A PETRI NET BASED MODEL FOR AEB SYSTEMS CONSIDERING VEHICLE AND PEDESTRIAN/CYCLIST IN A CERTAIN AREA

A Thesis

Submitted to the Faculty

of

Purdue University

by

Wensen Niu

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

December 2017

Purdue University

Indianapolis, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Stanley Yung-Ping Chien, Chair

Department of Electrical and Computer Engineering

Dr. Lingxi Li

Department of Electrical and Computer Engineering

Dr. Sohel Anwar

Department of Mechanical Engineering

Approved by:

Dr. Brian King

Head of the Graduate Program

To my family, for their love and support.

ACKNOWLEDGMENTS

The research work of my thesis is completed by the patient guidance and direction of Dr. Stanley Chien and Dr. Lingxi Li. Dr. Li introduced the methodology and application of Petri net in a fancy way not only with some lectures but also with some anecdotes among his experience. I greatly thank Dr. Chien, for his constant support throughout my research work and Dr. Sohel Anwar, for taking time out of his busy schedule to be a part of my advisor committee. I would like to thank Ms. Sherrie Tucker for assistance throughout my Master program.

Last but not the least, I would like to thank my group members and the best friends, Qiwen Deng and Dan Shen, who are genius PhD students and experienced researchers in various fields, for their innovative suggestions on my research work.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABBREVIATIONS

AEB	automatic emergency braking
PAEB	pedestrian automatic emergency braking
CAEB	cyclist automatic emergency braking
RCS	radar cross section

ABSTRACT

Niu, Wensen. M.S.E.C.E., Purdue University, December 2017. A Petri Net Based Model for AEB Systems Considering Vehicle and Pedestrian/Cyclist in a Certain Area. Major Professor: Lingxi Li.

For AEB performance testing, surrogates and testing systems have been developed by the Transportation Active Safety Institute (TASI). A set of tests, both for performance of vehicle equipped with AEB systems and for harmonization of surrogates, have been conducted under different scenarios with different variables which include cyclist speed, vehicle speed, cyclist moving directions, and so on.

There are several braking patterns described in this thesis, which provide the possibility to control the distance of an emergency braking. With different braking distance and steering angle during the emergency braking, choosing the final place of a vehicle becomes possible.

This thesis considers vehicle and pedestrian/cyclist together to avoid a crash in a certain area rather than predicting the collision point. Petri net models were built for both vehicle and pedestrian/cyclist in potential collision area and crossing road scenarios. Then, controllers were designed for Petri net models and all possible states were calculated.

1. INTRODUCTION

1.1 Background and Motivation

Automatic Emergency Braking (AEB) systems can avoid or mitigate a crash by detecting a potential forward crash, alerting driver to take actions, and applying brakes [1]. AEB is also called precrash system, pre-collision system, and others. AEB systems attract attention by both government and manufactures who announced to make AEB as a standard feature on new vehicles from 2022 [2]. Pedestrian Automatic Emergency Braking (PAEB) systems provide alerts and emergency braking when pedestrian was detected by the systems via in-vehicle camera and radar [3], [4]. Cyclist Automatic Emergency Braking (CAEB) systems have been developed by many automotive manufactures and have started to be equipped on vehicles [5].

For improving the AEB performance on different road situations, some other advanced driving-assistance systems, such as vehicle-to-vehicle (V2V) communication system, may be integrated with the AEB system [6]. Other methodologies can also be integrated for PAEB and CAEB systems to improve their performance. To avoid the vehicle crash with pedestrian/cyclist on the road, the problem can be considered as a mutual exclusion problem in a certain area. In order to consider vehicle and pedestrian/cyclist as a mutual exclusion problem, Petri net models including vehicle and pedestrian/cyclist in a certain area should be developed and analyzed. Also, the braking strategies should be studied under certain testing environment, given a vehicle equipped with AEB system.

1.2 Literature Survey

Research in automatic braking systems has been conducted over several decades. Chandler and Wood's paper published in 1977 discussed the fundamentals to design operation radar sensors for pedestrian detection that can help automobiles automatically activate the braking systems [7]. On the other hand, a vision based real time detection algorithm of vehicles and pedestrians was introduced by A. T. Ali and E. L. Dagless [8], which had the Transputer-Image Processing System (TIPS), image sensing, and geometric transformation. Moving Object Detectors (MODs), such as edge detection, interframe differencing, and so on, were developed [8]. Furthermore, the online vehicle and pedestrian detections based on sign pattern was introduced in [9]. And C. Curio et al separated human walking model into 12 phrase and detected walking pedestrian by checking those walking modes [10]. D. M. Gavrila introduced a pedestrian recognition system which was integrated with three pedestrian recognition systems, video-based, detection-verification framework, and stereo vision provided by Region of Interest (ROI) [11]. After these, the vision based pedestrian detection had been greatly developed in [12], [13], which include pedestrian detection, tracking, and protection with real time vision and night vision.

Studies in recent ten years showed a significant progress and impressive results in modeling, detection and control of cyclist/pedestrian autonomous emergency braking systems and related research, such as pedestrian/cyclist detection, pedestrian behavior and cyclist path prediction, AEB models including braking and evasive steering, and driving behavior. Some research focus on pedestrian/cyclist detection which output a set of pedestrian/cyclist candidates, detect with partial occlusion, overcome some potential false, detect without high computational cost, and introduce vision-based pedestrian/cyclist counting algorithm [14], [15], [16], [17], [18]. C. Zhou provided a stochastic optimization method to extend pedestrian tracking efficiently [15]. Pedestrian/cyclist behavior was modeled in [19], [20] and pedestrian/cyclist prediction was modeled in [21], [22]. And M. K. Park et al calculated precise warning

distance [23] and V. R. Garate et al summarized the use of a software packages with ability to cover all critical aspects of vulnerable road users [24]. Braking and evasive steering were modeled and analyzed in [25], [26]. AEB system improved via considering driving behavior and combining with V2X technology in [27], [28].

1.3 Problem Statement and Main Contributions

The purpose of this research is to find possible places for vehicle and pedestrian/cyclist to crash in certain area. This problem considered the braking strategies in AEB system. Petri net models were built for the vehicle and pedestrian/cyclist as a mutual exclusion problem in certain area for the AEB system. The prototype of this model defines the layout of the potential collision area and the controller has been designed to avoid crash between vehicle and pedestrian/cyclist. Specifically, the certain area includes the area when pedestrian/cyclist cross road.

In this thesis, Chapter 2 reviews the concepts of Petri net which include the terminology of Petri nets and mathematical descriptions of Petri net structure. At the same time, examples are provided in Chapter 2 for those concepts. Chapter 3 introduces the CAEB testing, such as testing equipment, surrogate, and testing scenarios. Then, the testing data are analyzed and braking patterns are shown, which are used for the design of certain area layout. The concept of potential collision area is proposed as the prototype of the certain area.

MATLAB codes are developed for processing the Petri nets model and calculating reachable states along with coverability tree, which are introduced in Section 4.1. And Petri net models are built for vehicle and pedestrian/cyclist and controllers has been designed in Chapter 4. Then, Chapter 5 provides conclusions and future works.

2. BACKGROUND ON PETRI NETS

The basic concepts of Petri nets and control of Petri nets are reviewed in this chapter. The mathematical background, definitions and terminology will be used in this thesis. Some examples are shown in this chapter. Graphical representation is a way to model the target system. These examples are graphical models with underlying mathematical analysis. More details about Petri nets can be found in [29].

This chapter is organized as follows. The basic Petri net concepts, such as the structure, the marking, the state space, incident matrices, the dynamics, and state equation are reviewed first. Followed those basic concepts, the control mechanism is reviewed.

2.1 Petri Net Models

Petri Net Structure

Petri net graph (structure) is weighted bipartite graph $N = (P, T, A, W)$ [29], where

P: a finite set of places.

T: a finite set of transitions.

A: the set of arcs from places to transitions and from transitions to places.

$$A \subseteq (P \times T) \cup (T \times P)$$

W: weight function on arcs.

There is no arcs between places and no arcs between transitions. $I(t_j)$ is used to denote the set of input places for transitions t_j; $O(t_j)$ to denote the set of output places from transitions t_j; $I(P_i)$ to denote the input transitions for place P_i; $O(P_i)$ to denote the output transitions from place P_i.

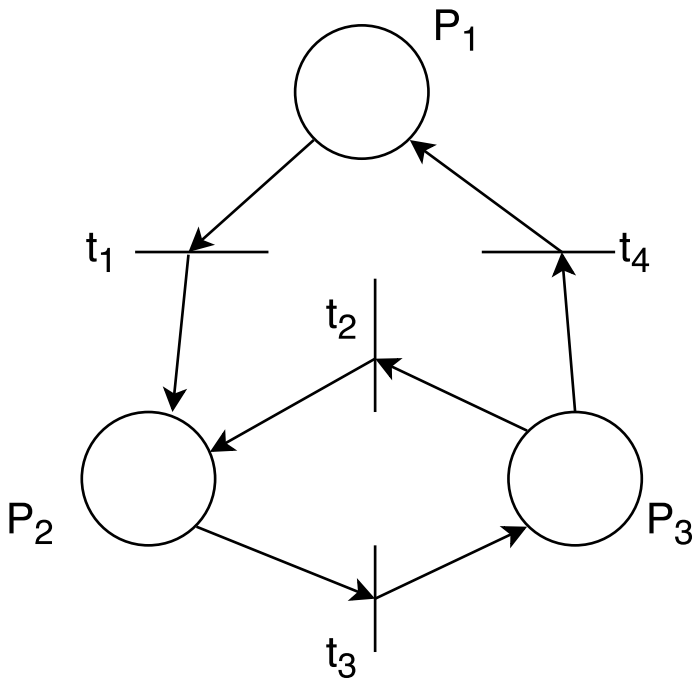

Fig. 2.1. A Petri net structure

Example 1 A Petri net structure is shown in Fig. 2.1.

In this simple Petri net structure, There is a set of places: $P = \{P_1,\ P_2,\ P_3\}$. The set of transitions is given by $T = \{t_1,\ t_2,\ t_3,\ t_4\}$, the set of arcs is by $A = \{(P_1,\ t_1),\ (t_1,\ P_2),\ (P_3,\ t_2),\ (t_2,\ P_2),\ (P_2,\ t_3),\ (t_3,\ P_3),\ (P_3,\ t_4),\ (t_4,\ P_1)\}$, and the arc weights by $W(P_1,\ t_1) = 1$, $W(t_1,\ P_2) = 1$, $W(P_3,\ t_2) = 1$, $W(t_2,\ P_2) = 1$, $W(P_2,\ t_3) = 1$, $W(t_3,\ P_3) = 1$, $W(P_3,\ t_4) = 1$, $W(t_4,\ P_1) = 1$,

Marking and State Space

Tokens (drawn as block dots) are assigned to places. The way in which tokens are assigned defines a marking [30].

$$M : P \Rightarrow 0, 1, 2, 3, \cdots$$

$M(P_j)$ denotes the numbers of tokens in place P_j. Subscripts for marking M are used to denote steps. For example, the initial marking M_0 denotes the initial markings for each place, where

$$M_0 = \begin{bmatrix} M_0(P_1) \\ M_0(P_2) \\ \vdots \end{bmatrix}$$

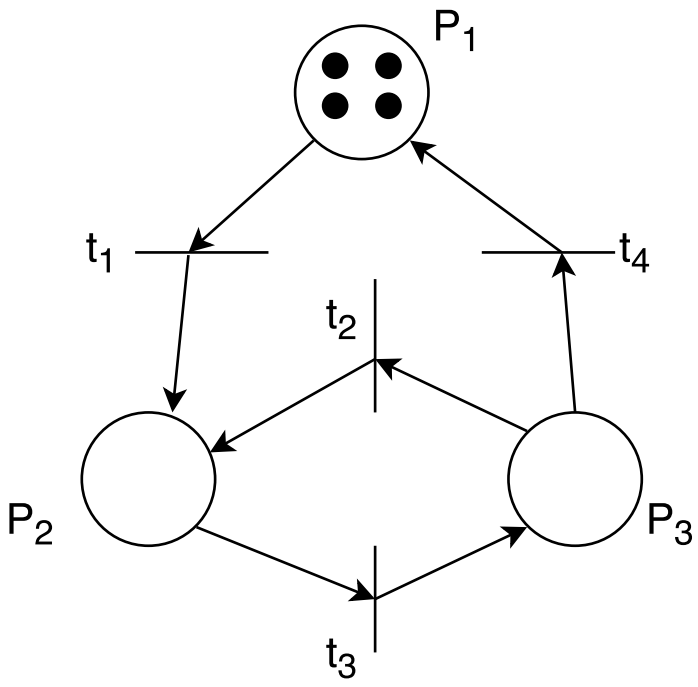

Fig. 2.2. The Petri net structure with a possible initial marking

Example 2 For the Petri net shown in Fig. 2.1, there is a possible initial marking $M_0 = [4\ 0\ 0]^\top$, which is shown in Fig. 2.2.

Incident Matrix

Giving a Petri net with n places and m transitions, input incident matrix, output incident matrix, and incident matrix can be defined uniquely in $n \times m$ dimensions as

follows. Output incident matrix B^+, captures arc weights from transitions to places; Input incident matrix B^-, captures arc weights from places to transitions; Incident matrix is defined as $B \triangleq B^+ - B^-$. In a Petri net, the incident matrices are structural properties, and they are independent of states (markings) of the system.

Example 3 For the Petri net shown in Fig. 2.1, there are some other possible initial markings. But they have identical incident matrices which are shown below (because they have the same Petri net structures).

The input incident matrix is given by

$$B^- = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

The output incident matrix is given by

$$B^+ = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

The incident matrix is given by

$$B \triangleq B^+ - B^- = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & 1 & -1 & 0 \\ 0 & -1 & 1 & -1 \end{bmatrix}$$

Dynamics (enabling and firing of transitions)

A transition $t_j \in T$ is said to be enabled if

$$M(P_i) \geq B^-(P_i, t_j)$$

for all $P_i \in I(t_j)$. In other words, transition t_j is enabled when the number of tokens in each input place P_i of t_j is at least as large as the arc weight from P_i to t_j.

If a transition is enabled, it can fire. When it fires, it removes as many tokens as the weight of the arc from each input place. It deposits as many tokens as the weight of the arc to each output place.

The state equation is

$$M_{k+1} = M_k + BV_k \tag{2.1}$$

where

M_{k+1} is the state at step $k+1$;

M_k is the state at step k;

B is the incident matrix;

V_k is the firing vector (with the dimension $m \times 1$).

Example 4 In the Petri net shown in Fig. 2.2, transition t_1 is enabled under marking $M_0 = [4\ 0\ 0]^\top$, which means it may fire. When transition t_1 fires,

$$V_0 = [1\ 0\ 0\ 0]^\top$$

Then, the new marking is

$$M_1 = M_0 + BV_0 = \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & 1 & -1 & 0 \\ 0 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix}$$

2.2 Control of Petri Nets

A state-based control is reviewed in this section. The control mechanism is to introduce additional places and arcs as controller to disable or enable transitions to avoid forbidden states. The specifications are in general in terms of constraints in the form of inequality as below

$$L \times M \leq b \tag{2.2}$$

For the inequality above, slack variable, M_c, is introduced as the state of the controller ($M_c \geq 0$) to make the left side of equation equal to the right one. The equation is shown as below

$$LM + M_C = b \tag{2.3}$$

which can be expressed in the form of matrix:

$$\begin{bmatrix} L & I \end{bmatrix} \begin{bmatrix} M \\ M_c \end{bmatrix} = b \tag{2.4}$$

where I is an identity matrix.

The initial state of the Petri net controller is obtained by

$$LM_0 + M_{c0} = b \tag{2.5}$$

Then, place invariant can be used to design a Petri net controller. Based on the definition of place invariant

$$X^\top B = 0 \tag{2.6}$$

$$\Rightarrow \quad \begin{bmatrix} L & I \end{bmatrix} \begin{bmatrix} B \\ B_c \end{bmatrix} = 0$$

The incident matrix of the controller is obtained by

$$LB + B_c = 0 \tag{2.7}$$

For a Petri net with constraint $LM \leq b$, to design a Petri net controller that satisfies constraints. The controller incident matrix is calculated by

$$B_c = -LB \tag{2.8}$$

The controller initial state is

$$M_{c0} = b - LM_0 \tag{2.9}$$

Example 5 Consider the Petri net shown in Fig. 2.2.

Design a Petri net controller to enforce the following constraint:

$$M(P_2) + M(P_3) \leq 2$$

To design the Petri net controller which satisfies the constraint, constraint was expressed in the form of $LM \leq b$, where

$$L = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix},\ M = \begin{bmatrix} M(P_1) \\ M(P_2) \\ M(P_3) \end{bmatrix},\ b = 2$$

For the Petri net structure shown in Fig. 2.2, the input incident matrix B^-, output incident matrix B^+ and the incident matrix B are given by

$$B^- = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \end{array},\ B^+ = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array},\ B = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & 1 & -1 & 0 \\ 0 & -1 & 1 & -1 \end{bmatrix}$$

Then the incident matrix and the initial state can be calculated for the controller as follows. The controller incident matrix is

$$B_c = -LB = -\begin{bmatrix} 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & 1 & -1 & 0 \\ 0 & -1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 1 \end{bmatrix},$$

and the controller initial state is

$$M_{c0} = b - LM = 2$$

The Petri net with its controller that satisfies the constraints are shown in Fig. 2.3. The control mechanism is to introduce the additional place, P_C, and arcs, $(P_C,\ t_1)$

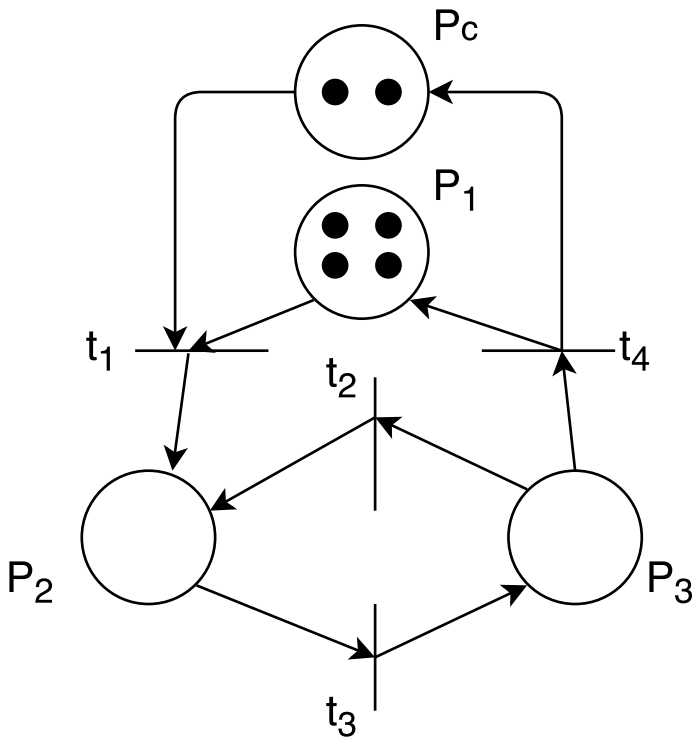

Fig. 2.3. The Petri net structure with controller for *Example 5*

and $(t_4,\ P_C)$, to the original Petri net which is shown in Fig. 2.2. The initial state of the controller $M(P_C) = 2$. And the weight of those arcs are $W(P_C,\ t_1) = 1$ and $W(t_4,\ P_C) = 1$ respectively.

