PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By KEERTHANAA RAMESH

Entitled
MODELING AND SIMULATION OF AN AUTOMATED PARALLEL PARKING SYSTEM USING HYBRID PETRI NETS

For the degree of Master of Science in Electrical and Computer Engineering

Is approved by the final examining committee:

DR. LINGXI LI
Chair

DR. BRIAN S. KING
Co-chair

DR. MAHER E. RIZKALLA
Co-chair

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): DR. LINGXI LI

Approved by: DR. BRIAN S. KING 10/30/2015
Head of the Departmental Graduate Program Date

MODELING AND SIMULATION OF AN AUTOMATED PARALLEL PARKING SYSTEM USING HYBRID PETRI NETS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Keerthanaa Ramesh

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

December 2015

Purdue University

Indianapolis, Indiana

To my family, for their love and support.

ACKNOWLEDGMENTS

I would like to dedicate this work to my parents, Usha Andal Ramesh and Ramesh Sowminarayanan and my brother Ekanth Ramesh. Words are not enough to describe the love and support as well as the unwavering faith they had in me throughout my life. I would also like to thank my husband, Hariram Satakopan, who has been my pillar of support throughout the 2.5 years of my graduate studies. His limitless patience and never ending faith in me was instrumental in enabling me to complete my masters degree with flying colors. I want to dedicate this work to him as well, as without him all this would not have been possible. Another thanks goes to my father in law Dr. V.N.Satakopan and my mother in law Hema Satakopan, for their encouragement from time to time.

I would like to thank my advisor Dr. Lingxi Li for his valuable help and guidance in completing this thesis successfully. Inspite of his hectic schedule, he was always there to help me out when I wanted to meet with him and his valuable suggestions from time to time helped me to better my work. I would also like to thank my committee members Dr. Brian King and Dr. Maher Rizkalla for their suggestions during the preparation of this work. A special thank you goes to Dr. Brian King, the Interim Chair of the ECE department, for his kind words of constant encouragement and guidance from the beginning of my graduate studies till date. I am thankful to Sherrie Tucker for her time and help in formatting my work. Her helpful reminders about the deadlines that have to be met kept me on track with my work. I would also thank my friends Cansu Sener and Mayuresh Bhasagre for helping me out, especially Mayuresh without whom it would have taken another month to finish this report. I would finally like to thank the department staffs and the professors for their guidance and help throughout the two and half years of my masters degree.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Ramesh, Keerthanaa M.S.E.C.E, Purdue University, December 2015. Modeling and simulation of an automated parallel parking system using Hybrid Petri nets. Major Professor: Lingxi Li.

In recent years, there have been a lot of technology innovations to automate the day to day processes done by every person. These days the automobile manufacturers introduce new features in their cars, in order to improve customer experience, like *Adaptive cruise control, Parallel park assist*, etc. The objective of this thesis is to model an automated parallel parking system and to simulate the system behavior, by taking into account the high level events which happen when a car is parallel parked. The tool used in this thesis to model and simulate the system is Hybrid Petri net (HPN), which is versatile to model the real life systems. Chapter 1 deals with a brief introduction of the related work in Hybrid Petri net modeling of real life systems, automatic parallel parking systems and how the concept for modeling the parallel parking system was developed. Chapter 2 deals with the general introduction about Discrete, Continuous and Hybrid Petri nets and their dynamics which are essential for understanding this thesis. Chapter 3 deals with the development of the model and the various stages in the model development. Errors encountered in each stage is briefly discussed and the improvements are discussed in the next stage of development. This chapter concludes with the final integrated model and operation of the model. Chapter 4 deals with the discussion of results obtained when the model is tested in MATLAB and SIMHPN (which is a Matlab embedded simulation program). The results are compared, the system behavior is observed and the purpose of the thesis is justified. In Chapter 5, a conclusion is provided to summarize the entire thesis.

1. INTRODUCTION

1.1 An introduction to parallel parking

Parallel parking is a kind of parking procedure where the vehicle is parked in parallel to the directon of the road and also in line with other vehicles. The generic procedure is as follows. If the parking space is between two cars, the driver either,

- moves into the space from behind the parking space or,
- drives slightly forward to the space, parallel to the car in front of the parking space and then backs into the space, adjusting the position to complete the parallel parking procedure.

1.2 Literature review

After gaining persective on the parallel parking procedure, a detailed literature review was done on the existing research in the field of automated parallel parking systems and Hybrid Petri net modelling of various real life systems, for example, manufacturing systems, traffic control systems etc. In [14], [15] automated parallel parking systems employing various algorithms for maneuver planning were discussed. The mathematical models of the vehicles and geometric representations were used in the algorithm to perform collision free maneuvering planning. Infra-red and ultra sound sensors were used to scan the parking area to search for a suitable parking spot. The data from the sensors was also used in the path planning. The control logic/algorithm generated was then used in the embedded microprocessor used for the automated parallel parking process. These algorithms were tested using a car like robot prototype. In [20], known reference trajectory (represented by a fifth order polynomial) was adopted and a fuzzy controller was designed to assist the car like mobile robot to compute a trajectory close to the reference trajectory. The car like mobile robot followed the trajectory computed and the parking procedure was completed.

In [16] and [17], an iterative algorithm was used in feedback and planning motion control procedure, which localizes a vehicle's loaction based on the data from range measurement systems. Then proper control algorithms were used which provide an approximate path to a designated location. Feedback was received after processing the range data, after the motion has been executed. This feedback was then used to decide whether the current motion is leading to the desired trajectory. Then this algorithm was continued iteratively till the parallel parking procedure is complete and the vehicle (CLMR) has been parked in the desired location. In [19], further improvements were done to the existing path planning algorithms and a neuro fuzzy behavior based controller was designed for the steering control, whose inputs are the distances of the car and the parking space. It is different from the existing fuzzy logic based algorithms in the sense, the car like vehicle (robot) need not know the reference trajectory for path planning. Only the initial configuration of the robot is necessary for it to track it's own path based on the positions measured at discrete intervals of time. Finally, [18] deals with a complete mechatronics system to automate the parallel parking procedure for reverse parallel parking (where the vehicle parallel parks from the front of the parking space) and the forward parallel parking (where the vehicle parallel parks from behind the parking space) based on a fuzzy controller output whose inputs are the the angles between the vehicle frame orientations and the X axis.

In [12], the authors have developed a way to model the parallel parking system using discrete Petri nets and then design fault tolerant supervisory controller for the parallel parking system. In this paper, initially a parallel parking system was modelled using the sensors as places and the high level events as transitions and a supervisory controller was developed to prevent any unnecessary events (transitions). Then, to prevent any failure in the controller itself (such as failure due to the faulty data from the sensors), another redundant controller was developed which allows for the fault-free operation of the initial supervisory controller and also maintain its basic functionality. This paper provided an insight on modeling using Petri nets and an idea to adapt this approach in this thesis.

In [13], technique of automating the parallel parking approach using artificial intelligence approach was analysed. Also, the parking space was divided into several sections and the parking space is modelled by Petri nets with the places denoting the sections of the space. This Petri net flow chart was then used to perfom an optimal path planning using genetic algorithm, for collision avoidance. Based on the results of the algorithm, the fuzzy rules were created and then a fuzzy controller was developed for controlling the steering angle of the vehicle. This paper proved the effectiveness of Petri net in modeling the systems with discrete event formalisms/sequences.

In [11], the authors have modeled a car safety controller along with modeling the vehicle dynamics by using Fluid Stochastic Petri nets (Stochastic Hybrid Petri nets). Fluid Stochastic Petri nets (FSPN) are a class of Hybrid Petri nets to model random behaviour of system dynamics. In this paper, FSPNs having continuous transitions with exponential firing rates and discrete transitions and places were used to model a car safety controller. The continuous quantities modelled by the continuous places and transitions were distance and position. The quantity distance is deterministic at times and stochastic at other times. The model was simulated using initial values for different parameters like speed of the truck, distance when the truck (travelling in front of the car in the tunnel) stops, and the initial speed of the car and its initial distance from the truck in the front. The probability density functions of the distance of the two vehicles at various instants of time were plotted to check at what instant of time the truck stops and when the car stops, indicating an accident has occured. This paper proved that Hybrid (Fluid) Petri net (HPN) is an apt formalism for modeling systems with complex dynamics, a car safety controller in this case. Similar attempts have been made in [21] to model and simulate a transportation system using Hybrid Petri Nets where the continuous dynamics of the passenger flow were modeled by the Continuous Petri net part of the HPN. [22], [23], [24], [25] dealt with modeling and simulation of traffic control systems with HPN where the continuous behavior modeled is the traffic flow. Finally, [27] dealt with developing a simulation model for evaluating the design of railway transit stations based on Hybrid Petri nets.

In [4], [5] and [9] the authors have provided an insight to the modeling, simulation and analysis of Hybrid Dynamic Systems (HDS) using Hybrid Petri nets. In [9], the authors have dealt with modeling a real life control system (Tank Fire Control System) using Hybrid Petri nets. Hence, it is evident that Petri nets, especially Hybrid Petri nets can be used to model real life control systems and is very versatile in modeling and simulation of such systems. So, based on this fact, this thesis is an attempt to model and simulate the automated parallel parking system and to understand the system behavior.

1.3 Thesis organization

The reminder of this thesis is organized as follows: Chapter 2 deals with the introduction to Petri nets, Discrete Petri nets, Continuous Petri nets, Hybrid Petri nets and also their dynamics. Chapter 3 presents the development of the model using high level events. Various versions of the models are discussed. The drawbacks of each version is discussed in detail and the next consecutive version of the model is designed so that it overcomes the drawbacks discussed. The final model is discussed in detail and then used to perform the simulation. Simulations are done using MATLAB, using the state equations of the Petri net firing, and defining the incidence matrices and firing vectors for the model and SIMHPN, a matlab embedded simulation tool. Chapter 4 discusses the results obtained from simulating the model in MATLAB and SIMHPN. Comparisons are made and system behavior is observed and the purpose of this thesis is justified. Chapter 5 offers a summary of the entire thesis, concluding remarks and future work to expand this thesis.

2. INTRODUCTION TO PETRI NETS

2.1 Notations and definitions

Petri net also called as Place/Transition net is a mathematic modeling tool/language used to represent the structural information of real life discrete/hybrid systems. They are a graphical and intuitive way to describe systems and complex control algorithms associated with them. Petri net was discovered by Carl Adam Petri in early 1960's to describe chemical processes. They are composed of two distinct nodes namely,

- Transitions - represented by vertical bar/box. The transitions will depict the events will occur in the system.

- Places - represented by a circle. The places will depict the conditions that have to be met for a particular event to take place. They can have a discrete number of tokens, indicating whether the condition/resource necessary to enable the occurrence of a particular event is present.

The nodes are connected to each other by means of weighted directed arcs. These arcs can connect two different nodes only, that is, only a place and a transition. Arcs cannot connect two places or two transitions. That is why Petri nets are referred to as weighted bipartite graphs [6]. The Fig. 2.1 depicts the pictorial representations of the notations discussed.

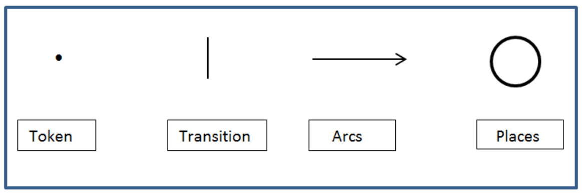


Fig. 2.1. Notations

A Petri net graph could be represented as 4-tuple follows,

$$N = (P, T, A, w) \tag{2.1}$$

where,

- P denotes a finite set of places (represented by circles). It is written as,

$$P = \{p_1, p_2, ..., p_m\} \tag{2.2}$$

- T denotes a finite set of transitions (represented by vertical bars). It is written as,

$$T = \{t_1, t_2, ..., t_n\} \tag{2.3}$$

- A denotes a set of arcs from places to transitions or transitions to place. It is mathematically represented as,

$$A \subseteq (P \times T) \cup (T \times P) \tag{2.4}$$

- w denotes the weight assigned to each arc connecting two different nodes (a place and a transition). The default value of the arc weight is assumed to be 1 if not mentioned explicitly. An arc weight of more than one, for example four denotes that there are four arcs of weight one between the two nodes, and instead of adding four arcs, we can represent them with single arc of weight four. w is represented as,

$$w \colon A \rightarrow \{1, 2, 3, ...\} \tag{2.5}$$

The set of input places to a particular transition t_j is denoted by $I(t_j)$. Mathematically this is represented as,

$$I(t_j) = \{p_i \in P : (p_i, t_j) \in A\} \tag{2.6}$$

The set of output places from transition t_j is denoted by $O(t_j)$. This is represented by,

$$O(t_j) = \{p_i \in P : (t_j, p_i) \in A\} \tag{2.7}$$

Similar notations can be used for input and output transitions as well. They are represented as follows,

$$I(p_i) = \{t_j \in T : (t_j, p_i) \in A\} \tag{2.8}$$

$$O(p_i) = \{t_j \in T : (p_i, t_j) \in A\} \tag{2.9}$$

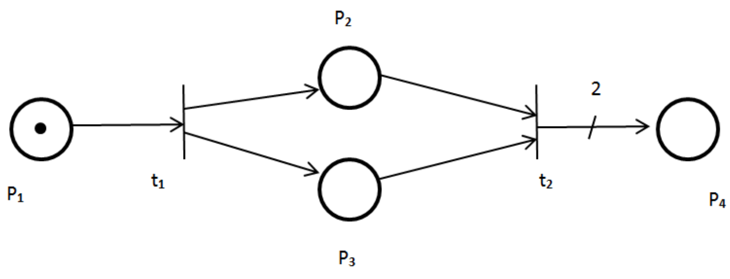


Fig. 2.2. A simple Petri net model

Fig. 2.2 shows a simple Petri net. We will discuss the notations/definitions we have seen so far for this Petri net.

$$P = \{p_1, p_2, p_3, p_4\}$$
$$T = \{t_1, t_2\}$$
$$A = \{(p_1, t_1), (p_2, t_2), (p_3, t_2), (t_1, p_2), (t_1, p_3), (t_2, p_4)\}$$
$$w = \{1, 1, 1, 1, 1, 2\}$$

$$I(t_1) = \{p_1\}, I(t_2) = \{p_2, p_3\}$$
$$I(p_1) = \emptyset, I(p_2) = \{t_1\}, I(p_3) = \{t_1\}, I(p_4) = \{t_2\}$$
$$O(t_1) = \{p_2, p_3\}, O(t_2) = \{p_4\}$$
$$O(p_1) = \{t_1\}, O(p_2) = \{t_2\}, O(p_3) = \{t_2\}, O(p_4) = \emptyset$$

2.2 Marking of a Petri net

A Petri net is usually used to graphically represent the system dynamics. In order to do so, the sequence of events that the system undergoes at various instants of time have to be depicted. In order to do that, we need a pictorial representation of the state in which the system is after a particular event has occured or whether the conditions for such an event to occur has been met. For this purpose, Petri net uses a concept of *tokens*. The Fig. 2.1 shows the graphical representation of a token. It is represented by a black dot. A place can have non negative number of tokens. The unique assignment of tokens in a Petri net is denoted as the *marking*. Mathematically, we represent this as,

$$M : P \rightarrow \{0, 1, 2...\} \tag{2.10}$$

The marking of a Petri net is a column vector called *marking vector M* where each element represents the number of tokens in each place. The number of rows/elements in this vector is equal to the number of places in the Petri net. If the place does not have tokens in it, the value corresponding to the particular place is 0 in the marking vector. $m(p_i)$ denotes marking of a place p_i (i.e, the number of tokens in the place p_i) where, i is the number of the place. For example, for the Petri net mentioned in the Fig. 2.1, $m(p_1)$=1, $m(p_2)$=$m(p_3)$=$m(p_4)$=0. It has to be noted that, places P_2, P_3, P_4 are devoid of tokens and hence have a marking of 0. Hence the marking of a place need not necessarily be an integer value greater than 0. A marking of 0 will indicate the absence of a conditon for a particular event to occur or that, the system is not present in the state indicated by the particular place.

M_0 represents the initial marking of the Petri net before any transition has fired (or any event has occurred). The initial marking of the Petri net in the Fig. 2.1 is,

$$M_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$$

Similarly, M_1 is the next marking after an event has occured (a transition has fired) and so on. We shall cover the dynamics and firing of transitions of the Petri nets in detail in the next section.

2.3 Dynamics of Petri nets

In a Petri net, the transitions depict the events that can occur in a system. To model and simulate a system behavior it is necessary to show the sequence of the events happening in the system and depicting them pictorially at various instants of time, by showing the various states the system transitions through. In Petri net, this can be shown by enabling of transitions and transfering tokens from one place to another based on the event sequence. Specific rules are required to indicate when the particular event can happen, or more specifically, when a particular transition can be enabled/fired [6].

