IUPUI

SCHOOL OF ENGINEERING AND TECHNOLOGY

A PURDUE UNIVERSITY SCHOOL

Indianapolis

ECE – 53801

Discrete Event Dynamic Systems

Programming Assignment-The cat and mouse problem

SUBMITTED BY:

Roshen Gajanan Borkar (University ID: 2000007125)

Sreeram Venkitachalam (University ID: 2000082784)

1. INTRODUCTION

A Petri net, also known as a place/transition (PT) net, is one of several mathematical modeling languages for the description of distributed systems. It is a class of discrete event dynamic system. [2] A petri net N can be represented as a weighted directed graph represented by N= (P, T, A, W) where P represents a set of finite places, T represents a set of transitions, A represents arcs from places to transitions and W is the weight function of the graph. The Fig. 1 represents an example of a petri net.

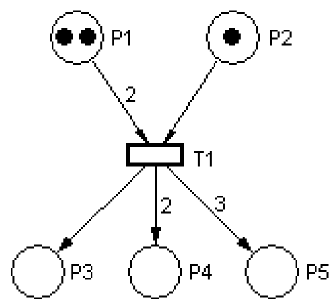

Fig .1: A petri-net example [1]

These graphs can be used to determine the reachable states of the system and to eliminate any duplicate states generated by the system.

Based on the application, there might occur a situation where a state might be reachable but not necessarily desirable. This creates constraints on the system for specific combinations. If these constraints can be represented as constraint equations using the petri net data, we can design petri net controllers by introducing additional places and arcs to enable/disable a transition satisfying the required constraints. This project is an example of one such situation.

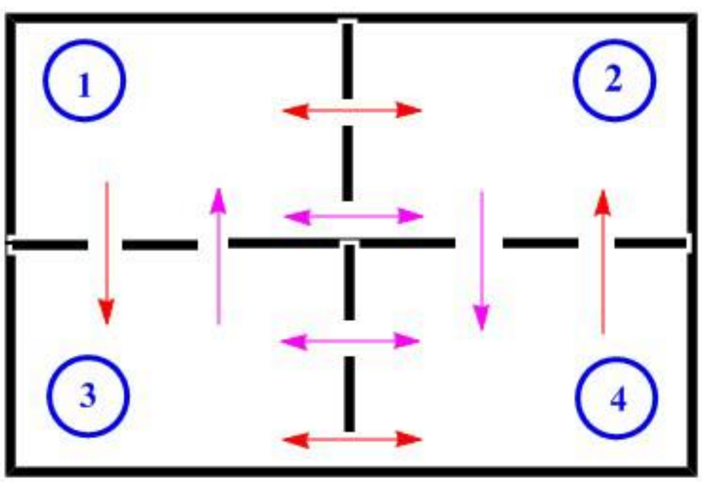

Fig .2: Layout of the house

2. Problem Statement

The project can be represented as an image in Fig .2. In this project we are dealing with the "Cat and Mouse" problem. The house contains four rooms 1, 2, 3, 4. The owner has a pet cat and a pet mouse. The cat and the mouse can move following the directions of the arrows in the house (Red for cat, Purple for mouse). Now assume that initially the cat is in Room 4 and the mouse is in Room 1 [3].

The following questions are to be understood and answered by the completion of the project.

- Build a Petri net model for the movement of the cat.
- Build a Petri net model for the movement of the mouse.
- Design a Petri net controller to guarantee that the cat and mouse can never be in the same room.
- Write a computer program (preferably in MATLAB or C) to calculate all possible reachable states of the Controlled Petri net. [3]

This can be achieved by first developing individual petri net graphs for the cat's and the mouse's movements and then combining them with slack places/variables which enables or disables the transitions in such a way that at no given state can you find the cat and the mouse in the same room. So, in this project we are first going develop the petri net graphs for the cat and the mouse movements by hand. Next, we are going to develop a controller so that the cat and the mouse are never in the same room. Finally, we are going to develop a code to realize all the reachable states of the controlled Petri net.

3. Results

3.1. Calculations for the petri net design

Based on the problem statement 2 independent petri-net models where generated, first representing the movement of the cat and the second representing the movement of the mouse. Fig .3 represents the petri net model for the movement of the cat. Fig .4 represents the petri net model for the movement of the mouse. This was done by hand based on the discussion of the controller of petri-nets in class. The calculated data for the petri net and its controller can be found on the Appendix. The Controller was calculated based on the constraints given by *Equations 1,2,3 and 4*.