The state-based control can also be used to constrain places separately, which is shown in the example below.

Example 6 Consider the Petri net shown in Fig. 2.4. Design a Petri net controller to enforce the following constraint:

$$M(P_2) \leq 2$$

$$M(P_3) \leq 1$$

Similar as the example above, constraints were expressed in the form of $LM \leq b$ as the matrix equation below

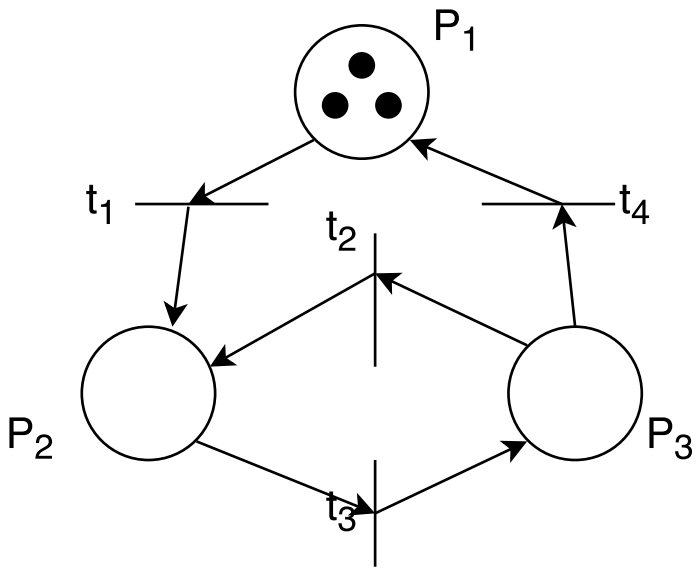

Fig. 2.4. The Petri net structure for *Example 6*

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M(P_1) \\ M(P_2) \\ M(P_3) \end{bmatrix} \leq \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Since the Petri net has the same Petri net structure with the Petri net in Fig. 2.2, the incident matrices are the same as the incident matrices in the example above.

Then the incident matrix and initial state were calculated for the controller as follow. The controller incident matrix and the controller initial state are

$$B_c = -LB = \begin{bmatrix} -1 & -1 & 1 & 0 \\ 0 & 1 & -1 & 1 \end{bmatrix},$$

$$M_{c0} = b - LM = \begin{bmatrix} 2 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

The Petri net with its controller that satisfies the constraints are shown in Fig. 2.5. Places, P_{C1} and P_{C2}, and arcs, $(P_{C1},\ t_1)$, $(t_4,\ P_{C2})$, $(P_{C1},\ t_2)$, $(t_2,\ P_{C2})$, $(P_{C2},\ t_3)$, $(t_3,\ P_{C1})$ are introduced to the original Petri net shown in Fig. 2.4. The weights of the arcs are $W(P_{C1},\ t_1) = 1$, $W(t_4,\ P_{C2}) = 1$, $W(P_{C1},\ t_2) = 1$, $W(t_2,\ P_{C2}) = 1$, $W(P_{C2},\ t_3) = 1$, $W(t_3,\ P_{C1}) = 1$ respectively.

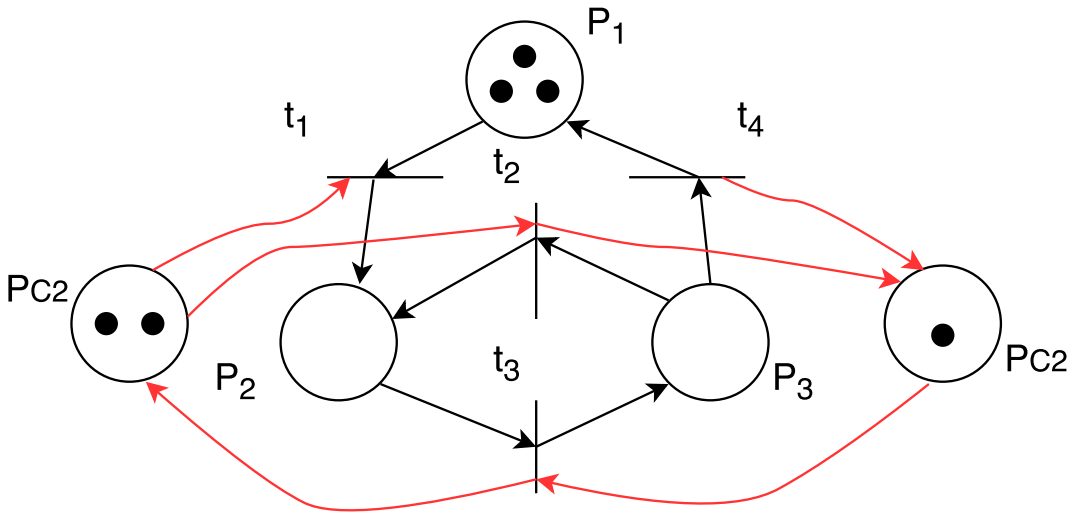

Fig. 2.5. The Petri net structure with Controller for *Example 6*

2.3 Summary

This chapter reviews the basic concepts of Petri net and the design processes of the Petri net controller with certain constraints. Those concepts of Petri net will be used in the following chapters.

3. DATA COLLECTION AND ANALYSIS

In this chapter, data collection is introduced first. Section 3.1 will introduce equipment, surrogates and scenarios. During the AEB process, braking strategies were analyzed. In Section 3.2, the typical test results are shown and braking patterns are abstracted from those results. In the following section, the concept of potential collision area was proposed. This potential collision area will be discussed in next chapter to build Petri net model and design its controller. As an extension of the potential collision area, the crossing road scenarios will be discussed in next chapter as well.

3.1 Data Collection

Equipment

An Oxford RT 3002 DGPS was used to record the vehicle location, vehicle speed, and vehicle acceleration. The warning light for the pre-collision system was used as the input to record the warning time and duration. The tail brake light was used as the input to record the braking start time and duration. The brake paddle was equipped with a contact sensor, which was used to record the time and duration of braking which was applied by driver.

The surrogate cyclist was dragged by a carrier triggered with wireless signal from the IR sensor, which detects vehicles passing certain positions in test scenarios. The motion of the carrier was controlled by wireless signal from a joystick. There is also a paddling motion of the cyclist, which was also controlled by the wireless signal from IR sensor.

Surrogate

Two surrogate cyclists were used in this cyclist AEB test. They are designed based on the size of US cyclist rider population and the size of Europe cyclist rider population, respectively. As a surrogate cyclist for AEB test, the surrogate design includes size, radar cross-section (RCS), and the motion system. The side view of two cyclist surrogates are shown in Fig. 3.1. The front and back view of two cyclists are shown in Fig. 3.2.

Fig. 3.1. Side view of cyclists

(a) Front view (b) Back view

Fig. 3.2. Front and back view of cyclists

Test site and scenario

There is a flat taxiway with old asphalt at a decommissioned airport that we use as a test site. All cyclist AEB tests were conducted on this test site. The road surface and the background is shown in Fig. 3.3. In this AEB test, there is a vehicle equipped with AEB system, which is shown in Fig. 3.4.

Fig. 3.3. Test road surface with background

Fig. 3.4. Vehicle for cyclist AEB test

Among the cyclist AEB test, there are three kinds of scenarios, moving along road, crossing road, and crossing road with 45°. These three scenarios are shown in Fig. 3.5. Cyclist icons in the figure show the start place of the cyclist for each scenario.

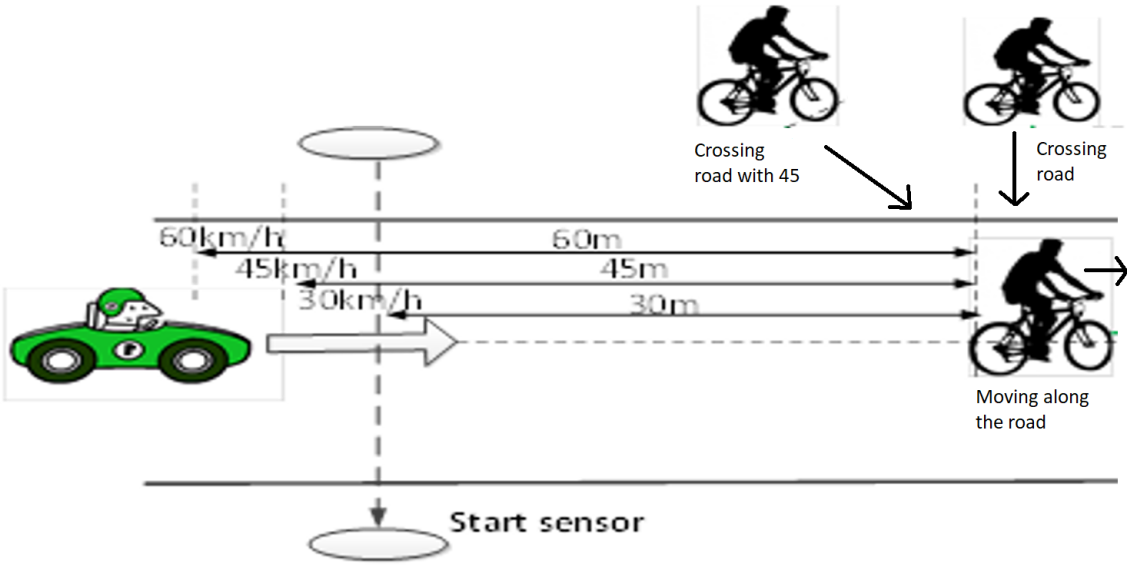

Fig. 3.5. Cyclist AEB test scenarios

In these three scenarios, the vehicle initial speed are $30km/h$, $45km/h$, $60km/h$, respectively, while the distance between start sensor and the origin are $30m$, $45m$, $60m$ respectively. For the moving along road scenario, the cyclist speeds is $15km/h$, or $20km/h$. For the crossing road scenario and the 45° crossing road scenario, the cyclist speed is $15km/h$ and the distance from start point to collision point is $19m$. There are several groups of stationary tests, where cyclist speed equals to $0km/h$ in tests with the crossing road and moving along road positions.

3.2 Braking Patterns

In the research of braking patterns, stationary tests data were analyzed, where AEB system works and avoids the collision. The braking patterns of stationary cyclist AEB tests show that the braking strategies can be used for pedestrian AEB, since

they would not be affected by the speed difference between pedestrians and cyclists. Stationary tests data of cyclist AEB show that there are different braking patterns for the AEB system equipped on the vehicle. In the main pattern shown in Fig. 3.6, there is an initial transient phase until the peak time t_m, when the braking force achieves the maximum braking force. After that there is a second transient time until the settling time t_r, and then a constant ratio braking phase until the vehicle stops.

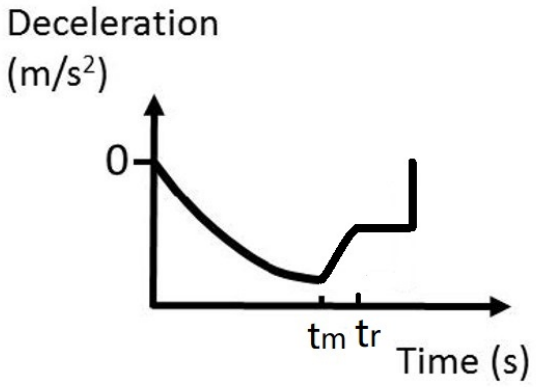

Fig. 3.6. Braking pattern A

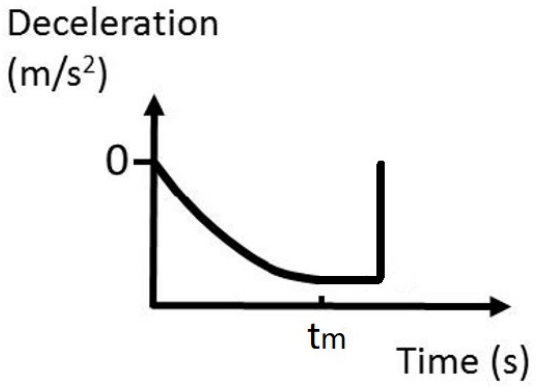

Fig. 3.7. Braking pattern B

There are two more braking patterns. One is shown in Fig. 3.7. The first phase is the transient phase until the maximum braking force has been applied. After the

transient phase, the maximum braking force was applied as a constant braking force until the vehicle stops. The other one is called pattern C which is shown in Fig. 3.8, the brake was released after the first phase in a slow speed and the vehicle stops right in front of the cyclist.

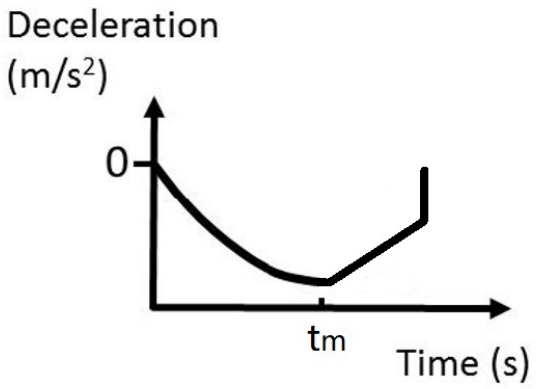

Fig. 3.8. Braking pattern C

The occurrence frequencies of the patterns mentioned above are shown in Fig. 3.9. The most frequent braking pattern is A. The control strategy of braking pattern A provides a phase with ratio which offers a possibility to control the barking distance and braking time duration by adjusting the ratio values.

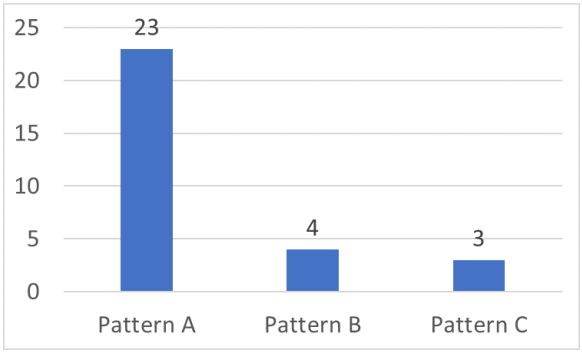

Fig. 3.9. Braking patterns frequency

3.3 Certain Area and Potential Collision Area

The common vehicle to pedestrian/cyclist encounters are shown in Fig. 3.10. Vehicle is moving on the road straight forward in the figure. Then, these common encounter scenarios can be defined as follows

- Pedestrian/cyclist crossing road from either side of the road/lane;
- Pedestrian/cyclist moving along the road/lane with/against the vehicle direction.

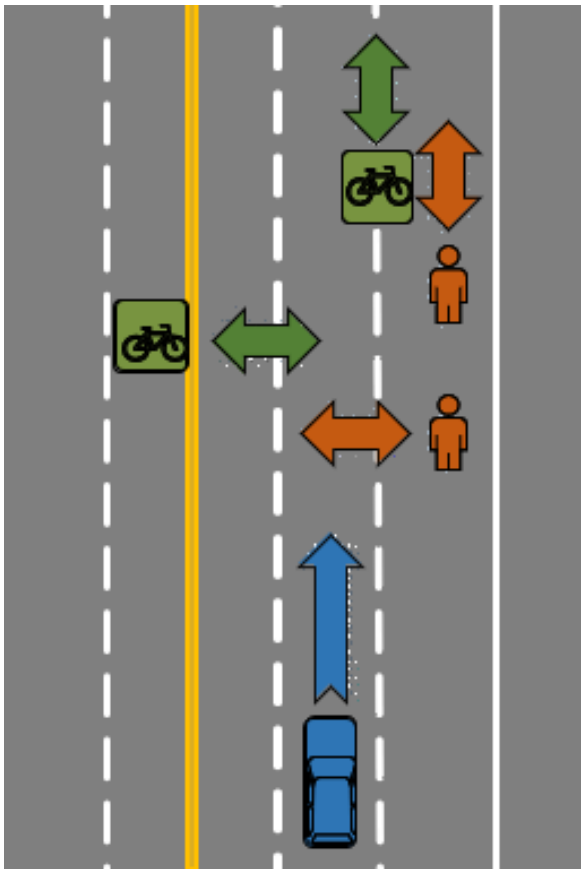

Fig. 3.10. Common vehicle to pedestrian/cyclist encounters

Certain Area: The possible area includes the possible places considering the moving ability of pedestrian/cyclist or the braking strategies of vehicle.

In each of the scenarios, pedestrian/cyclist can perform different moving paths. In this thesis, the concept of certain area is to consider pedestrian/cyclist moving in a area rather than to study the prediction of pedestrian/cyclist paths. For pedestrian/-cyclist in the Fig. 3.10, a certain area based on the pedestrian/cyclist moving ability is shown as the thick rectangle in Fig. 3.11. This area can both include several lanes or be included in one lane. It shows the possible moving area of pedestrian/cyclist for each of those scenarios.

