Contents

ECE595 Project 1 Report

Niu,Wensen and Shen,Dan

November 14, 2015

1 About This Project

Firstly, we recall the problem and tasks of this project in this chapter. Instantaneously, we introduce what we did for this project.

1.1 Conditions

"Cat and Mouse" problem is a typical problem of mutual exclusion problem which can be solved by the application of Petri net. In this project, we consider that there are 4 Rooms in a house and the house layout is shown as Figure 1.

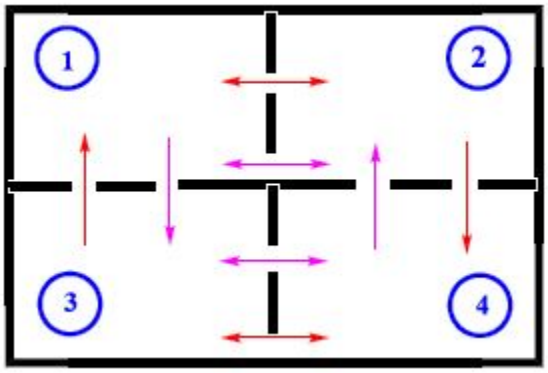

Figure 1 House layout

The cat can move following the direction of red arrows. And the mouse can move following the direction of purple arrows. Initially, the cat is in Room 4 and the mouse is in Room 1.

1.2 Tasks

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

2 Build Petri Net Model

To apply the Petri net on this kind "Cat and Mouse" problem, we built Petri net model for the movement. In this Chapter, we build Petri net for the movement of the cat and for the movement of the mouse and design a Petri net controller for this problem to guarantee that the cat and mouse can never be in the same room for that the initial state is that cat is in room 4 and mouse is in room 1.

2.1 Petri net model for the movement of the cat

A Petri net graph is weighted bipartite graph()For building Petri net model for the movement of the cat, we build place set, transition set and arcs set of the Petri net model.
Place set:

$$P_{cat} = \{P_{cat1},\ P_{cat2},\ P_{cat3},\ P_{cat4}\};$$

Transition set:

$$T_{cat} = \{t_{12C},\ t_{21C},\ t_{24},\ t_{31},\ t_{34C},\ t_{43C}\};$$

Arcs set: $A_{cat} =$

$$\begin{array}{cccc} \{(P_{cat1},\ t_{12C}), & (t_{12C},\ P_{cat2}), & (P_{cat2},\ t_{21C}), & (t_{21C},\ P_{cat1}), \\ (P_{cat2}, t_{24}), & (t_{24},\ P_{cat2}), & (P_{cat3},\ t_{31C}), & (t_{31C},\ P_{cat1}), \\ (P_{cat3},\ t_{34C}), & (t_{34C},\ P_{cat4}), & (P_{cat4},\ t_{43C}), & (t_{43C},\ P_{cat3})\}. \end{array}$$

Secondly, we calculate incident matrix of this Petri net. This Petri net has 4 places, 6 transitions. We can unique define input incident matrix B^-_{Cat}, captures are weights from places to transitions. The dimension of input incident matrix is 4×6.
The weights from places to transitions are:

	t_{12C}	t_{21C}	t_{24}	t_{31}	t_{34C}	t_{43C}
P_{cat1}	1	0	0	0	0	0
P_{cat2}	0	1	1	0	0	0
P_{cat3}	0	0	0	1	1	0
P_{cat4}	0	0	0	0	0	1

Input incident matrix is:

$$B^-_{Cat} = \begin{array}{c} \\ P_{cat1} \\ P_{cat2} \\ P_{cat3} \\ P_{cat4} \end{array} \begin{array}{c} \begin{array}{cccccc} t_{12C} & t_{21C} & t_{24} & t_{31} & t_{34C} & t_{43C} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Also, we can unique define output incident matrix B^+_{Cat}, captures are weights from transitions to places. The dimension of output incident matrix is 4×6.
The weights from transitions to places are:

	t_{12C}	t_{21C}	t_{24}	t_{31}	t_{34C}	t_{43C}
P_{cat1}	0	1	0	1	0	0
P_{cat2}	1	0	0	0	0	0
P_{cat3}	0	0	0	0	0	1
P_{cat4}	0	0	1	0	1	0

The output incident matrix is:

$$B^{+}_{Cat} = \begin{array}{c} \\ P_{cat1} \\ P_{cat2} \\ P_{cat3} \\ P_{cat4} \end{array} \begin{array}{c} \begin{array}{cccccc} t_{12C} & t_{21C} & t_{24} & t_{31} & t_{34C} & t_{43C} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

The incident matrix is:

$$B_{Cat} = B^{+}_{Cat} - B^{-}_{Cat} = \begin{array}{c} \\ P_{cat1} \\ P_{cat2} \\ P_{cat3} \\ P_{cat4} \end{array} \begin{array}{c} \begin{array}{cccccc} t_{12C} & t_{21C} & t_{24} & t_{31} & t_{34C} & t_{43C} \end{array} \\ \begin{bmatrix} -1 & 1 & 0 & 1 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & 1 & -1 \end{bmatrix} \end{array}$$

The Petri net graph for the Petri net model for movement of cat is shown as Figure 2.