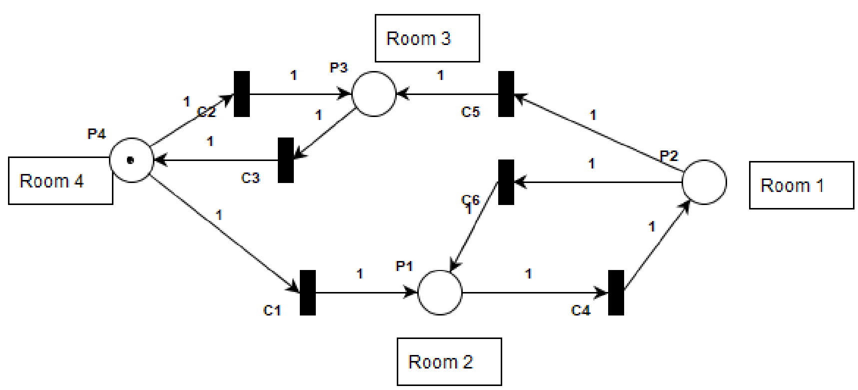

Fig .3: Petri net model for cat movement

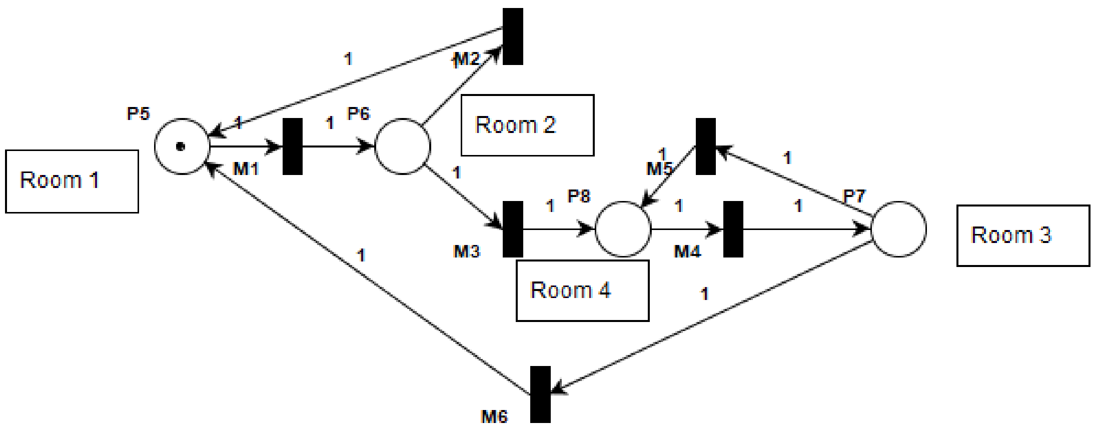

Fig .4: Petri net Model for Mouse Movement

M(P1) +M(P5) <= 1 ------------------------------→ *1*

M(P2) +M(P6) <= 1 ------------------------------→ *2*

M(P3) +M(P7) <= 1 ------------------------------→ *3*

M(P4) +M(P8) <= 1 ------------------------------→ *4*

Using the data of the slacks we can combine the two petri nets with its controller to get the combined petri nets which satisfy the constraints which were mentioned above. Fig .5 represents the combined movement of the cat and mouse which is controlled using 4 slacks/controllers.

