MATLAB TOOLBOX FOR PETRI NETS

Martina Svádová, Zdeněk Hanzálek
Center for Applied Cybernetics, DCE FEE
Czech Technical University in Prague

Introduction

Petri Nets offer profound mathematical background originating namely from linear algebra and graph theory. Various Petri Net tools offer convenient graphical environment and sometimes they provide complex simulation and analysis of various high level Petri Net classes. On the other hand these tools have very limited possibility of extensions to problems specifically needed for given application. Some of them are accessible in source code, but these software projects are relatively large, difficult to modify and platform dependent.

Matlab is probably one of the most popular software tools in the area of applied mathematics. Namely in control engineering it became the standard environment to interchange ideas implemented in algorithms. Why do the students, researchers and practitioners appreciate this language and associated environment so much? To answer this question it is important to notice the following:

- Even for beginners it is quite easy to implement and run simple algorithm (e.g. continuous space process simulation). Matlab manipulations are usually taught in basic subjects from control engineering, so students are already familiar with this environment when they come to the subjects dealing with Petri Nets.
- There is a wide range of toolboxes helping to implement very sophisticated algorithms. As these toolboxes are mainly designed for continuous space systems it is very attractive to use the Matlab solution for hybrid Petri Nets.
- Major part of the control theory is already implemented in the form of Matlab functions. So it helps to "standardize" the terminology and it gives opportunity to the students and practitioners to play with it. This practical standardization procedure would be also appreciated in Petri Net community.
- Matlab runs on many platforms (Windows, Unix).

On the other hand Matlab does not cover (with exception of Stateflow) discrete event systems.

Approach adopted in this article is based on the three steps:

- Petri Net modeling in convenient graphical design tool (e.g. PM Editeur or PIPE2)
- export to matrix representation in Matlab
- Matlab based Petri Net analysis and visualization of simulation results.

This approach helps to organize a work in a modular way, to use standard libraries and to build own tools. In other words one is no more using a 'universal' tool but he/she is programming his/her own tool with support in a modeling and visualization stage. This is quite more convenient because no tool is universal enough.

Example 1 - deadlock avoidance*: Figure 1 shows a Petri Net model of two communicating computers in deadlock situation. The core of the problem is that both computers are ready to receive but none is ready to send.*

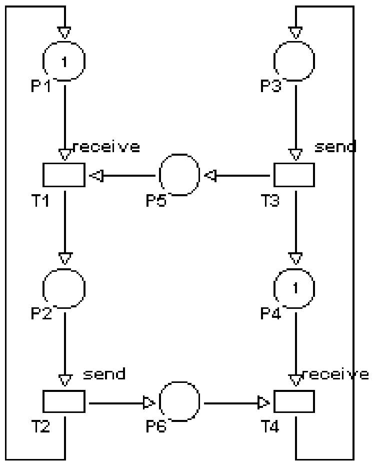

Figure 1: Two communicating computers in deadlock

Matlab based analysis of PN designed in Pmediteur is done by the following procedure:
1) In PM Editeur environment save the net as file deadlock.rdp
2) Import the net to the Matlab using function rdp.m:

```
[Pre, Post, M0] = rdp('deadlock.rdp')
```

3) Calculate incidence matrix C:

```
C = Post-Pre
```

4) Find a set of minimal standardized P-invariants using function silva.m:

```
G = silva(C)
```

5) Check whether each minimal standardized P-invariant contains at least one token:

```
d = G * M
```

In the case of the marked graph in Figure 1, there is a deadlock as there is at least one zero element in d'=(1 1 0). In other words the P-invatriant corresponding to the conservative component {P3,P5,P2,P6} does not contain any token.

This article is organized in two principal sections. The first one explains separate functions of the toolbox. The next section shows various examples used in laboratory exercises of the subject Distributed Control Systems.

1. Petri Net toolbox functions

Functions of Matlab Toolbox 1.0 can be downloaded from web site
http://dce.felk.cvut.cz/cak/research/PN/Matlab_Toolbox

1.1. Matlab import functions

Matlab Toolbox does not contain own graphical editor. Petri Nets under assumption is drawn in graphical editor PM Editeur. Functions *rdp.m* and *rdp2stpn.m* import structure and parameters of Petri Nets created in graphical editor PM Editeur. Function *rdp2stpn.m* is used to import stochastic timed Petri Net and function *rdp.m* is used in the case of autonomous Petri Nets.

Syntax:

[Pre, Post, M0] = **rdp**(filename)

[Pre,Post,M0,TimeT,TypeT]=**rdp2stpn**(filename)

Input parameter:

Filename	-	name of file, in which Petri net, drawn in graphical toolbox PM Editeur, is saved.

Output parameters:

Pre	-	matrix of pre-conditions
Post	-	matrix of post-conditions
C	-	incidence matrix
M0	-	column vector of initial marking of Petri Net
TimeT	-	column vector of time associated to the transitions
TypeT	-	column vector of transitions types: 0 - zero timed transition 1 - timed transition 2 - stochastic time transition with uniform distribution

1.2. The graph of reachable markings

The graph of reachable markings can be use only for analysis of properties of bounded net. Functions for construction of this graph are two. The first function *reach_graph.m* finds all reachable markings given marked PN and the second function *disp_gr.m* displays the graph of reachable markings. Note: *reach_graph.m* was previously named *graph.m* (*graph* is now reserved in Matlab).