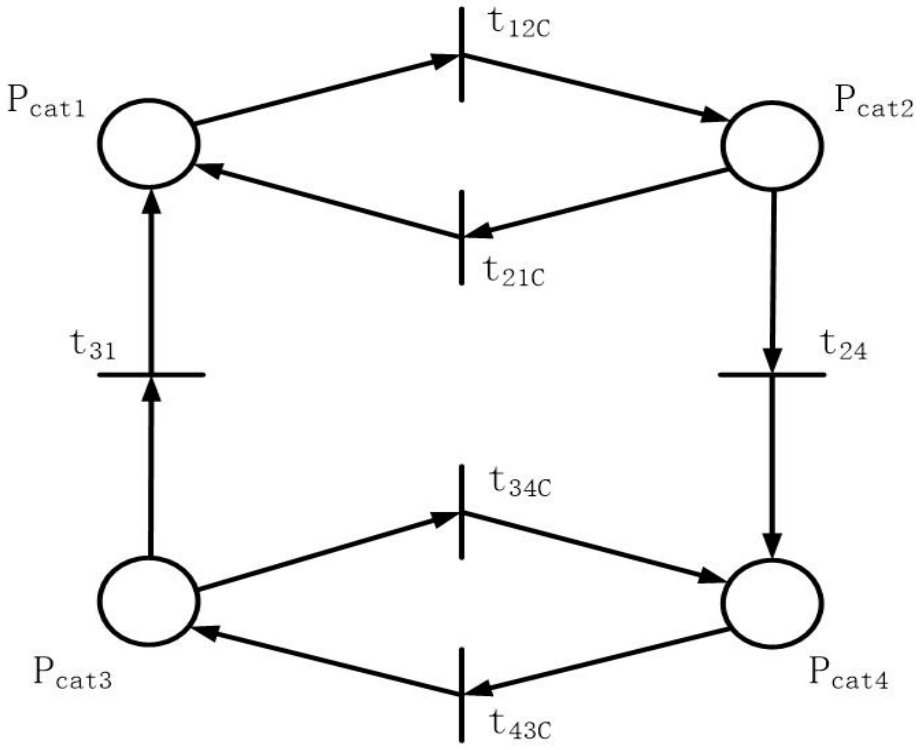

Figure 2 Petri net graph for movement of cat.

2.2 Petri net model for the movement of the mouse

Same as we built place set, transition set, and arcs set for building Petri net model for the movement of cat. For building the petri net model for movement of mouse, we also build place set, transition set and arcs set at first:
Place set:

$$P_{mouse} = \{P_{mouse1},\ P_{mouse2},\ P_{mouse3},\ P_{mouse4}\};$$

Transition set:

$$T_{mouse} = \{t_{12M},\ t_{21M},\ t_{13},\ t_{34M},\ t_{43M},\ t_{42}\};$$

Arcs set: $A_{mouse} =$

$$\begin{array}{cccc} \{(P_{mouse1},\ t_{12M}), & (t_{12M},\ P_{mouse2}), & (P_{mouse2},\ t_{21M}), & (t_{21M},\ P_{mouse1}), \\ (P_{mouse1}, t_{13}), & (t_{13},\ P_{mouse3}), & (P_{mouse3},\ t_{34M}), & (t_{34M},\ P_{mouse4}), \\ (P_{mouse4},\ t_{43M}), & (t_{43M},\ P_{mouse3}), & (P_{mouse4},\ t_{42M}), & (t_{42M},\ P_{mouse2})\}. \end{array}$$

And then, we calculate incident matrix of this Petri net model for movement of mouse. It also has 4 places and 6 transitions. The unique input incident matrix B^{-}_{Mouse}, captures are weights from places to transitions with 4×6 dimension. And the output incident matrix B^{+}_{Mouse}, captures are weights from transitions to places with 4×6 dimension.
The weights from places to transitions of the Petri net for movement of mouse are:

	t_{12M}	t_{21M}	t_{13}	t_{34M}	t_{43M}	t_{42}
P_{mouse1}	1	0	1	0	0	0
P_{mouse2}	0	1	0	0	0	0
P_{mouse3}	0	0	0	1	0	0
P_{mouse4}	0	0	0	0	1	1

Input incident matrix is:

$$B^{-}_{Mouse} = \begin{array}{c} \\ P_{mouse1} \\ P_{mouse2} \\ P_{mouse3} \\ P_{mouse4} \end{array} \begin{array}{c} \begin{array}{cccccc} t_{12M} & t_{21M} & t_{13} & t_{34M} & t_{43M} & t_{42} \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

The weights from transitions to places are:

	t_{12M}	t_{21M}	t_{13}	t_{34M}	t_{43M}	t_{42}
P_{mouse1}	0	1	0	0	0	0
P_{mouse2}	1	0	0	0	0	1
P_{mouse3}	0	0	1	0	1	0
P_{mouse4}	0	0	0	1	0	0

output incident matrix is:

$$B^{+}_{Mouse} = \begin{array}{c} \\ P_{mouse1} \\ P_{mouse2} \\ P_{mouse3} \\ P_{mouse4} \end{array} \begin{array}{c} \begin{array}{cccccc} t_{12M} & t_{21M} & t_{13} & t_{34M} & t_{43M} & t_{42} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

The incident matrix is:

$$B_{Mouse} = B^{+}_{Mouse} - B^{-}_{Mouse} = \begin{array}{c} \\ P_{mouse1} \\ P_{mouse2} \\ P_{mouse3} \\ P_{mouse4} \end{array} \begin{array}{c} \begin{array}{cccccc} t_{12M} & t_{21M} & t_{13} & t_{34M} & t_{43M} & t_{42} \end{array} \\ \begin{bmatrix} -1 & 1 & -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 \end{bmatrix} \end{array}$$

The Petri net graph for this Petri net model is shown as Figure 3.

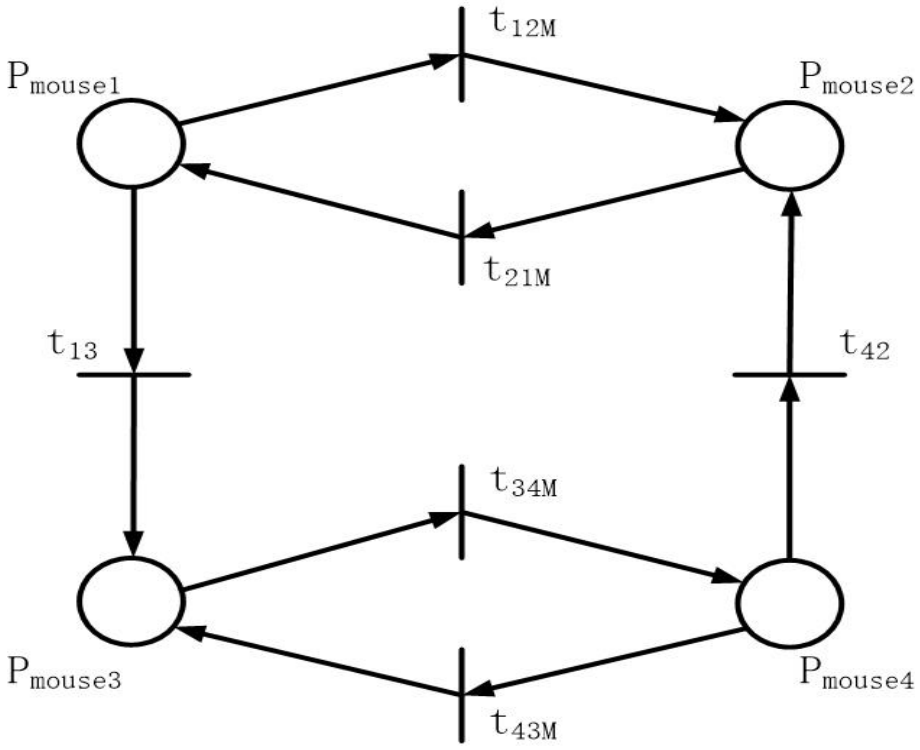

Figure 3 Petri net graph for movement of mouse.