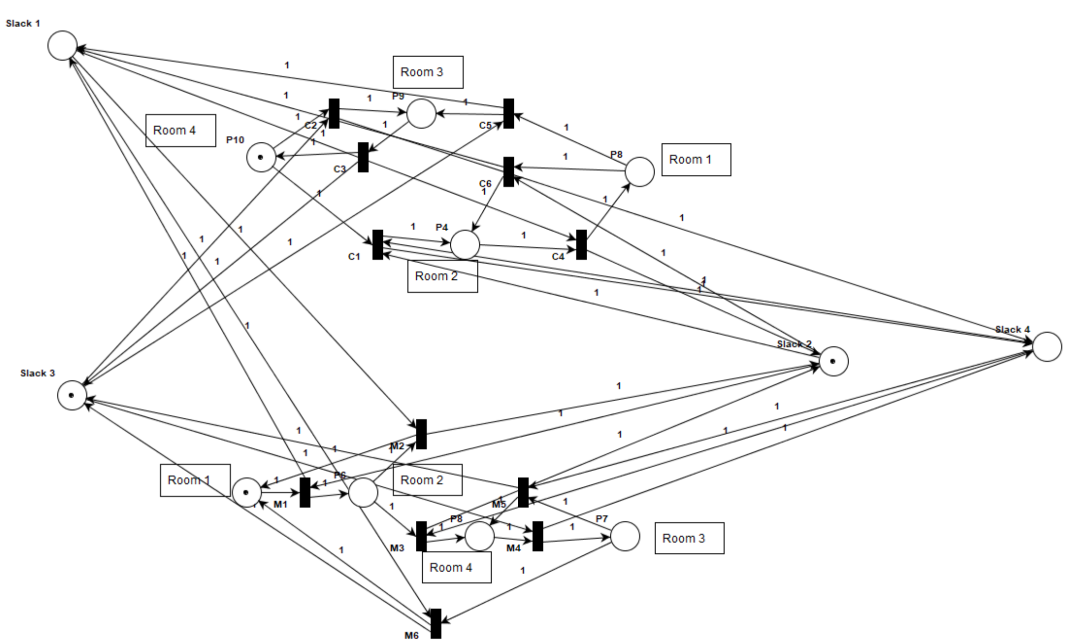

Fig .5: Combined movement of cat and mouse using controller

3.2. <u>Output from MATLAB</u>

The code is attached in the zip file attached along with report. The result was represented in the form of an image which is updated at every one second. The Transitions are represented as follows *[C1 C2 C3 C4 M1 M2 M3 M4 S1 S2 S3 S4]*. An example of one of the marking state of the new controller is shown in Fig .6. The Marking State represented in Fig .6 is [0 0 1 0 0 1 0 0 1 0 0 1]. The red square represents a cat (which is in room 3) and the blue square represents a mouse (which is in room 2). The Current Marking State was shown to the user in the form of print statements in the command window of MATLAB.

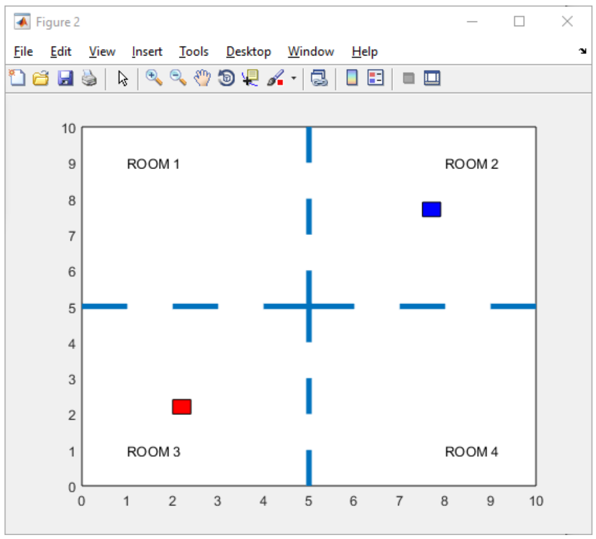

Fig .6: Cat and mouse representation in MATLAB

There can be different Marking States which may not be a reachable state as they maybe a duplicate of some previous reachable state. Thus, the petri net is said to have attained all its reachable states only when, firing the enabled transitions gives either a duplicate state or a terminal state. These marking states are displayed, and the enabled transitions are checked only if the Marking state is not a Duplicate state or a terminal state. However, these are still represented in the image diagram for the user to view. The final reachable states are shown in the command window for the user to view once the program is completed.

The step by step output is shown in a separate file, ECE53801_Output.dox. Below is the representation of the final output in the command window.

```
Marking state [0 0 0 1 1 0 0 0 0 1 1 0]Enabled transition C1 C2 M3
Marking state [0 0 0 1 0 1 0 0 1 0 1 0]Enabled transition C2 M4
Marking state [0 0 0 1 1 0 0 0 0 1 1 0]DUPLICATE NODE
Marking state [0 0 1 0 0 1 0 0 1 0 0 1]Enabled transition C3 M4
Marking state [0 0 1 0 0 0 0 1 1 1 0 0]Enabled transition
NO ENABLED STATES TERMINAL NODE


Marking state [0 0 1 0 1 0 0 0 0 1 0 1]Enabled transition C3 M3
Marking state [0 0 1 0 0 1 0 0 1 0 0 1]DUPLICATE NODE
Marking state [0 0 0 1 1 0 0 0 0 1 1 0]DUPLICATE NODE
Marking state [0 0 0 1 0 1 0 0 1 0 1 0]DUPLICATE NODE
Marking state [0 0 1 0 1 0 0 0 0 1 0 1]DUPLICATE NODE
Marking state [0 1 0 0 1 0 0 0 0 0 1 1]Enabled transition
NO ENABLED STATES TERMINAL NODE


The final reachable states in the order of firing:
     0     0     0     0     0     0
     0     0     0     0     0     1
     0     0     1     1     1     0
     1     1     0     0     0     0
     1     0     0     0     1     1
     0     1     1     0     0     0
     0     0     0     0     0     0
     0     0     0     1     0     0
     0     1     1     1     0     0
     1     0     0     1     1     0
     1     1     0     0     0     1
     0     0     1     0     1     1

>>
```

