Graduate School ETD Form 9
(Revised 12/07)

# PURDUE UNIVERSITY
## GRADUATE SCHOOL
## Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Omar Seddeq Omar Yaqub

Entitled
MODELING, ANALYSIS, AND SIMULATION OF TWO CONNECTED INTERSECTIONS USING DISCRETE AND HYBRID PETRI NETS

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

Is approved by the final examining committee:

Lingxi Li
Chair

Yaobin Chen

Maher Rizkalla

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Lingxi Li

Approved by: Yaobin Chen 12/03/2012
Head of the Graduate Program Date

Graduate School Form 20
(Revised 9/10)

# PURDUE UNIVERSITY
## GRADUATE SCHOOL

## Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:

MODELING, ANALYSIS, AND SIMULATION OF TWO CONNECTED INTERSECTIONS USING DISCRETE AND HYBRID PETRI NETS

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

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Executive Memorandum No. C-22,* September 6, 1991, *Policy on Integrity in Research.**

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Omar Seddeq Omar Yaqub

Printed Name and Signature of Candidate

12/03/2012

Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/c_22.html

# MODELING, ANALYSIS, AND SIMULATION OF TWO CONNECTED INTERSECTIONS USING DISCRETE AND HYBRID PETRI NETS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Omar Seddeq Omar Yaqub

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

December 2012

Purdue University

Indianapolis, Indiana

To my father, mother, brothers, and sister.

## ACKNOWLEDGMENTS

I would like to recognize my advisor, Dr. Lingxi Li, for his unceasing assistance since I started the Masters program at IUPUI. In particular, I greatly appreciate his support on this thesis. I also want to thank the other members of my thesis committee, Dr. Yaobin Chen and Dr. Maher Rizkalla for their comments and advice. Additionally, I would like to thank my friend, David Johnson, for his assistance formatting my thesis in Latex. Also, I am most thankful for all the feedback given by the other members in Dr. Li research group, especially Dr. Maria Cabasino for her encouragement and advice. Lastly, I am grateful for the assistance of Sherrie Tucker and all other faculty and staff of the ECE department for which the completion of this work would not be possible without.

# TABLE OF CONTENTS

# LIST OF TABLES

## LIST OF FIGURES

## ABSTRACT

Yaqub, Omar Seddeq Omar. M.S.E.C.E., Purdue University, December 2012. Modeling, Analysis, and Simulation of Two Connected Intersections Using Discrete and Hybrid Petri Nets. Major Professor: Lingxi Li.

In recent decades, Petri nets (PNs) have been used to model traffic networks for different purposes, such as signal phase control, routing, and traffic flow estimation, etc. Because of the complex nature of traffic networks where both discrete and continuous dynamics come into play, the Hybrid Petri net (HPN) model becomes an important tool for the modeling and analysis of traffic networks. In Chapter 1 a brief historical summery about traffic systems control and then related work is mentioned followed by the major contributions in this research. Chapter 2 provides a theoretical background on Petri nets. In Chapter 3, we develop a HPN model for a single signalized intersection first, then we extend this model to study a simple traffic network that consists of two successive intersections. Time delays between different points of network are also considered in order to make the model suitable for analysis and simulation. In addition to HPN models, we also consider discrete Petri nets where their modeling simplicity enables the characterization of the occurrences of all events in the system. This discrete PN is particularly useful to give a higher-level representation of the traffic network and study its event occurrences and correlations. In Chapter 4, we build a discrete PN model to represent a traffic network with two successive intersections. However, we find that the model leads to unbounded places which cannot accurately reflect the dynamics of the traffic in terms of event occurrences. Hence, we introduce the Modified Binary Petri nets (MBPN) to overcome the limitation and resolve the confliction problem when we design our controllers. This MBPN model is a powerful tool and can be useful for the modeling and analysis of many other ap-

plications in traffic networks. Chapter 5 gives a summary for each chapter, provides conclusion and discusses future work for both discrete and hybrid Petri nets.

# 1. INTRODUCTION

The traffic congestion problem appeared along with the evolution of car industry in the 1960s. Since then, it has been getting worse especially in large and fast-growing cities. Congestions in traffic systems lead to severe consequences such as reducing the efficiency and safety of traffic systems, degradation of traffic network infrastructure, and increasing the environmental pollutions. Therefore, system model, analysis, simulating, control, and optimization of traffic networks have been studied extensively [1].