Syntax:

[A, B] = **reach_graph**(Pre, Post, M0)

Input parameter: same as output parameters above

Output parameters:

A	-	adjacency matrix; matrix, whose elements *A(i,j)* mean oriented arc from vertex *i* to vertex *j*. The value of numbers in this matrix denotes indexes of fired transitions.
B	–	matrix of reachable; each column of matrix represents marking of one statee

Syntax:

[XX]= **disp_gr**(A, B)

Input parameter: same as output parameters above

Output parameters:

XX	-	which contains information necessary to display the graph of reachable markings. Information is obtained by processing of matrix **A**.

Matrix XX contains ten items:
- state identity number, which corresponds to number of column in matrix **B** *(state);*
- parent of this state *(parent);*
- the number of subsequent states *(subs);*
- index of fired transition *(trans)*
- initial and final X,Y positions needed for displaying the graph of reachable markings *(Xi, Xf, Yi, Yf)*
- number of column of **XX** matrix corresponding to parent of processing state *(col)*
- level of the graph of reachable markings *(level)*

1.3. Token player for T-timed and stochastic net

Stochastic Timed Petri nets used in this article contain either zero, timed or stochastic timed transitions with uniform distribution. There are two types of non-zero time transition behavior given in literature. The definition used in the article, considers that transition does not reserve the tokens in input places. When "reservation" behavior is needed, then zero time transition should precede timed or stochastic transition.
The functionality of the model under consideration is fully specified by the interpretation of the token player *playstpn.m* given in Matlab source code.
In the case of effective conflict, no token reservation is assumed. It means that the first fireable transition wins.
In the case of the actual conflict representing the system non-determinism (two and more fireable transitions in conflict) the winning transition is chosen in random manner. It means that one possible firing sequence is chosen among several ones.

Syntax:

[Seq] = **playstpn**(Pre,Post,M0,TimeT,TypeT,ticks)

Input parameter: same as output parameters of function rdp2stpn.m

Ticks - integer number of simulation ticks

Output parameters:

Seq – matrix **Seq** has 2 rows. The first row contains time, when some of transitions was fired. The second row is number of the transition, which was fired in time given in the first row.

M - marking vector of net after simulation

1.4. Finding of minimal standardized P-invariant

Finding of minimal standardised P-invariants is based on the algorithm published by Martinez&Silva.

Syntax:

[P]= **silva**(C)

Input parameter:

C - incidence matrix

Output parameters:

P - this matrix is a nonnegative matrix, such that:

1) each positive P-invariant could be done as a lambda combination of the rows of **P**
2) no row of **P** could be done as a lambda combination of other rows of **P**.

2. Case studies

2.1. Token Ring

Figure 2 shows two nodes accessing the media with the use of token ring access method. The media activity is represented by P9 and by P10 (Bus_Idle and Bus_busy). These two places are in fact implicit - they do not contribute to the system behavior. The following reasoning programmed in Matlab proves this.

- from the minimal standartized P-invariants

$M(P1)+M(P5)+M(P10) = 1$

$M(P1)+M(P3)+M(P5)+M(P7) = 1$

- Does the following hold?

$M(P10) \geq M(P3)$

- by substitution

$1-M(P1)-M(P5) \geq 1-M(P1)-M(P5)-M(P7)$

$M(P7) \geq 0$

- in similar way

$M(P10) \geq M(P7)$

$1-M(P1)-M(P5) \geq 1-M(P1)-M(P3)-M(P5)$

$M(P3) \geq 0$

- as consequence P10 is implicit
- in similar way P9 is implicit too

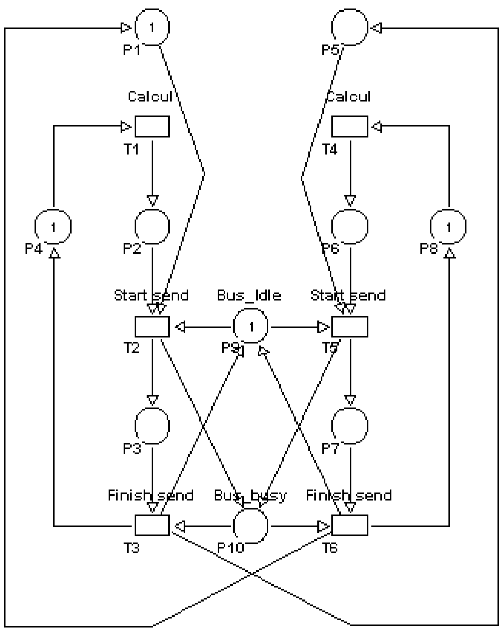

Figure 2: Token Ring access method

Next chapters show procedure how uses Matlab Toolbox functions for finding properties of Petri Net.