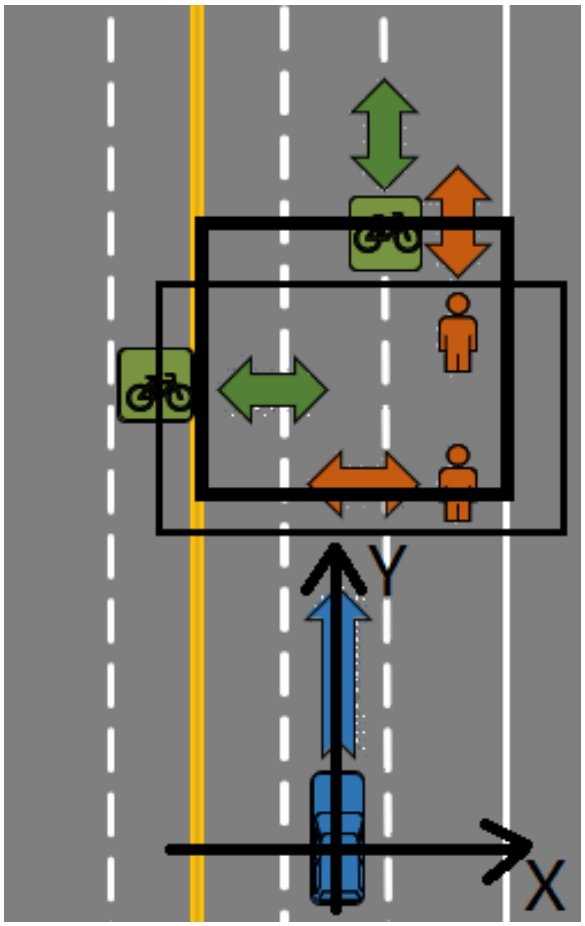

Fig. 3.11. A certain area of vehicle and pedestrian/cyclist

For a vehicle equipped with AEB system, steering can change the final vehicle place in x axis which is shown in Fig. 3.11. Based on the braking patterns discussed in the previous section, vehicle is able to control the braking distance and braking

time duration, which means that the final vehicle place can change in Y axis shown in Fig. 3.11. Then, there is a certain area for vehicle to stop, which is shown as the thin rectangle in Fig. 3.11.

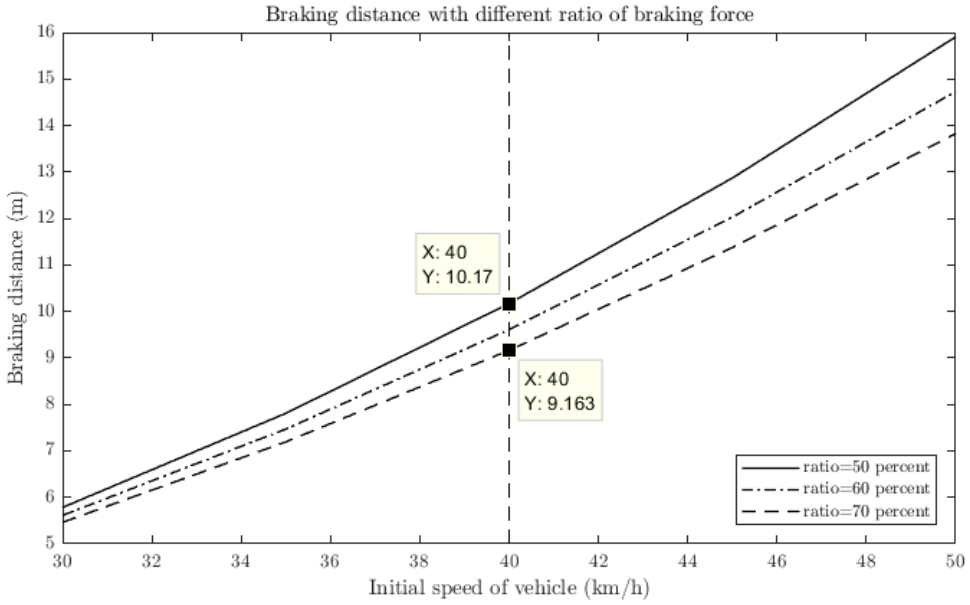

Fig. 3.12. Braking distance with different ratio of braking force

Not only pedestrian/cyclist can change their places in their certain area, vehicle can select the place for final stop. Based on the different strategies and properties among vehicles, the certain area can be changed. The certain area of vehicle can also be included in one lane or include several lanes.

If a vehicle takes braking pattern A but with different ratio of braking force in the last braking phase, it has different braking distances for different initial speed, which is shown in Fig. 3.12. This figure shows a simulation of the braking distance changing with different ratio of braking force under initial speed from 30 km/h to 50 km/h. For different vehicle, these curves can be different because of the differences among braking strategy and braking performance. In the figure, if the vehicle has 40 km/h as the initial speed and it's braking strategy allows 50% to 70% ratio of braking force

in the last phase with braking pattern A, it can provide near 1 *meter* range for its certain area in the direction of axis Y.

Potential collision area: The area is the intersection area between certain area of vehicle and certain area of pedestrian/cyclist, where vehicle and pedestrian/cyclist can meet such that potential crashes can happen.

In this situation, vehicle and pedestrian/cyclist encounter problem in certain area can be considered as a mutual exclusion problem. To solve this problem, potential collision area which is shown in Fig. 3.13 is proposed as a prototype of the certain area. It is defined in this section and be analyzed in next chapter. Furthermore, there are some other areas that can be analyzed. Potential collision area is an area with several places, or spaces, which means the area that vehicle and pedestrian/cyclist can meet so potential crashes can happen in this area.

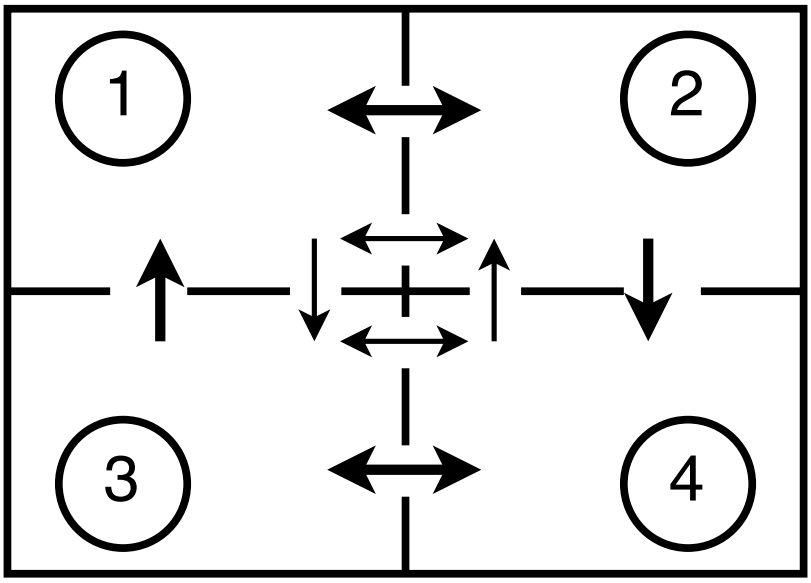

Fig. 3.13. A potential area with four places

A potential area can be designed in different ways, which means it can be divided into several subareas as needed. In this section a potential collision area is provided which is shown as prototype in Fig. 3.13. In this prototype, it is assumed that vehicle and pedestrian/cyclist have same certain area so that potential collision area is the

certain area of vehicle and pedestrian/cyclist. Four subareas are designed in this potential area as four places for Petri net modeling. It is defined that vehicle can choose the final places following the direction of thick arrows, and the pedestrian/cyclist can choose the final places following the direction of those thin arrows.

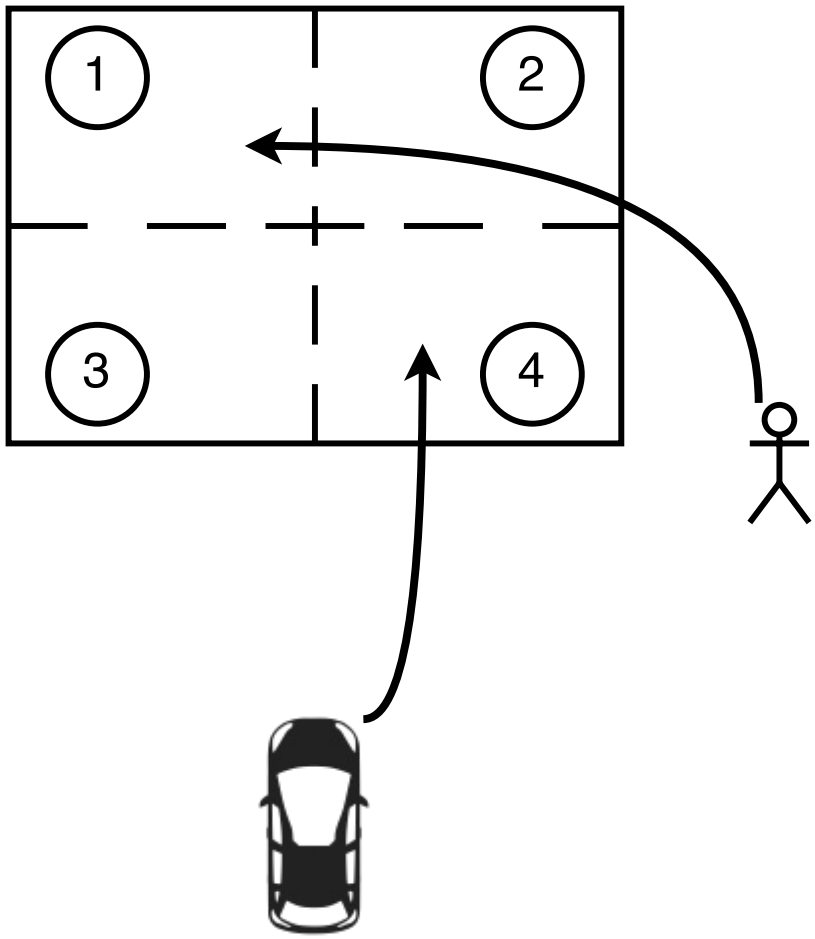

Fig. 3.14. Choosing place to stop in potential collision area

When AEB is needed, they can choose their final places, which is shown as the indication of arrows in Fig. 3.14. And they can change their final places. To avoid a collision, they should not stop finally in the same place of that area, which is a mutual exclusion problem between a vehicle and pedestrian/cyclist in the potential area. If the pedestrian/cyclist change the idea that choosing place 4 as the final place

rather than choosing place 1, the vehicle can change its final place from place 4 to place 3. It should change both its steering angle and the ratio of braking force in last phase of the braking, for changing the final place from place 4 to place3.

Petri nets will be used as a tool to build the model of vehicle and pedestrian/cyclist and to design the controller to make sure they would not stop in the same place. When the AEB system is needed, a potential collision area is taken into consideration which includes places as the potential final stop place for vehicle and pedestrian/cyclist. To avoid a collision in a potential collision area, the Petri net models for vehicle and pedestrian/cyclist in their certain area should be built first. Then the Petri net controller should be designed for the combined Petri net models for vehicle and pedestrian/cyclist. The controller is needed to avoid vehicle and pedestrian/cyclist going to the same place during the dynamic process of the combined Petri nets.

3.4 Summary

This chapter introduces the testing equipment for control and data collection. Then the surrogate, vehicle, and testing scenarios are also introduced. Braking patterns were analyzed from the original data and the frequency is shown in Section 3.2. After investigating braking patterns, a certain area is proposed with the potential collision area as its prototype.

4. PETRI NET MODEL FOR AEB SYSTEM

In this chapter, the MATLAB codes used for analysis are introduced in three subsections, inputs, functions and results. Section 4.2 includes the modeling and analysis of vehicle and pedestrian/cyclist in the potential area. Specifically, the modeling and analysis of vehicle and pedestrian/cyclist in crossing road scenario are illustrated in the following section.

4.1 MATLAB Codes

This section introduces the MATLAB codes, input and functions, for design Petri net controller and calculation of the reachable states for the Petri net with controller. The calculation is followed by the results in both command window and an text file, which include the coverability tree and the node type for each state.

4.1.1 Inputs

There is a main file as the input, which is named "*condition.m*". The file includes the basic parameters and constraints of the Petri net. Then it calculates parameters of the controller. Finally, it calculates the coverability tree for the Petri net and outputing the results as a file. An example condition file is shown in appendix *An Example Condition File*.

4.1.2 Functions

Functions include methods to calculate the parameters of original Petri net and controller for the specific constraints, such as incident matrices of original Petri net, initial states of the controller, initial incident matrix of the controller, all states in

coverability tree. These functions are used to process the original input file which describes the parameters and constraints for designing the controller. There are five functions, *incident function*, *inicon function*, *controlledpetri function*, *transition function*, and *petricon function*, to complete the whole task. Each function has a help document in MATLAB, which can be found by "help *Function name*" in the command window. In the last line of each help document, it will show the relative function name for searching those help documents for convenience. All Functions are shown in appendix *Functions*.

Incident Function

Incident matrices are properties of a Petri net structure, which is a basic information of the Petri net. This function treats the input incident matrix and output incident matrix, which are included in the condition file, as the input. And it checks the dimension of both input incident matrix and output incident matrix. After the check, it will calculate the incident matrix $B \triangleq B^+ - B^-$ if the dimensions of the two matrix are the same; or it will show the error information to let you check the incident input matrix and output incident matrix in the input file.

Inicon Function

Based on the incident matrix, initial marking, and the constraints of the Petri net, the initial state and markings of the Petri net controller can be calculated for the original Petri net. The inputs of this function are matrices L and b which are in the constraints $LM \leq b$, the initial marking of the Petri net M_0, and the incident matrix of Petri net. The output of this function are initial markings M_{C0} of the Petri net controller and initial incident matrix B_C of the Petri net controller.

Controlledpetri Function

The parameters of the Petri net with the controller are needed to calculate all states and the node type in the coverability tree of the Petri net with controller. The inputs for this function are the incident matrices and initial marking of both Petri net and the controller. Then it calculates those matrices for the incident matrix and initial markings of the new Petri net which include the places and arcs being introduced by the controller.

Example 7 Considering the example condition file which describes the Petri net structure in Fig. 2.2 and the constraint in *Example 5*. The results of calling *controlledpetri* function are shown as below

$$B_{cpinput} = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_C \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{array}, \; B_{cpoutput} = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_C \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

$$BB_{co} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \; M0M_{co} = \begin{bmatrix} 4 \\ 0 \\ 0 \\ 2 \end{bmatrix}$$

$B_{cpinput}$ is the input matrix of the Petri net with controller; $B_{cpoutput}$ is the output matrix of the Petri net with controller; BB_{co} is the incident matrix of the Petri net with controller; $M0M_{co}$ is the initial markings of the Petri net with controller.

Transition Function

Coverability tree is a finite tree with possibly infinite number of states. In this function, node type, Petri net markings, and transition sequence are calculated. Transition sequences are recorded in T_{all}, and each marking is stored in M_{all}.

Node type is defined as variable DT. If $DT = 0$, it is a state with transition to be enabled; if $DT = 1$, it is a duplicate node; if $DT = 2$, it is a terminate node in the coverability tree. In this function,$DT = 1$ is the initial value each time and it checks if there is any transition that can be enabled.

Petricon Function

Petricon function is called by the input file, *condition.m*, to calculate all the results by calling functions above as needed among the process of calculating parameters for controller, building the Petri net with controller, and generating the coverability tree.

4.1.3 Results

Results of these codes include those matrices shown in the command window and a text file shown as in the Fig 4.1.

Example 8 After calling the functions above, the results of the Petri net with the controller (Fig. 2.3.) are shown as below. The first line of M_{all} is the initial marking. It has transitions to be enabled since the nodes type is 0. In the DT matrix, $DT(i) = 1$ indicates the state is a duplicate node. There are five reachable states for the Petri net with controller.

wensenExample.txt

	node type	each states	transition sequences
1	node type	each states	transition sequences
2	0	4 0 0 2	0 0 0 0 0
3	0	3 1 0 1	1 0 0 0 0
4	0	2 2 0 0	1 1 0 0 0
5	0	3 0 1 1	1 3 0 0 0
6	0	2 1 1 0	1 1 3 0 0
7	1	2 1 1 0	1 3 1 0 0
8	1	3 1 0 1	1 3 2 0 0
9	1	4 0 0 2	1 3 4 0 0
10	1	2 2 0 0	1 1 3 2 0
11	0	2 0 2 0	1 1 3 3 0
12	1	3 1 0 1	1 1 3 4 0
13	1	2 1 1 0	1 1 3 3 2
14	1	3 0 1 1	1 1 3 3 4
15			

Fig. 4.1. Results text file as an example

$$D_T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \; M_{all} = \begin{bmatrix} 4 & 0 & 0 & 2 \\ 3 & 1 & 0 & 1 \\ 2 & 2 & 0 & 0 \\ 3 & 0 & 1 & 1 \\ 2 & 1 & 1 & 0 \\ 2 & 1 & 1 & 0 \\ 3 & 1 & 0 & 1 \\ 4 & 0 & 0 & 2 \\ 2 & 2 & 0 & 0 \\ 2 & 0 & 2 & 0 \\ 3 & 1 & 0 & 1 \\ 2 & 1 & 1 & 0 \\ 3 & 0 & 1 & 1 \end{bmatrix}, \; T_{all} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 \\ 1 & 1 & 3 & 0 & 0 \\ 1 & 3 & 1 & 0 & 0 \\ 1 & 3 & 2 & 0 & 0 \\ 1 & 3 & 4 & 0 & 0 \\ 1 & 1 & 3 & 2 & 0 \\ 1 & 1 & 3 & 3 & 0 \\ 1 & 1 & 3 & 4 & 0 \\ 1 & 1 & 3 & 3 & 2 \\ 1 & 1 & 3 & 3 & 4 \end{bmatrix}$$

4.2 Petri Net Modeling and Controller Design

The modeling and analysis for vehicle and pedestrian/cyclist in the potential collision area are discussed in this section. The Petri net graphs are designed for both vehicle and pedestrian/cyclist; Then, the vehicle and pedestrian/cyclist model is combined and a Petri net controller is designed for it. After that, the combined Petri net model is analyzed in detail.

To avoid a collision, a controller is needed for the combined Petri nets, which shows in the Section 4.2.3, *controller designing*. The Petri net controller is designed to guarantee that the vehicle and pedestrian/cyclist will never choose to stop in the same place as the initial state, vehicle stops in place 4 and pedestrian/cyclist stops in place 1.