- Transition $t_j \in T$ can be enabled if the

$$M(p_i) \geq w(p_i, t_j), \forall p_i \in I(t_j) \tag{2.11}$$

  where, $I(t_j)$ is denoted as set of the input places to transition t_j. In other words,transition t_j is enabled when the number of tokens in each input place p_i of t_j denoted by $M(p_i)$ is greater than or equal to the arc weight from p_i to t_j, denoted by $w(p_i, t_j)$.

- When a transition is enabled, it can fire at any time. This leads to a change in the marking of the system.

- When a transition t_j fires, from each input place p_i tokens equal to the weight of the arc connecting it to the transition t_j is removed. Then to each output place p_0 the number of tokens equal to the weight of the arc connecting the transition t_j to the place is deposited.

- According to [6], the state transition function $f\colon \mathbb{N}^n \times T \to \mathbb{N}^n$ is defined only for the transitions that are enabled. Mathematically,

$$M'(p_i) = M(p_i) - w(p_i, t_j) + w(t_j, p_i); i = 1, 2, ..., n \tag{2.12}$$

  This equation denotes that when a transition is fired, from each input place p_i tokens equal to the weight of the arc connecting it to the transition t_j is removed $(w(p_i, t_j))$ and to each output place the number of tokens equal to the weight of the arc connecting the transition to the place $(w(t_j, p_i))$ is deposited.

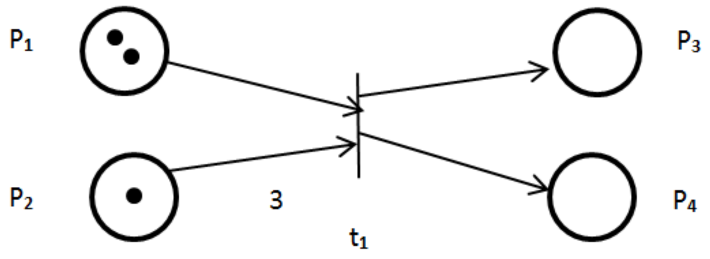


Fig. 2.3. Firing of transitions (a)

Fig. 2.3 demonstrates the firing dynamics of a Petri net. The number of tokens in place p_2 is less than the weight of the arc connecting it to the transition t_1. Hence, even if the place p_1 had tokens greater than the weight of the arc connecting it to the transition t_1, the transition t_1 is not enabled as not all of its places satisfy Equation 2.11.

Now, consider Fig. 2.3. The number of tokens in the places p_1 and p_2 are now greater than the corresponding arc weights connecting those places to the transition

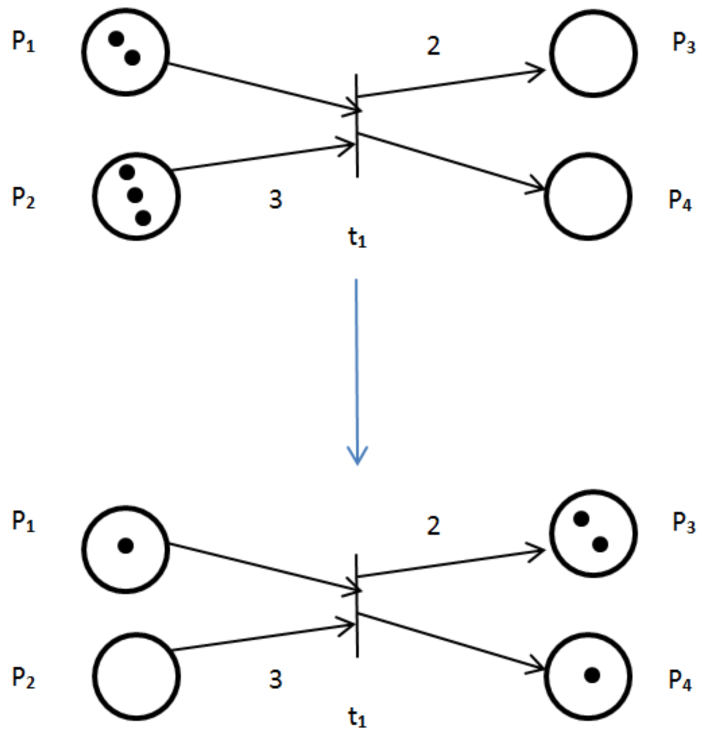


Fig. 2.4. Firing of transitions (b)

t_1. Now, all the places satisfy Equation 2.11 and the transition t_1 will be enabled/fired. After the transitions fire, the number of tokens deposited in each output place will be equal to the weight of the arc connecting the transition to the output place.

For example in Fig. 2.4 even if the place p_1 has two tokens, when the transition t_1 fires, only one token will be removed from it. Three tokens will be removed from place p_2 and two tokens will be deposited in the place p_3 (since the arc weight from t_1 to p_3 is 2) and one token will be deposited in place p_4. It can be observed that the total number of tokens removed from the input places need not be equal to the number of tokens deposited to the output places. Token removal and deposition are functions of the corresponding arc weights alone. So there is no conservation of tokens during the firing of a transition. Note that, when a single place functions as an input place to two different transitions, even if both the transitions are enabled only one can fire at a time. This enables us to observe system behavior through a sequence of events.

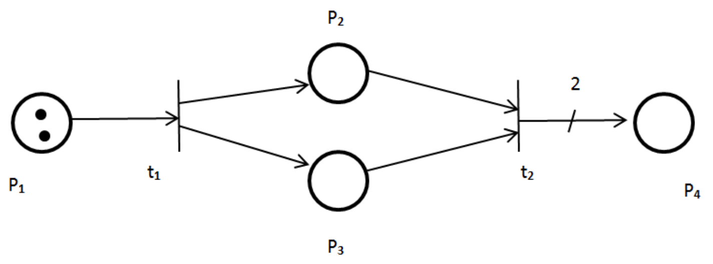


Fig. 2.5. Example of Petri net firing

Refer to Fig. 2.5. The sequence of transitions which will be fired, corresponding markings and then finally the reachability tree/coverability tree for this Petri net will be discussed further.The total possible reachable states of a system has to be pictorially/graphically depicted to find out if the system can be in a particular state after an event has occurred. Reachability tree is the graphical representation of the reachable states of the system. According to [6], it is denoted by,

$$R[(P, T, A, w, m)] := \{y \in \mathbb{N}^n : \exists s \in T^*(f(m, s) = y)\}$$

It starts from the initial marking/state/node and ends in the final state/node also known as the *terminal node* connected by directed arcs which indicate the transitions fired.

1. The initial marking of the system is $m_0 = \begin{bmatrix} 2 & 0 & 0 & 0 \end{bmatrix}^T$.

2. According to Equation 2.11, the transition t_1 will be fired. One token will be removed from the place p_1 and one token will be deposited in each of the places p_2 and p_3. Now, the marking of the Petri net would be $m_1 = \begin{bmatrix} 1 & 1 & 1 & 0 \end{bmatrix}^T$.

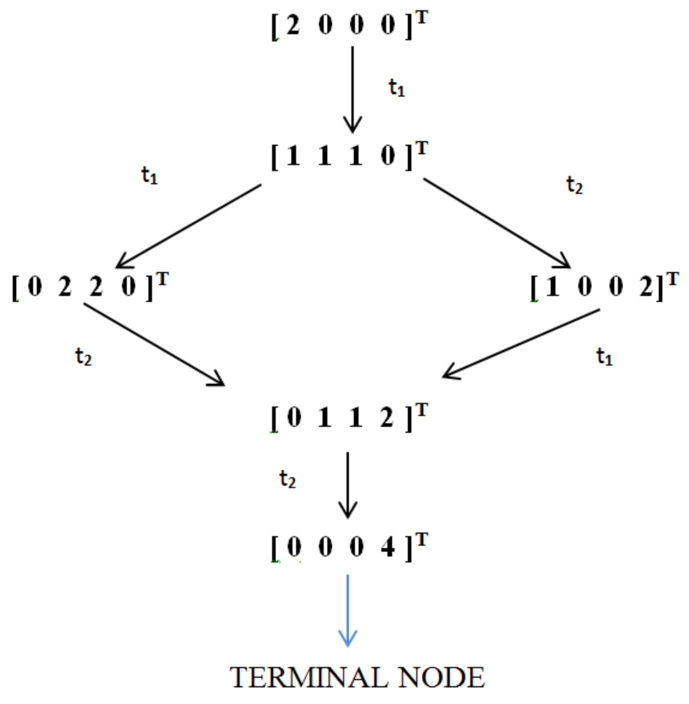


Fig. 2.6. Reachability tree

3. Then, again according to Equation 2.11 both t_1 and t_2 will be enabled. Since only one of them can fire, this will lead to two separate sequences of transitions. $S = \{t_1 t_2 t_2\}$ and $S = \{t_2 t_1 t_2\}$.

4. Considering the first sequence, when t_1 is fired again, one token will be removed from the place p_1 and one token will be deposited again in each of the places p_2 and p_3. Now, the marking of the Petri net would be $m_2 = \begin{bmatrix} 0 & 2 & 2 & 0 \end{bmatrix}^T$. After t_2 is fired, one token will be removed from each of the places p_2 and p_3 and two tokens will be deposited in the place p_4 (since the arc weight is two). Now the marking is $m_3 = \begin{bmatrix} 0 & 1 & 1 & 2 \end{bmatrix}^T$. Finally after t_2 is fired again, once again one token will be removed from each of the places p_2 and p_3 and two tokens will be deposited in the place p_4. The final marking is $m_4 = \begin{bmatrix} 0 & 0 & 0 & 4 \end{bmatrix}^T$. This node is called the *terminal node* and no more transitions can be fired and this is the end state of the system.

5. Considering the second sequence, we arrive at the state $m_4 = \begin{bmatrix} 0 & 0 & 0 & 4 \end{bmatrix}^T$. The reachability tree describing the firing dynamics discussed so far is as shown in Fig. 2.6.

It has to be noted that sometimes the sequence of events may repeate themselves for unlimited number of times forming an infinite loop and a reachability tree of infinite number of markings/nodes. Refer to Fig. 2.7 for an example.

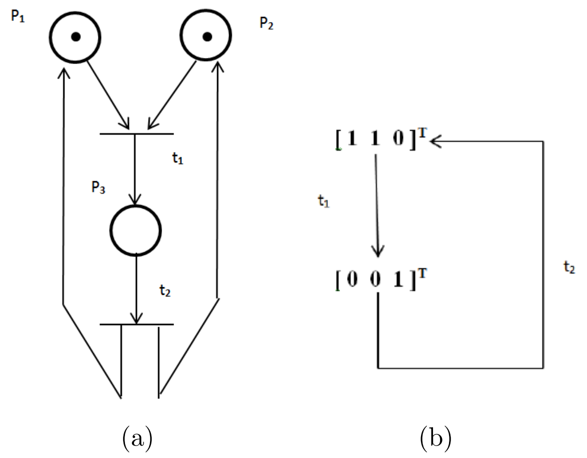


Fig. 2.7. (a) Petri net with a loop. (b) Reachability tree.

2.4 Incidence matrices and state equation of a Petri net

2.4.1 Incident matrices

The structure of a Petri net can be uniquely determined by the incident matrices. For a Petri net with m places and n transitions, we can define three different incident matrices as follows.

- Input Incident Matrix, B^- is an $m \times n$ matrix, which captures the arc weight from the input place p_i to the transition t_j.
- Output Incident Matrix, B^+ is an $m \times n$ matrix, which captures the arc weight from the transition t_j to the output place p_i.

- Incident Matrix, B is also an $m \times n$ matrix given by the mathematical equation

$$B = B^{+} - B^{-} \tag{2.13}$$

It has to be noted that in an input incident matrix, if there is no arc from a place p_i to a transition t_j, the corresponding entry would be denoted by a zero. In a similar sense, in an output incident matrix, when there is no arc from a transition t_j to a place p_i, then the corresponding entry would be zero. These matrices are just an indication of the structural property of the Petri net and are independent of the number of tokens and the states of the system. Let us discuss these concepts for the PN in Fig. 2.2.

$$B^{+} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 2 \end{bmatrix} \qquad B^{-} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$B = B^{+} - B^{-} = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 1 & -1 \\ 0 & 2 \end{bmatrix}$$

2.4.2 State equation

In case if complex systems modelled by complicated Petri nets, analysis of the Petri net/system using reachability tree will become very cumbersome and it will become harder to determine the next state of the Petri net. So, a mathematical approximation has to be used to solve this problem. Hence, we can use an algebraic mathematical representation to determine the state of the system, known as a *state* *equation.* By using this equation, we can find out the reachable states and the state of the system at any particular instant. The state equation is given by,

$$M_{k+1} = M_k + Bv_k \tag{2.14}$$

In the previous equation,

- M_{k+1} is an m×1 vector, which denotes the marking of the Petri net at a an instant of time $k + 1$.
- M_k is an m×1 vector, which denotes the marking of the Petri net at a an instant of time k (previous instant of time).
- B is the incident matrix of the Petri net.
- v_k is called the firing vector. It has a dimension of n×1 and has only one nonzero entry at any instant of time indicating that only one transition (no more than one transition) can fire at any instant of time. The entries can have a value of one or zero indicating which transition will be fired in the next instant of time.

For example, we can take the Petri net in the Fig. 2.5 and verify this state equation. When t_1 fires with an initial state of $M_0 = \begin{bmatrix} 2 & 0 & 0 & 0 \end{bmatrix}^T$, the next state according to the Equation 2.14 will be,

$$M_1 = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 1 & -1 \\ 0 & 2 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 \end{bmatrix}^T$$

Similarly after M_1, the state of the Petri net after the transition t_2 has fired (M_2) can be deduced by making the entry corresponding to the transition t_2 equal to one and the rest of the entries as zero in the firing vector and using the Equation 2.14. Hence as seen from these calculations, the usage of the state equation simplifies the process of deciphering the reachable states of a system. Thus, the concept of a mathematical *state equation* makes it simpler to find out the reachability tree of a highly complex system/Petri net where manual/pictorial calculations using tokens and arc weights becomes cumbersome.

2.5 Continuous Petri nets

So far, we have discussed in detail about Discrete Petri nets, which model the system behavior, by modeling the discrete sequence of events which occur in the system. Hence, Discrete Petri nets are event driven. But most of the systems, in real time, change their state/behavior continuously with respect to time, for example, the traffic/vehicle flow at an intersection, manufacturing of items in a continuous manufacturing of a product in the plant to name a few. These systems cannot be modelled by event driven Discrete Petri nets. A solution to this issue is the more flexible *Continuous Petri nets*(CPN), which are driven by time rather than by events. The definitions/notations discussed for Discrete Petri nets still hold good for the Continuous Petri nets. However there are major differences between a Discrete and a Continuous Petri net. They are as follows:

- The weight of the arcs connecting the places to the transitions and vice versa need not be an integer but can be any real number.
- Structure wise, the places are denoted by double circles and the transitions as represented by a rectangular box.
- The number of tokens in a continuous place need not be an integer and can be any real number. For example, the real number of tokens in a continuous place could denote the amount of materials required to manufacture a particular product in a manufacturing assembly.
- Each transition is associated with a specific firing quantity (a non negative real number).

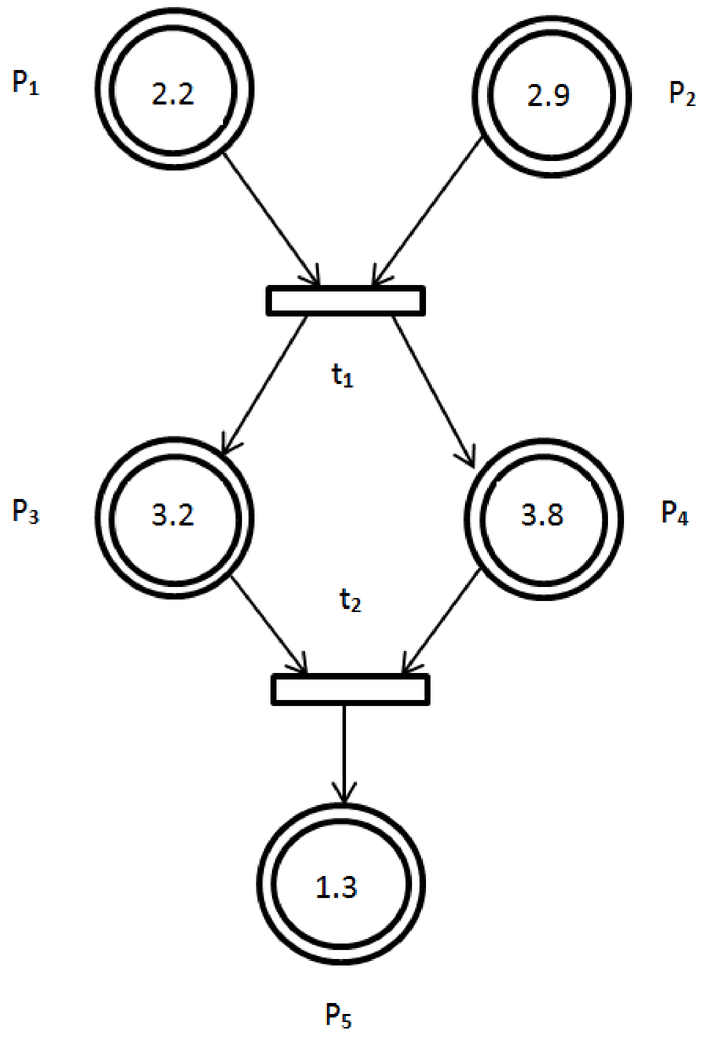


Fig. 2.8. Example of a Continuous Petri net

2.5.1 Notations and firing dynamics

The definitions and rules pertaining to the firing of transitions in a Continuous Petri net are discussed in detail in this subsection.