2.3 Design a Petri net controller

In this part, we design a Petri net controller to guarantee that cat and mouse can never be in the same room with that initially the cat is in Room 4 and the mouse is in Room 1. We combine the Petri net model for the movement of cat and mouse to build a Petri net for this "Cat and Mouse" problem. After combined, the place set, transition set, and arc set of the new Petri net is:
Place set:

$$P = \{P_{cat1},\ P_{cat2},\ P_{cat3},\ P_{cat4},\ P_{mouse1},\ P_{mouse2},\ P_{mouse3},\ P_{mouse4}\};$$

Transition set:

$$T = \{t_{12C},\ t_{21C},\ t_{24},\ t_{31},\ t_{34C},\ t_{43C},\ t_{12M},\ t_{21M},\ t_{13},\ t_{34M},\ t_{43M},\ t_{42}\};$$

Arc set: $A =$

$$\begin{array}{cccc}
\{(P_{cat1},\ t_{12C}), & (t_{12C},\ P_{cat2}), & (P_{cat2},\ t_{21C}), & (t_{21C},\ P_{cat1}), \\
(P_{cat2}, t_{24}), & (t_{24},\ P_{cat2}), & (P_{cat3},\ t_{31C}), & (t_{31C},\ P_{cat1}), \\
(P_{cat3},\ t_{34C}), & (t_{34C},\ P_{cat4}), & (P_{cat4},\ t_{43C}), & (t_{43C},\ P_{cat3}) \\
(P_{mouse1},\ t_{12M}), & (t_{12M},\ P_{mouse2}), & (P_{mouse2},\ t_{21M}), & (t_{21M},\ P_{mouse1}), \\
(P_{mouse1}, t_{13}), & (t_{13},\ P_{mouse3}), & (P_{mouse3},\ t_{34M}), & (t_{34M},\ P_{mouse4}), \\
(P_{mouse4},\ t_{43M}), & (t_{43M},\ P_{mouse3}), & (P_{mouse4},\ t_{42M}), & (t_{42M},\ P_{mouse2})\}.
\end{array}$$

The input incident matrix is:

$$B^- = \begin{array}{c} \\ P_{cat1} \\ P_{cat2} \\ P_{cat3} \\ P_{cat4} \\ P_{mouse1} \\ P_{mouse2} \\ P_{mouse3} \\ P_{mouse4} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} t_{12C} & t_{21C} & t_{24} & t_{31} & t_{34C} & t_{43C} & t_{12C} & t_{21C} & t_{24} & t_{31} & t_{34C} & t_{43C} \end{array} \\ \left[\begin{array}{cccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{array}\right] \end{array}$$

The output incident matrix is:

$$B^+ = \begin{array}{c} \\ P_{cat1} \\ P_{cat2} \\ P_{cat3} \\ P_{cat4} \\ P_{mouse1} \\ P_{mouse2} \\ P_{mouse3} \\ P_{mouse4} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} t_{12C} & t_{21C} & t_{24} & t_{31} & t_{34C} & t_{43C} & t_{12C} & t_{21C} & t_{24} & t_{31} & t_{34C} & t_{43C} \end{array} \\ \left[\begin{array}{cccccccccccc}
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{array}\right] \end{array}$$

Because of that cat is in Room 4 and mouse is in Room 1 initially, the initial state of this Petri net is:

$$M_0^\top = \begin{array}{c} \begin{array}{cccccccc} P_{cat1} & P_{cat2} & P_{cat3} & P_{cat4} & P_{mouse1} & P_{mouse2} & P_{mouse3} & P_{mouse4} \end{array} \\ \left[\begin{array}{cccccccc} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}\right] \end{array}$$

we transfer the constraint to the form:

$$LM \leq b,$$

Where

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad M = \begin{bmatrix} M(P_{cat1}) \\ M(P_{cat2}) \\ M(P_{cat3}) \\ M(P_{cat4}) \\ M(P_{mouse1}) \\ M(P_{mouse2}) \\ M(P_{mouse3}) \\ M(P_{mouse4}) \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

According to the input incident matrix and output incident matrix, the incident matrix is:

$$B = B^+ - B^- = \begin{array}{c} \\ P_{cat1} \\ P_{cat2} \\ P_{cat3} \\ P_{cat4} \\ P_{mouse1} \\ P_{mouse2} \\ P_{mouse3} \\ P_{mouse4} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} t_{12C} & t_{21C} & t_{24} & t_{31} & t_{34C} & t_{43C} & t_{12C} & t_{21C} & t_{24} & t_{31} & t_{34C} & t_{43C} \end{array} \\ \begin{bmatrix} -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \end{bmatrix} \end{array}$$

For $LM \leq b$, we introduce a slack variable, M_c, the state of the controller($M_c \geq 0$). Such that

$$LM + M_c = b$$

$$\left[\begin{array}{c|c} L & I \end{array} \right] \begin{bmatrix} M \\ \hline M_c \end{bmatrix} = b$$

Use place invariant to design controller:
Based on the definition of place invariant, we have

$$X^\top B = 0 \;\Rightarrow\; LB + B_c = 0$$

$$\Rightarrow\; B_c = 0 - LB = -LB$$

$$B_c = \begin{array}{c} \\ P_{C1} \\ P_{C2} \\ P_{C3} \\ P_{C4} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} t_{12C} & t_{21C} & t_{24} & t_{31} & t_{34C} & t_{43C} & t_{12C} & t_{21C} & t_{24} & t_{31} & t_{34C} & t_{43C} \end{array} \\ \begin{bmatrix} 1 & -1 & 0 & -1 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & 1 \end{bmatrix} \end{array}$$

The initial state of the Petri net controller is obtained by

$$LM + M_c = b \;\Rightarrow\; LM_0 + M_{C0} = b$$

The initial state is:

$$M_{C0} = b - LM_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

The Petri net graph for "Cat and Mouse" problem with controller is shown in Figure 4.

