

Main Ideas

For this project the paper titled, *Feedback Control of Petri Nets Based on Place Invariants* was read. The paper presented a few examples of how to take a Petri Net model of a system and develop a controller to dictate the performance of said system. This topics introduced in the paper were shown to be especially useful to industrial processes, and various constraint methods were introduced.

In what follows, the ideas of that paper are extended with new examples, and applications. In the first example a new petri net and corresponding controller are developed as a sequel to the drunken walk problem. In this example the drunkard's path is modeled with a petri net instead of a makarov chain. The main purpose of this example is to convey the information contained in *Feedback Control of Petri Nets Based on Place Invariants*.

The second example is an extension of the Cat & Mouse problem. This example shows how multiple controllers can be placed on the same Petri Net model. It also illustrates how to make a controller more robust.

Example 1: Recovering Drunkard

Uncontrolled Petri Net

In discrete event system literature, there exists a well-known problem known as the Drunkard's walk. This problems exists as a Markov Chain, which is based on probabilities. In the Drunkards' walk, a drunk person is free to move any direction they choose and enter as many bars as they like until they arrive at either their house, or a detox center.

In this new example, the drunk man has moved to a new town, and is trying to get his life together, and is a recovering drunkard. In the new town he gets a great new job, girlfriend, and things are looking up. The only bad thing is that as the man typically moves between home, work, and his girlfriend's house he has numerous opportunities to enter bars along the way, especially after a long stressful day at work.

So in this new model presented below, the man is leaving work and has the opportunity to either go to this house, his girlfriend's, or two either of two bars along the way. In this example, a feedback Petri Net controller is to be developed to keep the man on the wagon, and out of the bars.

The new model is based as close as possible to the Drunkard's Walk Markov chain, but some modifications did need to be made such as adding dummy states, and transitions at the two end states, as self-loops are not good in a Petri Net, and paths from the starting place to the final places are added to accommodate the controller design.

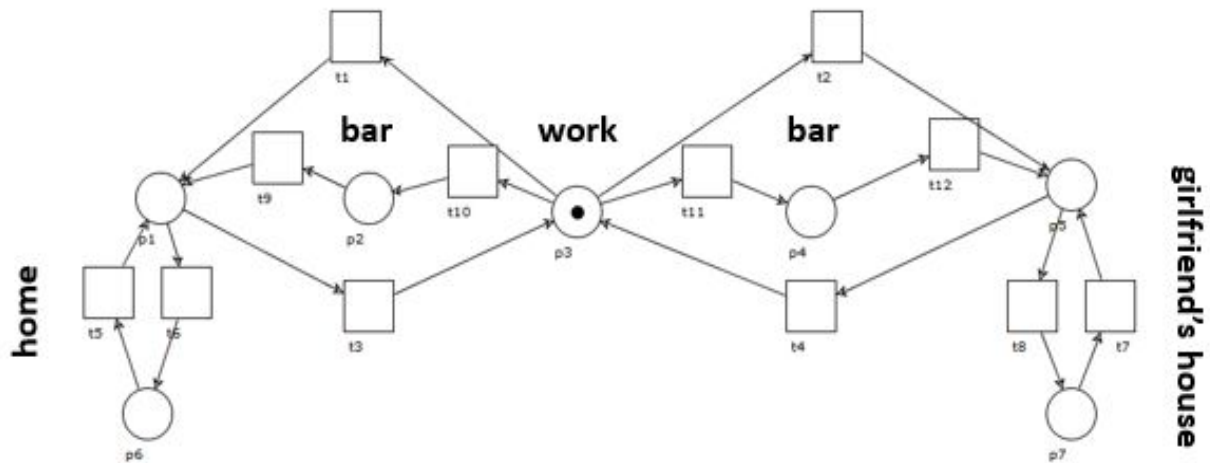


Figure 1- Petri Net Model for recovering drunkard

The new model is faulty in the sense that the recovering man is free to move to places 2 and 4, which are bars. In the figure below, the highlighted green boxes show which transitions are enabled at a given instant. As can be seen in the figure, the transition leading to the bars are enabled, which we would like to design a controller to make them not be enabled to prevent the man from reaching those places in the net.