## 1.1 Traffic Management Strategies

Back in 1960s, research regarding control and optimization of vehicles motilities inside cities is known as traffic-signaling control strategy. It mainly depends on changing some parameters of traffic-signal with the objective to enhance the throughput of a single intersection or a traffic network and guarantee the safety operation of the traffic network. Generally, a specific performance index is chosen to be optimized such as maximizing the number of vehicles that use the traffic network during a deterministic time period, or minimizing and equalizing the queues lengths at the intersections, such as the problem studied by Lei and Ozguner [2]. According to the number of intersections considered, traffic-signaling control strategies are divided into isolated strategies and coordinating strategies. It is obvious that isolated strategies are those related to problems of controlling a single intersection while coordinating strategies are applied to the problems of traffic networks (which consist at least of two intersections and the roads connecting them). Furthermore, according to the time response feature, traffic signaling control strategies are classified as fixed-time and traffic-response strategies [1, 3, 4].

In fixed-time strategies, historical data and statistics of vehicles flow at specific points in traffic network during deterministic time periods are used to solve a bunch of mathematical equations (linear or non-linear) to determine the optimal values of control parameters in order to optimize the chosen performance index. For single intersection problems, splits (green light durations) and the phase plan (how many phases shall be applied and what vehicle movements are allowed during each phase) are the main controlling parameters. For traffic network problems, in addition to these parameters, the offsets between intersections traffic light cycles are also considered. Clearly, these parameters would be different for different chosen time periods because vehicle behavior will not be the same; vehicle flow at a specific location in traffic network is not the same during rush hours and off-peak time. The most known fixed-time strategy is traffic network signal tool (TRANSYT) which first introduced by Robertson [5]. In TRANSYT, vehicle travel time in a bounded zone of traffic network and the number of stops are used as performance indices to be optimized. According to Papageorgiou et al. [1], the first practical implementation of TRANSYT leads to minimize vehicle average travelling time in a traffic network by 16% [1,3,5,6].

Instead of using historical data, responsive-time strategy is adopted based on real-time measurements. The controlling process in responsive-time strategy is performed through the continuous acquiring of measurements from upstream intersections inductive loops and digital cameras, and sending them to a central computer for process. Dynamic programming strategies are used to obtain the optimal values of splits and offsets [7]. SCOOT (split cycle offset optimization technique) [8] is the most known and usable responsive-time strategy. It is considered as the responsive-time version of TRANSYT that was introduced by Robertson et al. [1, 3, 8, 9].

Recently, the traffic management problem has considered many other objectives other than improving the vehicle mobility, for instance, saving gas and energy, reducing pollutions, etc. These problems are now addressed in the framework of intelligent transportation systems (ITS), where a variety of objectives have been considered, such as improving the mobility of special vehicles (such as public transportations and emergency vehicles), reducing the environmental impact of traffic systems, and minimizing the gas usage of vehicles.

## 1.2 Petri Net Modeling of Traffic Systems (Previous Related Work)

Recently, much attention has been given to discrete event systems for the modeling and analysis of traffic systems and networks. As a powerful graphical and mathematical tool for the modeling and analysis of practical complex systems, Petri nets have been used to model traffic network in different ways for a variety of purposes. It can be concluded that when vehicle flow has been studied, hybrid Petri net (HPN) [10–12] is a suitable representation tool because it consists of both continuous and discrete nodes that are working together to reflect the dynamics of the entire traffic system. Continuous nodes are suitable for modeling continuous dynamics such as vehicles flow, while discrete nodes are used to represent discrete events such as changing in traffic signal and/or enabling/disabling vehicle movement because of emergent events as incase of accidents and the blocking roads. Febbraro and Sacone [13] developed a general HPN model for transportation system. Traffic flow was described by continuous nodes and the events that affect the traffic dynamics were modeled through discrete nodes. Febbraro et al. [14] used a simple HPN to model the intersection of two one-way streets, while Andzquez et al. [15] used a continuous Petri net to model a non-signalized intersection. Adding discrete nodes were essential to represent a four-way intersection with two phase traffic light through a HPN. Febbraro et al. [3] have also developed the HPN model to improve the performance of special vehicles such as buses and emergency vehicles.

In aforementioned works, HPN models were used since they were more accurate to reflect the dynamics for the entire traffic network for certain applications. On the other hand, in some traffic network applications such as in control and monitoring, only events are the desired objectives to be studied and analyzed. For these problems discrete Petri nets (DPN) are the suitable representation tools. List and Cetin [16] proposed a DPN model which described a traffic light signals phase change for an intersection. Qu et al. [17] used a DPN model for a small transportation system to estimate the optimal travel route between the starting and destination points. Wu et al. [18] used a DPN model to represent control of a simple two-way intersection used by intelligent vehicles.