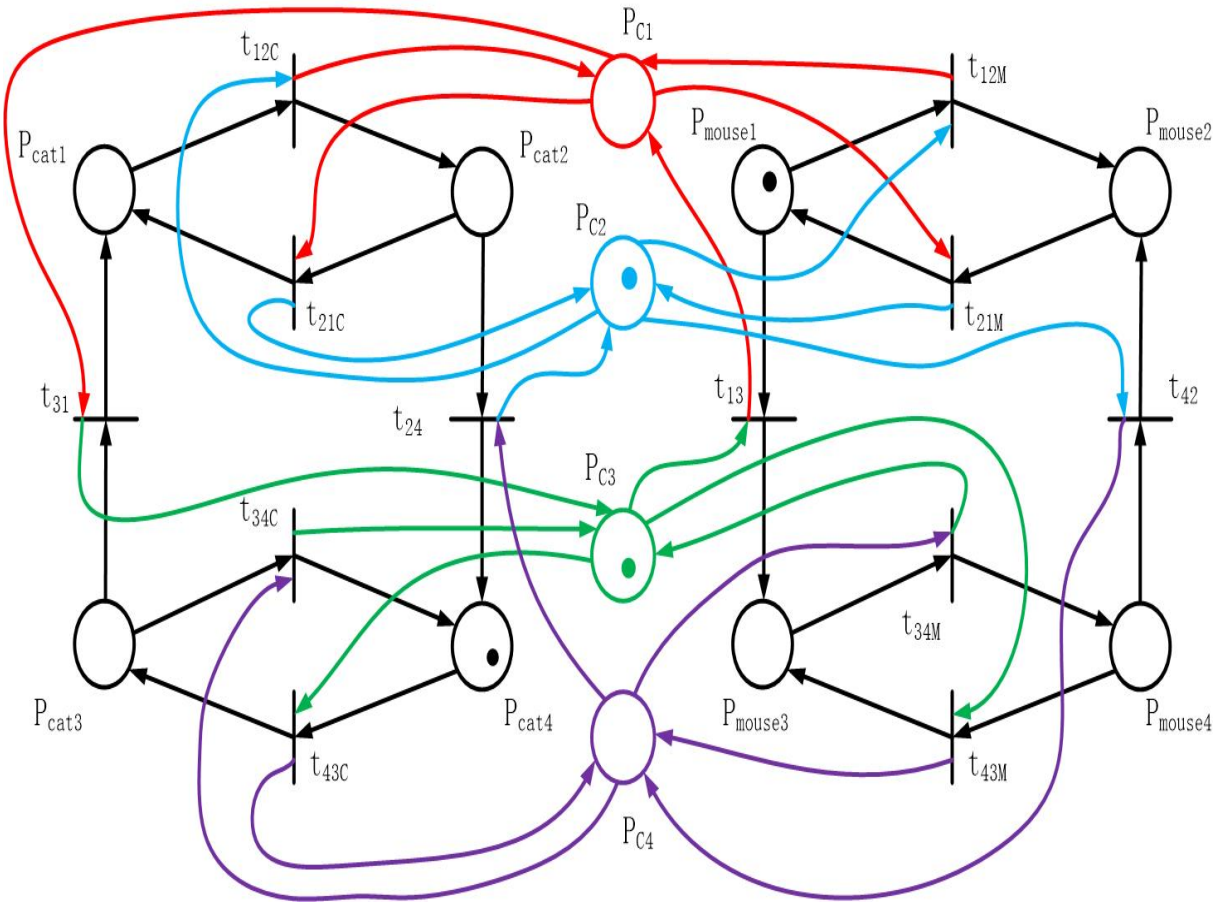

Figure 4 Petri net graph for the Petri net with controller.

3 Computer program

This chapter introduces a computer program designed for calculate the reachable states of a Petri net with controller. The program using MATLAB includes five functions to calculate and a m-file for input initial conditions that define the Petri net to be calculated.
We assume that a controller need to be designed for a marked Petri net. From the definition of marked Petri net, it is a five-tuple (P, T, A, ω, x) where (P, T, A, ω) is a Petri net graph and x is a marking of the set of places P; $x = [x_{p1}, x_{p2}, ..., x_{pn}] \in \mathbb{N}^n$ is the row vector associated with x. For programming, we treat a Petri net with constraint as a five-tuple (B^-, B^+, L, b, M_0) where B^- and B^+ is the input and output incident matrix, M_0 is the initial states of Petri net, and L with b form the constraint matrix of the Petri net which can be used to design controller as state-based controll.
For this "Cat and Mouse" problem,
The input incident matrix is:

$$B^- = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \\ P_8 \end{array} \begin{array}{c} \begin{array}{cccccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} & t_{11} & t_{12} \end{array} \\ \left[\begin{array}{cccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right] \end{array}$$

The output incident matrix is:

$$B^+ = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \\ P_8 \end{array} \begin{array}{c} \begin{array}{cccccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} & t_{11} & t_{12} \end{array} \\ \left[\begin{array}{cccccccccccc} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \end{array}$$

The constraint of this Petri net:

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad M = \begin{bmatrix} M(P_1) \\ M(P_2) \\ M(P_3) \\ M(P_4) \\ M(P_5) \\ M(P_6) \\ M(P_7) \\ M(P_8) \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Input all of these function as initial conditions into the section for inputting conditions of condition.m. Then it call functions to calculate initial states of controller and result in a "controlled Petri net" with satisfactory behavior as follow.

$$B_c = \begin{array}{c} \\ P_{C1} \\ P_{C2} \\ P_{C3} \\ P_{C4} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} & t_{11} & t_{12} \end{array} \\ \begin{bmatrix} 1 & -1 & 0 & -1 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & 1 \end{bmatrix} \end{array} M_{C0} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

The results of "controlled Petri net" show below:

$$B_{cpinput} = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \\ P_8 \\ P_{C1} \\ P_{C2} \\ P_{C3} \\ P_{C4} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} & t_{11} & t_{12} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

$$B_{cpoutput} = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \\ P_8 \\ P_{C1} \\ P_{C2} \\ P_{C3} \\ P_{C4} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} & t_{11} & t_{12} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

$$B_{cp} = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \\ P_8 \\ P_{C1} \\ P_{C2} \\ P_{C3} \\ P_{C4} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} & t_{11} & t_{12} \end{array} \\ \left[\begin{array}{cccccccccccc} -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \\ 1 & -1 & 0 & -1 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & 1 \end{array} \right] \end{array}$$