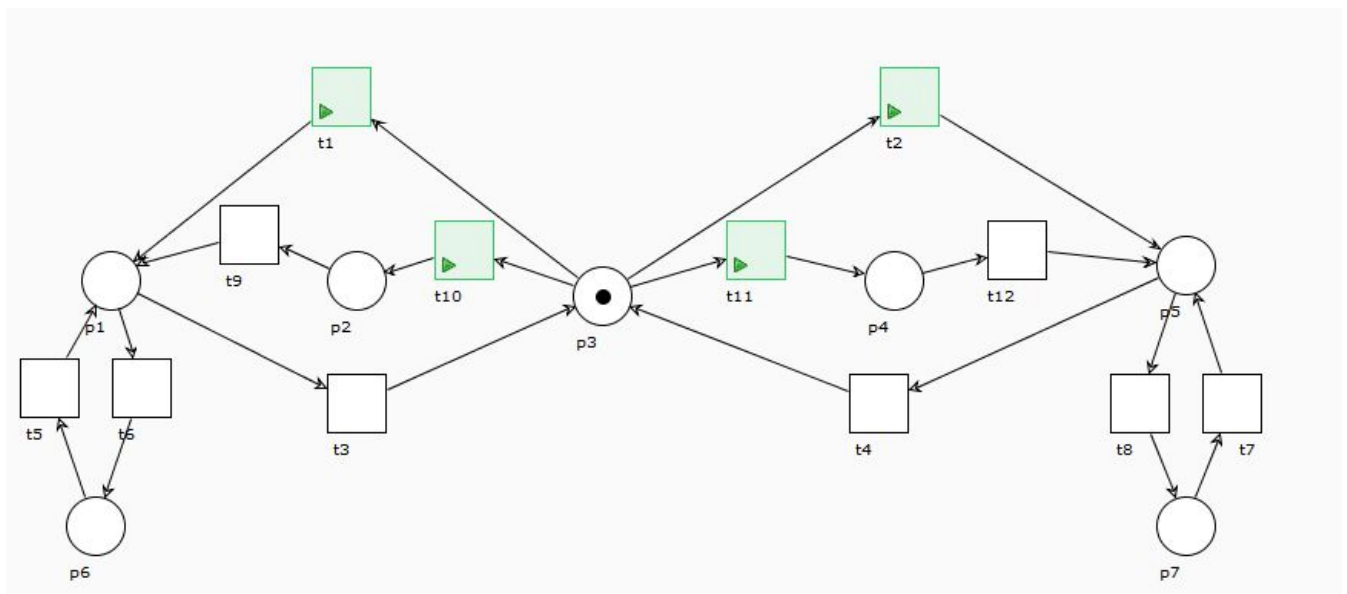


Figure 2- Uncontrolled system - allowed to enter unwanted places p2 and p3

Controller Design Process

The controller design process begins with determining the input and out incident matrices for the uncontrolled system. The input matrix, B_p , and the output matrix, B_m are copied from Matlab and listed below

```
% Bp =  
%  
%      1      0      0      0      1      0      0      0      1      0      0      0  
%      0      0      0      0      0      0      0      0      0      1      0      0  
%      0      0      1      1      0      0      0      0      0      0      0      0  
%      0      0      0      0      0      0      0      0      0      0      1      0  
%      0      1      0      0      0      0      1      0      0      0      0      1  
%      0      0      0      0      0      1      0      0      0      0      0      0  
%      0      0      0      0      0      0      0      1      0      0      0      0
```

```
% Bm =  
%  
%      0      0      1      0      0      1      0      0      0      0      0      0  
%      0      0      0      0      0      0      0      0      1      0      0      0  
%      1      1      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      1      0      0      0      1      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
```

For designing the controller, the difference of these two matrices is found, which gives us the incidence matrix for this uncontrolled system

```
% B =  
%  
%      1      0     -1      0      1     -1      0      0      1      0      0      0  
%      0      0      0      0      0      0      0      0     -1      1      0      0  
%     -1     -1      1      1      0      0      0      0      0     -1     -1      0  
%      0      0      0      0      0      0      0      0      0      0      1     -1  
%      0      1      0     -1      0      0      1     -1      0      0      0      1  
%      0      0      0      0     -1      1      0      0      0      0      0      0  
%      0      0      0      0      0      0     -1      1      0      0      0      0
```

The initial marking of the uncontrolled system

```
% Mo.' =  
%  
%      0      0      1      0      0      0      0
```

Based on the desired performance of the Petri Net, the constraint for this system is determined to be

$$\sum LM(p_i) = b \rightarrow M(p_2) + M(p_4) = 0 \rightarrow L = [0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0], \text{ and } b = 0$$

As there is one constraint on the system, the controller will contain one new place. The controller incidence matrix and initial markings for the controller are determined as

$$B_C = -L^*B$$

```
% Bc =
%
%      0      0      0      0      0      0      0      0      1     -1     -1      1
```

$$M_{Co} = b - L^*M_o$$

```
% Mco =
%
%      0
```

From the values just determined, the new controller place will add arcs from the controller place, pc1 to transitions t10 and t11. The new controller place will also have arcs from t9 and t12. The Mco shows there will be no tokens placed within the new controller place.