According to [8] a Continuous Petri net graph is a pair ($\mathbb{N}$, m_0) where,

- $\mathbb{N}$ is the Petri net structure.
- $m_0 : P \rightarrow \Re \geq 0$ is the initial marking of the system.

- A continuous transition $t_j \in T$ will be enabled at any marking m, if and only if,
$$\forall p_i \in \bullet t_j, m_i > 0. \tag{2.15}$$

- The enabling degree of a particular transition t_j is
$$(t_j, m) = \min_{p_i \in \bullet t_j} \left(\frac{m_i}{B^-(p_i, t_j)} \right) \tag{2.16}$$
.

  For example, in the Fig. 2.8, the transition t_1 is 2.2 enabled.

- Any transition t_j which is enabled can fire in any real quantity, $\alpha \in \Re^+$, which means that the transition t_j will fire at α tokens per unit time. Mathematically, $0 \leq \alpha \leq enab(t, m)$ [8]. Refer again to Fig. 2.8 which consists of five continuous places and two continuous transitions. Let the firing rate of t_1 be 0.1, then from an initial marking of $m_0 = \begin{bmatrix} 2.2 & 2.9 & 3.2 & 3.8 & 1.3 \end{bmatrix}^T$ after t_1 is fired, $m_1 = \begin{bmatrix} 2.1 & 2.8 & 3.3 & 3.9 & 1.3 \end{bmatrix}^T$ will be obtained. Thus it means that 0.1 tokens will be removed from each of the places P_1 and P_2 and deposited at each one of the two output places P_3 and P_4. Hence, the state equation for the Continuous Petri nets is given by,
$$M' = M + BV \tag{2.17}$$

  Here,
  M is the initial marking.
  M' is the new marking reached from the initial marking.
  B is the incidence matrix/token flow matrix.
  V is the firing vector. This vector is a $n \times 1$ for a Continuous Petri net with n number of transitions. The entries of this vector are either zero or a non zero firing rate, corresponding to the transition. This firing rate can be any real number as discussed earlier.

Now, further discussing about Continuous Petri nets, the topic to be discussed is the reachability tree. It should be noted that the Continuous Petri net can have infinite number of reachable markings as the markings change constantly, due to constant firing rates. Hence it is difficult to predict the total number of reachable states using a reachability tree. Hence we use a concept known as *macro marking* to depict the possible general reachable states of the CPN. Mathematically, if a CPN has n number of places, then the total number of macro markings would be 2^n. For example, consider the CPN in Fig. 2.9. The number of continuous places is 3.

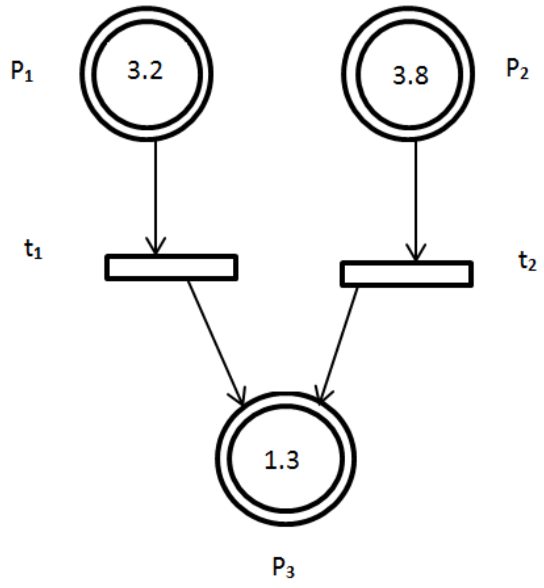


Fig. 2.9. A Continous Petri net

Now according to the previous discussion, the number of marco markings is $2^3 = 8$ where, $n = 3$. These markings would be $\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$, $\begin{bmatrix}0 & 0 & m_3\end{bmatrix}^T$, $\begin{bmatrix}0 & m_2 & 0\end{bmatrix}^T$, $\begin{bmatrix}0 & m_2 & m_3\end{bmatrix}^T$, $\begin{bmatrix}m_1 & 0 & 0\end{bmatrix}^T$, $\begin{bmatrix}m_1 & 0 & m_3\end{bmatrix}^T$, $\begin{bmatrix}m_1 & m_2 & 0\end{bmatrix}^T$, $\begin{bmatrix}m_1 & m_2 & m_3\end{bmatrix}^T$. It has to be noted that the marking of P_3 is initially non zero and there are no transitions to remove tokens from this place. Hence the marking of P_3 can never be zero, and it will always be a non zero, non negative real number. Hence the states/macro markings having m(p_3) as zero are not taken into consideration and can be avoided. These states would be $\begin{bmatrix}m_1 & m_2 & 0\end{bmatrix}^T$, $\begin{bmatrix}0 & m_2 & 0\end{bmatrix}^T$, $\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$ and $\begin{bmatrix}m_1 & 0 & 0\end{bmatrix}^T$. After removing these states, we can get the final generalized macro marking for the given CPN in Fig. 2.9. Refer to the Fig. 2.10 for the corresponding macro marking of the CPN in the Fig. 2.9 [5].

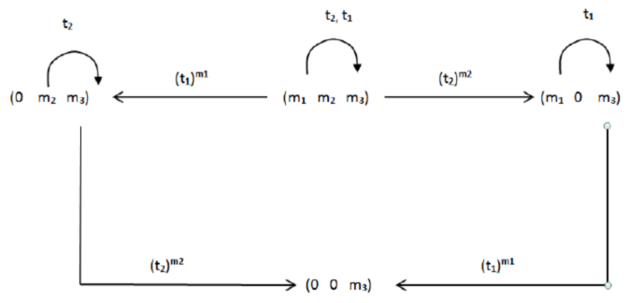


Fig. 2.10. Macro marking

2.6 Hybrid Petri nets

In the previous sections, we have discussed Discrete Petri nets which are used to model the discrete events in a system and Continuous Petri nets which are used to model the continuous dynamics of a system. Most of the systems in the real world are hybrid in the sense, they have a sequence of events which are discrete and also have certain processes which have continuous dynamics and are not event based. For example, if we proceed to model a transportation system or a traffic system, the flow of passengers or vehicles would be the continuous processes happening in the system, while the state of the traffic lights, or the transportation system are the discrete events which govern the system. So, in order to model/simulate the hybrid systems, we need to incorporate both discrete and continuous Petri net models together and proceed. We differentiate the discrete and continuous places and transitions by a superscript "c" or "d". A superscript of "c" will indicate if the particular place/transition is continuous. A superscript of "d" will indicate the particular place/transition is discrete.

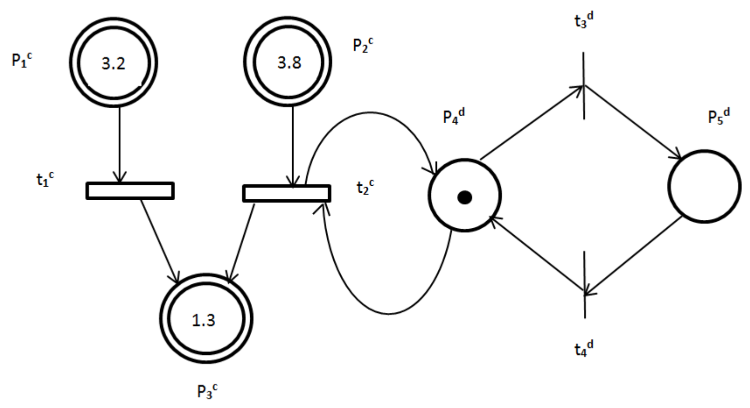


Fig. 2.11. Example of a Hybrid Petri net

2.6.1 Notations

[2] A Hybrid Petri net is a 6-tuple (P,T,Pre,Post,m_0,h), where,

- $P = \{p^d \cup p^c\}$ is a set of places with finite number of places, continuous and discrete. For the Hybrid Petri net in Fig. 2.11, $P = \{p_1^c, p_2^c, p_3^c, p_4^d, p_5^d\}$.
- $T = \{t^d \cup t^c\}$ is a set of places with finite number of transitions, both continuous and discrete. In Hybrid Petri net shown in the Fig. 2.11, $T = \{t_1^c, t_2^c, t_3^d, t_4^d\}$.
- The term *Pre* refers to the input incidence matrix, B^-.
- The term *Post* refers to the input incidence matrix, B^+.
- m_0 is the initial marking of the Petri net. For example, in the Hybrid Petri net shown in the Fig. 2.11, the initial marking is $m_0 = \begin{bmatrix} 3.2 & 3.8 & 1.3 & 1 & 0 \end{bmatrix}^T$.
- h is the *hybrid function* which indicates whether the particular node is a discrete or a continuous node. Mathematically, $h : P \cap T \rightarrow \{D, C\}$ [2]. For example,

p^c, t^c are continuous place and transition respectively.

p^d, t^d are discrete place and transition respectively.

An important rule used in modeling is that, the arc weight connecting an input discrete place p_i to a continuous transition t_j should be the same as the arc weight connecting a continuous transition t_j to an output discrete place p_i. This rule is followed strictly so that the marking of a discrete places is not altered and is strictly an integer. So, when $p_i \in p^d$ and $t_j \in t^c$, mathematically, we can say that,

$$B^-(p_i^d, t_j^c) = B^+(p_i^d, t_j^c)$$

The conditions for firing or enabling of a transition depends on whether the transition is continuous or discrete. If it is discrete, then the condition for firing is the same as Equation 2.11. But if it is continuous, there are certain conditions for enabling it and these conditions depend on the input places attached to the continuous transition. The conditions are,

- For each input place $p_i \in p^d$, $m(p_i) \geq Pre(p_i, t_j^c)$.
- For each input place $p_i \in p^c$, $m(p_i) > 0$.

The state equation of the HPN is given by

$$M_1 = M_0 + Bs \tag{2.18}$$

Here, M_0 is the initial state of the Petri Net, M_1 is the next reachable state and B is the incidence matrix. 's' is the *characteristic vector.* This is practically the same as the firing vector discussed previously, but its elements can be an integer or a real number depending on whether the transition is discrete or continuous. If the given transition is continuous, the corresponding entry would be the firing rate of the said continuous transition. If it is discrete, the entry would be either a '1' or '0' depending on whether it is fired or not. In the Fig. 2.11, if the firing rate of t_1^c is 0.1 and that of t_2^c is 0.2, then from the initial marking is $m_0 = \begin{bmatrix}3.2 & 3.8 & 1.3 & 1 & 0\end{bmatrix}^T$, the next marking would be $\begin{bmatrix}3.3 & 3.6 & 1.6 & 1 & 0\end{bmatrix}^T$.

3. HYBRID PETRI NET MODEL FOR THE AUTOMATED PARALLEL PARKING SYSTEM

The modeling in this thesis was done on the assumption that the vehicle of interest is a car. In the United States and other countries where driving is done in the right lanes, parallel parking is done mostly in the right lane closer to the curb. Hence, if the parking space is between two cars on the right side of the lane, there are two ways to parallel park, as discussed in the introduction in Section 1. The two main methods are,

- *Reverse parallel parking*, where the driver drives slightly forward to the space, parallel to the car in front of the parking space and then backs right into the space and then moves forward right into the parking space, thereby adjusting the position to complete the parallel parking procedure. Refer to Fig. 3.1(a).
- *Forward parallel parking*, where the driver moves forward right into the parking space from behind it and the backs right and finishes parking in the desired position. This is shown in the Fig. 3.1(b).

In real life scenarios, these two methods would require multiple gear shifts to complete parallel parking. But here we are discussing about automated parallel park assist systems where the backing right and moving forward right can happen only once, and the parallel parking is done at the end of one cycle of these events. Only these two events/actions require gear shifts and hence there are only two gear shifts. So, to summarize, the two main events identified in the parallel parking process are moving forward right and backing right. These two are the high level events which we consider while starting the modeling process. The other minor events and the states in which the system resides are also taken into account to closely approximate the modeling to an automated parallel park assist. The safety distance assumed between the car of

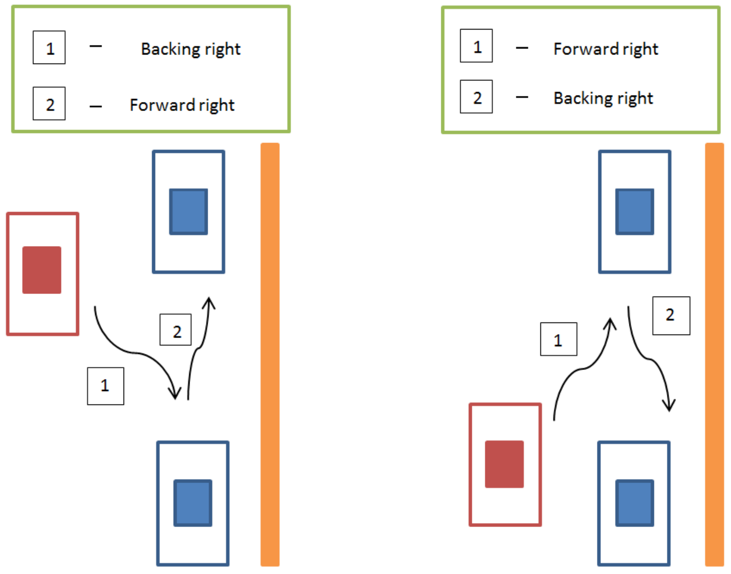


Fig. 3.1. (a) Reverse parallel parking (b) Forward parallel parking

interest and the car at the front of the parking space is two feet. Similarly, the safety distance assumed between the car of interest and the car at the back of the parking space is two feet as shown in the Fig. 3.2. The detailed procedure of modeling the parallel parking system is discussed in the upcoming subsections.

3.1 Development of the Discrete Petri net model

The Discrete Petri net part of the model will be used to depict the high level events that occur in the parallel parking system. The various versions of the model is discussed in this section with the explanation of the drawbacks of each model and how it was overcome in the next version.

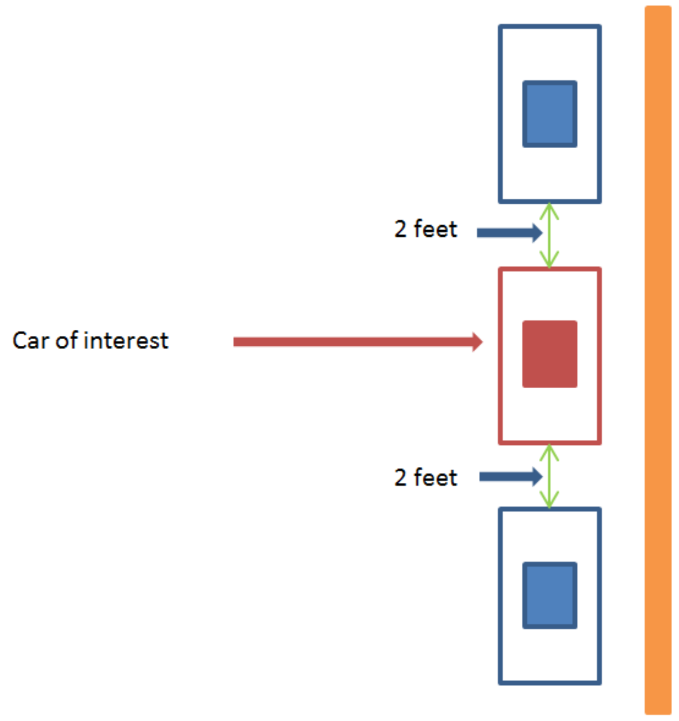


Fig. 3.2. Safety distance between the cars

3.1.1 Version 1

Description of the model

First, the working of the model for a reverse parallel parking procedure is explained in detail. Refer to Fig. 3.3. The car is initially at the place P_{front}, which indicates that the car is in front of the parking space. The transition t_1 indicating the car has is ready to back right fires depositing a token in place P_1. Now the transition t_2 indicating that the car has started backing right fires, depositing one token each in places P_2 and P_5 respectively. P_5 is the check place which denotes that the car has started backing right. P_2 is the place denoting state the car is in before it finishes backing right. Now the transition t_3 indicating the car has finished backing right and is ready to move forward right fires, depositing 1 token each in places P_{inter1} and P_3.