Initial state of controlled Petri net:

$$M_0 M_{C0}^{\top} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

And then the program calculate each state in Coverability Tree.

$$DT = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \\ 1 \\ 0 \\ 1 \\ 1 \\ 2 \\ 1 \\ 1 \end{bmatrix}, \quad M_{all} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array} \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}, \quad Tall = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array} \begin{bmatrix} 0 & 0 & 0 \\ 6 & 0 & 0 \\ 7 & 0 & 0 \\ 9 & 0 & 0 \\ 6 & 5 & 0 \\ 6 & 7 & 0 \\ 7 & 6 & 0 \\ 7 & 8 & 0 \\ 6 & 7 & 4 \\ 6 & 7 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

Each line of M_all is a state, the first line of M_all is the initial state of "Controlled Petri net".
When $DT_i = 1$, $M_all_i$is a duplicate node;When $DT_i = 2$, $M_all_i$is a terminal node;
For this "controlled Petri net", it has 6 reachable state:

$$R_{all} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 6 \\ 9 \end{array} \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Each line of R_{all} is a reachable state of this problem.

A CONDITION.M

```
clc
clearvars
%%
% input initial conditions for petri net in this section
% input incident matrix of Petri net model for movement of cat
Bcatinput=[1 0 0 0 0 0;0 1 1 0 0 0;0 0 0 1 1 0;0 0 0 0 0 1];
% output incident matrix of Petri net model for movement of cat
Bcatoutput=[0 1 0 1 0 0;1 0 0 0 0 0;0 0 0 0 0 1;0 0 1 0 1 0];
% input incident matrix of Petri net model for movement of mouse
Bmouseinput=[1 0 1 0 0 0;0 1 0 0 0 0;0 0 0 1 0 0;0 0 0 0 1 1];
%  output incident matrix of Petri net model for movement of
%  mouse
Bmouseoutput=[0 1 0 0 0 0;1 0 0 0 0 1;0 0 1 0 1 0;0 0 0 1 0 0];
% generating a zeros matrix which has same dimension with above
% matrix's dimension
zeros46=zeros(4,6);
% calculate input incident matrix of "Cat and Mouse" problem
Binput=[Bcatinput,zeros46;zeros46,Bmouseinput]
% calculate output incident matrix of "Cat and Mouse" problem
Boutput=[Bcatoutput,zeros46;zeros46,Bmouseoutput]
% generate a 4 by 4 identical matrix
eye4=eye(4);
% input coefficient matrix of constraint
L=[eye4,eye4]
% generate a 4 by 1 ones matrix
ones41=ones(4,1);
% input column vector of constraint
b=ones41
% input initial state of "Cat and mouse" Petri net
M0=[0 0 0 1 1 0 0 0]'

%%
% calculate initial states of controller.
% problem and reachable state of the "Controlled Petri net"
% Call functions, petricon.
petricon=petricon(Boutput,Binput,L,b,M0);
% parameters of controller
Bc=petricon.controller.Bc
Mco=petricon.controller.Mco
% incident matrix of "Controlled Petri net"
Bcpinput=petricon.controlledpetri.Bcpinput
Bcpoutput=petricon.controlledpetri.Bcpoutput;
BBco=petricon.controlledpetri.BBco
```

```
44 M0Mco=petricon.controlledpetri.M0Mco
45 % all states, all transition sequences, and the kind of state
46 Mall=petricon.transition.Mall%each row is a marking state
47 Tall=petricon.transition.Tall
48 % each row of Tall represents a transition sequence to the
49 % corresponding row in Mall.
50 DT=petricon.transition.DT
51 % each row of DT represents the kind of state to the
52 % corresponding row in Mall
53 %%
54 % This section generate a txt file for the results
55 %%%%%%%%%%show data%%%%%%%%%%
56 sizeDT=size(DT);
57 head=' node       each                        transition \n kinds
       states                        sequences';
58 fid=fopen('cat and mouse.txt','wt');
59 fprintf(fid,[head '\n']);
60 for i=1:sizeDT(1)
61     fprintf(fid,' %d         %d %d %d %d %d %d %d %d %d %d %d %d %d
          %d %d %d \n',...
62         DT(i,1),Mall(i,:),Tall(i,:));
63 end
64 fclose(fid);
```