The figure below shows the new controller Petri Net. If we step through the transitions that are enabled, it will be easy to see that the bars in p2 and p4 will never be reached. It is also easy to see that the goal was obtained, as with the new control place adding arcs to t10 and t11 the transtions permitting movement to the bars will never be enabled as all input places do not contain tokens.

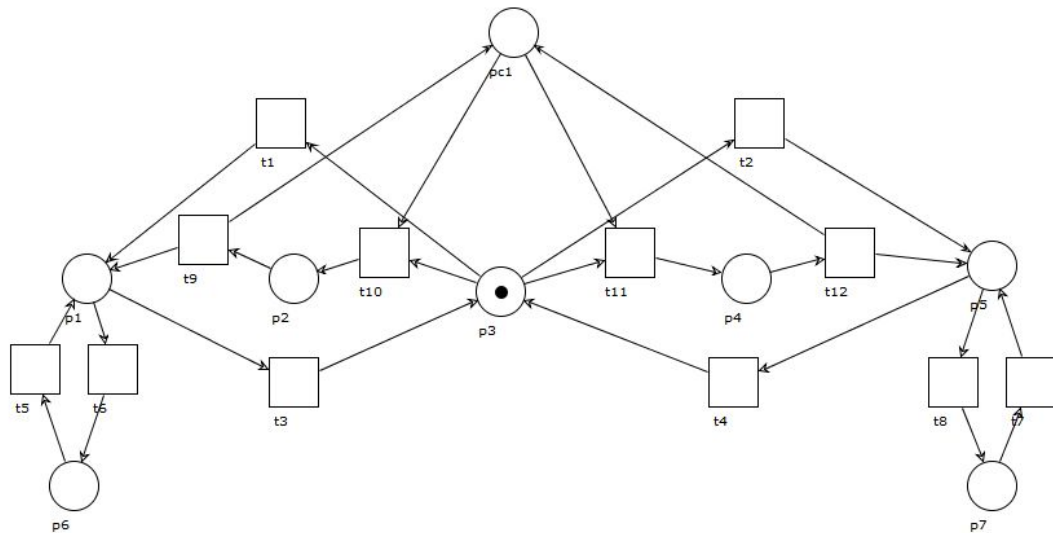


Figure 3 - Controlled Petri Net

Uncontrolled Petri Net

$$M0=[1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0].'$$

Controller 1: Keeping the Animals Separate

The first controller is designed to prevent the cat and the mouse being in the same room. Since a similar controller was designed in project 1 and the process is shown in example 1, the design process will not be reiterated here. The controller matrix and the resulting petri net are below.

Bc1=

$$\begin{bmatrix} 1 & -1 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & -1 & 1 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 1 & 0 & -1 & 1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & -1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \end{bmatrix}$$