4.2.1 Petri Net Model of Pedestrian/Cyclist

To build the Petri net model of pedestrian/cyclist in the potential collision area, some basic parameters of the Petri net, such as places set, transition set, and arcs set, are designed. For the potential area shown in Fig. 3.13 and the direction of the thin arrows, the places of pedestrian/cyclist are given by

$$P_{PC} = \{P_{PC1},\ P_{PC2},\ P_{PC3},\ P_{PC4}\}$$

where P with subscript represents to No. 1, 2 ,3 ,4 pedestrian/cyclist places in the potential area, respectively.

The transitions set is defined as $T_{PC} = \{t_{12PC},\ t_{21PC},\ t_{13},\ t_{34PC},\ t_{43PC},\ t_{42}\}$, where t with the subscripts ij represents that only pedestrian/cyclist has those transitions from i to j and for some transition from i to j not only has transition for pedestrian/cyclist, those subscripts $ijPC$ means the transition from i to j for pedestrian/cyclist.

In the potential collision area shown in Fig. 3.13, we define the pedestrian/cyclist moving between place 1 and place 2 or between place 3 and place 4 as the crossing

road direction and the pedestrian/cyclist moving from place 1 to place 3 or from place 4 to place 2 as moving along road direction. The event of each transition are shown in Table. 4.1.

Table 4.1.
Events for Pedestrian/Cyclist Transitions in Potential Collision Area

Transition	**Event**
t_{12PC}	Crossing road moving from place1 to place2
t_{21PC}	Crossing road moving from place2 to place1
t_{13}	Along road moving from place1 to place3
t_{34PC}	Crossing road moving from place3 to place4
t_{43PC}	Crossing road moving from place4 to place3
t_{42}	Along road moving from place4 to place2

Arcs set is given by

$$\begin{aligned} A_{PC} = \{&(P_{PC1},\ t_{12PC}),\ (t_{12PC},\ P_{PC2}),\ (P_{PC2},\ t_{21PC}),\ (t_{21PC},\ P_{PC1}),\\ &(P_{PC1}, t_{13}),\ (t_{13},\ P_{PC3}),\ (P_{PC3},\ t_{34PC}),\ (t_{34PC},\ P_{PC4}),\\ &(P_{PC4},\ t_{43PC}),\ (t_{43PC},\ P_{PC3}),\ (P_{PC4},\ t_{42PC}),\ (t_{42PC},\ P_{PC2})\} \end{aligned}$$

It is assumed the weight of each arc is 1. The weight from places to transitions and the weight from transitions to places are given by:

	t_{12PC}	t_{21PC}	t_{13}	t_{34PC}	t_{43PC}	t_{42}
P_{PC1}	1	0	1	0	0	0
P_{PC2}	0	1	0	0	0	0
P_{PC3}	0	0	0	1	0	0
P_{PC4}	0	0	0	0	1	1

	t_{12PC}	t_{21PC}	t_{13}	t_{34PC}	t_{43PC}	t_{42}
P_{PC1}	0	1	0	0	0	0
P_{PC2}	1	0	0	0	0	1
P_{PC3}	0	0	1	0	1	0
P_{PC4}	0	0	0	1	0	0

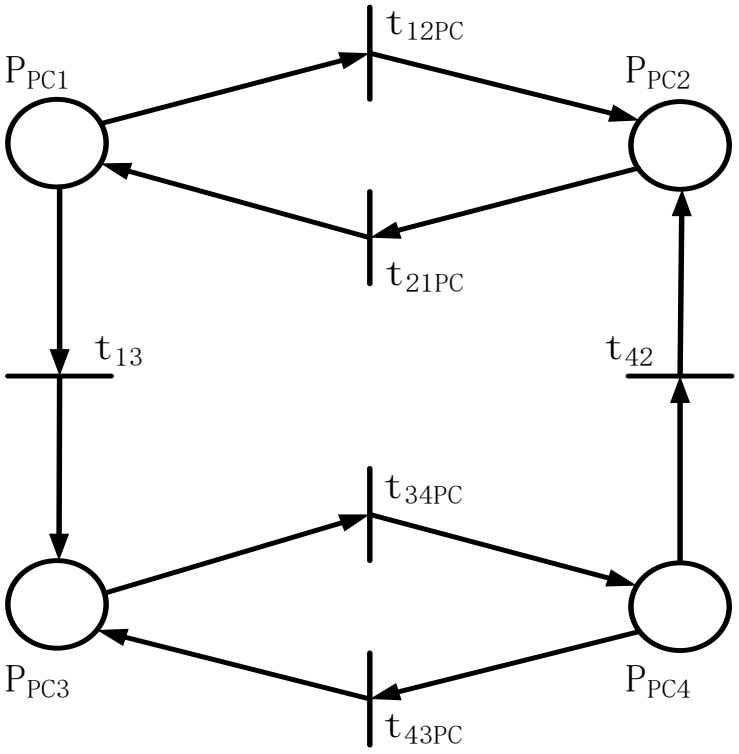

Fig. 4.2. Petri net graph for pedestrian/cyclist

The Petri net graph for pedestrian/cyclist Petri net is shown in Fig. 4.2

For the Petri net structure of pedestrian/cyclist Petri net, the incident matrices, input incident matrix B^-_{PC} and output incident matrix B^+_{PC} are given by

$$B^-_{PC} = \begin{array}{c} \\ P_{PC1} \\ P_{PC2} \\ P_{PC3} \\ P_{PC4} \end{array} \begin{array}{c} \begin{array}{cccccc} t_{12PC} & t_{21PC} & t_{13} & t_{34PC} & t_{43PC} & t_{42} \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

$$B^{+}_{PC} = \begin{array}{c} \\ P_{PC1} \\ P_{PC2} \\ P_{PC3} \\ P_{PC4} \end{array} \begin{array}{c} \begin{array}{cccccc} t_{12PC} & t_{21PC} & t_{13} & t_{34PC} & t_{43PC} & t_{42} \end{array} \\ \left[\begin{array}{cccccc} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \end{array}$$

$$B_{PC} = \begin{array}{c} \\ P_{PC1} \\ P_{PC2} \\ P_{PC3} \\ P_{PC4} \end{array} \begin{array}{c} \begin{array}{cccccc} t_{12PC} & t_{21PC} & t_{13} & t_{34PC} & t_{43PC} & t_{42} \end{array} \\ \left[\begin{array}{cccccc} -1 & 1 & -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 \end{array}\right] \end{array}$$

4.2.2 Petri Net Model of Vehicle

For building Petri net model of the vehicle, the places set, transition set and arcs set are defined first. The places set is $P_V = \{P_{V1},\ P_{V2},\ P_{V3},\ P_{V4}\}$, where the subscript Vi indicates the places for vehicle.

Transition set is $T_V = \{t_{12V},\ t_{21V},\ t_{24},\ t_{31},\ t_{34V},\ t_{43V}\}$ and the arcs set is

$$\begin{aligned} A_V = \{&(P_{V1},\ t_{12V}),\ (t_{12V},\ P_{V2}),\ (P_{V2},\ t_{21V}),\ (t_{21V},\ P_{V1}),\\ &(P_{V2}, t_{24}),\ (t_{24},\ P_{V2}),\ (P_{V3},\ t_{31V}),\ (t_{31V},\ P_{V1}),\\ &(P_{V3},\ t_{34V}),\ (t_{34V},\ P_{V4}),\ (P_{V4},\ t_{43V}),\ (t_{43V},\ P_{V3})\} \end{aligned}$$

Then, the incident matrices of vehicle Petri net model are calculated. For this Petri net model, there are four places and six transitions. The weight from the places to transitions and the weight from transitions to places are given by

	t_{12V}	t_{21V}	t_{24}	t_{31}	t_{34V}	t_{43V}
P_{V1}	1	0	0	0	0	0
P_{V2}	0	1	1	0	0	0
P_{V3}	0	0	0	1	1	0
P_{V4}	0	0	0	0	0	1

	t_{12V}	t_{21V}	t_{24}	t_{31}	t_{34V}	t_{43V}
P_{V1}	0	1	0	1	0	0
P_{V2}	1	0	0	0	0	0
P_{V3}	0	0	0	0	0	1
P_{V4}	0	0	1	0	1	0

In the potential collision area, the vehicle moving direction is defined as the same direction moving from place 3 to place 1. Places P_{V1} to P_{V4} represent four potential places at which vehicle would stop. To define the events for vehicle transitions, F_{RMB1} to F_{RMB4} are defined as the final braking force which would make vehicle stopped at Places P_{V1} to P_{V4} respectively.

Table 4.2.
Events for Vehicle Transitions in Potential Collision Area

Transition	**Event**
t_{12v}	Steering right and changing the final braking force from F_{RMB1} to F_{RMB2}
t_{21v}	Steering left and changing the final braking force from F_{RMB2} to F_{RMB1}
t_{24}	Change the final braking force from F_{RMB2} to F_{RMB4}
t_{31}	Change the final braking force from F_{RMB3} to F_{RMB1}
t_{34v}	Steering right and changing the final braking force from F_{RMB3} to F_{RMB4}
t_{43v}	Steering left and changing the final braking force from F_{RMB4} to F_{RMB3}

For this vehicle Petri net, the incident matrices are in 4×6 dimension. Input incident matrix B_V^-, output incident matrix B_V^+, and incident matrix B_V are given by

$$B_V^- = \begin{array}{c} \\ P_{V1} \\ P_{V2} \\ P_{V3} \\ P_{V4} \end{array} \begin{array}{c} \begin{array}{cccccc} t_{12V} & t_{21V} & t_{24} & t_{31} & t_{34V} & t_{43V} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

$$B_V^+ = \begin{array}{c} \\ P_{V1} \\ P_{V2} \\ P_{V3} \\ P_{V4} \end{array} \begin{array}{c} \begin{array}{cccccc} t_{12V} & t_{21V} & t_{24} & t_{31} & t_{34V} & t_{43V} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

$$B_V = B_V^+ - B_V^- = \begin{array}{c} \\ P_{V1} \\ P_{V2} \\ P_{V3} \\ P_{V4} \end{array} \begin{array}{c} \begin{array}{cccccc} t_{12V} & t_{21V} & t_{24} & t_{31} & t_{34V} & t_{43V} \end{array} \\ \begin{bmatrix} -1 & 1 & 0 & 1 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & 1 & -1 \end{bmatrix} \end{array}$$

The Petri net graph for the Petri net structure is shown in Fig. 4.3

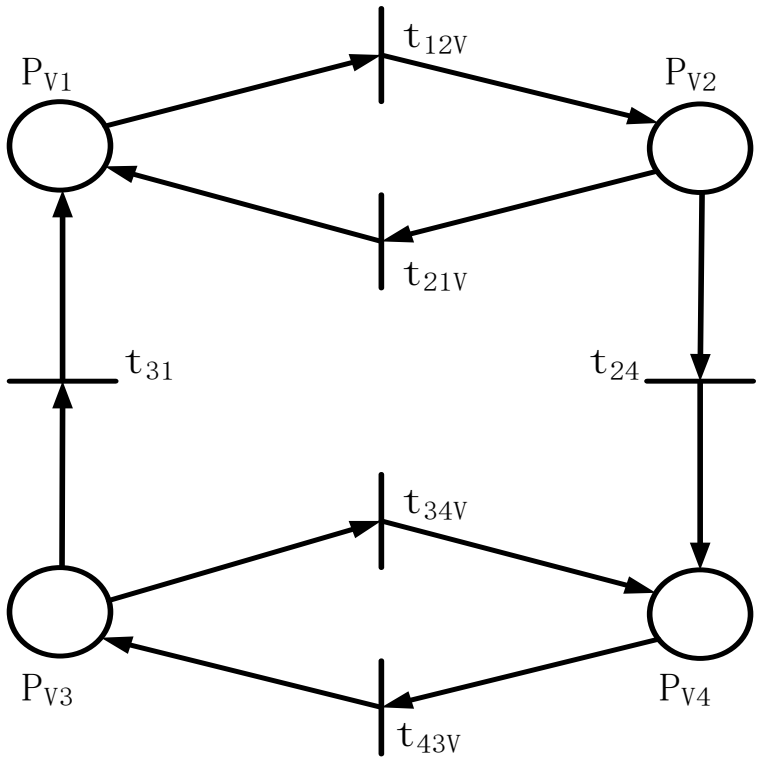

Fig. 4.3. Petri net graph for vehicle in potential collision area

4.2.3 Controller Design

The two Petri net models are to be combined into one Petri net for this mutual exclusion problem in the potential collision area.

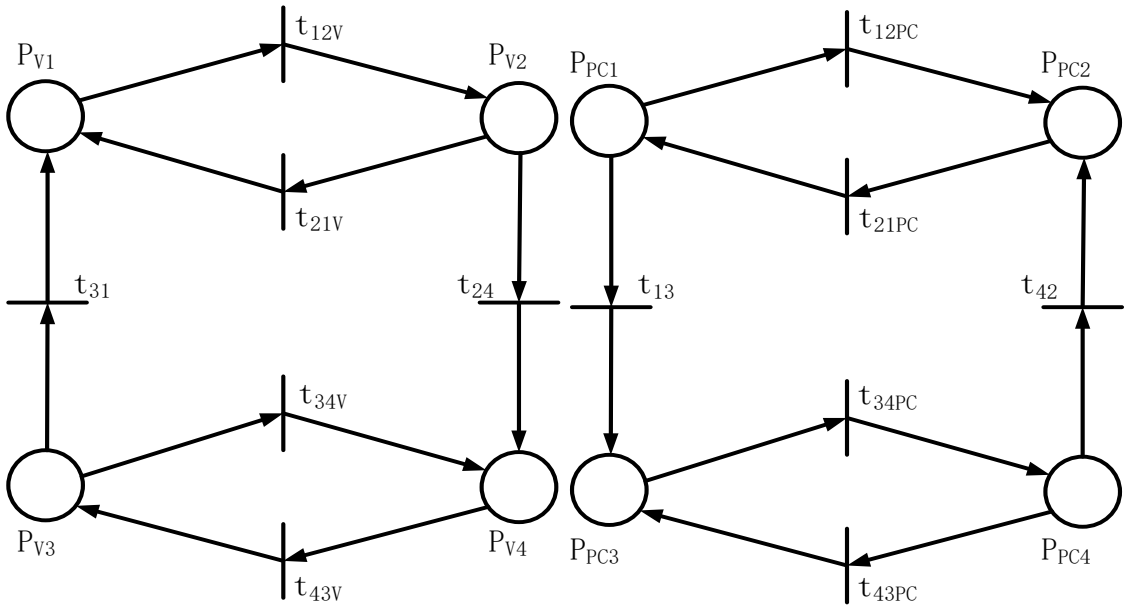

Fig. 4.4. Combined Petri net graph in potential collision area

After the combination, the Petri net graph is shown as Fig. 4.4. A Petri net controller is needed to guarantee that vehicle and pedestrian/cyclist will never choose the same place as the final stop place. For example, initially vehicle stops in place 4 and pedestrian/cyclist stops in Place 1.

There are new place set P, transition set T, and arcs set$A$ as follows

$$P = \{P_{V1},\ P_{V2},\ P_{V3},\ P_{V4},\ P_{PC1},\ P_{PC2},\ P_{PC3},\ P_{PC4}\};$$

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

$$A = \{(P_{V1},\ t_{12V}),\ (t_{12V},\ P_{V2}),\ (P_{V2},\ t_{21V}),\ (t_{21V},\ P_{V1}),$$
$$(P_{V2}, t_{24}),\ (t_{24},\ P_{V2}),\ (P_{V3},\ t_{31V}),\ (t_{31V},\ P_{V1}),$$
$$(P_{V3},\ t_{34V}),\ (t_{34V},\ P_{V4}),\ (P_{V4},\ t_{43V}),\ (t_{43V},\ P_{V3}),$$
$$(P_{PC1},\ t_{12PC}),\ (t_{12PC},\ P_{PC2}),\ (P_{PC2},\ t_{21PC}),\ (t_{21PC},\ P_{PC1}),$$
$$(P_{PC1}, t_{13}),\ (t_{13},\ P_{PC3}),\ (P_{PC3},\ t_{34PC}),\ (t_{34PC},\ P_{PC4}),$$
$$(P_{PC4},\ t_{43PC}),\ (t_{43PC},\ P_{PC3}),\ (P_{PC4},\ t_{42PC}),\ (t_{42PC},\ P_{PC2})\}.$$

The incident matrix is calculated based on the input incident matrix and output incident matrix as follows. The input incident matrix, output incident matrix, and the incident matrix are shown below

$$B^- = \begin{array}{c} \\ P_{V1} \\ P_{V2} \\ P_{V3} \\ P_{V4} \\ P_{PC1} \\ P_{PC2} \\ P_{PC3} \\ P_{PC4} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} t_{12V} & t_{21V} & t_{24} & t_{31} & t_{34V} & t_{43V} & t_{12PC} & t_{21PC} & t_{13} & t_{34PC} & t_{43PC} & t_{42} \end{array} \\ \left[\begin{array}{cccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right] \end{array}$$

$$B^+ = \begin{array}{c} \\ P_{V1} \\ P_{V2} \\ P_{V3} \\ P_{V4} \\ P_{PC1} \\ P_{PC2} \\ P_{PC3} \\ P_{PC4} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} t_{12V} & t_{21V} & t_{24} & t_{31} & t_{34V} & t_{43V} & t_{12PC} & t_{21PC} & t_{13} & t_{34PC} & t_{43PC} & t_{42} \end{array} \\ \left[\begin{array}{cccccccccccc} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \end{array}$$

$$B = B^+ - B^- = \begin{bmatrix} -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \end{bmatrix}$$

For this problem, the initial state of the Petri net is

$$M_0^\top = \begin{matrix} P_{V1} & P_{V2} & P_{V3} & P_{V4} & P_{PC1} & P_{PC2} & P_{PC3} & P_{PC4} \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

The controller is designed to guarantee the marking of each place are never greater than 1, which can be transferred into the form $LM \leq b$, where

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad M = \begin{bmatrix} M(P_{V1}) \\ M(P_{V2}) \\ M(P_{V3}) \\ M(P_{V4}) \\ M(P_{PC1}) \\ M(P_{PC2}) \\ M(P_{PC3}) \\ M(P_{PC4}) \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

A slack variable, $M_C (M_C \geq 0)$, is introduced to the constraints $LM \leq b$, such that

$$LM + M_c = b$$

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

The initial state for the Petri net controller of the combined Petri net is obtained by

$$LM_0 + M_{C0} = b$$

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

The Petri net controller has the incident matrix, $B_C = -LB$, which indicates the relationship between the controller places and the transitions of combined Petri net

$$B_C = \begin{array}{c} \\ P_{C1} \\ P_{C2} \\ P_{C3} \\ P_{C4} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} t_{12V} & t_{21V} & t_{24} & t_{31} & t_{34V} & t_{43V} & t_{12PC} & t_{21PC} & t_{13} & t_{34PC} & t_{43PC} & t_{42} \end{array} \\ \begin{bmatrix} 1 & -1 & 0 & -1 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & 1 \end{bmatrix} \end{array}$$

In the first section of condition file, *conditionPotentialCollisionArea.m*, incident matrices are given by *Bvinput*, *Bpcinput*, *Bvoutput*, *Bpcoutput*, respectively, to represent input incident matrices and output incident matrices of vehicle and pedestrian/cyclist Petri net model. For the combined Petri net, incident matrices are *Binput* and *Boutput* calculated by those incident matrices mentioned above. *M0* indicates the initial state of the combined Petri net, and the constraint of the combined Petri net is expressed as matrices L and b.