P_{inter1} is also the check place to check if the car has finished backing right and has started initiating forward right event. Now the transition t_5 indicating that the car has started moving forward right is fired and deposits 1 token each in places P_6 and P_4 respectively. P_6 is the check place to check if the car has finished moving forward right. P_4 is the place denoting the state the car is in before it finishes moving forward right. Now the transition t_6 fires, depositing 1 token each in P_1 and P_{inter2}. P_{inter2} is also the check place to check if the car has finished moving forward right and has started initiating backing right event. Now the final transition t_7 is fired and a token is deposited in place P_{final} indicating the car has been parallel parked successfully. The same sequence of events will happen accordingly (order reversed) when forward parallel parking is done and the car is located at the back of the parking space P_{back}.

Drawbacks of the model

The model described above had a looping issue. The two events backing right and forward right will happen continuously, since there is no condition to prevent the firing of t_3 and t_6 for the second time, which eventually deposit tokens on P_3 and P_1, necessary to initiate the firing of t_2 and t_5 respectively, indicating the events backing right and moving forward right. So these events will happen continuously and will form a never ending infinite loop of events, comprising of the events backing right and forward right.

3.1.2 Version 2

Description of the model

Necessary changes were made to fix the problem of the infinite looping. The Fig. 3.4 shows the second version of the discrete part of the model. The model is basically the same except for the addition of two places P_{check1} and P_{check2} and their corresponding arcs and the removal of the two places P_{inter1} and P_{inter2} and their arcs.

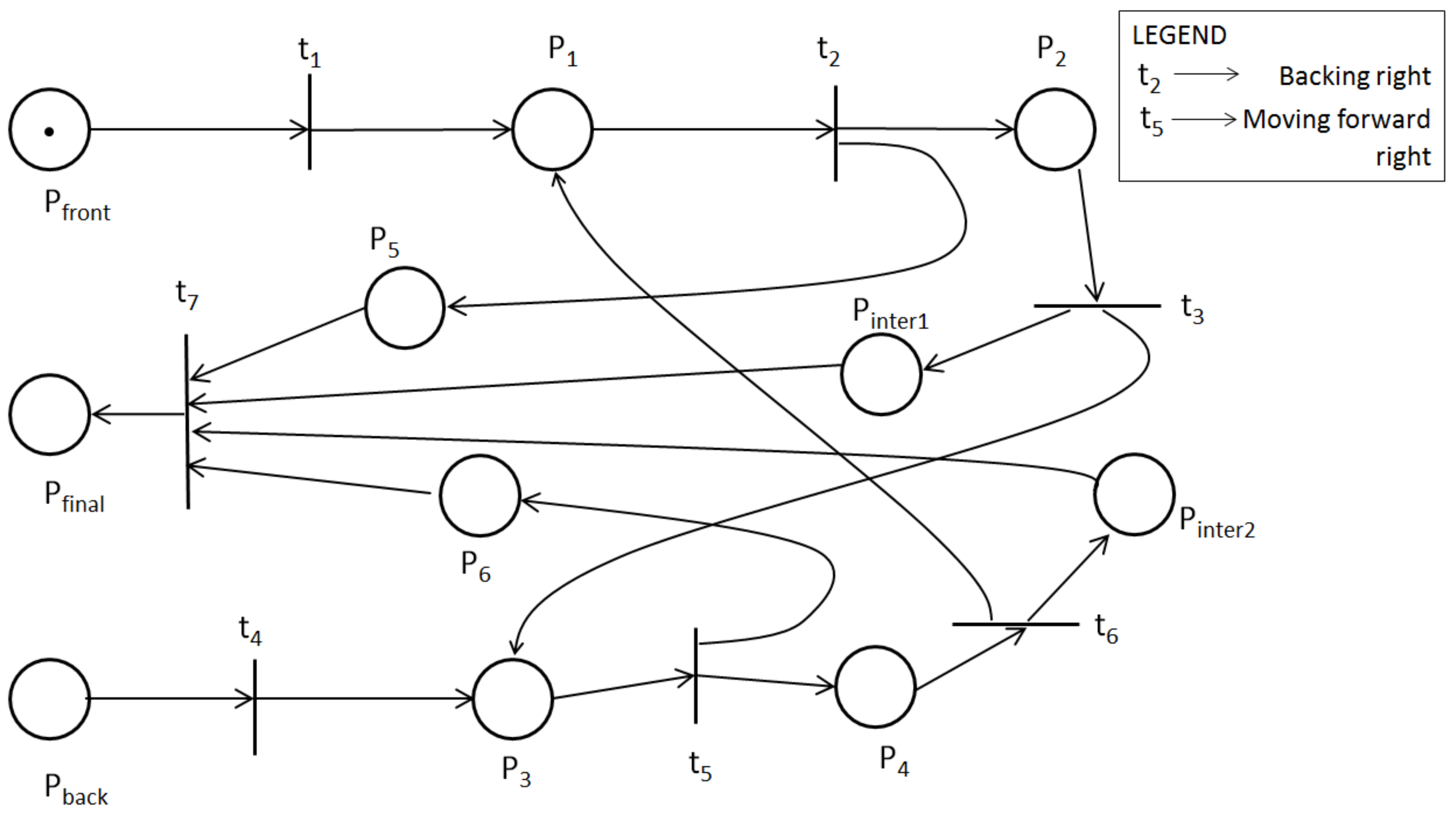


Fig. 3.3. Version 1 of the Discrete Petri net model

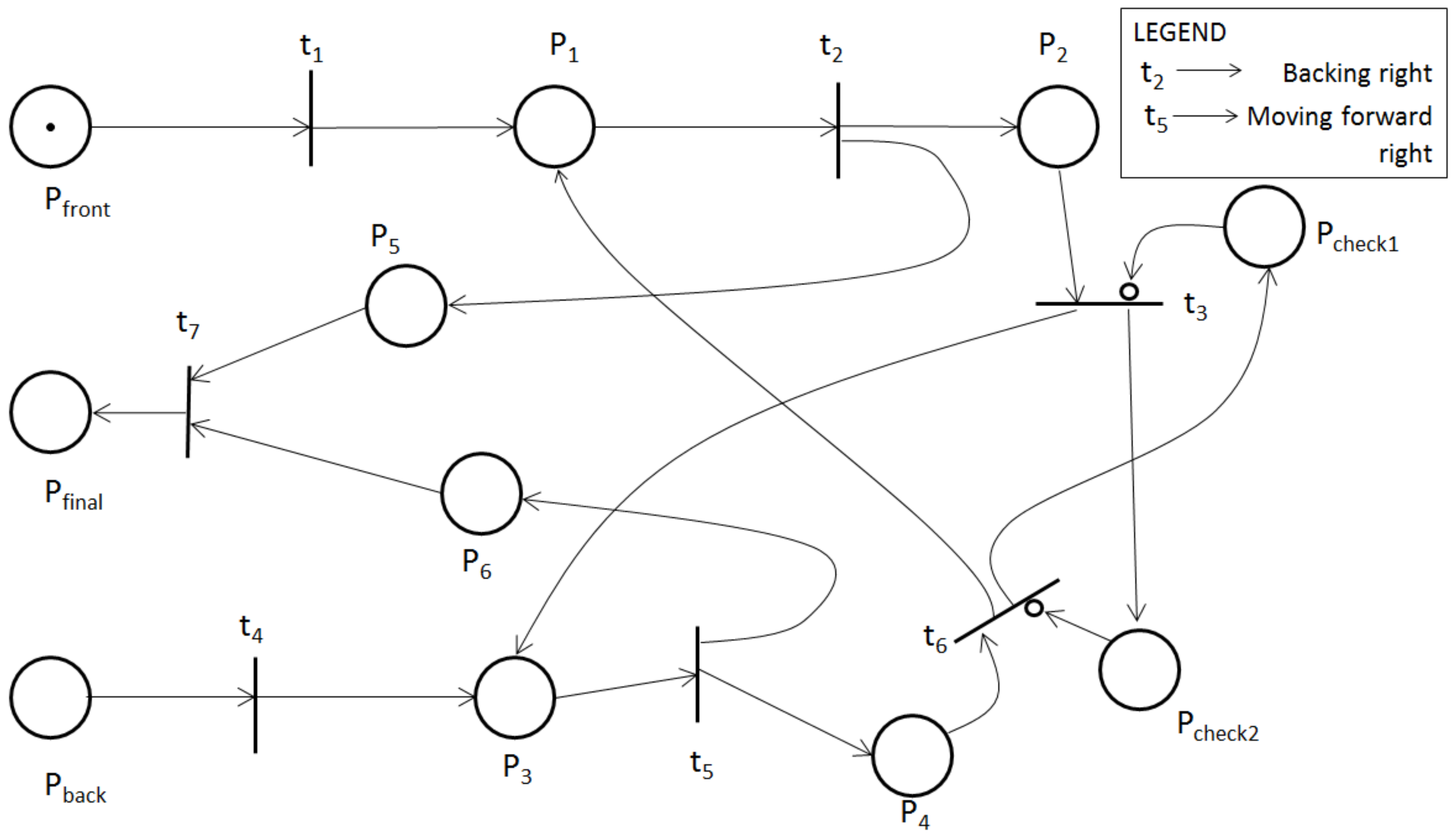


Fig. 3.4. Version 2 of the Discrete Petri net model

- P_{check1} this place is used as a checking place to make sure that the event moving forward right does not happen the second time when it has already been completed.
- P_{check2} this place is used as a checking place to make sure that the event backing right does not happen the second time when it has already been completed.

The working of the model is exactly the same with a few minor changes. So, when the place P_{check2} has a token after the transition t_3 has fired. It has an inhibitor arc connecting it to the transition t_6. Hence, transition t_6 which is essential for depositing tokens to place P_1, initiating the backing right event will not be fired if the place P_{check2} has a token (indicating Backing right event is already complete). Similarly, when P_{check1} has a token, then going forward right will be prevented from happening twice.

Drawbacks of the model

There are two major drawbacks of this model. First, the usage of inhibitor arc will limit the automatic nature of the algorithm and needs hard coding when it is tested in matlab, as additional conditions have to be written for the firing of the transitions associated with it. Also, when this model is tested in SimHPN, there will be problems in automating and simulating this model as SimHPN does not permit the usage of inhibitor arcs or test arcs in the model. So, an alternative solution had to be sought for the looping problem without the usage of inhibitor arcs.

3.1.3 Version 3

Description of the model

The next stage of the development was to solve the looping problem without the usage of inhibitor arcs. The model shown in Fig. 3.5 is a solution. The changes made to the model are the removal of the places P_{check1} and P_{check2} and the inhibitor

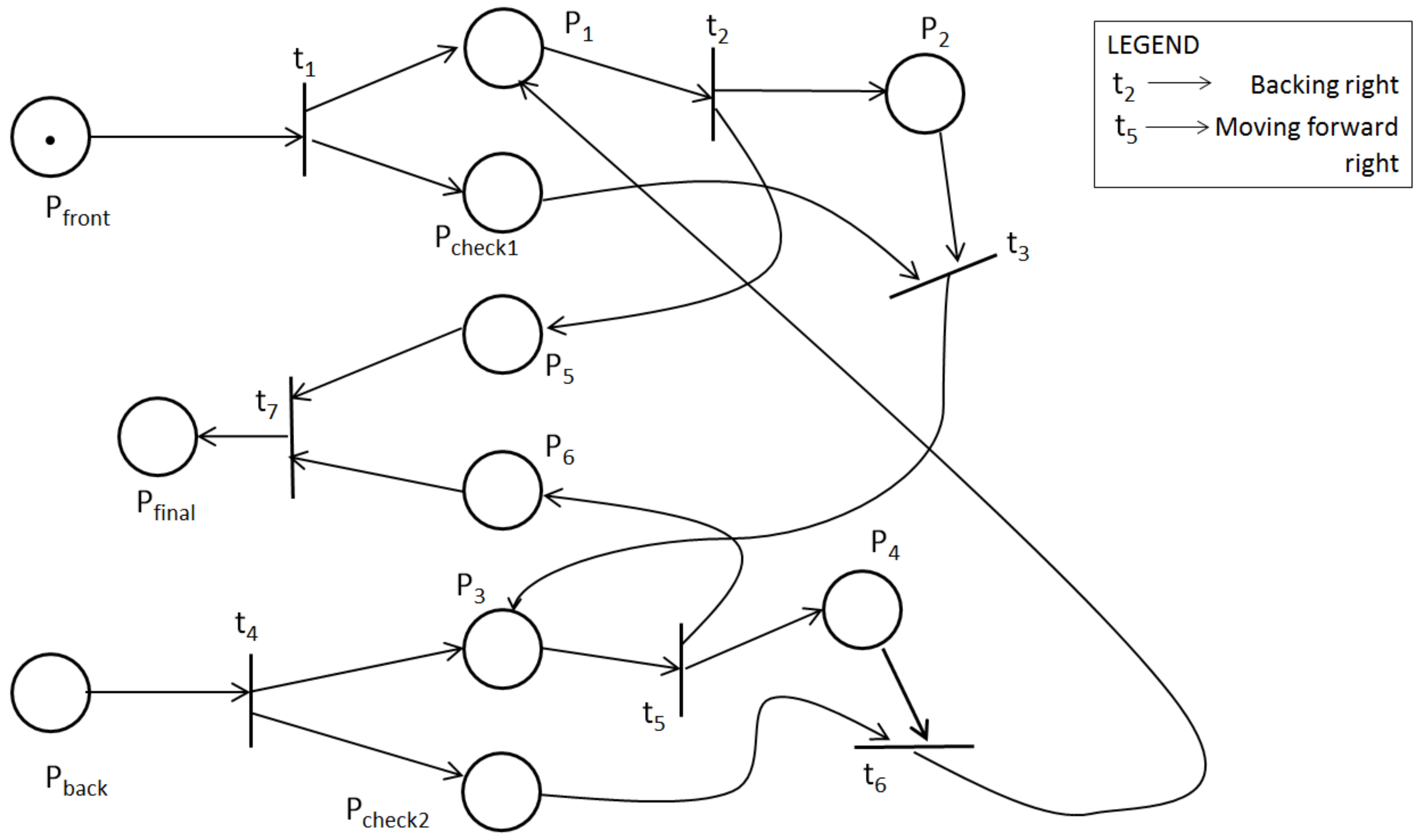


Fig. 3.5. Version 3 of the Discrete Petri net model

arcs associated with them. These places are then assigned as the output places of transitions t_1 and t_4. Place P_{check1} is then the input place of transition t_3 and place P_{check2} is the input place of the transition t_6. Hence, the transition t_3 can fire only for the first time when the place P_{check1} has a token (indicating that the car has started parallel parking process from the initial position at the front of the parking space). Similarly, t_6 can fire only for the first time when the place P_{check2} has a token (indicating that the car has started parallel parking process from the initial position at the back of the parking space). For example, after the events backing right and moving forward right has been completed (indicated by the firing of transitions t_2 and t_5), P_4 will have a token. But P_{check2} will not have a token and so, t_6 will not be enabled, preventing an infinite loop of these two major events, backing right and moving forward right.