2.2. Construction of the graph reachable markings

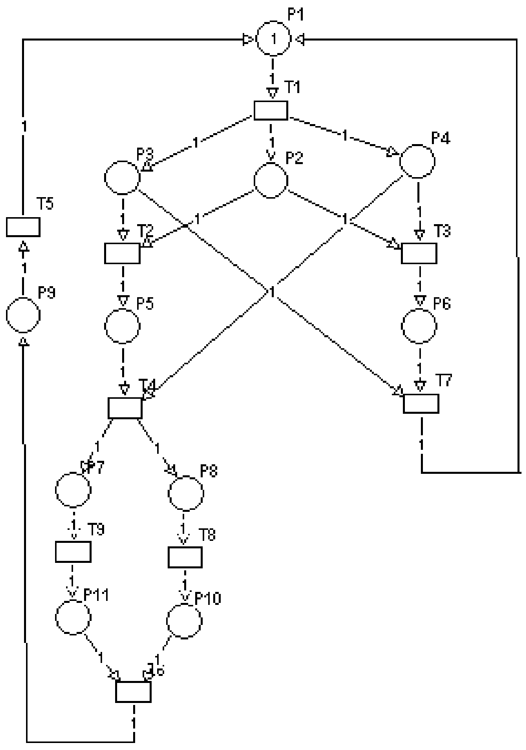

Figure 3

1) Save the net as file *net.rdp* in *PM Editeur* environment
2) Import the net structure using function *rdp.m*

[Pre, Post, M0] = rdp('net.rdp')

3) Construction adjacency matrix **A** and matrix reachable markings **B**

[A, B] = reach_graph(Pre, Post, M0)

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{array}{c|cccccccccc} \text{M} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \\ \hline & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{B} = & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}$$

4) Display of the graph of reachable markings

[XX] = disp_gr(A, B)

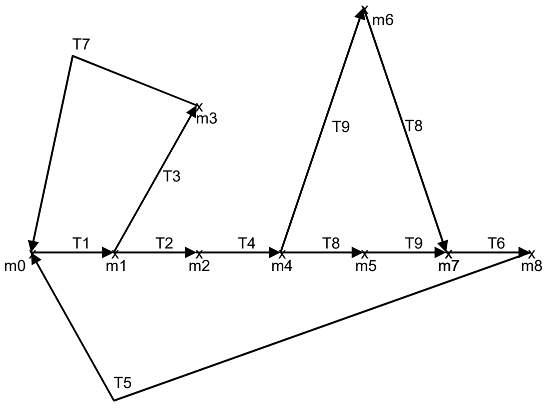

Figure 5 - The graph of reachable markings

Properties of Petri Nets showed in the fig.5: net is safe, bounded, live and reversible, has no deadlocks and contains three repetitive component T1T3T7 and T1T2T4T8T9T6T5 and T1T2T4T9T8T6T5.

2.3. Using of token player

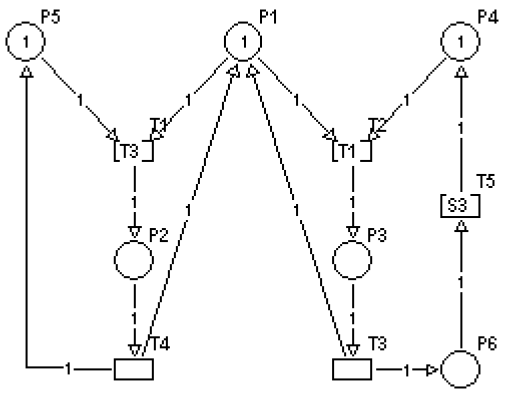

Figure 4

1) *Save the net as file stpnpl.rdp in PM Editeur environment*

2) *Import the net structure using function rdp2stpn.m*

[Pre,Post,M0,TimeT,TypeT]=rdp2stpn(*stpnpl.rdp*)

3) *Using function of token player palystpn.m*

[Seq] =playstpn(Pre,Post,M0,TimeT,TypeT,ticks)

$$\mathbf{Seq} = \begin{pmatrix} 1 & 1 & 2 & 3 & 3 & 6 & 6 & 6 & 7 & 7 & 10 & 10 & 10 & 11 & 11 \\ 2 & 3 & 5 & 2 & 3 & 5 & 1 & 4 & 2 & 3 & 5 & 1 & 4 & 2 & 3 \end{pmatrix}$$

$$\mathbf{M'} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

3. References

[1] Tadao Murata: “Petri Nets: Properties, Analysis and Applications” Proceedings of the IEEE, vol. 77, No. 4, April 1989.

[2] Zhen Liu, "Performance Analysis of Stochastic Timed Petri Nets using Linear Programing Approach, INRIA 1997 ISSN 0249-6399

[3] J. Martinez, M. Silva: A Simple and Fast Algorithm to Obtain All Invariants of a Generalised Petri Net,
in: C. Girault, W. Reisig (eds): Application and Theory of Petri Nets, Informatik Fachberichte No.52, Springer (1982), 301-310.

[4] F. Kruckeberg, M. Jaxy: Mathematical Methods for Calculating Invariants in Petri Nets, in: G. Rozenberg (ed): Advances in Petri Nets, LNCS 266, Springer (1987) 104-131.