Building the controlled Petri net and calculating the reachable states of the Petri net is discussed below. *petricon* function is called with five input tuple, *Boutput*, *Binput*, L, b, $M0$, for incident matrix and initial state of the Petri net controller, which guarantees the vehicle and pedestrian/cyclist will never stop in the same place.

Incident matrices of the combined Petri net with controller are calculated by calling *petricon.controlledpetri.BBco* method. The initial state of the controlled Petri net is given by calling *petricon.controlledpetri.M0Mco* method. After calculation, the results of $BBco$ and $M0Mco$ are

$$BBco = \begin{array}{c} \\ P_{V1} \\ P_{V2} \\ P_{V3} \\ P_{V4} \\ P_{PC1} \\ P_{PC2} \\ P_{PC3} \\ P_{PC4} \\ P_{C1} \\ P_{C2} \\ P_{C3} \\ P_{C4} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} t_{12V} & t_{21V} & t_{24} & t_{31} & t_{34V} & t_{43V} & t_{12PC} & t_{21PC} & t_{13} & t_{34PC} & t_{43PC} & t_{42} \end{array} \\ \left[\begin{array}{cccccccccccc} -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \\ 1 & -1 & 0 & -1 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & 1 \end{array}\right] \end{array}$$

$$M0Mco^{\top} = \begin{array}{c} \begin{array}{cccccccccccc} P_{V1} & P_{V2} & P_{V3} & P_{V4} & P_{PC1} & P_{PC2} & P_{PC3} & P_{PC4} & P_{C1} & P_{C2} & P_{C3} & P_{C4} \end{array} \\ \left[\begin{array}{cccccccccccc} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}\right] \end{array}$$

With the incident matrix and initial state calculated above, the structure of controlled Petri net and the initial markings can be drawn as shown in Fig. 4.5. There are four controller places to be introduced to the original Petri net, which would guarantee the vehicle and pedestrian/cyclist will never stop in the same place.

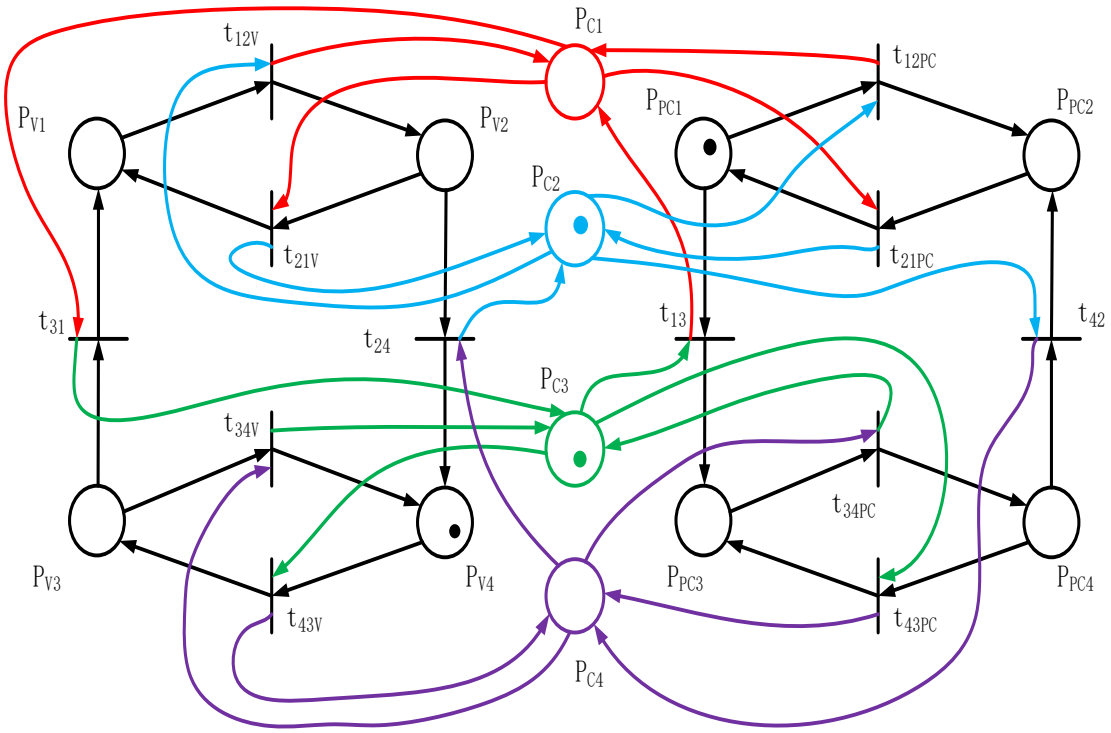

Fig. 4.5. Controlled Petri net with initial state in potential collision area

Then, a reachable states, transition sequences, and the nodes type are calculated by calling methods, *petricon.transition.Mall*, *petricon.transition.Tall*, and *petricon.transition.DT*. The results are listed as follows

$$M_{all} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$DT = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \\ 1 \\ 0 \\ 1 \\ 1 \\ 2 \\ 1 \\ 1 \end{bmatrix}, \; Tall = \begin{bmatrix} 0 & 0 & 0 \\ 6 & 0 & 0 \\ 7 & 0 & 0 \\ 9 & 0 & 0 \\ 6 & 5 & 0 \\ 6 & 7 & 0 \\ 7 & 6 & 0 \\ 7 & 8 & 0 \\ 6 & 7 & 4 \\ 6 & 7 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

In the last section of the condition file, a text file is generated as the results by printing the nodes type, reachable states, and transition sequences in that file. Each line of state matrix, M_{all}, is a reachable state. The first line of the matrix is the initial state which is equal to the initial state, $M0Mco$, of controlled Petri net. Because 1 indicates a duplicate node and 2 indicates a terminal node. For this controlled Petri net in the potential collision area, there are 6 reachable states including the initial state, $M_0^\top = [0\ 0\ 0\ 1\ 1\ 0\ 0\ 0]$. R_{all} denotes the reachable states

$$R_{all} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

4.3 Crossing Road Scenarios

In the previous section, the Petri net model of vehicle and pedestrian/cyclist was built and the controller was designed. In addition to this, the MATLAB code was used to build the Petri net with controller and to calculate the reachable states.

This section focuses on the situation of crossing road scenarios. The layout was defined as shown in Fig. 4.6. which includes the movement of pedestrian/cyclist and the potential stop places for vehicle. There are eight places in this layout of the crossing road scenario. We define the driving direction of vehicle the same as the direction from place 6 to place 4. And the direction between place 4 and place 5 or among place 1 to place 3 are crossing road direction. In the layout of the crossing road scenario, thin arrows indicate the abstract of the pedestrian/cyclist movement. Thick arrows present the final places where vehicle would stop finally.

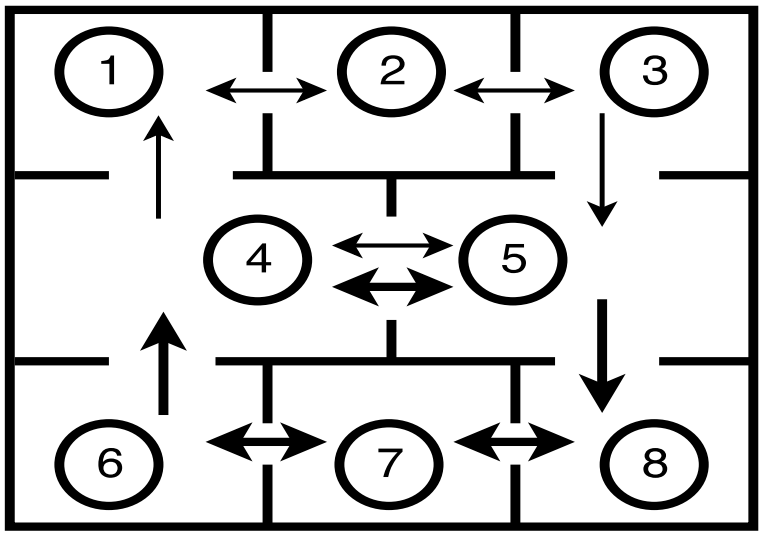

Fig. 4.6. Layout of crossing road scenarios

In the following subsections, Petri net model of pedestrian/cyclist and vehicle are introduced for this scenario, as well as the controller design and analysis process.

4.3.1 Vehicle Petri net model in Crossing Road Scenarios

In the crossing road scenario layout which is shown in Fig. 4.6, there are five places for vehicle Petri net model. Those places are denoted by P_{Vi}, where the subscript V means it is a place for vehicle and the subscript i indicates the number of the place in the layout of the crossing road scenario. For the vehicle Petri net model, the set of places is $P_V = \{P_{V4},\ P_{V5},\ P_{V6},\ P_{V7},\ P_{V8}\}$. The transitions in vehicle Petri net model are denoted by T_{ijV} for those transitions between places which are shared by pedestrian/cyclist Petri net model in the certain area layout of crossing road scenario, and T_{ij} for those transitions only for vehicle Petri net model. The transition set is given by $T_V = \{t_{45V},\ t_{54V},\ t_{64},\ t_{58},\ t_{67},\ t_{76},\ t_{78},\ t_{87}\}$.

Between these places and transitions, there are arcs for the vehicle Petri net model. The arcs set is shown as below. The weights between transitions and places are given as follows

$$\begin{aligned} A_V = \{&(P_{V4},\ t_{45V}),\ (t_{45V},\ P_{V5}),\ (P_{V5},\ t_{54V}),\ (t_{54V},\ P_{V4}),\\ &(P_{V6}, t_{64}),\ (t_{64},\ P_{V4}),\ (P_{V6},\ t_{67}),\ (t_{67},\ P_{V7}),\\ &(P_{V7}, t_{76}),\ (t_{76},\ P_{V6}),\ (P_{V7},\ t_{78}),\ (t_{78},\ P_{V8}),\\ &(P_{V8},\ t_{87}),\ (t_{87},\ P_{V7}),\ (P_{V4},\ t_{58}),\ (t_{58},\ P_{V8})\} \end{aligned}$$

	t_{45V}	t_{54V}	t_{64}	t_{58}	t_{67}	t_{76}	t_{78}	t_{87}
P_{V4}	1	0	0	0	0	0	0	0
P_{V5}	0	1	0	1	0	0	0	0
P_{V6}	0	0	1	0	1	0	0	0
P_{V7}	0	0	0	0	0	1	1	0
P_{V8}	0	0	0	0	0	0	0	1

	t_{45V}	t_{54V}	t_{64}	t_{58}	t_{67}	t_{76}	t_{78}	t_{87}
P_{V4}	0	1	1	0	0	0	0	0
P_{V5}	1	0	0	0	0	0	0	0
P_{V6}	0	0	0	0	0	1	0	0
P_{V7}	0	0	0	0	1	0	0	1
P_{V8}	0	0	0	1	0	0	1	0

To satisfy the final stopping place changing of the vehicle in crossing road scenario, event of each transition is designed in Table 4.3 which is similar to Table 4.2. Steering and braking with different ratio braking force is the event to change one final stopping place to another. The braking forces in final phase of the braking duration are defined as $F_V = \{F_{RMB4},\ F_{RMB5},\ F_{RMB6},\ F_{RMB7},\ F_{RMB8}\}$. The subscript RMB is the abbreviation of *Rational Maximum Braking force* and i after RMB indicate the ratio braking force needed to stop at each place.

Based on the relationship among the transitions and places in vehicle Petri net, the vehicle Petri net model in the crossing road scenario layout can be expressed in graph shown in Fig. 4.7. Places are drawn as circles and transitions are drawn as bars. There is no number on those arcs from transitions to places and from places to transitions, which means those arcs are with default weight of 1.

For this vehicle Petri net model, the input incident matrix and output incident matrix are given by

$$B_V^- = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},\ B_V^+ = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Table 4.3.
Events for Vehicle Transitions in Crossing Road Scenarios

Transition	**Event**
t_{45v}	Steering right and changing the final braking force from F_{RMB4} to F_{RMB5}
t_{54v}	Steering left and changing the final braking force from F_{RMB5} to F_{RMB4}
t_{64}	Keeping the steering angle or steering right and change the final braking force from F_{RMB6} to F_{RMB4}
t_{58}	Keeping the steering angle or steering right and change the final braking force from F_{RMB5} to F_{RMB8}
t_{67}	Steering right and changing the final braking force from F_{RMB6} to F_{RMB7}
t_{76}	Steering left and changing the final braking force from F_{RMB7} to F_{RMB6}
t_{78}	Steering right and changing the final braking force from F_{RMB7} to F_{RMB8}
t_{87}	Steering left and changing the final braking force from F_{RMB8} to F_{RMB7}

$$B_V = B_V^+ - B_V^- = \begin{array}{c} \\ P_{V4} \\ P_{V5} \\ P_{V6} \\ P_{V7} \\ P_{V8} \end{array} \begin{array}{c} \begin{array}{cccccccc} t_{45V} & t_{54V} & t_{64} & t_{58} & t_{67} & t_{76} & t_{78} & t_{87} \end{array} \\ \begin{bmatrix} -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & -1 \end{bmatrix} \end{array}$$

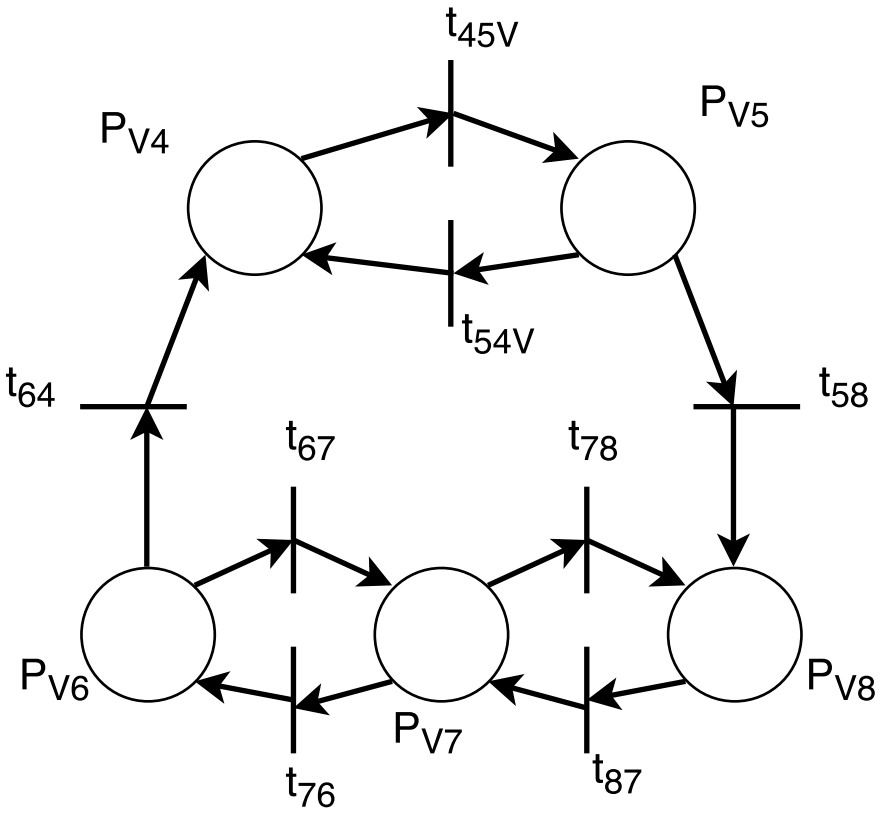

Fig. 4.7. Vehicle Petri net graph in crossing road scenarios

In this subsection, the vehicle Petri net model is developed for the layout of the crossing road scenario. The places, transitions, and events of the vehicle Petri net have been defined. Furthermore, the properties and incident matrices are calculated for vehicle Petri net graph which is shown in Fig. 4.7.

4.3.2 Pedestrian/Cyclist Petri net model in Crossing Road Scenarios

In Fig. 4.6, five places are defined for the pedestrian/cyclist Petri net model, where the place set is $P_{PC} = \{P_{PC1},\ P_{PC2},\ P_{PC3},\ P_{PC4},\ P_{PC5}\}$. The subscript PC means the places for pedestrian/cyclist and the subscript i indicates the place number in the layout.

To change the place from one to another, there are transitions in pedestrian/cyclist Petri net model as the transition set: $T_{PC} = \{t_{12},\ t_{21},\ t_{23},\ t_{32},\ t_{41},\ t_{35},\ t_{45PC},\ t_{54PC}\}$. The subscript ij means it moves the token from place i to j if the transition is enabled. The subscript PC indicates the transition is included in pedestrian/cyclist Petri net model.