B CONTROLLEDPETRI.M

```
1  function controlledpetri=controlledpetri(Boutput,Binput,Bc,Mco,M0)
2  %%
3  % CONTROLLEDPETRI Generate a "controlled petrinet"
4  %    controlledpetri=controlledpetri(Boutput,Binput,Bc,Mco)
5  %    calculate the BBco, Bcpinput, Bcpoutput, and M0Mco of a
6  %    "controlled Petri net"
7  %      Def:
8  %          BBco:The incident matrix of the "controlled Petri net"
9  %          Bcpinput: The input incident matrix of the "controlled
10 %          Petri net"
11 %          Bcpoutput: The output incident matrix of the
12 %          "controlled Petri net"
13 %          M0Mco: The initial state of the "controlled Petri net"
14 %
15 %    see also incident inicon petricon transition
16 %    Copyright Wensen CNboy @2015
17
18 %%
19 % Jusge number of input arguments for this function.
20 % If the number of input argument great than 5, program feedbacks
```

```
21 % the error message below.
22 if nargin>5
23     error('Too many input arguments')
24 end
25 % If the number of output argument less than 5, program feedbacks
26 % the error message below.
27 if nargin<5
28     error('Five input arguments should be given for this function')
29 end
30 %%
31 if nargin==5%If the number of input arguments equals to 5
32     sizeBoutput=size(Boutput);%check the size of Boutput
33     sizeBinput=size(Binput);%check the size of Binput
34     sizeBc=size(Bc);%check the size of Bc
35     boolean.A=sizeBoutput==sizeBinput;
36 %     Does size of Boutput equals to size of Binput?
37     boolean.B=sizeBc(2)==sizeBinput(2);
38 %     Does columns of Bc equals to columns of Binput?
39     if boolean.A==0
40 %         If size of Boutput does not equal to size of Binput,
41 %         program feedbacks the error message below.
42         error('Boutput and Binput should have same dimension')
43     elseif boolean.B==0
44 %         If columns of Bc does not equal to columns of Binput,
45 %         programs feedbacks the error message below.
46         error('Bc should has same columns with Binput and Boutput')
47     else
48         signBc=sign(Bc);%checking sign of each entry of Bc
49 %         Generating two zeros matrix with the dimension same as
50 %         Bc's dimension.
51         controlledpetri.Bcinput=zeros(sizeBc);
52         controlledpetri.Bcoutput=zeros(sizeBc);
53 %         A "for" loop for setting value for entries of Bcinput,
54 %         the input incident matrix of controller, and Bcoutput,
55 %         the output incident matrix of controller.
56         for i=1:sizeBc(1)%"for" loop for rows
57             for j=1:sizeBc(2)%"for" loop for columns
58                 if signBc(i,j)==-1
59                     controlledpetri.Bcinput(i,j)=controlledpetri.
                           Bcinput(i,j)-Bc(i,j);
60                     controlledpetri.Bcoutput(i,j)=controlledpetri.
                           Bcoutput(i,j);
61                 elseif signBc(i,j)==0
62                     controlledpetri.Bcinput(i,j)=controlledpetri.
                           Bcinput(i,j);
```

```
63                      controlledpetri.Bcoutput(i,j)=controlledpetri.
                            Bcoutput(i,j);
64                  elseif signBc(i,j)==1
65                      controlledpetri.Bcinput(i,j)=controlledpetri.
                            Bcinput(i,j);
66                      controlledpetri.Bcoutput(i,j)=controlledpetri.
                            Bcoutput(i,j)+Bc(i,j);
67                  end
68              end
69          end
70  %           Calculating Bcpinput, Bcpoutput, B, BBco, and M0Mco and
71  %           puting them into structure controlledpetri.
72          controlledpetri.Bcpinput=[Binput;controlledpetri.Bcinput];
73          controlledpetri.Bcpoutput=[Boutput;controlledpetri.Bcoutput];
74          controlledpetri.B=incident(Boutput,Binput);
75          controlledpetri.BBco=[controlledpetri.B;Bc];
76          controlledpetri.M0Mco=[M0;Mco];
77  %           For this function, it will return structure
78  %           controlledpetri to its main program.
79      end
80  end
```

C INCIDENT.M

```
1   function incident=incident(Boutput,Binput)
2   %%
3   % INCIDENT Incident matrix of petri net
4   %    incident=incident(Boutput,Binput) calculate the incident
5   %    matrix
6   %      Def:
7   %          B=Boutput-Binput
8   %          B represents the incident matrix
9   %          Binput represents the input incident matrix
10  %          Boutput represents the output incident matrix
11  %
12  %    see also controlledpetri inicon petricon transition
13  %    Copyright Wensen CNboy @2015
14
15  %%
16  % Checking number of input arguments for this function
17  % If the number of input arguments great than 2, program
18  % feedbacks the error message below
19  if nargin>2
20      error('Too many input arguments')
21  end
22  % If the number of input arguments less than 2, program feedbacks
```

```
% the error message below.
if nargin<2
    error('Two matrix should be given for this function')
end
%%
if nargin==2%If the number of input arguments equals to 2
    sizeBoutput=size(Boutput);%check the size of Boutput
    sizeBinput=size(Binput);%check the size of Binput
%     Checking the size of Boutput and the size of Binput.
%     If they are not equaled, program feedbacks the error
%     message below.
    if sizeBoutput~=sizeBinput
        error('Check dimensions of Binput and Boutput and keep them
            same')
    end
%     If the size of Boutput and Binput are equal, calculate the
%     incident matrix of the Petri net defined by Boutput and
%     Binput.
    if sizeBoutput==sizeBinput
%         For a Petri net, the incident matrix equals the output
%         incident matrix minus the input incident matrix.
%         For this function, it will return incident value to its
%         main program.
        incident=Boutput-Binput;
    end
end
```

D INICON.M

```
function inicon=inicon(L,b,M0,B)
%%
% INICON initial state and marking of Petri net controller
%     inicon=inicon(L,b,M0,B)
%         is used to calculate parameters of petri net controller
%         parameters:
%         inicon.Bc represents the incident matrix of petri net
%         controller
%         inicon.Mco represents the initial states of petri net
%         controller
%
%         Inputs:
%         L and b: L*M<=b;
%         M0: the initial state of petri net.
%         B: the incident matrix of petri net
%
%     see also controlledpetri incident petricon transition
```

```
18 %      Copyright Wensen CNboy @2015
19
20 %%
21 % Checking number of input arguments for this function
22 % If the number of input arguments great than 4, program
23 % feedbacks the error message below.
24 if nargin>4
25     error('Too many input arguments');
26 end
27 % If the number of input arguments less than 4, program feedbacks
28 % the error message below
29 if nargin<4
30     error('This function should have four inputs');
31 end
32 %%
33 if nargin==4%If the number of input arguments equals to 4
34 %     Saving size value of L, b, M0, and B to it corresponding
35 %     variable prepare for the logic check.
36     sizeL=size(L);%check the size of L
37     sizeb=size(b);%check the size of b
38     sizeM0=size(M0);%check the size of M0
39     sizeB=size(B);%check the size of B
40 %     Getting logic value for error checking
41     inicon.boolean.logisignA1=sizeM0(1)==sizeB(1);
42 %     Does rows of M0 equals to rows of B?
43     inicon.boolean.logisignA2=sizeM0(2)==1;
44 %     Does M0 is a column vector?
45     inicon.boolean.logisignA=inicon.boolean.logisignA1&&inicon.boolean.
         logisignA2;
46 %     A equals to "A1 and A2"
47     inicon.boolean.logisignB=sizeL(2)==sizeB(1);
48 %     Does columns of L equal to rows of B?
49     inicon.boolean.logisignC1=sizeb(1)==sizeL(1);
50 %     Does rows of b equals to rows of L?
51     inicon.boolean.logisignC2=sizeb(2)==sizeM0(2);
52 %     Does b is a column vector?
53     inicon.boolean.logisignC=inicon.boolean.logisignC1&&inicon.boolean.
         logisignC2;
54 %     C equals to "C1 and C2"
55 %     Checking logic values.
56     if inicon.boolean.logisignA==0
57 %         If rows of M0 does not equal to rows of B, program
58 %         feedbacks the error message below. If M0 is not a
59 %         column vector, program also feedbacks the error message
60 %         below.
```

```
61          error('Rows of column vector M0 should equal to as rows of B')
62      elseif inicon.boolean.logisignB==0
63  %           If columns of L does not equal to rows of B, program
64  %           feedbacks the error message below
65          error('Columns of coefficient matrix of constraint should equal
                to rows of B')
66      elseif inicon.boolean.logisignC==0
67  %           If rows of b does not equal to rows of L or b is not a
68  %           column vector, program feedbacks the error message
69  %           below.
70          error('b should be a scale or a column vector with rows same as
                rows of L')
71      else
72  %           According to state-based control of Petri nets,
73  %           calculate the incident matrix of controller and initial
74  %           state of controller to design a controller for a Petri
75  %           net.
76          inicon.Bc=-L*B;
77          inicon.Mco=b-L*M0;
78  %           For this function, it will return the inicon structure
79  %           for its main program. The inicon structure include the
80  %           incident matrix of controller that called inicon.Bc and
81  %           the initial state of controller that called inicon.Mco.
82
83      end
84  end
```