3.1.4 Final Discrete Petri net model

Table 3.1.
List of transitions and connections for the final DPN model

Transitions	**Connections**
t_1	$P_{front} \to P_1$
t_2	$P_1 \to P_2\&P_5$
t_3	$P_2 \to P_3\&P_{inter1}$
t_4	$P_{back} \to P_3$
t_5	$P_3 \to P_4\&P_6$
t_6	$P_4 \to P_1\&P_{inter2}$
t_7	$P_5,P_6,P_{inter1}\&P_{inter2} \to P_5$

The final discrete part of the model is as shown in Fig. 3.6. The changes made to the existing model are with reference to removing of the residual tokens. Hence there are additional transitions to remove the residual tokens from the places P_2 and P_4. The transitions t_7 and t_8 fire, and remove the residual token from places P_2 and

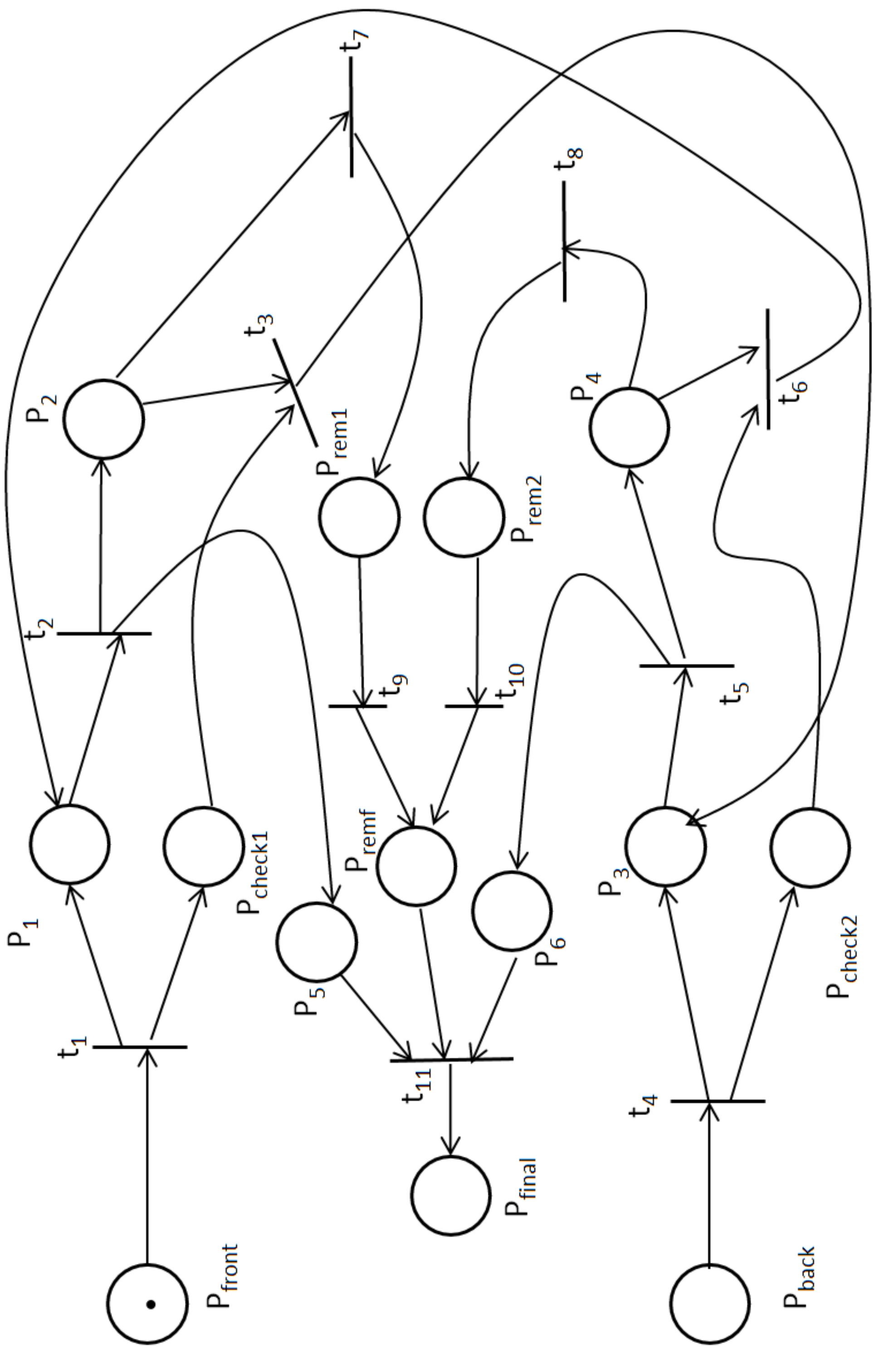


Fig. 3.6. Final Discrete Petri net model.

P_4 respectively, and deposit them in P_{rem1} and P_{rem2} respectively. It has to be noted that only P_2 or P_4 can have a token at a time, so only P_{rem1} or P_{rem2} has a token. The transitions t_9 and t_{10} are in charge of removing the token from either of these places and depositing it in P_{remf}. The final transition t_{11} fires when there are tokens present, one each in the places P_5 (to indicate that the backing right event has been started), P_6 (to indicate that the moving forward right event has been started), and P_{remf} (residual token collecting place). The transition t_{11} fires and placing a token in P_{final}, indicating the parallel parking process is complete.

3.2 Integrated Hybrid Petri net model development

3.2.1 Continuous Petri net part of the model

As described in the previous chapters, the Continuous Petri nets (continuous places) can be used to model the continuous dynamics in any control system/real time system. The continuous dynamics taken into account in this model is the distance. The Continuous Petri net part of the model which is to be integrated with the discrete part is as shown in Fig. 3.7. The continuous place P_{c1} denotes the distance

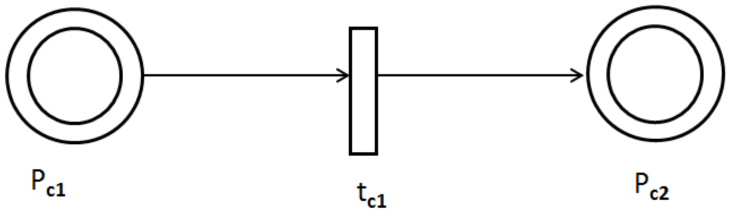


Fig. 3.7. Continuous Petri net modelling of the continuous dynamics

between the car of interest and the car at the front/back of the parking lot. The continuous transition t_{c1} fires with the firing rate of 1 (1 foot). The continuous place P_{c2} keeps on collecting the tokens/distance covered by the car as it moves. When the distance/marking of the place P_{c2} reaches 4, a discrete transition at the output of this place fires and deposits a token at an output place designated, to indicate that the moving forward right or backing right motion has stopped. The continuous transition t_{c1} will keep on firing and depositing tokens in the place P_{c2}, which will then hold the distance remaining at the end of a particular event, say moving forward right or backing right, which is 2 feet in our case. This means that the distance between the car of interest and the back of the car in the front of the parking space, and the car of interest and the front of the car at the back of the parking space is 2 feet each.

3.2.2 Version 1

The first step towards developing an integrated model for simulation is to include the Continuous Petri net dynamics in the Discrete Petri net part of the model. Refer to the Fig. 3.2.2. Here, the continuous dynamics was added after the firing of the transition t_1 (which denotes the car has started backing right) and after transition t_7 (which denotes the car has started moving forward right). The working of the model is basically the same as in the previously discussed models except the inclusion of the continuous dynamics/ Continuous Petri net dynamics. So, after the car has started backing right (firing of the transition t_1), it deposits 6 tokens in the place P_{c1} (indicating the 6 feet distance between the car of interest and the car at the back of the parking space as taken into account in this model) and one token in the check place P_5, which denotes that the backing right motion has begun. As discussed previously, the continuous transition t_{c1}, keeps firing with a firing quantity of 1 and P_{c2} keeps collecting the distance between the car of interest and the car at the back of the parking space. When this distance reaches 4 feet, the transition t_3 fires, indicating that the car has stopped backing right and deposits a token in the discrete place P_2, which serves as a check to see if the car has stopped backing right. The transition t_4, which is the intermediate transition between the two main events backing right and moving forward right, fires and deposits a token in place P_3. Then the transition t_7 fires, and deposits 6 tokens in the pace P_{c3} (indicating the 6 feet distance between the

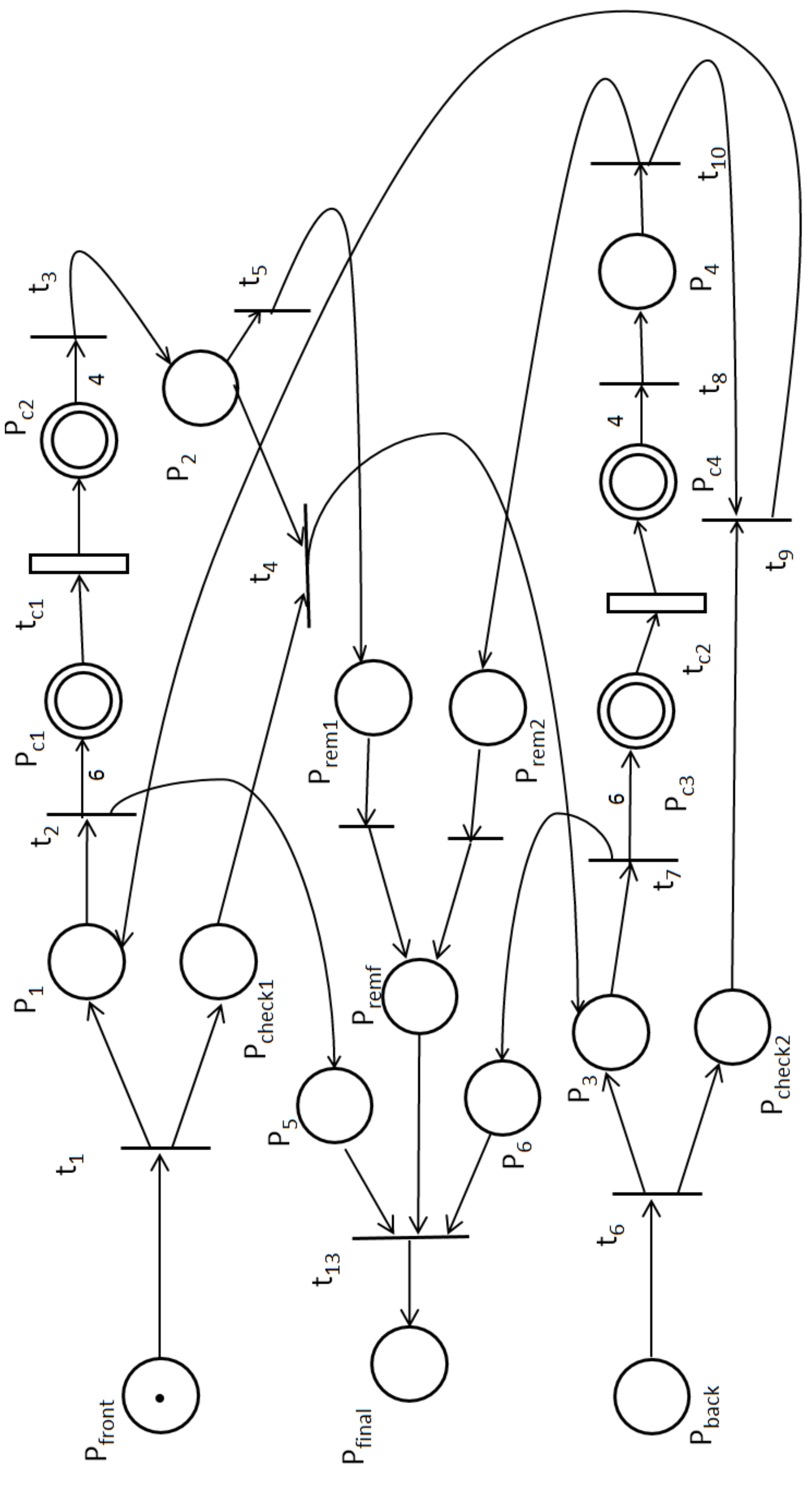


Fig. 3.8. Version 1 of the integrated Hybrid Petri net model

car of interest and the car at the front of the parking space as taken into account in this model) and one token at the check place P_6, which indicates that the moving forward right motion has begun. Then, as discussed previously, the continuous transition t_{c2}, keeps firing with a firing quantity of 1 and P_{c4} keeps collecting the distance between the car of interest and the car at the front of the parking space. When this distance reaches 4 feet, the transition t_8 fires, indicating that the car has stopped moving forward right and deposits a token in the discrete place P_4, which serves as a check to see if the car has stopped backing right. Finally, the continuous place P_{c2} and P_{c4} will hold a distance of 2 feet (since the continuous transition t_{c1} and t_{c2} keeps on firing), which denotes the final distance between the car of interest and the car at the front and back of the parking space respectively. Now, the transition t_{10} fires and deposits the residual token at place P_{rem2} and t_{12} again fires depositing it in P_{remf}. Finally, the transition t_{13} fires, since the events have been completed and tokens are deposited in places indicating so. A token is deposited at P_{final}, indicating the parallel parking process is successfully completed. Similar sequence of events happens when the car starts the parallel parking process when its initial position is at the back of the parking space, namely P_{back}.

3.2.3 Final Hybrid Petri net model of an automated parallel parking system

We can optimize the model by reducing the redundant discrete places and keep the number of discrete and continuous places to a minimum.We start by removing the transitions concerned with the removal of residual tokens in the places P_2 and P_4. These transitions are removed since, the two parallel transitions t_4, t_5 and t_9, t_{10} cannot happen simultaneously and conditions have to be specified to indicate which transition has to fire. This will affect the dynamic nature of the firing of the Hybrid Petri net model and also the sequence in which the events occur in the model. Hence, following the said approach, the number of discrete places was reduced to 13. We

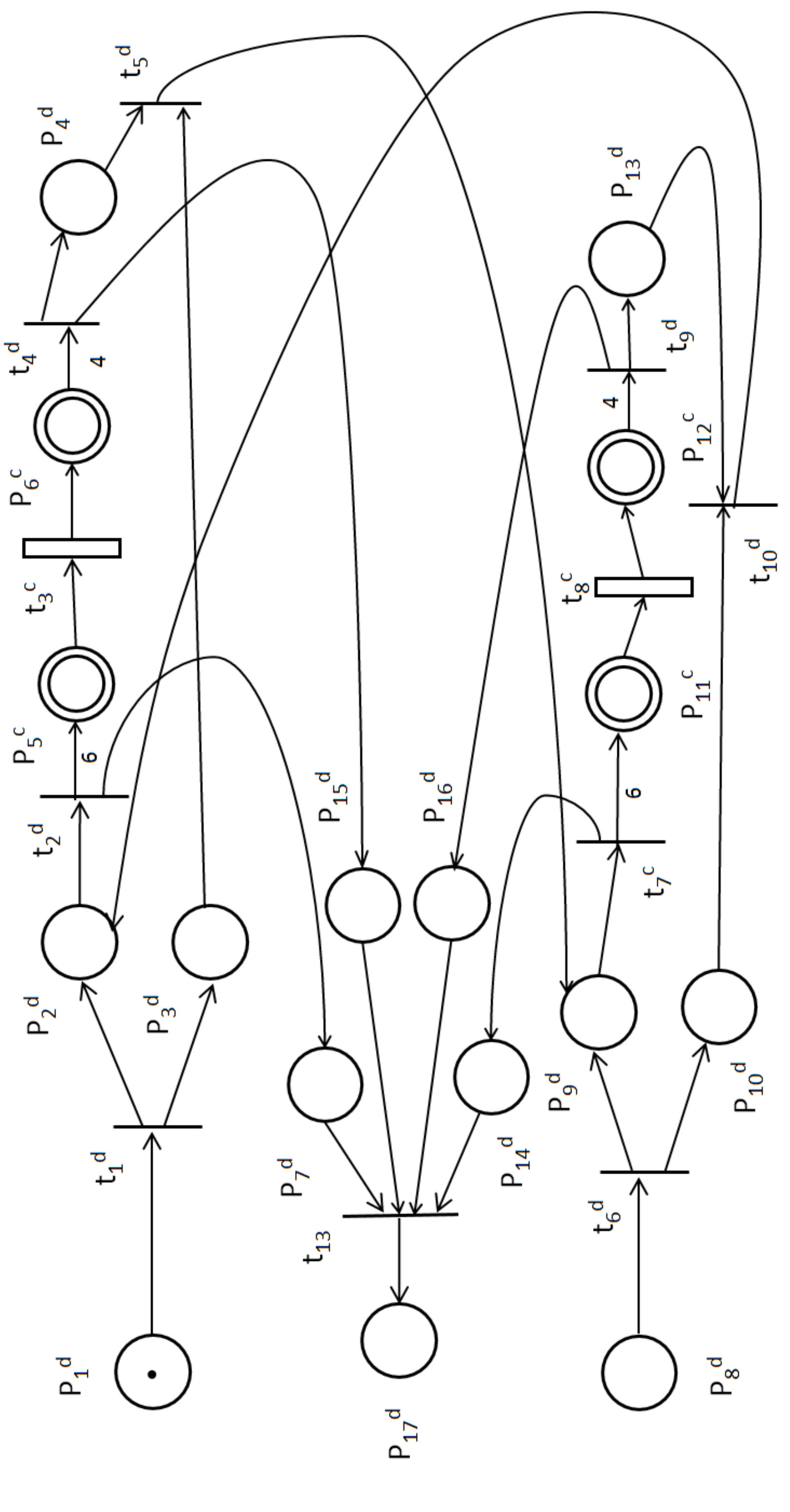


Fig. 3.9. Hybrid Petri net model of the parallel parking system

change the nomenclature of the places to differentiate the continuous and discrete nodes and also to suit the sequence of the events. The descriptions of discrete and continuous places are as follows:

Discrete places:

- P_1^d, P_8^d - Places indicating the car is in the front and back of the parking space respectively.
- P_2^d, P_9^d - Places indicating the car has initiated backing right and going forward right respectively.
- P_3^d, P_{10}^d- Check points for checking if the car has inititated backing right and going forward right respectively.
- P_4^d - Place indicating that the car has finished backing up right and is now ready to go forward right.
- P_{13}^d - Place indicating that the car has finished going forward right and is now ready to back up right.
- P_{15}^d, P_{16}^d - Check points for checking if the backing right and forward right motions have been completed.
- P_7^d, P_{14}^d - Places to indicate that the car has started the process of backing right and going forward right.
- P_{17}^d - Place to indicate that the car has been parked at the final parking spot.

Continuous places: In this modelling we consider distance as a continuous parameter. The continuous places are described as follows:

- P_5^c - Indicates the distance between the car of interest and the car parked at the back of the parking spot.