Arcs are among these transitions and places of the pedestrian/cyclist Petri net model, including those from transitions to places and from places to transitions. The arcs set is given by

$$\begin{aligned} A_{PC} = \{&(P_{PC1},\ t_{12}),\ (t_{12},\ P_{PC2}),\ (P_{PC2},\ t_{21}),\ (t_{21},\ P_{PC1}),\\ &(P_{PC2}, t_{23}),\ (t_{23},\ P_{PC3}),\ (P_{PC3},\ t_{32}),\ (t_{32},\ P_{PC2}),\\ &(P_{PC4}, t_{41}),\ (t_{41},\ P_{PC1}),\ (P_{PC3},\ t_{35}),\ (t_{35},\ P_{PC5}),\\ &(P_{PC4},\ t_{45PC}),\ (t_{45PC},\ P_{PC5}),\ (P_{PC5},\ t_{54PC}),\ (t_{54PC},\ P_{PC4})\} \end{aligned}$$

The weights of the arcs are shown below

	t_{12}	t_{21}	t_{23}	t_{32}	t_{41}	t_{35}	t_{45PC}	t_{54PC}
P_{PC1}	1	0	0	0	0	0	0	0
P_{PC2}	0	1	1	0	0	0	0	0
P_{PC3}	0	0	0	1	0	1	0	0
P_{PC4}	0	0	0	0	1	0	1	0
P_{PC5}	0	0	0	0	0	0	0	1

	t_{12}	t_{21}	t_{23}	t_{32}	t_{41}	t_{35}	t_{45PC}	t_{54PC}
P_{PC1}	0	1	0	0	1	0	0	0
P_{PC2}	1	0	0	1	0	0	0	0
P_{PC3}	0	0	1	0	0	0	0	0
P_{PC4}	0	0	0	0	0	0	0	1
P_{PC5}	0	0	0	0	0	1	1	0

Based on the weight of arcs, the input incident matrix B^- which indicates the weight of those arcs from places to transitions and the output incident matrix B^+ which indicates the weight of those arcs from transitions to places are given by

$$B_{PC}^{-} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \; B_{PC}^{+} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

With the input incident matrix and output incident matrix, incident matrix can be calculated for the pedestrian/cyclist Petri net structure.

$$B_{PC} = B_{PC}^{+} - B_{PC}^{-} = \begin{array}{c} \\ P_{PC1} \\ P_{PC2} \\ P_{PC3} \\ P_{PC4} \\ P_{PC5} \end{array} \begin{array}{c} \begin{array}{cccccccc} t_{12} & t_{21} & t_{23} & t_{32} & t_{41} & t_{35} & t_{45PC} & t_{54PC} \end{array} \\ \begin{bmatrix} -1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix} \end{array}$$

Then, the pedestrian/cyclist model can be drawn as shown in Fig. 4.8, which includes all the places in the place set, transitions in the transition set, and arcs in the arc set of pedestrian/cyclist Petri net model.

As the moving direction defined at the beginning of this section, pedestrian/cyclist were considered mainly focus on the moving in the crossing road direction, which means that pedestrian/cyclist can move freely among place 1 to place 3 and place 4 to place 5 in dual directions but can move in one direction from place 4 to place 1 and from place 3 to place 5. In this pedestrian/cyclist Petri net model in the crossing road scenario, the event is defined for each transition in Table 4.4, which can satisfy the model developed for the crossing road scenario.

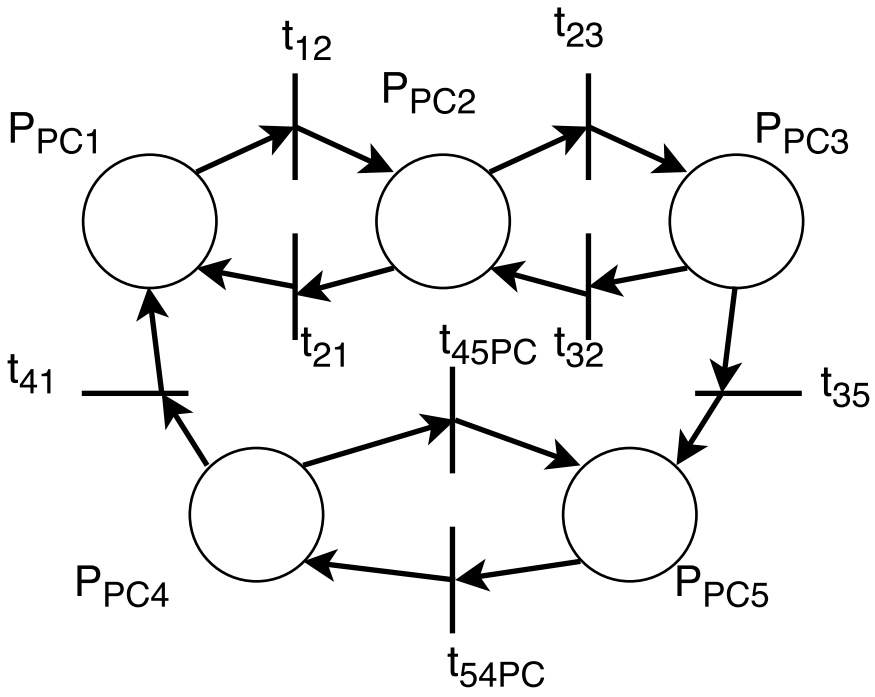

Fig. 4.8. Pedestrian/cyclist Petri net graph in crossing road scenarios

Table 4.4.
Events for Pedestrian/Cyclist Transitions in Potential Collision Area

Transition	**Event**
t_{12}	Crossing road moving from place1 to place2
t_{21}	Crossing road moving from place2 to place1
t_{23}	Crossing road moving from place2 to place3
t_{32}	Crossing road moving from place3 to place2
t_{41}	Along road moving from place4 to place1
t_{35}	Along road moving from place4 to place2
t_{45PC}	Crossing road moving from place4 to place5
t_{54PC}	Crossing road moving from place5 to place4

4.3.3 Controller and Results

The combined Petri net graph is shown in Fig. 4.9. For this combined Petri net

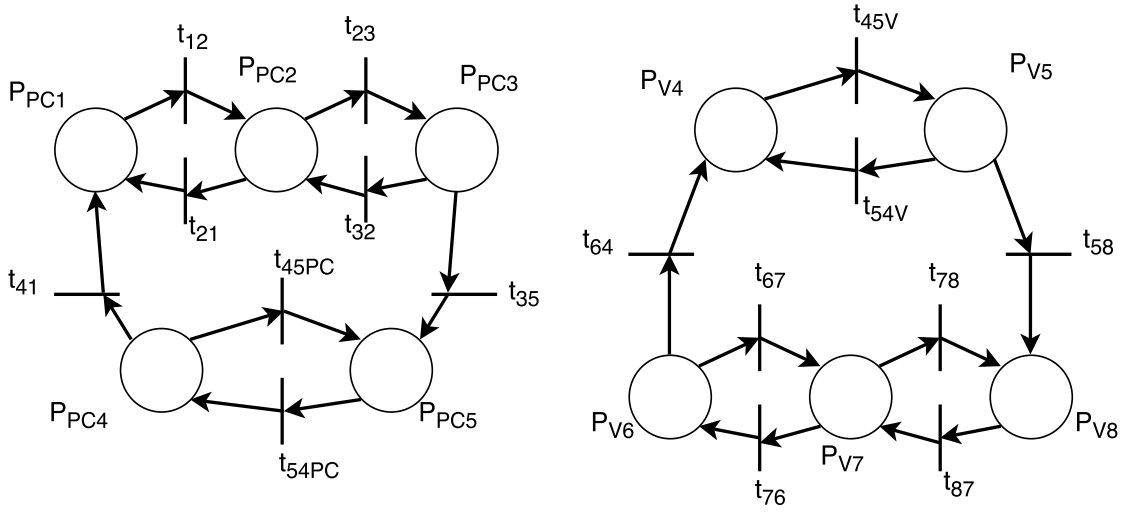

Fig. 4.9. Combined Petri net in crossing road scenarios

model, the place set $P = \{P_V, P_{PC}\}$, the transition set $T = \{T_V,\ T_{PC}\}$, and the arcs $A = \{A_V,\ A_{PC}\}$, where,

$$P = \{P_{V4},\ P_{V5},\ P_{V6},\ P_{V7},\ P_{V8},\ P_{PC1},\ P_{PC2},\ P_{PC3},\ P_{PC4},\ P_{PC5}\},$$

$$T = \{t_{45V},\ t_{54V},\ t_{64},\ t_{58},\ t_{67},\ t_{76},\ t_{78},\ t_{87},\ t_{12},\ t_{21},\ t_{23},\ t_{32},\ t_{41},\ t_{35},\ t_{45PC},\ t_{54PC}\},$$

$$\begin{aligned} A = \{&(P_{V4},\ t_{45V}),\ (t_{45V},\ P_{V5}),\ (P_{V5},\ t_{54V}),\ (t_{54V},\ P_{V4}),\\ &(P_{V6}, t_{64}),\ (t_{64},\ P_{V4}),\ (P_{V6},\ t_{67}),\ (t_{67},\ P_{V7}),\\ &(P_{V7}, t_{76}),\ (t_{76},\ P_{V6}),\ (P_{V7},\ t_{78}),\ (t_{78},\ P_{V8}),\\ &(P_{V8},\ t_{87}),\ (t_{87},\ P_{V7}),\ (P_{V4},\ t_{58}),\ (t_{58},\ P_{V8}),\\ &(P_{PC1},\ t_{12}),\ (t_{12},\ P_{PC2}),\ (P_{PC2},\ t_{21}),\ (t_{21},\ P_{PC1}),\\ &(P_{PC2}, t_{23}),\ (t_{23},\ P_{PC3}),\ (P_{PC3},\ t_{32}),\ (t_{32},\ P_{PC2}),\\ &(P_{PC4}, t_{41}),\ (t_{41},\ P_{PC1}),\ (P_{PC3},\ t_{35}),\ (t_{35},\ P_{PC5}),\\ &(P_{PC4},\ t_{45PC}),\ (t_{45PC},\ P_{PC5}),\ (P_{PC5},\ t_{54PC}),\ (t_{54PC},\ P_{PC4})\} \end{aligned}$$

To guarantee the vehicle and pedestrian/cyclist will never stop in the same place for this crossing road scenario, a Petri net controller was designed for the combined Petri net, including the vehicle Petri net and pedestrian/cyclist Petri net. The constraint of this combined Petri net is $LM \leq b$, where

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, M = \begin{bmatrix} M(P_{V4}) \\ M(P_{V5}) \\ M(P_{V6}) \\ M(P_{V7}) \\ M(P_{V8}) \\ M(P_{PC1}) \\ M(P_{PC2}) \\ M(P_{PC3}) \\ M(P_{PC4}) \\ M(P_{PC5}) \end{bmatrix}, b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

The condition file is shown in appendix *Crossing Road Scenarios M-File*. The constraint matrices L and b are used as input variables in the condition file. Incident matrices of both vehicle Petri net model pedestrian/cyclist Petri net model are also used as input variables. Incident matrices of the combined Petri net are calculated in the condition file as follows

$$B^- = \left[\begin{array}{c|c} B_V^- & \mathbf{0} \\ \hline \mathbf{0} & B_{PC}^- \end{array}\right], \; B^+ = \left[\begin{array}{c|c} B_V^+ & \mathbf{0} \\ \hline \mathbf{0} & B_{PC}^+ \end{array}\right]$$

Then, the condition file called functions to calculate the initial state of the controller. The initial state of combine Petri net is defined as

$$M_0^\top = [0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0].$$

Then controller parameters are calculated. The incident matrix Bc and initial state Mco of the controller are shown below

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

$$Mco = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

To calculate the coverability tree of the combined Petri net with controller, the incident matrix and initial state are calculated as follows

$$BBco = \left[\frac{B^+ - B^-}{Bc}\right], M0Mco = \left[\frac{M_0}{Mco}\right]$$

which means $BBco =$

$$\left[\begin{array}{cccccccccccccccc} -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \\ 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 \\ -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 \end{array}\right]$$

$$M0Mco^\top = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

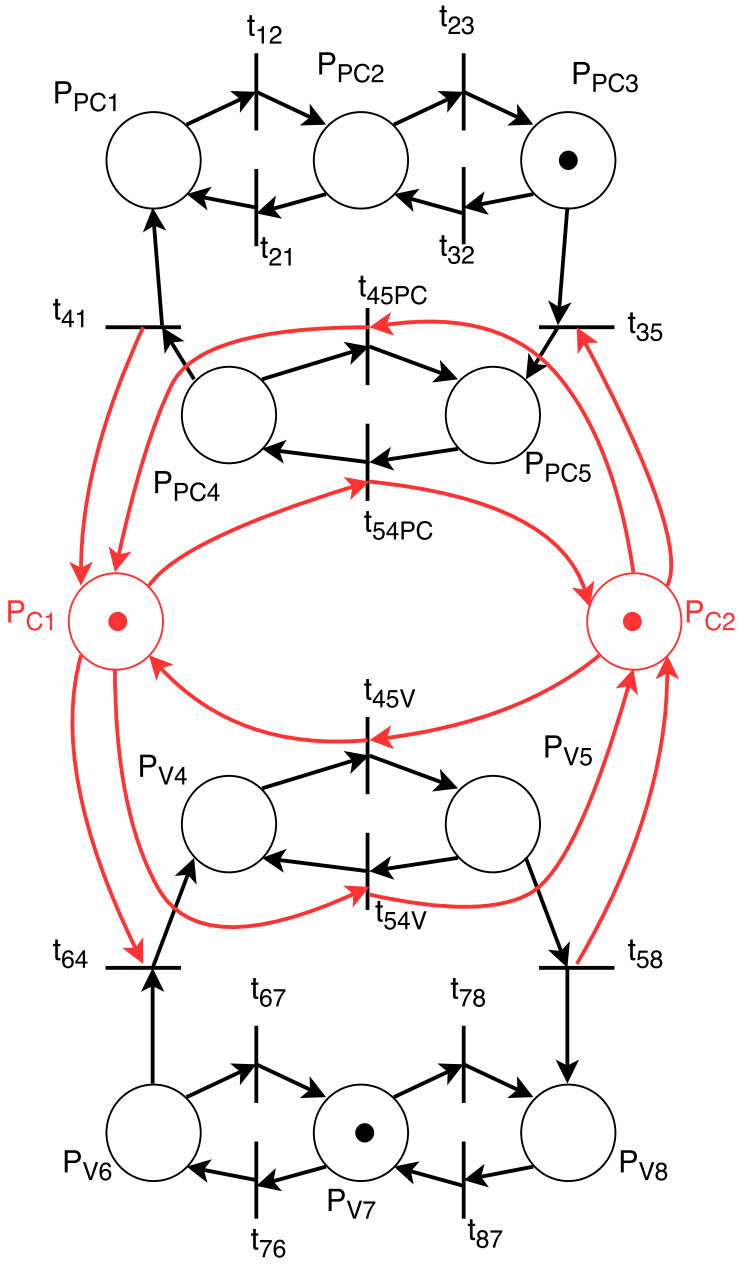

Fig. 4.10. Crossing road scenario Petri net model with controller and initial state

The structure of the combined Petri net with controller and the initial markings are shown in Fig. 4.10. There are two controllers introduced to the combined Petri net to guarantee the vehicle and pedestrian/cyclist will never stop at the same place.

After calling the functions, the coverability tree is calculated, which is shown in Fig. 4.11. The reachable states are recorded as *Rall*, where

$$Rall = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}$$

The reachable state matrix includes all states that can be reached and guarantees vehicle and pedestrian/cyclist to stop in different places. Column 1 to column 10

indicate places P_{V4} to P_{V8} and places P_{PC1} to P_{PC5}, respectively. The first line of *Rall* represents the initial state that vehicle will stop at place 7 and pedestrian/cyclist will stop at place 3.

With the initial state $[0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0]^\top$, transitions t_{76}, t_{78}, t_{32}, and t_{35} can be enabled. If transition t_{76} fires, the state changes to $[0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0]^\top$; if transition t_{78} fires, the state changes to $[0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0]^\top$; if transition t_{32} fires, the state changes to $[0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0]^\top$; if transition t_{35} fires, the state changes to $[0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1]^\top$. Following the transition sequences in the result shown in Fig. 4.11, those states in the coverability tree can be reached.

4.4 Summary

In this chapter, the MATLAB functions were introduced first. A input file, *condition file*, was used to call functions for each problem. In the input files, input variables included incident matrices to describe the Petri net structure, and constraint matrices to indicate which states will be avoided with the controller. Then, the initial state of controller and the incident matrix of the controller were be calculated. The condition file calculated the coverability tree for the combined Petri net with the controller and provided the results including the parameters of the controller and the coverability tree as text file.

In addition, the mutual exclusion problem between vehicle and pedestrian/cyclist was discussed in the potential collision area which was proposed in Fig. 3.13. Petri net model for vehicle and pedestrian/cyclist was built and the controller was designed for the combined Petri net model in the potential collision area. The parameters of the controller and the results were shown in Section 4.2

In crossing road scenario, the movement of the pedestrian/cyclist and the possible stopping area of vehicle do not overlap. For this kind of scenario, Petri net model was built and the controller was designed for the combined model in Section 4.3.