Doc. 1: Output representation in MATLAB Command window

4. REFERENCES

1. https://www.techfak.uni-bielefeld.de/~mchen/BioPNML/Intro/pnfaq_files/image003.gif
2. https://en.wikipedia.org/wiki/Petri_net
3. Problem Statement given
4. ECE53801_lecture17.pptx
5. https://www3.nd.edu/~pantsakl/Publications/155-Automaticab.pdf

1. APPENDIX

Calculation for the Cat and mouse Petri net models

Input Incident Matrix:

$\boldsymbol{B}^{-}$ = [0 0 0 0 1 1 0 0 0 0 0 0;
0 0 0 1 0 0 0 0 0 0 0 0;
0 0 1 0 0 0 0 0 0 0 0 0;
1 1 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 1 0 0 0 0 0;
0 0 0 0 0 0 0 1 1 0 0 0;
0 0 0 0 0 0 0 0 0 0 1 1;
0 0 0 0 0 0 0 0 0 1 0 0]

Output Incident Matrix:

$\boldsymbol{B}^{+}$ **=** [0 0 0 1 0 0 0 0 0 0 0 0;
1 0 0 0 0 1 0 0 0 0 0 0;
0 1 0 0 1 0 0 0 0 0 0 0;
0 0 1 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 1 0 0 0 1;
0 0 0 0 0 0 1 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 1 0 0;
0 0 0 0 0 0 0 0 1 0 1 0]

Incident Matrix:

B = [0 0 0 1 -1 -1 0 0 0 0 0 0;
1 0 0 -1 0 1 0 0 0 0 0 0;
0 1 -1 0 1 0 0 0 0 0 0 0;
-1 -1 1 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 -1 1 0 0 0 1;
0 0 0 0 0 0 1 -1 -1 0 0 0;
0 0 0 0 0 0 0 0 0 1 -1 -1;
0 0 0 0 0 0 0 0 1 -1 1 0]

Constraints

M(P1) +M(P5) <= 1
M(P2) +M(P6) <= 1
M(P3) +M(P7) <= 1
M(P4) +M(P8) <= 1

M(P1) +M(P5) +M(C1)= 1
M(P2) +M(P6) +M(C2) = 1
M(P3) +M(P7) +M(C3) = 1
M(P4) +M(P8) +M(C4) = 1

Calculations for the petri net controllers

The L matrix and the initial state is given by:

L = [1 0 0 0 1 0 0 0;
0 1 0 0 0 1 0 0;
0 0 1 0 0 0 1 0;
0 0 0 1 0 0 0 1]

M0 = [0; 0; 0; 1; 1; 0; 0; 0;]

Incident Matrix of the controller:

Bc = [0 0 0 -1 1 1 1 -1 0 0 0 -1;
-1 0 0 1 0 -1 -1 1 1 0 0 0;
0 -1 1 0 -1 0 0 0 0 -1 1 1;
1 1 -1 0 0 0 0 0 -1 1 -1 0]

The B vector and the initial state of the controller is given by:

B = [1; 1; 1; 1]

$\boldsymbol{M_c}$ = B-L*M0

$\boldsymbol{M_c}$ = [0; 1; 1; 0]

Combined controller and petri net

Incident Matrix of the controlled petri net

B = [0 0 0 1 -1 -1 0 0 0 0 0 0;
1 0 0 -1 0 1 0 0 0 0 0 0;
0 1 -1 0 1 0 0 0 0 0 0 0;
-1 -1 1 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 -1 1 0 0 0 1;
0 0 0 0 0 0 1 -1 -1 0 0 0;
0 0 0 0 0 0 0 0 0 1 -1 -1;
0 0 0 0 0 0 0 0 1 -1 1 0;
0 0 0 -1 1 1 1 -1 0 0 0 -1;
-1 0 0 1 0 -1 -1 1 1 0 0 0;
0 -1 1 0 -1 0 0 0 0 -1 1 1;
1 1 -1 0 0 0 0 0 -1 1 -1 0;]

Initial state of the combined petri net:

M0 = [0; 0; 0; 1; 1; 0; 0; 0; 0; 1; 1; 0]