- P_6^c - Initially indicates the distance covered when the car is backing right. Finally, it will indicate the distance between the car of interest and the back car after the parking has been completed.
- P_{11}^c - Indicates the distance between the car of interest and the car parked in the front of the parking spot.
- P_{12}^c - Initially indicates the distance covered when the car is going forward right. Finally, it will indicate the distance between the car of interest and the front car after the parking has been completed.

Table 3.2.
List of transitions and connections for the integrated HPN model

Transitions	**Connections**
t_1^d	$P_1^d \rightarrow P_2^d \& P_3^d$
t_2^d	$P_2^d \rightarrow P_5^c \& P_7^d$
t_3^c	$P_5^c \rightarrow P_6^c$
t_4^d	$P_6^c \rightarrow P_4^d \& P_{15}^d$
t_5^d	$P_3^d \& P_4^d \rightarrow P_9^d$
t_6^d	$P_8^d \rightarrow P_9^d \& P_{10}^d$
t_7^d	$P_9^d \rightarrow P_{11}^c \& P_{14}^d$
t_8^c	$P_{11}^c \& P_{14}^d$
t_9^d	$P_{12}^c \rightarrow P_{13}^d \& P_{16}^d$
t_{10}^d	$P_{13}^d \& P_{10}^d \rightarrow P_2^d$
t_{11}^d	$P_7^d \& P_{16}^d \& P_{15}^d \& P_{14}^d \rightarrow P_{17}^d$

Working of the model

The working of the integrated model is as follows. As discussed earlier, there are two different methods of parallel parking. The car can be at the front of the parking space or at the back corresponding to these two scenarios. Hence, the car can be at the place P_1^d (reverse parallel parking) or P_8^d (forward parallel parking).

Case 1: Reverse parallel parking

The car is initially at the place P_1^d. The transition t_1^d fires, indicating the car is ready to initiate backing right. Then the place P_2^d and P_3^d have one token each. The place P_2^d has a token, indicating that the car is ready to proceed to the next state of backing right. The place P_3^d has a token which serves a check to indicate that the car has initiated backing right. Now, the transition t_2^d fires and six tokens are deposited at place P_5^c and one token is deposited at place P_7^d. Place P_7^d has a token indicating that the car has started backing right. The continuous place P_5^c (denotes distance) has six tokens indicating that the distance between the car of interest and the front of the back car is 6 feet. The continuous transition t_3^c keeps on firing with a firing rate of 1 foot. The continuous place P_6^c keeps collecting the tokens, and also indicates the distance travelled by the car of interest. When the distance travelled reached 4 feet, the distance between the car and the front of the car at the back is 2 feet. This is the safety limit considered in the model to prevent the car from colliding with the car at the back. So, now the transition t_4^d indicating the car has stopped backing right fires, depositing one token each in places P_{15}^d and P_4^d. The place P_{15}^d has a token indicating the car has stopped backing right. The place P_4^d has a token which serves as a check to see if the car has stopped backing right and is ready to proceed to the next state. Now, the transition t_5^d which indicates that the car has finished backing right and is ready to start moving forward right is fired. This event deposits a token to the place P_9^d. A token in P_9^d would indicate that the car is ready to move forward right. Now the transition t_7^d indicating that the car has started moving forward right will be fired. Six tokens will be deposited in the continuous place P_{11}^c. The continuous transition t_8^c will now start to fire. The continuous place P_{12}^c keeps collecting the tokens in a similar way to place P_6^c. When the distance travelled by the car of interest reaches 4 feet, the safety limit of 2 feet between the car and the car in the front of the parking space is reached. So, the transition t_9^d indicating that the car has stopped moving forward right fires and deposits a token each at places P_{13}^d and P_{16}^d. Finally, the continuous place P_6^c and P_{12}^c will hold a distance of 2 feet (since the continuous transitions t_3^c and t_8^c keep on firing), which denotes the final distance between the car of interest and the car at the back and front of the parking space respectively. Now all the check places, P_7^d, P_9^d, P_{15}^d and P_{16}^d have tokens in them indicating the car has finally finished the sequence of events to complete the parallel parking process in the spot. The final transition t_{11}^d is then fired. The place P_{17}^d will now have a token indicating that the car has been parked in the final desired spot.

Case 2: Forward parallel parking

The car is initially at the place P_8^d. The transition t_6^d fires, indicating the car is ready to initiate backing right. Then the place P_9^d and P_{10}^d have one token each. The place P_9^d has a token, indicating that the car is ready to proceed to the next state of going forward right. The place P_{10}^d has a token which serves a check to indicate that the car has initiated going forward right. Now, the transition t_7^d fires and six tokens are deposited at place P_{11}^c and one token is deposited at place P_{14}^d. Place P_{14}^d has a token indicating that the car has started going forward right. The continuous place P_{10}^d (denotes distance) has six tokens indicating that the distance between the car of interest and the back of the car in front of the parking space is 6 feet. The continuous transition t_8^c keeps on firing with a firing rate of 1 foot. The continuous place P_{12}^c keeps collecting the tokens, and also indicates the distance travelled by the car of interest. When the distance travelled reached 4 feet, the distance between the car and the back of the car in front of the parking space is 2 feet. This is the safety limit considered in the model to prevent the car from colliding with the car at the front. So, now the transition T9 indicating the car has stopped going forward right fires, depositing one token each in places P_{13}^d and P_{16}^d. The place P_{16}^d has a token indicating the car has stopped going forward right. The place P_{13}^d has one token which serves as a check to see if the car has stopped going forward right and is ready to proceed to the next state. Now, the transition t_{10}^d which indicates that the car has finished going forward right and is ready to start backing right is fired. This event deposits a token to the place P_2^d. A token in P_2^d would indicate that the car is ready to back right. Now the transition t_2^d indicating that the car has started backing right will be fired. Six tokens will be deposited in the continuous place P_5^c. The continuous transition t_3^c will now start to fire. The continuous place P_6^c keeps collecting the tokens in a similar way to place P_{12}^c. When the distance travelled by the car of interest reaches 4 feet, the safety limit of 2 feet between the car and the car in the back of the parking space is reached. So, the transition t_4^d indicating that the car has stopped moving forward right fires and deposits a token each at places P_4^d and P_{15}^d. Finally, the continuous place P_6^c and P_{12}^c will hold a distance of 2 feet (since the continuous transitions t_3^c and t_8^c keep on firing), which denotes the final distance between the car of interest and the car at the back and front of the parking space respectively. Now, all the check places, P_7^d, P_9^d, P_{15}^d and P_{16}^d have tokens in them indicating the car has finally finished the sequence of events to complete the parallel parking process in the spot. The final transition t_{11}^d is then fired. The place P_{17}^d will now have a token indicating that the car has been parked in the final desired spot.

4. RESULTS

In order to observe the behavior of the Hybrid Petri net model of an automated parallel parking system, simulations should be carried out. This section talks about the simulaton tool SimHPN, simulations that were carried out both in MATLAB and SimHPN , the results obtained and the justification of the results.

4.1 Testing using MATLAB

The final discrete and the Hybrid Petri net model is initially tested using the MATLAB software. The state equations and the rules for firing are incorporated while coding the algorithm in MATLAB. The incident matrices, the firing vectors, and the initial state of the system is defined and the state equation is used to find out the reachable states of the system and to check if the model portrays the sequence of events correctly. Using Equation 2.13 we can find the incident matrix B from the input incident matrix and output incident matrix. The transitions t_3^c and t_8^c fire with a firing rate of 1 foot. So the corresponding entries are equal to 1. The rest of the entries are 1 to indicate the corresponding discrete transition will be fired. The firing vector V is given by:

$$V = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T \tag{4.1}$$

As discussed, the incidence matrix B for the HPN model will be calculated from the input incidence and output incidence matrices B^- and B^+ respectively. These will be incorporated in the MATLAB code for simulating the model.

$$B^+ = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&1&0\\ 1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0\\ 0&6&0&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&6&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&1 \end{bmatrix} \quad B^- = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&4&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&4&0&0\\ 0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix} \quad B = \begin{bmatrix} -1&0&0&0&0&0&0&0&0&0&0\\ 1&-1&0&0&0&0&0&0&0&1&0\\ 1&0&0&0&-1&0&0&0&0&0&0\\ 0&0&0&1&-1&0&0&0&0&0&0\\ 0&6&-1&0&0&0&0&0&0&0&0\\ 0&0&1&-4&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&-1\\ 0&0&0&0&0&-1&0&0&0&0&0\\ 0&0&0&0&1&1&-1&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&-1&0\\ 0&0&0&0&0&0&6&-1&0&0&0\\ 0&0&0&0&0&0&0&1&-4&0&0\\ 0&0&0&0&0&0&0&0&1&-1&0\\ 0&0&0&0&0&0&1&0&0&0&-1\\ 0&0&0&1&0&0&0&0&0&0&-1\\ 0&0&0&0&0&0&0&0&1&0&-1\\ 0&0&0&0&0&0&0&0&0&0&1 \end{bmatrix}$$

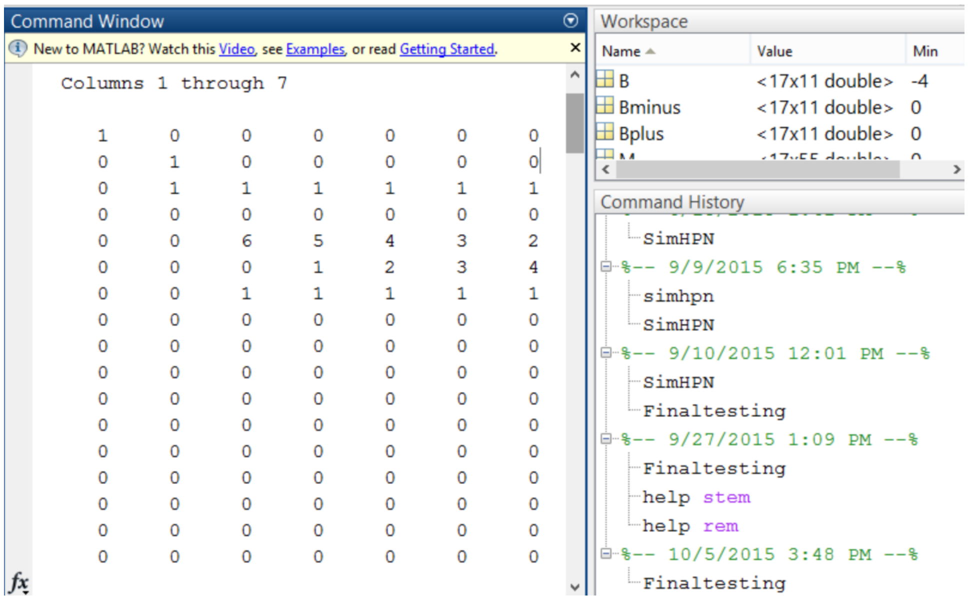
Fig. 4.1. Incidence matrices

4.1.1 Case 1 : Reverse parallel parking

In this case the car is present initially at the front of the parking space. So, one token is present at the place P_1^d. Hence the initial marking of the system, M_0 is given by,

$$M_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.2}$$

The MATLAB results showing the reachability tree is shown as follows. The reachable states are stored in a matrix to make the checking of the reachability tree simpler. The first column is the first state, the second column is the next state and so on. The reachable states are calculated by using the state equation given by Equation 2.14. Refer to Fig. 4.2. The transition t_1^d fires, indicating the car is ready to initiate backing right. Then the place P_2^d and P_3^d have one token each. Then, the transition t_2^d fires and six tokens are deposited at place P_5^c and one token is deposited at place P_7^d. This process continues and the rest of the reachable states can be checked from the columns of this matrix.

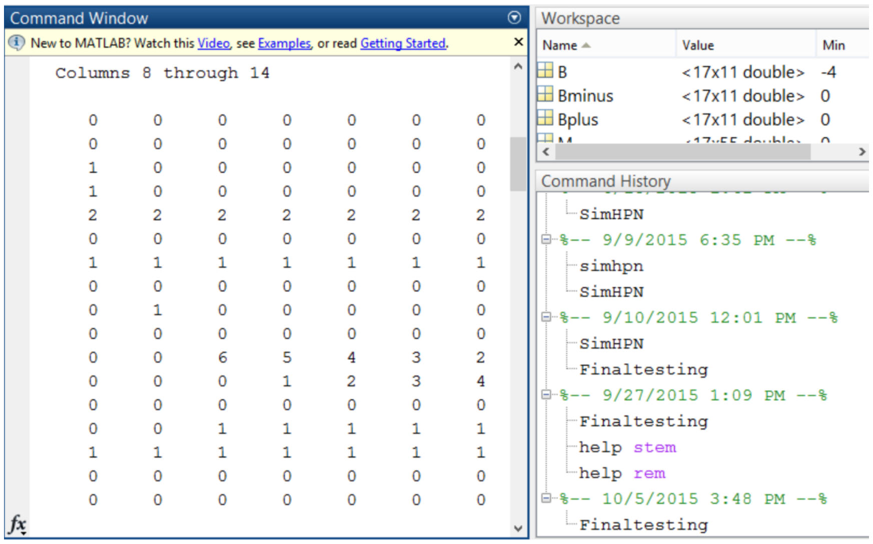

Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.
Workspace
Command History

```
  Columns 1 through 7

     1     0     0     0     0     0     0
     0     1     0     0     0     0     0
     0     1     1     1     1     1     1
     0     0     0     0     0     0     0
     0     0     6     5     4     3     2
     0     0     0     1     2     3     4
     0     0     1     1     1     1     1
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

Name	Value	Min
B	<17x11 double>	-4
Bminus	<17x11 double>	0
Bplus	<17x11 double>	0

```
  SimHPN
%-- 9/9/2015 6:35 PM --%
  simhpn
  SimHPN
%-- 9/10/2015 12:01 PM --%
  SimHPN
  Finaltesting
%-- 9/27/2015 1:09 PM --%
  Finaltesting
  help stem
  help rem
%-- 10/5/2015 3:48 PM --%
  Finaltesting
```

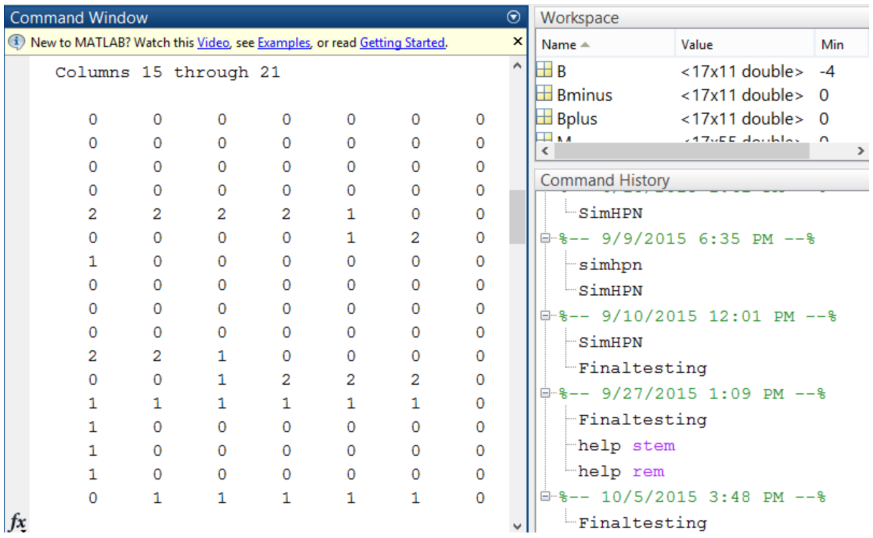
Fig. 4.2. First seven reachable states of the system.



```
Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

  Columns 8 through 14

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     1     0     0     0     0     0     0
     1     0     0     0     0     0     0
     2     2     2     2     2     2     2
     0     0     0     0     0     0     0
     1     1     1     1     1     1     1
     0     0     0     0     0     0     0
     0     1     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     6     5     4     3     2
     0     0     0     1     2     3     4
     0     0     0     0     0     0     0
     0     0     1     1     1     1     1
     1     1     1     1     1     1     1
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

```
Workspace
Name        Value            Min
B           <17x11 double>   -4
Bminus      <17x11 double>   0
Bplus       <17x11 double>   0

Command History
  SimHPN
%-- 9/9/2015 6:35 PM --%
  simhpn
  SimHPN
%-- 9/10/2015 12:01 PM --%
  SimHPN
  Finaltesting
%-- 9/27/2015 1:09 PM --%
  Finaltesting
  help stem
  help rem
%-- 10/5/2015 3:48 PM --%
  Finaltesting
```

Fig. 4.3. The next seven reachable states of the system.



```
Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

  Columns 15 through 21

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     2     2     2     2     1     0     0
     0     0     0     0     1     2     0
     1     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     2     2     1     0     0     0     0
     0     0     1     2     2     2     0
     1     1     1     1     1     1     0
     1     0     0     0     0     0     0
     1     0     0     0     0     0     0
     1     0     0     0     0     0     0
     0     1     1     1     1     1     0
```

```
Workspace
Name        Value            Min
B           <17x11 double>   -4
Bminus      <17x11 double>   0
Bplus       <17x11 double>   0

Command History
  SimHPN
%-- 9/9/2015 6:35 PM --%
  simhpn
  SimHPN
%-- 9/10/2015 12:01 PM --%
  SimHPN
  Finaltesting
%-- 9/27/2015 1:09 PM --%
  Finaltesting
  help stem
  help rem
%-- 10/5/2015 3:48 PM --%
  Finaltesting
```

Fig. 4.4. The last few reachable states of the system.