In this chapter, vehicle and pedestrian/cyclist Petri net models were built for both potential collision area and crossing road scenario. The controllers were designed with state-based control, which guaranteed the vehicle and pedestrian/cyclist will never stop in the same place to avoid the collision between vehicle and pedestrian/cyclist.

	node kinds	each states	transition sequences
1	0	0 0 0 1 0 0 0 0 1 0 0 1 1	0 0 0 0 0 0
2	0	0 0 1 0 0 0 0 0 1 0 0 1 1	6 0 0 0 0 0
3	0	0 0 0 0 1 0 0 1 0 0 0 1 1	7 0 0 0 0 0
4	0	0 0 0 1 0 0 0 1 0 0 0 1 1	12 0 0 0 0 0 0
5	0	0 0 0 1 0 0 0 0 0 0 1 1 0	14 0 0 0 0 0 0
6	0	1 0 0 0 0 0 0 0 1 0 0 0 1	6 3 0 0 0 0
7	1	0 0 0 1 0 0 0 0 1 0 0 1 1	6 5 0 0 0 0
8	0	0 0 1 0 0 0 0 1 0 0 0 1 1	6 12 0 0 0 0
9	0	0 0 1 0 0 0 0 0 0 0 1 1 0	6 14 0 0 0 0
10	1	0 0 0 1 0 0 0 0 1 0 0 1 1	7 8 0 0 0 0
11	0	0 0 0 0 1 0 1 0 0 0 0 1 1	7 12 0 0 0 0
12	0	0 0 0 0 1 0 0 0 0 1 1 0	7 14 0 0 0 0
13	1	0 0 1 0 0 0 0 0 1 0 0 1 1	12 6 0 0 0 0
14	1	0 0 0 0 1 0 1 0 0 0 0 1 1	12 7 0 0 0 0
15	0	0 0 0 1 0 1 0 0 0 0 0 1 1	12 10 0 0 0 0
16	1	0 0 0 1 0 0 0 1 0 0 0 1 1	12 11 0 0 0 0
17	1	0 0 1 0 0 0 0 0 0 0 1 1 0	14 6 0 0 0 0
18	1	0 0 0 0 1 0 0 0 0 0 1 1 0	14 7 0 0 0 0
19	0	0 0 0 1 0 0 0 0 0 1 0 0 1	14 16 0 0 0 0
20	0	0 1 0 0 0 0 0 0 1 0 0 1 0	6 3 1 0 0 0
21	0	1 0 0 0 0 0 0 1 0 0 0 0 1	6 3 12 0 0 0
22	2	1 0 0 0 0 0 0 0 0 0 1 0 0	6 3 14 0 0 0
23	1	1 0 0 0 0 0 0 1 0 0 0 0 1	6 12 3 0 0 0
24	1	0 0 0 1 0 0 0 1 0 0 0 1 1	6 12 5 0 0 0
25	0	0 0 1 0 0 0 1 0 0 0 0 1 1	6 12 10 0 0 0
26	1	0 0 1 0 0 0 0 0 1 0 0 1 1	6 12 11 0 0 0
27	2	1 0 0 0 0 0 0 0 0 0 1 0 0	6 14 3 0 0 0
28	1	0 0 0 1 0 0 0 0 0 0 1 1 0	6 14 5 0 0 0
29	0	0 0 1 0 0 0 0 0 0 1 0 0 1	6 14 16 0 0 0
30	1	0 0 0 1 0 0 0 0 1 0 0 1 1	7 12 8 0 0 0
31	0	0 0 0 0 1 1 0 0 0 0 0 1 1	7 12 10 0 0 0
32	1	0 0 0 0 1 0 0 1 0 0 0 1 1	7 12 11 0 0 0
33	1	0 0 0 1 0 0 0 0 0 0 1 1 0	7 14 8 0 0 0
34	0	0 0 0 0 1 0 0 0 1 0 0 1 1	7 14 16 0 0 0
35	1	0 0 1 0 0 1 0 0 0 0 0 1 1	12 10 6 0 0 0
36	1	0 0 0 0 1 1 0 0 0 0 0 1 1	12 10 7 0 0 0
37	1	0 0 0 1 0 0 1 0 0 0 0 1 1	12 10 9 0 0 0
38	1	0 0 1 0 0 0 0 0 0 1 0 0 1	14 16 6 0 0 0
39	1	0 0 0 0 1 0 0 0 1 0 0 1 1	14 16 7 0 0 0
40	1	0 0 0 1 0 1 0 0 0 0 0 1 1	14 16 13 0 0 0
41	1	0 0 0 1 0 0 0 0 0 1 1 0	14 16 15 0 0 0
42	1	1 0 0 0 0 0 0 1 0 0 0 0 1	6 3 1 2 0 0
43	1	0 0 0 0 1 0 0 1 0 0 0 1 1	6 3 1 4 0 0
44	0	0 1 0 0 0 0 1 0 0 0 0 1 0	6 3 1 12 0 0
45	1	0 1 0 0 0 0 1 0 0 0 0 1 0	6 3 12 1 0 0
46	0	1 0 0 0 0 1 0 0 0 0 0 0 1	6 3 12 10 0 0
47	1	1 0 0 0 0 0 0 1 0 0 0 0 1	6 3 12 11 0 0
48	1	1 0 0 0 0 0 1 0 0 0 0 0 1	6 12 10 3 0 0
49	1	0 0 0 1 0 1 0 0 0 0 0 1 1	6 12 10 5 0 0
50	1	0 0 1 0 0 0 1 0 0 0 0 1 1	6 12 10 9 0 0
51	1	0 0 0 1 0 0 0 0 1 0 0 1 1	6 14 16 5 0 0
52	1	0 0 1 0 0 1 0 0 0 0 0 1 1	6 14 16 13 0 0
53	1	0 0 1 0 0 0 0 0 0 1 1 0	6 14 16 15 0 0
54	1	0 0 0 1 0 1 0 0 0 0 0 1 1	7 12 10 8 0 0
55	1	0 0 0 0 1 0 1 0 0 0 0 1 1	7 12 10 9 0 0
56	1	0 0 0 1 0 0 0 0 0 1 0 0 1	7 14 16 8 0 0
57	1	0 0 0 0 1 1 0 0 0 0 0 1 1	7 14 16 13 0 0
58	1	0 0 0 0 1 0 0 0 0 1 1 0	7 14 16 15 0 0
59	1	1 0 0 0 0 0 0 1 0 0 0 0 1	6 3 1 12 2 0
60	1	0 0 0 0 1 0 1 0 0 0 0 1 1	6 3 1 12 4 0
61	0	0 1 0 0 0 1 0 0 0 0 0 1 0	6 3 1 12 10 0
62	1	0 1 0 0 0 0 0 1 0 0 0 1 0	6 3 1 12 11 0
63	1	0 1 0 0 0 1 0 0 0 0 0 1 0	6 3 12 10 1 0
64	1	1 0 0 0 0 0 1 0 0 0 0 0 1	6 3 12 10 9 0
65	1	1 0 0 0 0 0 1 0 0 0 0 1 1	6 3 1 12 10 2
66	1	0 0 0 0 1 1 0 0 0 0 0 1 1	6 3 1 12 10 4
67	1	0 1 0 0 0 0 1 0 0 0 0 1 0	6 3 1 12 10 9

Fig. 4.11. Coverability tree

5. CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

In this thesis, a Petri net model is proposed considering the vehicle and pedestrian/cyclist in a certain area including potential collision area and crossing road scenarios for AEB system. In the layout of potential collision area and crossing road scenarios, the movement of pedestrian/cyclist and the possible stopping places of vehicle are defined to follow arrows marked in those layouts. The Petri net models were built and the controllers are designed. With the state-based control, developed controllers guarantee vehicle and pedestrian/cyclist can never stop in the same place. The reachable states and coverability tree are discussed for the Petri net with controllers for both potential collision area and crossing road scenario.

The basic concepts of Petri nets and the processes of controller design are reviewed in Chapter 2. Examples are provided to help explain the concepts and controller designing process. In Chapter 3, AEB testing, data collection, and data analysis are introduced. The concept of certain area are proposed with descriptions.

MATLAB codes are developed to solve problems in this thesis, which are introduced in the beginning of Chapter 4. Vehicle and pedestrian/cyclist Petri net model are built for both potential collision area and crossing road scenarios. The constraints are transferred into matrix form and controllers are designed for those combined Petri net with constraints.

Finally, those controllers design can avoid Petri nets to reach states which indicate collision between vehicle and pedestrian/cyclist. The Petri nets with controller are discussed to search the reachable states and coverability tree.

5.2 Future Work

There are some future directions to extend the research work in this thesis. One possible extension is to explore more specific layout in certain area for vehicle with different braking patterns. Based on the test drive of different vehicles, it can be easily found that there are different braking patterns, especially for different vehicles with different power systems, such as gasoline vehicles, hybrid vehicles, and plug-in electric vehicles. These power systems provide more strategies to brake when the AEB system is working.

Another interesting extension is to build a the model including more pedestrian/-cyclist and vehicle rather than one vehicle and one pedestrian/cyclist. The model including more pedestrians/cyclists and vehicles is more similar to the real world and can be used to solve some complex situations.

Apart from the models, real-time algorithms and approaches are also helpful to extend the research of this thesis. For the same state, there are one or more transition sequences that can reach the state, which may have different costs. The real-time algorithms and approaches can provide solutions for the mutual exclusion problem on the road to choose the final state with the lowest cost.

REFERENCES

REFERENCES

[1] safercar.gov Powered by NHTSA. Automatic emergency braking. Last date accessed: 11/15/2017. [Online]. Available: https://www.safercar.gov/Vehicle-Shoppers/Safety-Technology/AEB/aeb

[2] M. Moon. Automakers agree to make auto braking a standard by 2022. Last data accessed: 11/15/2017. [Online]. Available: https://www.engadget.com/2016/03/19/automakers-government-auto-braking-agreement/

[3] safercar.gov Powered by NHTSA. Pedestrian automatic emergency braking. Last date accessed: 11/15/2017. [Online]. Available: https://www.safercar.gov/Vehicle-Shoppers/Safety-Technology/paeb

[4] Toyota. Pre-collision system with pedestrian detection. Last date accessed: 11/15/2017. [Online]. Available: https://www.toyota.com/safety-sense/animation/pcspd

[5] D. Quick. Volvos auto-braking detection system adds cyclists to the mix. Last data accessed: 11/15/2017. [Online]. Available: https://newatlas.com/volvo-auto-braking-cyclist-detection-system/26536/

[6] H. Mun, G. Kim, and B. Kim, "Aeb system for a curved road considering v2v-based road surface conditions," pp. 8–13, April 2015.

[7] R. A. Chandler and L. E. Wood, "System considerations for the design of radar braking sensors," *IEEE Transactions on Vehicular Technology*, vol. 26, no. 2, pp. 151–160, May 1977.

[8] A. T. Ali and E. L. Dagless, "Vehicle and pedestrian detection and tracking," in *IEE Colloquium on Image Analysis for Transport Applications*, February 1990, pp. 5/1–5/7.

[9] H. Mori, N. M. Charkari, and T. Matsushita, "On-line vehicle and pedestrian detections based on sign pattern," *IEEE Transactions on Industrial Electronics*, vol. 41, no. 4, pp. 384–391, August 1994.

[10] C. Curio, J. Edelbrunner, T. Kalinke, C. Tzomakas, and W. von Seelen, "Walking pedestrian recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 3, pp. 155–163, September 2000.

[11] D. M. Gavrila, "Sensor-based pedestrian protection," *IEEE Intelligent Systems*, vol. 16, no. 6, pp. 77–81, November 2001.

[12] D. M. Gavrila, U. Franke, C. Wohler, and S. Gorzig, "Real time vision for intelligent vehicles," *IEEE Instrumentation Measurement Magazine*, vol. 4, no. 2, pp. 22–27, June 2001.

[13] F. Xu, X. Liu, and K. Fujimura, "Pedestrian detection and tracking with night vision," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 1, pp. 63–71, March 2005.

[14] X. Li, L. Li, F. Flohr, J. Wang, H. Xiong, M. Bernhard, S. Pan, D. M. Gavrila, and K. Li, "A unified framework for concurrent pedestrian and cyclist detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 2, pp. 269–281, February 2017.

[15] C. Zhou and J. Yuan, "Multi-label learning of part detectors for heavily occluded pedestrian detection," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2017.

[16] J. Lee, S. Lee, Y. Kim, J. Lee, and J. Kim, "Refine pedestrian detections by referring to features in different ways," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 418–423.

[17] A. D. Costea and S. Nedevschi, "Semantic channels for fast pedestrian detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 2360–2368.

[18] M. K. Kocamaz, J. Gong, and B. R. Pires, "Vision-based counting of pedestrians and cyclists," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2016, pp. 1–8.

[19] L. Huang, J. Wu, F. You, Z. Lv, and H. Song, "Cyclist social force model at unsignalized intersections with heterogeneous traffic," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 782–792, April 2017.

[20] A. Kassim, K. Ismail, and S. Woo, "Modeling cyclists speed at signalized intersections: Case study from ottawa, canada," in *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, June 2017, pp. 639–644.

[21] S. Zernetsch, S. Kohnen, M. Goldhammer, K. Doll, and B. Sick, "Trajectory prediction of cyclists using a physical model and an artificial neural network," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, June 2016, pp. 833–838.

[22] E. A. I. Pool, J. F. P. Kooij, and D. M. Gavrila, "Using road topology to improve cyclist path prediction," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 289–296.

[23] M. K. Park, S. Y. Lee, C. K. Kwon, and S. W. Kim, "Design of pedestrian target selection with funnel map for pedestrian aeb system," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 5, pp. 3597–3609, May 2017.

[24] V. R. Garate, R. Bours, and K. Kietlinski, "Numerical modeling of ada system for vulnerable road users protection based on radar and vision sensing," in *2012 IEEE Intelligent Vehicles Symposium*, June 2012, pp. 1150–1155.

[25] C. G. Keller, T. Dang, H. Fritz, A. Joos, C. Rabe, and D. M. Gavrila, "Active pedestrian safety by automatic braking and evasive steering," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1292–1304, December 2011.

[26] D. F. Llorca, V. Milanes, I. P. Alonso, M. Gavilan, I. G. Daza, J. Perez, and M. . Sotelo, "Autonomous pedestrian collision avoidance using a fuzzy steering controller," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 390–401, June 2011.

[27] S. Y. Gelbal, S. Arslan, H. Wang, B. Aksun-Guvenc, and L. Guvenc, "Elastic band based pedestrian collision avoidance using v2x communication," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 270–276.

[28] C. Su, W. Deng, H. Sun, J. Wu, B. Sun, and S. Yang, "Forward collision avoidance systems considering driver's driving behavior recognized by gaussian mixture model," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 535–540.

[29] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[30] L. Li, *Estimation, diagnosis, and control in discrete event systems in the presence of observability constraints and faults*. Ph.D. Dissertation University of Illinois at Urbana-Champaign, 2008.