$$MC10=[0\ 1\ 1\ 0\ 1\ 1].'$$

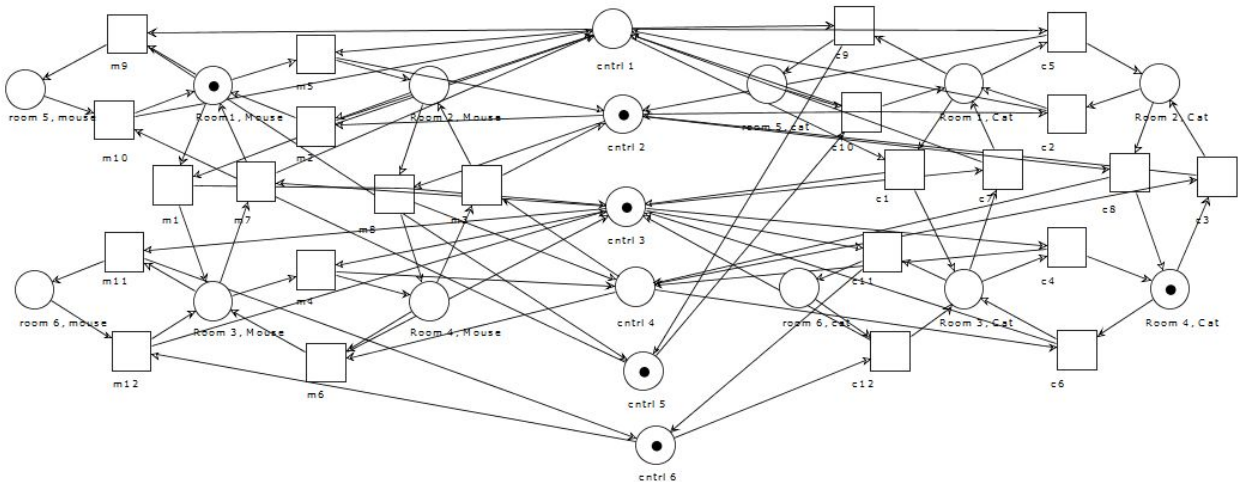


Figure 5: Petri Net with the First Controller

Controller 2: Protecting the Food

A controller needs to be made to prevent the pets from eating the wrong food. Room 5 contains the mouse's food and room 6 contains the cat's food. The new requirement can be stated as $M(\text{place6_mouse}) \leq 0$ and $M(\text{place5_cat}) \leq 0$. The resultant incident matrix is shown below. It is used with the first controller on the petri net in figure 6.

Bc2=

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \end{bmatrix}$$

MC20=[0 0].'

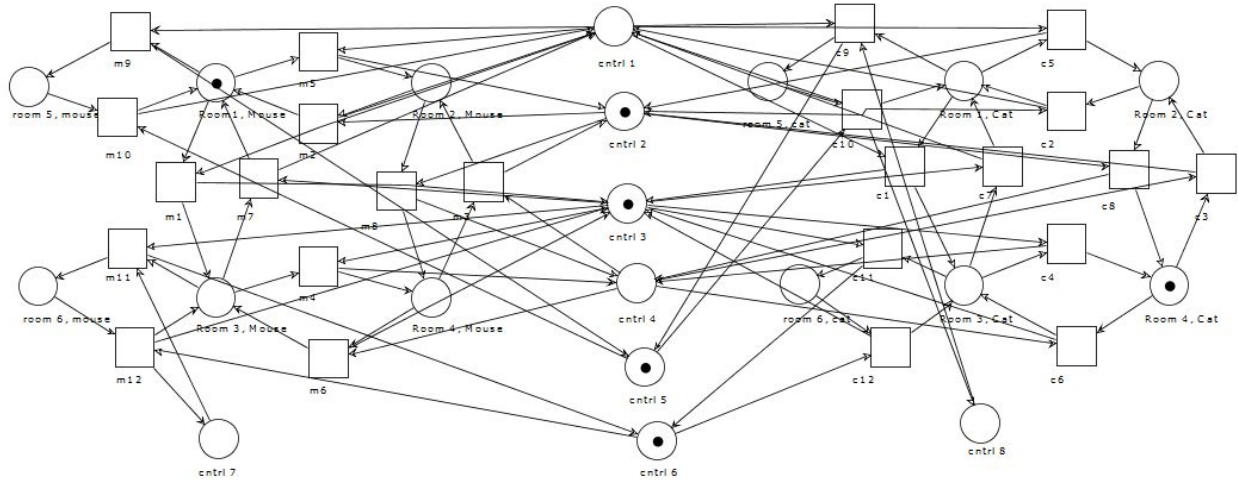


Figure 6: Petri Net with the First and Second Controller

Controller 3: Separate and Redundant Control

The third controller is a backup for the second controller. Since the owner of the cat and mouse values both pets they want to make sure that no problems arise. In order to cut costs, they decided not to make a redundant controller for the feeding system. This being the case, it is known that the incident matrix for the third controller is identical to the incident matrix of the first controller. The final incident matrix for all controllers is found by vertically concatenating all of the individual incident matrixes.

Bcontroller= [B; Bc1; Bc2; Bc3]

The complete incident matrix of the petri net is given in the appendix. There is no figure for the final petri net since the image is too cluttered to be of much benefit.

Conclusion

From the first example it can be concluded that both a makarov chain and a petri net can be used to help understand a situation. The detailed path of the alcoholic can be modeled as a makarov chain when he is drunk and a petri net while he is sober.

The second example shows how multiple controllers can be applied to the same petri net by concatenating the incident matrix with the incident matrix of the new controller. Adding more controllers on a petri net can also restrict the movement of the pets. If there are too many controls eventually the pets will never be able to move. Any movement of a marker must satisfy all of the controllers. It is also seen how the inclusion of a separate and redundant controller does not add any additional restrictions to the movements of the pets.

References

- [1] K. Yamalidou, J. Moody, M. Lemmon, and P. Antsaklis, "Feedback control of Petri nets based on place invariants," *Automatica*, vol. 32, no. 1, pp. 15-28, January 1996
- [2] L. Li, C. N. Hadjicostis, and R. S. Sreenivas, "Designs of bisimilar Petri net controllers with fault tolerance capabilities," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 38, no. 1, pp. 207-217, January 2008
- [3] Diagrams produced with WoPeD 3.2.0 software

Appendix:

The final incident matrix for the second example is shown below.

Bcontroller=

[illegible]