E PETRICON.M

```
1   function petricon=petricon(Boutput,Binput,L,b,M0)
2   %%
3   % PETRICON Petri net with controller
4   %   petricon=petricon(Boutput,Binput,L,b,M0)
5   %       is used to calculate parameters of petrinet with
6   %       controller Parameters:
7   %       petricon.petri.B represents the incident matrix
8   %       petricon.inicon is a structure that includes initial
9   %       states of controller
10  %       Boutput represents the output incident matrix
11  %
12  %   see also controlledpetri incident inicon transition
13  %   Copyright Wensen&Dan CNboys @2015
14
15  %%
16  % Checking number of input arguments.
17  if nargin~=5
```

```
18 %     If the number does not equal to 5, program feedbacks the
19 %     error message below.
20     error('Pleas check number of input arguments. It should be 5')
21 end
22 %%
23 if nargin==5%if the number of input arguments equals to 5
24     sizeBoutput=size(Boutput);%check the size of Boutput
25     sizeBinput=size(Binput);%check the size of Binput
26     sizeL=size(L);%check the size of L
27     sizeb=size(b);%check the size of b
28     sizeM0=size(M0);%check the size of M0
29 %     Getting logic value for error checking
30     petricon.boolean.logisignA1=sizeBinput(1)==sizeBoutput(1);
31 %     Does rows of Binput equals to rows of Boutput?
32     petricon.boolean.logisignA2=sizeBinput(2)==sizeBoutput(2);
33 %     Does columns of Binput equals to columns of Boutput?
34     petricon.boolean.logisignA=petricon.boolean.logisignA1&&petricon.
         boolean.logisignA2;
35 %     A equals to "A1 and A2"
36     petricon.boolean.logisignB=sizeL(2)==sizeBoutput(1);
37 %     Does columns of L equals to rows of Boutput?
38     petricon.boolean.logisignC=sizeL(1)==sizeb(1);
39 %     Does rows of L equals to rows of b?
40     petricon.boolean.logisignD=sizeM0(1)==sizeBoutput(1);
41 %     Does rows of M0 equals to rows of Boutput?
42 %     Checking logic values.
43     if petricon.boolean.logisignA==0
44 %         If size of Binput does not equals to size of Boutput,
45 %         program feedbacks the error message below.
46         error('Size of Binput should equals to size of Boutput');
47     elseif petricon.boolean.logisignB==0
48 %         If columns of L does not equal to rows of Boutput,
49 %         program feedbacks the error message below.
50         error('Columns of L should equals to rows of Boutput')
51     elseif petricon.boolean.logisignC==0
52 %         If rows of L does not equal to rows of b, program
53 %         feedbacks the error message below.
54         error('Rows of L should equals to rows of b')
55     elseif petricon.boolean.logisignD==0
56 %         If rows of M0 does not equal to rows of Boutput,
57 %         program feedbacks the error message below.
58         error('Please check initial state. It should be a column vector
             and has same row(s) with row(s) of Boutput')
59     else%If there is no error, continue to calculating
60 %         Calling incident function calculates incident matrix of
```

```
61 %           input incident matrix and output incident matrix of
62 %           Petri net model.
63           petricon.petri.B=incident(Boutput,Binput);
64 %           Calling inicon function calculates parameters of
65 %           controller for Petri net
66           petricon.controller=inicon(L,b,M0,petricon.petri.B);
67           petricon.controller.Bc;
68           petricon.controller.Mco;
69 %           Calling controlledpetri function calculates parameters
70 %           of "Controlled Petri net"
71           petricon.controlledpetri=controlledpetri(Boutput,Binput,
                petricon.controller.Bc,petricon.controller.Mco,M0);
72           petricon.controlledpetri.Bcpinput;
73           petricon.controlledpetri.Bcpoutput;
74           petricon.controlledpetri.BBco;
75           petricon.controlledpetri.M0Mco;
76 %           Calling transition function calculate Coverability Tree
77           petricon.transition=transition(petricon.controlledpetri.
                Bcpinput,petricon.controlledpetri.BBco,petricon.
                controlledpetri.M0Mco);
78       end
79   end
```