Finally, to confirm the results, we can check the states of the system towards the end of the parallel parking process. After the car has stopped backing right and going forward right, the continuous places P_6^c and P_{12}^c will collect the distance remaining between the car of interest and the cars at the front and back. The safety distance considered in this thesis is 2 feet. So, if we notice in the Fig. 4.4, the places P_6^c and

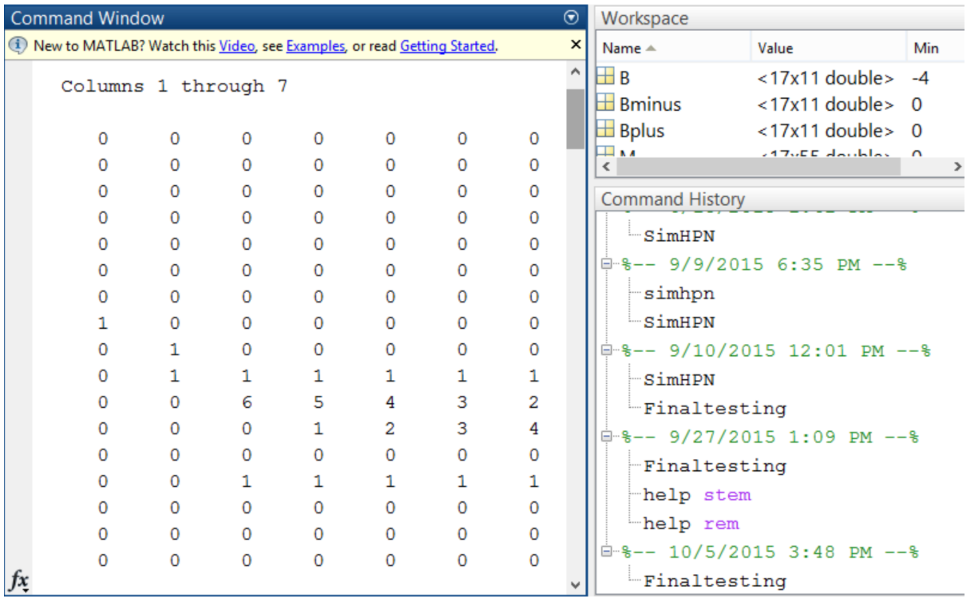
P^c_{12} will have 2 tokens each, the place P^d_{13} will have a residual token 1 (since we have removed the transitions associated with removal of residual tokens to enable testing in MATLAB and simulation in SimHPN) and finally, a token in the final place P^d_{17} to indicate the parking has been completed.

4.1.2 Case 2 : Forward parallel parking

In this case the car is present initially at the back of the parking space. So, one token is present at the place P^d_8. Hence the initial marking of the system, M_0 is given by,

$$M_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.3}$$

Finally, to confirm the results, we can check the states of the system towards the end of the parallel parking process. Similar to the reverse parallel parking scenario, in case of forward parallel parking, as seen in the last state in Fig. 4.7, the places P^c_6 and P^c_{12} will have 2 tokens each, the place P^d_{13} will have a residual token 1. Finally, a token in the final place P^d_{17} indicates that the parking has been completed.

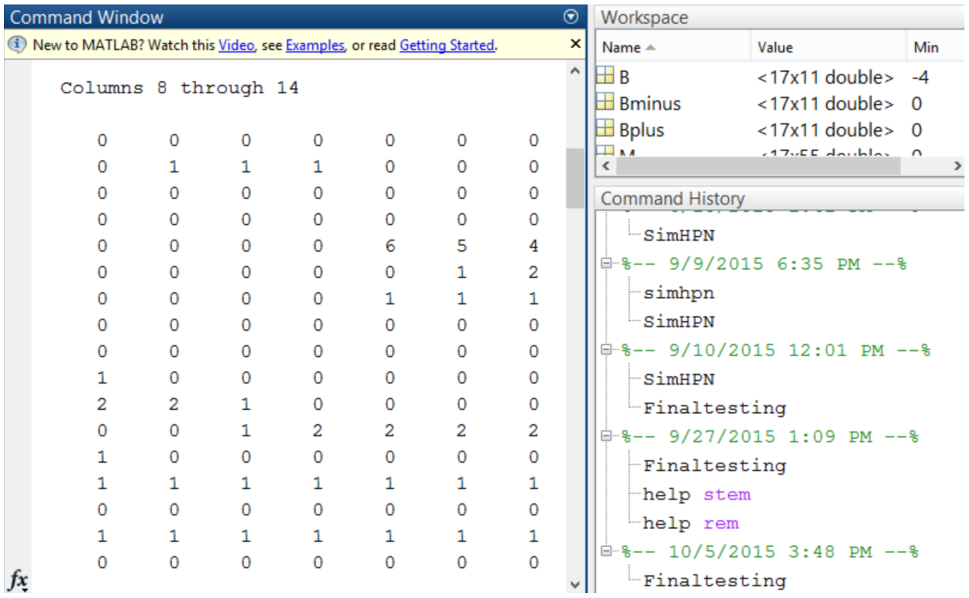

Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.
Workspace
Name
Value
Min
Command History

```
  Columns 1 through 7

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     1     0     0     0     0     0     0
     0     1     0     0     0     0     0
     0     1     1     1     1     1     1
     0     0     6     5     4     3     2
     0     0     0     1     2     3     4
     0     0     0     0     0     0     0
     0     0     1     1     1     1     1
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

```
B        <17x11 double>  -4
Bminus   <17x11 double>  0
Bplus    <17x11 double>  0
```

```
  SimHPN
%-- 9/9/2015 6:35 PM --%
  simhpn
  SimHPN
%-- 9/10/2015 12:01 PM --%
  SimHPN
  Finaltesting
%-- 9/27/2015 1:09 PM --%
  Finaltesting
  help stem
  help rem
%-- 10/5/2015 3:48 PM --%
  Finaltesting
```

Fig. 4.5. The first 7 reachable states of the system.



Command Window

New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
  Columns 8 through 14

     0     0     0     0     0     0     0
     0     1     1     1     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     6     5     4
     0     0     0     0     0     1     2
     0     0     0     0     1     1     1
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     1     0     0     0     0     0     0
     2     2     1     0     0     0     0
     0     0     1     2     2     2     2
     1     0     0     0     0     0     0
     1     1     1     1     1     1     1
     0     0     0     0     0     0     0
     1     1     1     1     1     1     1
     0     0     0     0     0     0     0
```

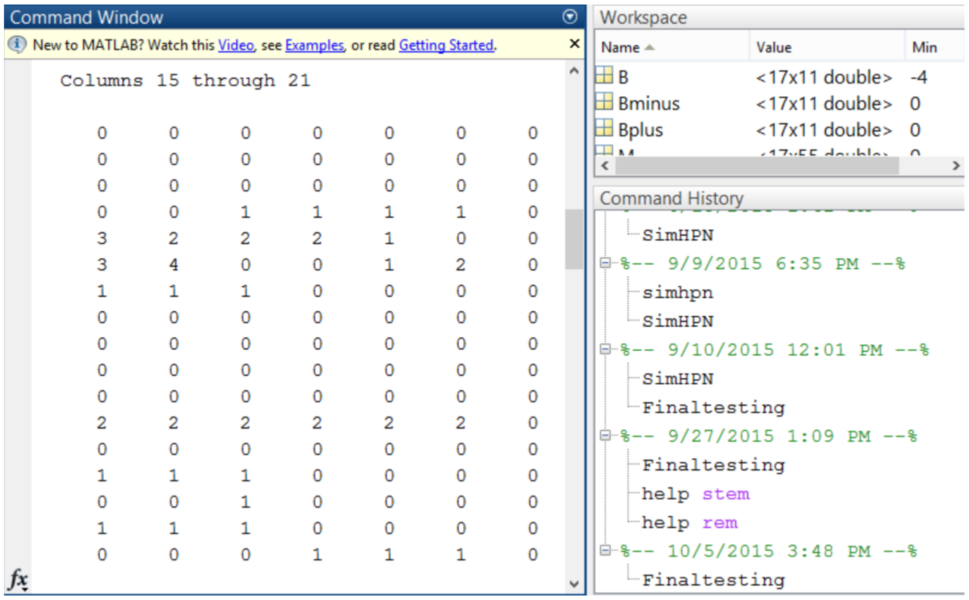
Workspace

Name	Value	Min
B	<17x11 double>	-4
Bminus	<17x11 double>	0
Bplus	<17x11 double>	0

Command History

```
  SimHPN
%-- 9/9/2015 6:35 PM --%
  simhpn
  SimHPN
%-- 9/10/2015 12:01 PM --%
  SimHPN
  Finaltesting
%-- 9/27/2015 1:09 PM --%
  Finaltesting
  help stem
  help rem
%-- 10/5/2015 3:48 PM --%
  Finaltesting
```

Fig. 4.6. The next 7 reachable states of the system.



Command Window

New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
  Columns 15 through 21

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     1     1     1     1     0
     3     2     2     2     1     0     0
     3     4     0     0     1     2     0
     1     1     1     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     2     2     2     2     2     2     0
     0     0     0     0     0     0     0
     1     1     1     0     0     0     0
     0     0     1     0     0     0     0
     1     1     1     0     0     0     0
     0     0     0     1     1     1     0
```

Workspace

Name	Value	Min
B	<17x11 double>	-4
Bminus	<17x11 double>	0
Bplus	<17x11 double>	0

Command History

