. M_1 matches by 0/1 to 0 :→ (0,β).
M_2 reacts as 1/1. M_1 cannot match the output 1. Stop.
Hence, there does not exist any bi-simulation relation B, as M_1 has failed to match M_2. Therefore, M_1 and M_2 are not bi-similar.

Problem 3. Verify Fischer's Mutual Exclusion protocol using UPPAAL. The example is available with UPPAAL tool. Measure the time required to verify the protocol for increasing number of participants in the protocol.

Post loading UPPAAL's demo *fischer.xml*, the block diagram showed up as:



This was followed by facing a problem in Verification tab:



Please enable network device(s) before proceeding.
License key:

Preferred license-device binding period in hours:
72

After which the application closed down due to non-availability of the liscence key.

I tried out FAQs Point 18 at UPPAAL Documentation, but this didn't work out.

Problem 4. (10 points) Work out Problem 2 in the Exercises of Chapter 16 in [LS15]:

```
void testFn(int *x, int flag) {
    while (flag != 1) {
        flag = 1;
        *x = flag;
    }
    if (*x > 0)
    *x += 2;
}
```

While answering the questions below, assume that x is not NULL.

1. Draw the control-flow graph of this program. Identify the basic blocks with unique IDs starting with 1. Note that we have added a dummy source node, numbered 0, to represent the entry to the function. For convenience, we have also introduced a dummy sink node, although this is not strictly required.
2. Is there a bound on the number of iterations of the while loop? Justify your answer.
3. How many total paths does this program have? How many of them are feasible, and why?
4. Write down the system of flow constraints, including any logical flow constraints, for the control-flow graph of this program.
5. Consider running this program uninterrupted on a platform with a data cache. Assume that the data pointed to by x is not present in the cache at the start of this function. For each read/write access to *x, argue whether it will be a cache hit or miss. Now, assume that *x is present in the cache at the start of this function. Identify the basic blocks whose execution time will be impacted by this modified assumption.

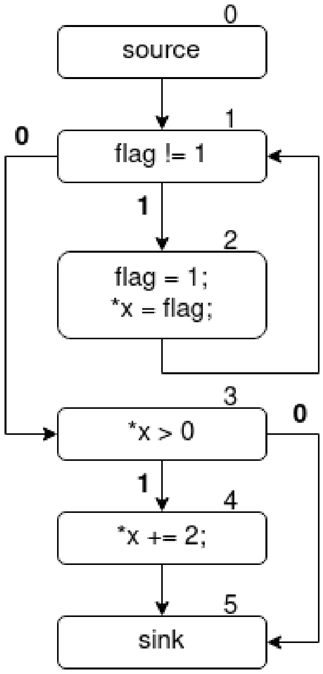


Figure 4: Control Flow Graph of *testFn*

The bound on while loop is 1, because of line 3, which makes the while condition false during the first iteration, if it isn't false already.

There seem to be a total of 4 paths, each of the following number represents the corresponding block:
Path 1: 0, 1, 2, 1, 3, 4, 5
Path 2: 0, 1, 2, 1, 3, 5
Path 3: 0, 1, 3, 4, 5
Path 4: 0, 1, 3, 5
Out of them, path 2 is infeasible because if the while block is executed, $*x = 1$, therefore block 4 must be executed.

The following are the flow constraints:

$$\begin{aligned} x_0 &= x_5 = 1 \\ x_1 &= d_{13} + d_{12} = d_{01} + d_{21} \\ x_2 &= d_{12} = d_{21} \\ x_3 &= d_{13} = d_{35} + d_{34} \\ x_4 &= d_{34} = d_{45} \\ x_5 &= d_{45} + d_{35} \end{aligned} \tag{2}$$

The following are Logical flow constraints:

$$\begin{gathered} x_2 \leq 1 \text{ (Loop Bound)} \\ d_{12} + d_{35} \leq 1 \text{ (Path Infeasibility)} \end{gathered} \tag{3}$$

There is read access in block 3 and write access in block 2 and 4.

1. *x is not present in the cache at the start of the function:
   If flag value is not equal to one, the write access in block 2 is always cache miss. Then, read access inblock 3 and write access in block 4 will always be cache hit.If flag value is one, then block 2 will not be executed.Therefore, read access in block 3 will be acache miss and write access in block 4 will be cache hit.

2. *x is present in the cache at the start of the function:
   Since, *x is already in the cache. Therefore, write acesses in block 2 and 4, and read access in block 3will always be cache hit.In this case, execution of block 2 and 3 will never be a cache miss.Therefore, the execution time ofblock 2 and 3 will be improved considerably by this modified assumption.