F TRANSITION.M

```
1   function transition=transition(Bcpinput,BBco,M0Mco)
2   %%
3   % TRANSITION Transition of a "Controlled Petri net"
4   %    transition=transition(Bcpinput,BBco,M0Mco) calculate all
5   %    states in Coverability Tree.
6   %      Def:
7   %           Bcpinput: input incident matrix of "Controlled Petri
8   %           net".
9   %           BBco: incident matrix of "Controlled Petri net".
10  %           M0Mco: initial state of "Controlled Petri net".
11  %
12  %    see also controlledpetri incident inicon petricon
13  %    Copyright Wensen CNboy @2015
14
15  %%
16  sizeBcpinput=size(Bcpinput);%checking the size of Bcpinput
17
18  % Generating a identity matrix with the dimension of columns of
19  % Bcpinput by the columns of Bcpinput
20  V=eye(sizeBcpinput(2));
21
```

```
22 % initialize M0McoT, which used to store state(s) for next
23 % time fire by enabled transition, to the initial state.
24 M0McoT=M0Mco';
25 sizeM0McoT=size(M0McoT);%checking the size of M0McoT
26 Mi=1;% initialize Mi to 1
27 % Mi is used as a pointer to point the row of Mall which help to
28 % store all state into correct place in Mall.
29 transition.Mall(Mi,:)=M0McoT;%store initial state into Mall
30
31 Ti=1;% initialize Ti to 1
32 % Ti is used as a pointer to point the row of Tall which help to
33 % store all transition sequence into correct place in Tall.
34 transition.Tall(Ti)=0;
35 % store 0 as there is no transition sequence to the initial state
36
37 %%
38 %DT(i)=0 represents the state is a fireable state.
39 %DT(i)=1 represents the state is a duplicate node.
40 %DT(i)=2 represents the state is a terminal node.
41 DTi=1;%initialize DTi to 1
42 % DTi is used as a pointer to point the row of DT which help to
43 % store kind of all state into correct place in DT.
44
45 % Flag is used as a flag to determine whether continue to next
46 % fire. If the flag equal to 0, program stop.
47
48 % Ddetect is used to store DT value in one transition, Which is
49 % used to determine whether to fire next transition for all state
50 % obtained in last time transition.
51
52 %%
53 % "for" loop is used to determin the initial value of
54 % transition.DT(DTi,:), Ddetect, and flag.
55 for i=1:sizeM0McoT(1)
56     for j=1:sizeBcpinput(2)
57         if M0McoT(i,:)'<Bcpinput(:,j)
58             transition.DT(DTi,:)=2;
59             Ddetect=1;
60             flag=0;
61         else
62             transition.DT(DTi,:)=0;
63             Ddetect=0;
64             flag=1;
65         end
66     end
```

```
67 end
68
69 % Dtistart is used to help store transition sequence to Tall.
70 % Initialize DTistart to 2, since we will not use it at very
71 % begin of this program.
72 DTistart=2;
73 %%
74 while flag~=0
75 %     initialize Tistart to DTistart in each time of loop
76     Tistart=DTistart;
77 %     check the size of M0McoT for each time of loop
78     sizeM0McoT=size(M0McoT);
79 %     n helps to store M to M0Mco, initialize it to 1 at begining
80 %     of each time of loop.
81     n=1;
82 % start a "for" loop from 1 to the rows of M0McoT
83     for i=1:sizeM0McoT(1)
84 % Use Ddetect to determine whether the state can be fire by any
85 % transition of the Petri net.
86         if Ddetect(i)==0
87 %             So the state can be fired by some transition
88             for j=1:sizeBcpinput(2)
89 %                 Start a "for" loop from 1 to the columns of
90 %                 Bcpinput, the input incident matrix of
91 %                 "Controlled Petri net.
92                 if M0McoT(i,:)'>=Bcpinput(:,j)
93 %                     If the transpose of state i can be fired by
94 %                     the j transition,
95                     sizeTallold=size(transition.Tall);
96                     Ti=Ti+1;
97                     sizeTall=size(transition.Tall);
98 %                     use flag to help storing transition
99 %                     sequence to Tall.
100                    if flag>=2
101                        for f=1:flag-1
102                            transition.Tall(Ti,f)=transition.Tall(
                                Tistart,f);
103                        end
104                    end
105                    transition.Tall(Ti,flag)=j;
106 %                     iterate DTistart
107                    sizeTallnew=size(transition.Tall);
108                    if sizeTallnew(2)==sizeTallold(2)+1
109                        DTistart=Ti;
110                    end
```

```
111 %                       storing M to Mall
112                        M=M0McoT(i,:)'+BBco*V(:,j);
113                        Mi=Mi+1;
114                        transition.Mall(Mi,:)=M';
115
116                        sizeMall=size(transition.Mall);
117 %                       iterate DTi
118                        DTi=DTi+1;
119                        transition.DT(DTi,:)=0;
120 %                       storing the value that represents the state
121 %                       kind to DT.
122                        for k=1:sizeMall(1)-1
123 %                          Checking whethe the state is a
124 %                          duplicate state. If it is a duplicate
125 %                          state, writing 1 to the corresponding
126 %                          place in DT.
127                            if M==transition.Mall(k,:)
128                                transition.DT(DTi,:)=1;
129                            end
130                        end
131 %                       initialization Mdetect to a zeros matrix,
132 %                       which is used to check whether the state is
133 %                       a terminal state. If it is a terminal
134 %                       state, writing 2 to the corresponding place
135 %                       in DT.
136                        Mdetect=zeros(1,sizeBcpinput(2));
137                        for m=1:sizeBcpinput(2)%"for" loop
138 %                          If M less than each column of Bcpinput,
139 %                          it is a terminal state. each m of M
140 %                          will be written to 1
141                            if M>=Bcpinput(:,m)
142                                Mdetect(1,m)=0;
143                            else
144                                Mdetect(1,m)=1;
145                            end
146                        end
147                        if Mdetect>=1
148                            transition.DT(DTi)=2;
149                        else
150                            transition.DT(DTi)=transition.DT(DTi);
151                        end
152                        M0Mco(:,n)=M;%store M to M0Mco
153                        n=n+1;
154                    end
155                end
```

```
156             end
157             Tistart=Tistart+1;
158         end
159         M0McoT=[];
160         M0McoT=M0Mco';
161         M0Mco=[];
162         Ddetect=[];
163         Ddetect=transition.DT(DTistart:DTi,:);
164         if Ddetect>=1
165             flag=0;
166         else
167             sizeTall=size(transition.Tall);
168             flag=flag+1;
169         end
170 end
```

G RESULT.TXT

cat and mouse.txt

	node kinds	each states	transition sequences
1			
2			
3	0	0 0 0 1 1 0 0 0 0 1 1 0	0 0 0
4	0	0 0 1 0 1 0 0 0 0 1 0 1	6 0 0
5	0	0 0 0 1 0 1 0 0 1 0 1 0	7 0 0
6	2	0 0 0 1 0 0 1 0 1 1 0 0	9 0 0
7	1	0 0 0 1 1 0 0 0 0 1 1 0	6 5 0
8	0	0 0 1 0 0 1 0 0 1 0 0 1	6 7 0
9	1	0 0 1 0 0 1 0 0 1 0 0 1	7 6 0
10	1	0 0 0 1 1 0 0 0 0 1 1 0	7 8 0
11	2	1 0 0 0 0 1 0 0 0 0 1 1	6 7 4
12	1	0 0 0 1 0 1 0 0 1 0 1 0	6 7 5
13	1	0 0 1 0 1 0 0 0 0 1 0 1	6 7 8

Figure 5 Result of Program.