```
  SimHPN
%-- 9/9/2015 6:35 PM --%
  simhpn
  SimHPN
%-- 9/10/2015 12:01 PM --%
  SimHPN
  Finaltesting
%-- 9/27/2015 1:09 PM --%
  Finaltesting
  help stem
  help rem
%-- 10/5/2015 3:48 PM --%
  Finaltesting
```

Fig. 4.7. The last few reachable states of the system.

4.2 Simulations using SimHPN

4.2.1 Introduction to SimHPN

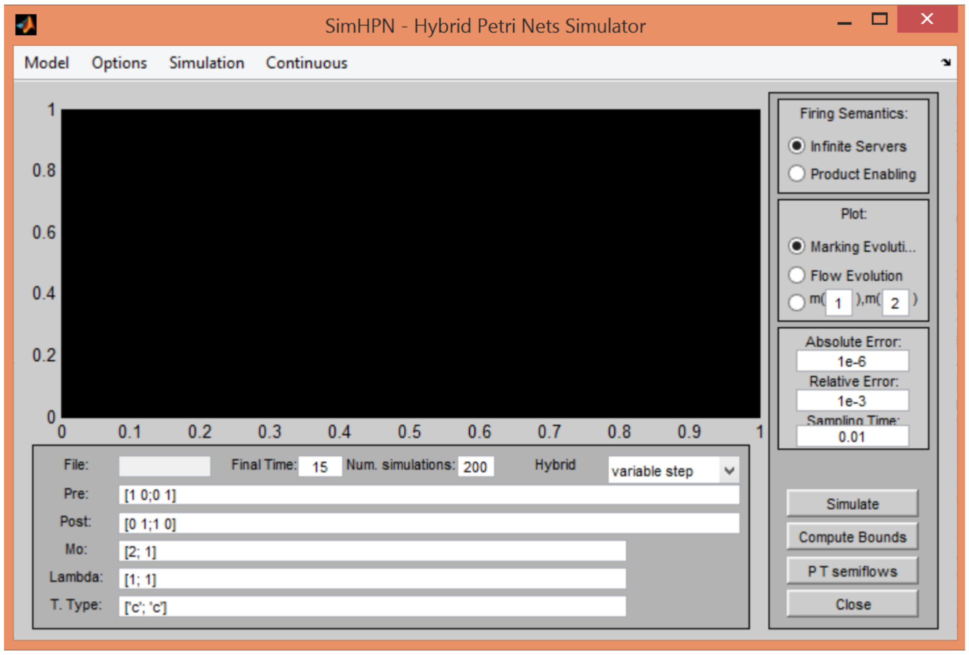
SimHPN is a software tool/program embedded in MATLAB, which is used in the simulation of Hybrid Petri net systems [7]. A short introduction to the SimHPN tool and its GUI interface is discussed in this section. The input to a SimHPN tool can be imported from a *.mat* file or given directly through the input GUI window of the simulator. Fig. 4.8 shows the GUI of the SimHPN simulator. The inputs are given to the fields named *Pre, Post, Lambda, M0, T.Type*. The description of the inputs is as follows:

- *Pre*: Indicates the input incident matrix of the Hybrid Petri net B^-.
- *Post*: Indicates the output incident matrix of the Hybrid Petri net B^+.
- *Lambda*: Indicates the firing vector of the Hybrid Petri net transitions, V.
- *T.Type*: Indicates the transition type of the Hybrid Petri net transitions. 'c' denotes that the particular transition is of continuous type. 'd' denotes the particular transition is of discrete type.
- *M0*: Indicates the initial marking/state of the Hybrid Petri net/system.

The output of the simulator is the marking evolution of the Hybrid Petri net system. It shows the marking of the system when it proceeds through a sequence of states.

4.2.2 Simulation results in SimHPN

This section details the results got by simulating the forward and reverse parallel parking process from the Hybrid Petri net model. As discussed previously, there are different initial markings for the HPN model, based on whether the parallel parking is forward parallel parking or reverse parallel parking.



Firing Semantics: [✓ Infinite Servers] Product Enabling

Plot: [✓ Marking Evoluti...] Flow Evolution m(1),m(2)

Absolute Error: 1e-6

Relative Error: 1e-3

Sampling Time: 0.01

File:

Final Time: 15

Num. simulations: 200

Hybrid: variable step

Pre: [1 0;0 1]

Post: [0 1;1 0]

Mo: [2; 1]

Lambda: [1; 1]

T. Type: ['c'; 'c']

Fig. 4.8. SimHPN simulator GUI

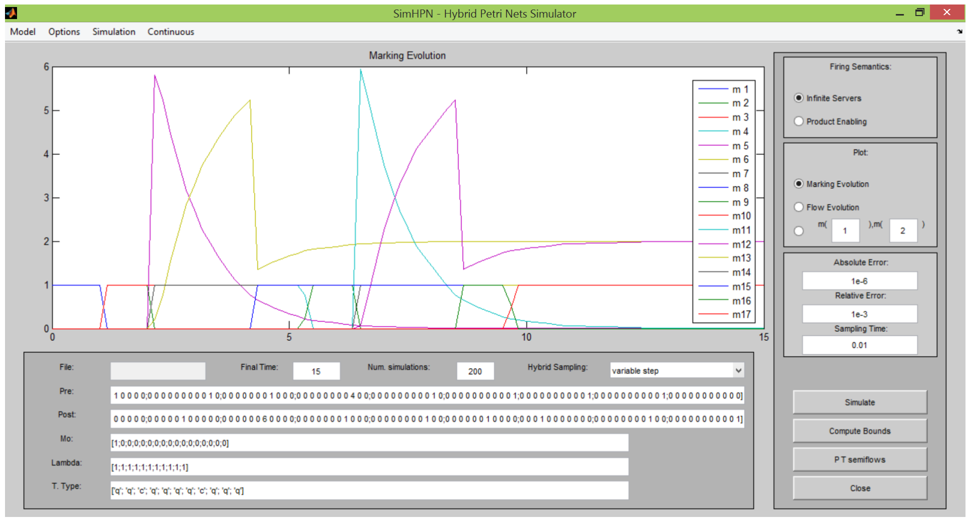
Reverse parallel parking

The initial marking for this case is,

$$M_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

Giving this as the input in M0 field of the GUI window and giving the rest of the inputs as discussed in the previous section, we can get the simulated result by clicking on the simulate button in the GUI. Fig. 4.9 shows the result of simulating the forward parallel parking with the HPN model. It shows the marking of all the places present in the model.

We can refine this result by plotting only the markings of the places which indicate the major events happening in the parallel parking system. So it is enough to monitor the marking evolution of P_5^c, P_6^c, P_{11}^c, P_{12}^c, P_{15}^d, P_{16}^d, P_{17}^d. The places P_5^c, P_6^c, P_{11}^c and P_{12}^c will depict the distance dynamics. P_{15}^d will indicate the end of backing right event. P_{16}^d will indicate the end of moving forward right event. P_{17}^d will indicate the



Firing Semantics: [✓ Infinite Servers] Product Enabling

Plot: [✓ Marking Evolution] Flow Evolution m(1),m(2)

Absolute Error: 1e-6

Relative Error: 1e-3

Sampling Time: 0.01

File:

Final Time: 15

Num. simulations: 200

Hybrid Sampling: variable step

Pre: 1 0 0 0 0;0 0 0 0 0 0 0 0 0 1 0;0 0 0 0 0 0 0 1 0 0 0;0 0 0 0 0 0 0 0 4 0 0;0 0 0 0 0 0 0 0 0 1 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 0 0 0 0 0 0 1;0 0 0 0 0 0 0 0 0 0 0]

Post: 0 0 0 0 0;0 0 0 0 0 1 0 0 0 0 0;0 0 0 0 0 0 6 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 0;0 0 0 0 0 0 1 0 0 0 0;0 0 1 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 1 0 0;0 0 0 0 0 0 0 0 0 0 1]

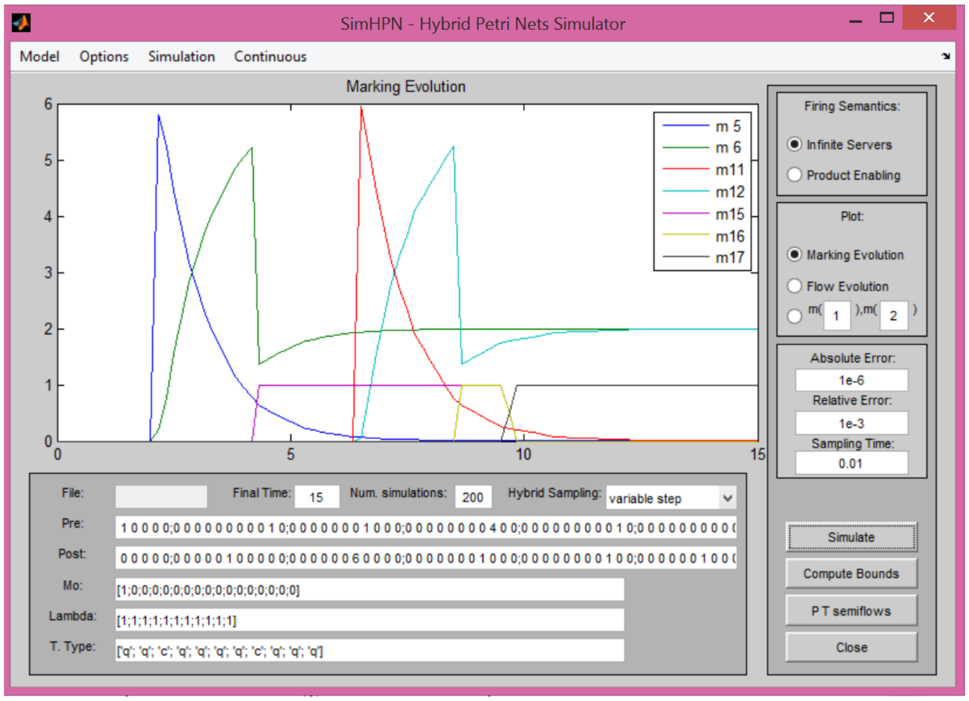
Mo: [1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0]

Lambda: [1;1;1;1;1;1;1;1;1;1;1]

T. Type: ['q'; 'q'; 'c'; 'q'; 'q'; 'q'; 'q'; 'c'; 'q'; 'q'; 'q']

Fig. 4.9. Simulation of the reverse parallel parking scenario

end of the parallel parking process. Fig. 4.10 shows the simulation result containing the marking evolution of places indicating only the major events. The place P_5^c has a marking of 6 initially. As the backing right event is initiated, the marking of the place P_6^c increases from the value 0 gradually. Finally, the marking of the place P_{15}^d reaches a discrete value 1 when the marking of the place P_6^c reaches a value 4. It has to be noted that the marking of the place P_6^c goes on increasing in the simulation , since there is no priority set to stop t_3^c from depositing tokens in the place P_6^c. Finally the marking of the place P_5^c will reach a value 0 and that of P_6^c will reach a value 2, indicating the safety distance of 2 feet at the end of the backing right event. Similarly, P_{11}^c and P_{12}^c will also reach the values of 0 and 2 respectively. P_{16}^d will reach a discrete value 1 to indicate the end of moving forward right event. Finally, the marking of the place P_{17}^d reaches 1 after the end of the major events backing right and moving forward right, to indicate the end of the parallel parking process.



Firing Semantics: [✓ Infinite Servers] Product Enabling

Plot: [✓ Marking Evolution] Flow Evolution m(1),m(2)

Absolute Error: 1e-6

Relative Error: 1e-3

Sampling Time: 0.01

File:

Final Time: 15

Num. simulations: 200

Hybrid Sampling: variable step

Pre: 1 0 0 0 0;0 0 0 0 0 0 0 0 0 1 0;0 0 0 0 0 0 0 1 0 0 0;0 0 0 0 0 0 0 0 4 0 0;0 0 0 0 0 0 0 0 0 1 0;0 0 0 0 0 0 0 0 0 0

Post: 0 0 0 0 0;0 0 0 0 0 1 0 0 0 0 0;0 0 0 0 0 0 6 0 0 0 0;0 0 0 0 0 0 0 1 0 0 0;0 0 0 0 0 0 0 0 1 0 0;0 0 0 0 0 0 0 1 0 0

Mo: [1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0]

Lambda: [1;1;1;1;1;1;1;1;1;1;1]

T. Type: ['q'; 'q'; 'c'; 'q'; 'q'; 'q'; 'q'; 'c'; 'q'; 'q'; 'q']

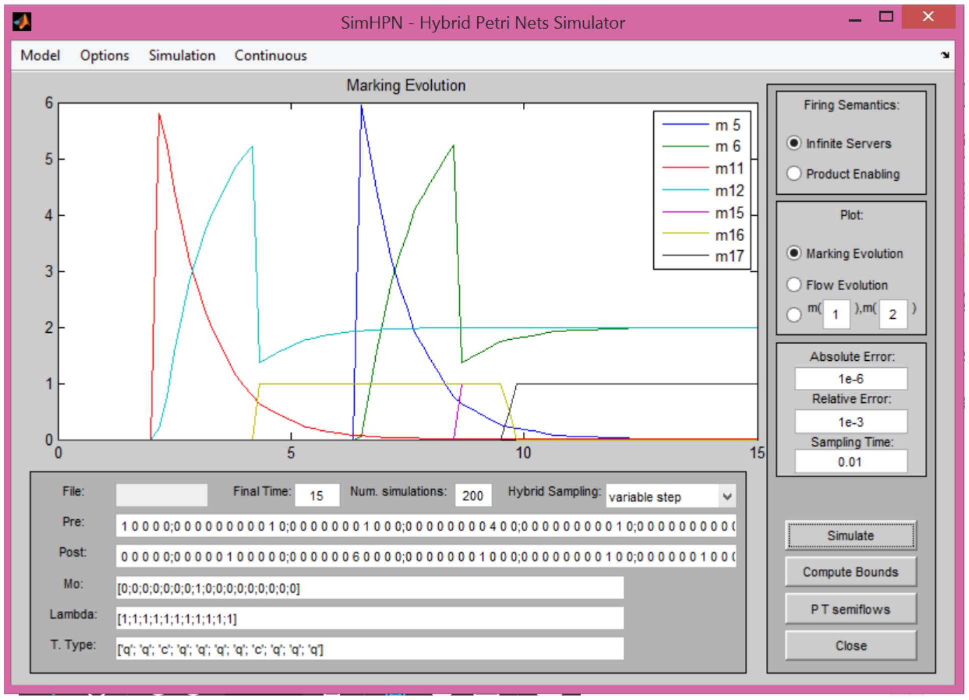
Fig. 4.10. Simulation of the reverse parallel parking indicating the high level events

Forward parallel parking

The initial marking for this case is,

$$M_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

Giving this as the input in M0 field of the GUI window and giving the rest of the inputs as discussed in the previous section, we can get the simulated result for the forward parallel parking process as shown in the Fig. 4.11.



Firing Semantics: [✓ Infinite Servers] Product Enabling

Plot: [✓ Marking Evolution] Flow Evolution m(1),m(2)

Absolute Error: 1e-6

Relative Error: 1e-3

Sampling Time: 0.01

File:

Final Time: 15

Num. simulations: 200

Hybrid Sampling: variable step

Pre: 1 0 0 0 0;0 0 0 0 0 0 0 0 0 1 0;0 0 0 0 0 0 0 1 0 0 0;0 0 0 0 0 0 0 0 4 0 0;0 0 0 0 0 0 0 0 0 1 0;0 0 0 0 0 0 0 0 0 0

Post: 0 0 0 0 0;0 0 0 0 0 1 0 0 0 0 0;0 0 0 0 0 0 6 0 0 0 0;0 0 0 0 0 0 0 1 0 0 0;0 0 0 0 0 0 0 0 1 0 0;0 0 0 0 0 0 0 1 0 0 0

Mo: [0;0;0;0;0;0;0;1;0;0;0;0;0;0;0;0;0]

Lambda: [1;1;1;1;1;1;1;1;1;1;1]

T. Type: ['q'; 'q'; 'c'; 'q'; 'q'; 'q'; 'q'; 'c'; 'q'; 'q'; 'q']

Fig. 4.11. Simulation of the forward parallel parking scenario indicating the high level events

5. CONCLUSION

5.1 Summary of the work

This thesis is an attempt to model the automated parallel parking system using Hybrid Petri net and simulate the behavior of the system. Petri net modeling concept simplified the process of graphically representing the complex dynamics of the system. So, initially, a Discrete Petri net model was developed, which modeled the major events which happen during the parallel parking process. There were several versions of this model, each with its own set of drawbacks. The drawbacks were rectified and the final model was tested using MATLAB algorithms to check if the model correctly portrays the sequence of events. Then the continuous dynamics of the system was discussed and was modelled by Continuous Petri net. This Continuous Petri net model was then integrated with the Discrete Petri net model and was tested again with MATLAB. Then SimHPN, a MATLAB embedded tool to simulate the Hybrid Petri nets was introduced and the results obtained from simulation using SimHPN were also discussed. Finally, the modeling using Hybrid Petri net is justified using the results from both MATLAB and SimHPN. The practical application of this thesis would be using this modeling approach to model control algorithms of various systems like adaptive cruise control, car safety systems, etc. and to evaluate their design and working mechanisms. This approach can be potentially used to evaluate every control system, and systems that involve complex dynamics. The algorithms governing these systems could be tested for errors. Debugging various components of the algorithm becomes simpler with this approach, as the working of the system could be visualized using simulations from SimHPN.

5.2 Future work

There is a broad scope for future work in this thesis. This work has taken into account the high level events/transitions happening in the parallel parking system. Hence this work can be expanded to include many low level or complex intermediate events, to closely approximate the real life parallel parking system. Another direction in which this work could be expanded is to include other continuous dynamics like steering angle, position of the car etc. and model them using Continuous Petri nets as well to simulate the behavior of the system. This would be a good approach to closely approximate a parallel parking system. Finally, the modeling procedure can be tried using Fluid Stochastic Petri nets (FSPN), Timed Hybrid Petri nets, which take into account the stochastic probabilities of the events and the time involved respectively. Modeling using these various types of Petri nets would also be a very good approach to simulate a real life parallel parking system.

REFERENCES

REFERENCES

[1] T. Murata, "Petri nets: Properties, analysis and applications", *Proceedings of the IEEE*, vol.77, issue 4, pp. 541 - 580, Apr 1989.

[2] R. David, H. Alla, "On Hybrid Petri Nets", (LAST DATE ACCESSED: October 08, 2015), http://webdiis.unizar.es/asignaturas/MSC/hybrid.pdf.

[3] R. David, H. Alla, "Discrete, Continuous, and Hybrid Petri Nets", *Springer*, first edition, September 2005.

[4] L. Ghomri, H. Alla, "Modeling and simulation by hybrid Petri nets" *Simulation Conference (WSC), Proceedings of the 2012 Winter. IEEE* pp. 1-8, 2012.

[5] L. Ghomri, H. Alla, "Modeling and analysis using hybrid Petri nets" *Nonlinear Analysis: Hybrid Systems* vol.1, no. 2, pp. 141-153, 2007.

[6] C. G. Cassandras, S. Lafortune, "Introduction to Discrete Event Systems", *Springer*, Second edition, 2008.

[7] J. Julves, C. Mahulea, "SimHPN: A MATLAB Toolbox for Hybrid Petri Nets USER MANUAL", (LAST DATE ACCESSED: October 08, 2015) http:webdiis.unizar.es/GISED/tools/contpn/SimHPNman.pdf vol. 1, 2012 .

[8] C. Mahulea, J. Julves, M. Silva and R. Laura, "On control of Continuous Petri nets", (LAST DATE ACCESSED: October 08, 2015) *http://www.diee.unica.it/ seatzu/slides_Mahulea.pdf* .

[9] R. David, "Modeling of hybrid systems using continuous and hybrid Petri nets.", *Proceedings of the IEEE Seventh International Workshop on Petri Nets and Performance Models*, pp. 47-58, 1997.

[10] D. Ling-Xun, D. Li-Hua, Y. Hong-Ju, and L. Hang, "Hybrid modeling of control system based on hybrid petri nets.", *IEEE International Conference on Control and Automation, ICCA 2007.*, pp. 2158-2162, 2007.

[11] A. Bobbio, M. Gribaudo, and A. Horvath. "Modeling a car safety controller in road tunnels using hybrid petri nets." *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pp. 1436-1441, 2006.

[12] Y. Qu, Li. Lingxi, Y. Chen, and Y. Dai. "Design of fault tolerant controllers in parallel parking systems." *12th International IEEE Conference on Intelligent Transportation Systems, ITSC'09*, pp. 1-6, 2009.

[13] C. K. Lee, C. L. Lin, and B. M. Shiu. "Autonomous vehicle parking using artificial intelligent approach.", *IEEE 4th International Conference on.Autonomous Robots and Agents, 2009. ICARA 2009*, pp. 496-501, 2009.

[14] Y. K. Lo, A. B. Rad, C. W. Wong, and M. L. Ho. "Automatic parallel parking.", *IEEE proceedings on Intelligent Transportation Systems*, vol. 2, pp. 1190-1193, 2003.

[15] K. Jiang, "A sensor guided parallel parking system for nonholonomic vehicles.", *IEEE proceedings on Intelligent Transportation Systems*, pp. 270-275, 2000.

[16] I. E. Paromtchik, and L. Christian, "Autonomous parallel parking of a nonholonomic vehicle.", *Proceedings of the 1996 IEEE symposium on Intelligent Vehicles*, pp. 13-18, 1996.

[17] I. E. Paromtchik, and L. Christian, "Automatic parallel parking and returning to traffic manoeuvres.", *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'97*, vol. 3, pp. V21-V23, 1997.

[18] C. W. Cheng, S. J. Chang, and T. H. S. Li, "Parallel-parking control of autonomous mobile robot.", *23rd International Conference on Industrial Electronics, Control and Instrumentation, IECON 97.* , vol. 3, pp. 1305-1310, IEEE, 1997.

[19] K. Demirli and M. Khoshnejad, "Autonomous parallel parking of a car-like mobile robot by a neuro-fuzzy sensor-based controller." *Fuzzy Sets and Systems*, vol. 160, no. 19, pp. 2876-2891. 2009.

[20] P. M. Moghri, M. R. Karami, and R. Ghaderi, "A real-time intelligent parallel parking system for a car like mobile robot.", *16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP)*, pp. 532-537, IEEE, 2012.

[21] F. Mahi, F. Debbat, A. Nait-Sidi-Moh, and M. F. Khelfi, "A multimodal transportation system: hybrid Petri net-based modeling and simulation.", *IEEE International Conference on Complex Systems (ICCS), 2012*, pp. 1-6, 2012.

[22] A. D. Febbraro, D. Giglio and N. Sacco, "Modular representation of urban traffic systems based on hybrid Petri nets.", *Proceedings of 2001 IEEE conference on Intelligent Transportation Systems*, pp. 866-871, 2001.

[23] A. D. Febbraro, D. Giglio and N. Sacco, "Urban traffic control structure based on hybrid Petri nets.", *Intelligent Transportation Systems, IEEE Transactions*, vol. 5, no. 4, pp. 224-237, 2004.

[24] L. G. Zhang, Z. L. Li, and Y. Z. Chen. "Hybrid petri net modeling of traffic flow and signal control.", *IEEE International Conference on Machine Learning and Cybernetics*, vol. 4, 2008.

[25] M. Dotoli, M. P. Fanti, and G. Iacobellis. "An urban traffic network model by first order hybrid Petri nets.", emphIEEE International Conference on Systems, Man and Cybernetics, SMC 2008, pp. 1929-1934, 2008.

[26] J. Wang, "Petri Nets for Dynamic Event-Driven System Modeling," *Handbook of Dynamic System Modeling*, Ed: Paul Fishwick, CRC Press, 2007.

[27] F. Kaakai, S. Hayat, and A. El Moudni. "A hybrid Petri nets-based simulation model for evaluating the design of railway transit stations.", *Simulation Modelling Practice and Theory*, vol. 15, no. 8, pp. 935-969, 2007.