APPENDICES

A. M-FUNCTIONS

```
function incident=incident(Boutput,Binput)
%%
% INCIDENT Incident matrix of petri net
%   incident=incident(Boutput,Binput) calculate the incident
%   matrix
%     Def:
%         B=Boutput-Binput
%         B represents the incident matrix
%         Binput represents the input incident matrix
%         Boutput represents the output incident matrix
%
%   see also controlledpetri inicon petricon transition
%   Copyright Wensen Niu for MSthesis @2017

%%
% Checking number of input arguments for this function
% If the number of input arguments great than 2, program
% feedbacks the error message below
if nargin>2
    error('Too many input arguments')
end
% 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


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
%     Copyright Wensen Niu for MSthesis @2017

%%
% Checking number of input arguments for this function
% If the number of input arguments great than 4, program
% feedbacks the error message below.
if nargin>4
    error('Too many input arguments');
end
% If the number of input arguments less than 4, program feedbacks
% the error message below
if nargin<4
    error('This function should have four inputs');
end
%%
if nargin==4%If the number of input arguments equals to 4
%     Saving size value of L, b, M0, and B to it corresponding
%     variable prepare for the logic check.
    sizeL=size(L);%check the size of L
    sizeb=size(b);%check the size of b
    sizeM0=size(M0);%check the size of M0
    sizeB=size(B);%check the size of B
%     Getting logic value for error checking
    inicon.boolean.logisignA1=sizeM0(1)==sizeB(1);
%     Does rows of M0 equals to rows of B?
    inicon.boolean.logisignA2=sizeM0(2)==1;
%     Does M0 is a column vector?
    inicon.boolean.logisignA=inicon.boolean.logisignA1&& ...
        inicon.boolean.logisignA2;
%     A equals to "A1 and A2"
    inicon.boolean.logisignB=sizeL(2)==sizeB(1);
```

```
%     Does columns of L equal to rows of B?
    inicon.boolean.logisignC1=sizeb(1)==sizeL(1);
%     Does rows of b equals to rows of L?
    inicon.boolean.logisignC2=sizeb(2)==sizeM0(2);
%     Does b is a column vector?
    inicon.boolean.logisignC=inicon.boolean.logisignC1&& ...
        inicon.boolean.logisignC2;
%     C equals to "C1 and C2"
%     Checking logic values.
    if inicon.boolean.logisignA==0
%         If rows of M0 does not equal to rows of B, program
%         feedbacks the error message below. If M0 is not a
%         column vector, program also feedbacks the error message
%         below.
        error('Rows of column vector M0 should equal to as rows of B')
    elseif inicon.boolean.logisignB==0
%         If columns of L does not equal to rows of B, program
%         feedbacks the error message below
        error('Columns of coefficient matrix of constraint should ...
            equal to rows of B')
    elseif inicon.boolean.logisignC==0
%         If rows of b does not equal to rows of L or b is not a
%         column vector, program feedbacks the error message
%         below.
        error('b should be a scale or a column vector with rows same ...
            as rows of L')
    else
%         According to state-based control of Petri nets,
%         calculate the incident matrix of controller and initial
%         state of controller to design a controller for a Petri
%         net.
        inicon.Bc=-L*B;
        inicon.Mco=b-L*M0;
%         For this function, it will return the inicon structure
%         for its main program. The inicon structure include the
```

```
%         incident matrix of controller that called inicon.Bc and
%         the initial state of controller that called inicon.Mco.

    end
end


function controlledpetri=controlledpetri(Boutput,Binput,Bc,Mco,M0)
%%
% CONTROLLEDPETRI Generate a "controlled petrinet"
%   controlledpetri=controlledpetri(Boutput,Binput,Bc,Mco,M0)
%   calculate the BBco, Bcpinput, Bcpoutput, and M0Mco of a
%   "controlled Petri net"
%     Def:
%         BBco:The incident matrix of the "controlled Petri net"
%         Bcpinput: The input incident matrix of the "controlled
%         Petri net"
%         Bcpoutput: The output incident matrix of the
%         "controlled Petri net"
%         M0Mco: The initial state of the "controlled Petri net"
%
%   see also incident inicon petricon transition
%   Copyright Wensen Niu for MSthesis @2017

%%
% Jusge number of input arguments for this function.
% If the number of input argument great than 5, program feedbacks
% the error message below.
if nargin>5
    error('Too many input arguments')
end
% If the number of output argument less than 5, program feedbacks
% the error message below.
if nargin<5
    error('Five input arguments should be given for this function')
```

```
end
%%
if nargin==5%If the number of input arguments equals to 5
    sizeBoutput=size(Boutput);%check the size of Boutput
    sizeBinput=size(Binput);%check the size of Binput
    sizeBc=size(Bc);%check the size of Bc
    boolean.A=sizeBoutput==sizeBinput;
%     Does size of Boutput equals to size of Binput?
    boolean.B=sizeBc(2)==sizeBinput(2);
%     Does columns of Bc equals to columns of Binput?
    if boolean.A==0
%         If size of Boutput does not equal to size of Binput,
%         program feedbacks the error message below.
        error('Boutput and Binput should have same dimension')
    elseif boolean.B==0
%         If columns of Bc does not equal to columns of Binput,
%         programs feedbacks the error message below.
        error('Bc should has same columns with Binput and Boutput')
    else
        signBc=sign(Bc);%checking sign of each entry of Bc
%         Generating two zeros matrix with the dimension same as
%         Bc's dimension.
        controlledpetri.Bcinput=zeros(sizeBc);
        controlledpetri.Bcoutput=zeros(sizeBc);
%         A "for" loop for setting value for entries of Bcinput,
%         the input incident matrix of controller, and Bcoutput,
%         the output incident matrix of controller.
        for i=1:sizeBc(1)%"for" loop for rows
            for j=1:sizeBc(2)%"for" loop for columns
                if signBc(i,j)==-1
                    controlledpetri.Bcinput(i,j)= ...
                        controlledpetri.Bcinput(i,j)-Bc(i,j);
                    controlledpetri.Bcoutput(i,j)= ...
                        controlledpetri.Bcoutput(i,j);
                elseif signBc(i,j)==0
```

```
                    controlledpetri.Bcinput(i,j)= ...
                        controlledpetri.Bcinput(i,j);
                    controlledpetri.Bcoutput(i,j)= ...
                        controlledpetri.Bcoutput(i,j);
                elseif signBc(i,j)==1
                    controlledpetri.Bcinput(i,j)= ...
                        controlledpetri.Bcinput(i,j);
                    controlledpetri.Bcoutput(i,j)= ...
                        controlledpetri.Bcoutput(i,j)+Bc(i,j);
                end
            end
        end
%         Calculating Bcpinput, Bcpoutput, B, BBco, and M0Mco and
%         puting them into structure controlledpetri.
        controlledpetri.Bcinput;
        controlledpetri.Bcoutput;
        controlledpetri.Bcpinput=[Binput;controlledpetri.Bcinput];
        controlledpetri.Bcpoutput=[Boutput;controlledpetri.Bcoutput];
        controlledpetri.B=incident(Boutput,Binput);
        controlledpetri.BBco=[controlledpetri.B;Bc];
        controlledpetri.M0Mco=[M0;Mco];
%         For this function, it will return structure
%         controlledpetri to its main program.
    end
end


 function transition=transition(Bcpinput,BBco,M0Mco)
%%
% TRANSITION Transition of a "Controlled Petri net"
%   transition=transition(Bcpinput,BBco,M0Mco) calculate all
%   states in Coverability Tree.
%     Def:
%         Bcpinput: input incident matrix of "Controlled Petri
%         net".
```

```
%         BBco: incident matrix of "Controlled Petri net".
%         M0Mco: initial state of "Controlled Petri net".
%
%   see also controlledpetri incident inicon petricon
%   Copyright Wensen Niu for MSthesis @2017

%%
sizeBcpinput=size(Bcpinput);%checking the size of Bcpinput

% Generating a identity matrix with the dimension of columns of
% Bcpinput by the columns of Bcpinput
V=eye(sizeBcpinput(2));

% initialize M0McoT, which used to store state(s) for next
% time fire by enabled transition, to the initial state.
M0McoT=M0Mco';
sizeM0McoT=size(M0McoT);%checking the size of M0McoT
Mi=1;% initialize Mi to 1
% Mi is used as a pointer to point the row of Mall which help to
% store all state into correct place in Mall.
transition.Mall(Mi,:)=M0McoT;%store initial state into Mall

Ti=1;% initialize Ti to 1
% Ti is used as a pointer to point the row of Tall which help to
% store all transition sequence into correct place in Tall.
transition.Tall(Ti)=0;
% store 0 as there is no transition sequence to the initial state

%%
%DT(i)=0 represents the state is a fireable state.
%DT(i)=1 represents the state is a duplicate node.
%DT(i)=2 represents the state is a terminal node.
DTi=1;%initialize DTi to 1
% DTi is used as a pointer to point the row of DT which help to
% store kind of all state into correct place in DT.
```

```
% Flag is used as a flag to determine whether continue to next
% fire. If the flag equal to 0, program stop.

% Ddetect is used to store DT value in one transition, Which is
% used to determine whether to fire next transition for all state
% obtained in last time transition.

%%
% "for" loop is used to determin the initial value of
% transition.DT(DTi,:), Ddetect, and flag.
for i=1:sizeM0McoT(1)
    for j=1:sizeBcpinput(2)
        if M0McoT(i,:)'<Bcpinput(:,j)
            transition.DT(DTi,:)=2;
            Ddetect=1;
            flag=0;
        else
            transition.DT(DTi,:)=0;
            Ddetect=0;
            flag=1;
        end
    end
end

% Dtistart is used to help store transition sequence to Tall.
% Initialize DTistart to 2, since we will not use it at very
% begin of this program.
DTistart=2;
%%
while flag≠0
%     initialize Tistart to DTistart in each time of loop
    Tistart=DTistart;
%     check the size of M0McoT for each time of loop
    sizeM0McoT=size(M0McoT);
```

```
%     n helps to store M to M0Mco, initialize it to 1 at begining
%     of each time of loop.
    n=1;
% start a "for" loop from 1 to the rows of M0McoT
    for i=1:sizeM0McoT(1)
% Use Ddetect to determine whether the state can be fire by any
% transition of the Petri net.
        if Ddetect(i)==0
%             So the state can be fired by some transition
            for j=1:sizeBcpinput(2)
%                 Start a "for" loop from 1 to the columns of
%                 Bcpinput, the input incident matrix of
%                 "Controlled Petri net.
                if M0McoT(i,:)' ≥Bcpinput(:,j)
%                   If the transpose of state i can be fired by
%                   the j transition,
                    sizeTallold=size(transition.Tall);
                    Ti=Ti+1;
                    sizeTall=size(transition.Tall);
%                   use flag to help storing transition
%                   sequence to Tall.
                    if flag≥2
                        for f=1:flag-1
                            transition.Tall(Ti,f)= ...
                                transition.Tall(Tistart,f);
                        end
                    end
                    transition.Tall(Ti,flag)=j;
%                   iterate DTistart
                    sizeTallnew=size(transition.Tall);
                    if sizeTallnew(2)==sizeTallold(2)+1
                        DTistart=Ti;
                    end
%                   storing M to Mall
                    M=M0McoT(i,:)'+BBco*V(:,j);
```

```
                Mi=Mi+1;
                transition.Mall(Mi,:)=M';

                sizeMall=size(transition.Mall);
%                 iterate DTi
                DTi=DTi+1;
                transition.DT(DTi,:)=0;
%                 storing the value that represents the state
%                 kind to DT.
                for k=1:sizeMall(1)-1
%                     Checking whethe the state is a
%                     duplicate state. If it is a duplicate
%                     state, writing 1 to the corresponding
%                     place in DT.
                    if M'==transition.Mall(k,:)
                        transition.DT(DTi,:)=1;
                    end
                end
%                 initialization Mdetect to a zeros matrix,
%                 which is used to check whether the state is
%                 a terminal state. If it is a terminal
%                 state, writing 2 to the corresponding place
%                 in DT.
                Mdetect=zeros(1,sizeBcpinput(2));
                for m=1:sizeBcpinput(2)%"for" loop
%                     If M less than each column of Bcpinput,
%                     it is a terminal state. each m of M
%                     will be written to 1
                    if M≥Bcpinput(:,m)
                        Mdetect(1,m)=0;
                    else
                        Mdetect(1,m)=1;
                    end
                end
                if Mdetect≥1
```

```
                        transition.DT(DTi)=2;
                    else
                        transition.DT(DTi)=transition.DT(DTi);
                    end
                    M0Mco(:,n)=M;%store M to M0Mco
                    n=n+1;
                end
            end
        end
        Tistart=Tistart+1;
    end
    M0McoT=[];
    M0McoT=M0Mco';
    M0Mco=[];
    Ddetect=[];
    Ddetect=transition.DT(DTistart:DTi,:);
    if Ddetect≥1
        flag=0;
    else
        sizeTall=size(transition.Tall);
        flag=flag+1;
    end
end


function petricon=petricon(Boutput,Binput,L,b,M0)
%%
% PETRICON Petri net with controller
%   petricon=petricon(Boutput,Binput,L,b,M0)
%       is used to calculate parameters of petrinet with
%       controller Parameters:
%       petricon.petri.B represents the incident matrix
%       petricon.inicon is a structure that includes initial
%       states of controller
%       Boutput represents the output incident matrix
```

```
%
%   see also controlledpetri incident inicon transition
%   Copyright Wensen Niu for MSthesis @2017

%%
% Checking number of input arguments.
if nargin≠5
%     If the number does not equal to 5, program feedbacks the
%     error message below.
    error('Pleas check number of input arguments. It should be 5')
end
%%
if nargin==5%if the number of input arguments equals to 5
    sizeBoutput=size(Boutput);%check the size of Boutput
    sizeBinput=size(Binput);%check the size of Binput
    sizeL=size(L);%check the size of L
    sizeb=size(b);%check the size of b
    sizeM0=size(M0);%check the size of M0
%     Getting logic value for error checking
    petricon.boolean.logisignA1=sizeBinput(1)==sizeBoutput(1);
%     Does rows of Binput equals to rows of Boutput?
    petricon.boolean.logisignA2=sizeBinput(2)==sizeBoutput(2);
%     Does columns of Binput equals to columns of Boutput?
    petricon.boolean.logisignA=petricon.boolean.logisignA1&& ...
        petricon.boolean.logisignA2;
%     A equals to "A1 and A2"
    petricon.boolean.logisignB=sizeL(2)==sizeBoutput(1);
%     Does columns of L equals to rows of Boutput?
    petricon.boolean.logisignC=sizeL(1)==sizeb(1);
%     Does rows of L equals to rows of b?
    petricon.boolean.logisignD=sizeM0(1)==sizeBoutput(1);
%     Does rows of M0 equals to rows of Boutput?
%     Checking logic values.
    if petricon.boolean.logisignA==0
%         If size of Binput does not equals to size of Boutput,
```

```
%         program feedbacks the error message below.
        error('Size of Binput should equals to size of Boutput');
    elseif petricon.boolean.logisignB==0
%         If columns of L does not equal to rows of Boutput,
%         program feedbacks the error message below.
        error('Columns of L should equals to rows of Boutput')
    elseif petricon.boolean.logisignC==0
%         If rows of L does not equal to rows of b, program
%         feedbacks the error message below.
        error('Rows of L should equals to rows of b')
    elseif petricon.boolean.logisignD==0
%         If rows of M0 does not equal to rows of Boutput,
%         program feedbacks the error message below.
        error('Please check initial state. It should be a column ...
           vector and has same row(s) with row(s) of Boutput')
    else%If there is no error, continue to calculating
%         Calling incident function calculates incident matrix of
%         input incident matrix and output incident matrix of
%         Petri net model.
        petricon.petri.B=incident(Boutput,Binput);
%         Calling inicon function calculates parameters of
%         controller for Petri net
        petricon.controller=inicon(L,b,M0,petricon.petri.B);
        petricon.controller.Bc;
        petricon.controller.Mco;
%         Calling controlledpetri function calculates parameters
%         of "Controlled Petri net"
        petricon.controlledpetri=controlledpetri(Boutput,Binput, ...
            petricon.controller.Bc,petricon.controller.Mco,M0);
        petricon.controlledpetri.Bcpinput;
        petricon.controlledpetri.Bcpoutput;
        petricon.controlledpetri.BBco;
        petricon.controlledpetri.M0Mco;
%         Calling transition function calculate Coverability Tree
        petricon.transition=transition( ...
```

```
            petricon.controlledpetri.Bcpinput, ...
            petricon.controlledpetri.BBco, ...
            petricon.controlledpetri.M0Mco);
    end
end
```

B. AN EXAMPLE CONDITION M-FILE

```
% This is An Example Condition File
clc
clearvars
Binput=[1 0 0 0;0 0 1 0;0 1 0 1]
Boutput=[0 0 0 1;1 1 0 0;0 0 1 0]
L=[0 1 1 ]
b=2
M0=[4 0 0]'

% Calculating parameters for Controller.
petricon=petricon(Boutput,Binput,L,b,M0);
Bc=petricon.controller.Bc
Mco=petricon.controller.Mco

% Building the Petri net with Controller.
Bcpinput=petricon.controlledpetri.Bcpinput
Bcpoutput=petricon.controlledpetri.Bcpoutput
BBco=petricon.controlledpetri.BBco
M0Mco=petricon.controlledpetri.M0Mco

%%%%%%%%%%results%%%%%%%%%%

% Calculate the Coverability Tree
Mall=petricon.transition.Mall
Tall=petricon.transition.Tall
DT=petricon.transition.DT

% output the result as a file.
```

```
sizeDT=size(DT);
head=' node      each         transition \n type    states        ...
   sequences';
fid=fopen('wensenExample.txt','wt');
fprintf(fid,[head '\n']);

for i=1:sizeDT(1)
    fprintf(fid,' %d        %d %d %d %d        %d %d %d %d %d\n',...
        DT(i,1),Mall(i,:),Tall(i,:));
end
fclose(fid);
```

C. POTENTIAL COLLISION AREA M-FILE

```
clc
clearvars
%%
% input initial conditions for petri net in this section
% input incident matrix of pedestrian/cyclist Petri net
Bvinput=[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 pedestrian/cyclist Petri net
Bvoutput=[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 vehicle Petri net model
Bpcinput=[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 vehicle Petri net model
Bpcoutput=[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 for combined Petri net
Binput=[Bvinput,zeros46;zeros46,Bpcinput]
% calculate output incident matrix for combined Petri net
Boutput=[Bvoutput,zeros46;zeros46,Bpcoutput]
% 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 combined 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
M0Mco=petricon.controlledpetri.M0Mco
% all states, all transition sequences, and the kind of state
Mall=petricon.transition.Mall%each row is a marking state
Tall=petricon.transition.Tall
% each row of Tall represents a transition sequence to the
% corresponding row in Mall.
DT=petricon.transition.DT
% each row of DT represents the kind of state to the
% corresponding row in Mall
%%
% This section generate a txt file for the results
%%%%%%%%%%show data%%%%%%%%%%
sizeDT=size(DT);
head=' node      each                        transition \n kinds    ...
   states                       sequences';
fid=fopen('PCA41.txt','wt');
fprintf(fid,[head '\n']);
RS=0;
for i=1:sizeDT(1)
    fprintf(fid,' %d        %d %d %d %d %d %d %d %d %d %d %d %d ...
                %d %d %d \n',...
```

```
        DT(i,1),Mall(i,:),Tall(i,:));
    if DT(i,1)≠1
        RS=RS+1;
    end
end
fclose(fid);
RS
```

D. CROSSING ROAD SCENARIO M-FILE

```
clc
clearvars
%%
% input initial conditions for petri net in this section
% input incident matrix of vehicle Petri net

Bvinput=[1 0 0 0 0 0 0 0;
    0 1 0 1 0 0 0 0;
    0 0 1 0 1 0 0 0;
    0 0 0 0 0 1 1 0;
    0 0 0 0 0 0 0 1;];
% output incident matrix of pedestrian/cyclist Petri net
Bvoutput=[0 1 1 0 0 0 0 0;
    1 0 0 0 0 0 0 0;
    0 0 0 0 0 1 0 0;
    0 0 0 0 1 0 0 1;
    0 0 0 1 0 0 1 0;];
% input incident matrix of vehicle Petri net model

Bpcinput=[1 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 1 0 1 0;
    0 0 0 0 0 0 0 1;];
%  output incident matrix of vehicle Petri net model
Bpcoutput=[0 1 0 0 1 0 0 0;
    1 0 0 1 0 0 0 0;
    0 0 1 0 0 0 0 0;
```

```
    0 0 0 0 0 0 0 1;
    0 0 0 0 0 1 1 0;];
% generating a zeros matrix which has same dimension with above
% matrix's dimension
zeros58=zeros(5,8);
% calculate input incident matrix for combined Petri net
Binput=[Bvinput,zeros58;zeros58,Bpcinput]
% calculate output incident matrix for combined Petri net
Boutput=[Bvoutput,zeros58;zeros58,Bpcoutput]
% generate a 2 by 2 identical matrix
eye2=eye(2);
% generate a 2 by 3 zero matrix
zero23=zeros(2,3);
% input coefficient matrix of constraint
L=[eye2,zero23,zero23,eye2]
% generate a 2 by 1 ones matrix
ones21=ones(2,1);
% input column vector of constraint
b=ones21
% input initial state of combined Petri net
M0=[0 0 0 1 0 0 0 1 0 0]'
% M0=[0 0 1 0 0 1 0 0 0 0]'
% M0=[0 0 1 0 0 1 0 0 0 0]'
% M0=[0 0 1 0 0 1 0 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
M0Mco=petricon.controlledpetri.M0Mco
% all states, all transition sequences, and the kind of state
Mall=petricon.transition.Mall%each row is a marking state
Tall=petricon.transition.Tall
% each row of Tall represents a transition sequence to the
% corresponding row in Mall.
DT=petricon.transition.DT
% each row of DT represents the kind of state to the
% corresponding row in Mall
%%
% This section generate a txt file for the results
%%%%%%%%%%show data%%%%%%%%%%
sizeDT=size(DT);
head=' node      each                          transition \n kinds    ...
   states                        sequences';
fid=fopen('CRS73.txt','wt');
fprintf(fid,[head '\n']);
RS=0;
j=1;
for i=1:sizeDT(1)
    fprintf(fid,' %d        %d %d %d %d %d %d %d %d %d %d %d %d ...
                %d %d %d %d %d %d \n',...
        DT(i,1),Mall(i,:),Tall(i,:));
    if DT(i,1)≠1
        RS=RS+1;
        Rall(j,:)=Mall(i,:);
        j=j+1;
    end
end
fclose(fid);
Rall
```