## 1.3 Thesis Contribution

The variety of applications of Petri nets in traffic networks in general was a motivating factor in developing two intersection models using both hybrid and discrete Petri nets. Regarding to hybrid Petri nets, this work objectives (i) the design of a general model for a signalized intersection that can be used to represent vehicles mobility across the intersection considering simulating details such as time intervals that vehicles spend to cross the intersection and time intervals between successive phases (if required). (ii) Extending intersection modeling to a network modeling that simulate and analyze a connection between two successive, an area that has not been researched using Petri nets. While dealing with the complexity of the HPN design, discrete Petri nets were used in the design of high level events models for traffic network (iii). Such a model will not offer much information such as the hybrid model. However, it will be a suitable representation for problems such as analyzing the affects of an event occurrence somewhere on the network on other events taking place somewhere else. In designing such a model some difficulties were experienced due to the limitation of traditional discrete Petri nets such us having unbounded places which were not desirable in the design. Using Petri net control equations was not a suitable or practical solution; it led conflict in transitions firing. Therefore, the modified binary Petri nets (MBPN) were introduced to overcome the problems faced in design of the event-model (iv).

## 1.4 Thesis Organizing

In this thesis, definitions, classifications and dynamics of Petri nets are introduced in Chapter 2 to give the reader the basic background to understand the following chapters. Definitions, classifications and dynamics of Petri nets are supported by simple examples. After this theoretical background, two Petri net models for two successive signalized intersections were proposed. The first proposed model, introduced in Chapter 3, is a Hybrid model. The motivation for designing such a model is to establish a general Petri net framework for traffic networks that can be used to tackle typical traffic problems such as the analysis and simulation of vehicle mobility, optimization, and routing problems for large-scale network. The aspect of having an accurate simulation for vehicle mobility in traffic networks requires the modeling of intersections as well as roads connecting them, and then using both discrete and continuous nodes to simulate the travelling time between any two points in the network (e.g., between the point where vehicles leave an intersection and the point where vehicles enter the following intersections). Also, the time that vehicles take to cross an intersection is considered. Then the model is simulated through a demo-version of HPN simulator issued this year. In Chapter 4 a discrete model is introduced. This model is based mainly on higher-level event occurrences without considering details of lower-level continuous dynamics. For instance, it can represent the event of vehicles crossing an intersection during a specific traffic light phase from a specific entrance to a known destination. However, this model does not consider information such as how many vehicles have crossed the intersection or how much time this process takes. The advantage is that this model gives an abstract view of the entire complex network and can be used to analyze the effect of the occurrence of a specific event in a specific node on other parts of the traffic network. After the DPN model was developed for two successive intersections, its disadvantage of having some unbounded places was identified and analyzed. Hence, a modified binary Petri net (MBPN) model was proposed to overcome this limitation since it is more suitable for the analysis and control of the connected intersections. Finally, Chapter 5 draws conclusions and presents the future work.

# 2. BACKGROUND ON PETRI NETS

## 2.1 Definitions and Discrete Petri Nets Structure

Petri nets are tools for modeling systems, which consist of a combined graphical and mathematical representation that offers supervision of the events and / or time evolution of systems and processes. A Petri net is a bipartite directed graph comprised of two types of vertices or nodes, called places and transitions, which are connected through edges that are called arcs. While places are represented by circles, transitions are represented by thick bars. A Petri net graph contains a finite but non-zero number of places, transitions, and arcs. Arcs can only connect different types of nodes together: a place to a transition or a transition to a place. Each arc has a specific weight, which should be a finite positive integer. For an arc, if this number is not shown on the Petri net diagram or its attached description, it is assumed to be one.

Generally, as described in other works [10, 19, 20], an unmarked Petri net (UPN) is structured based on four elements and is defined as

$$UPN = (\mathbf{P}, \mathbf{T}, B^-, B^+)$$

where

**P** is a finite and non-empty set of places of size $N$

**T** is a finite and non-empty set of transitions of size $M$

$B^-$ is the input incident matrix, which consists of the arcs weights directed from places to transitions such that $B^-(p_i, t_j)$ is the weight of the arc directed from the place $i$ to the transition $j$ . If there is no such arc, then the element $B^-(p_i, t_j)$ is zero.

$B^+$ is the output incident matrix which consists of the arcs weights directed from transitions to places such that $B^+(p_i, t_j)$ is the weight of the arc directed from the transition $j$ to the place $i$. If there is no such arc, then the element $B^+(p_i, t_j)$ is zero. The size of both incident and output matrices, $B^-$ and $B^+$, is $N \times M$.

Fig. 2.1 shows a Petri nets structure. This Petri net contains 4 places, $\mathbf{P} = \{p_1, p_2, p_3, p_4\}$, and three transitions, $\mathbf{T} = \{t_1, t_2, t_3\}$. As shown in Fig. 2.1, the weight of the arc connecting place $p_3$ to transition $t_3$ is two while other arcs have a weight of one since no other weights are shown. Thus, the input and output incident matrices $(B^-, B^+)$ can be written as:

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

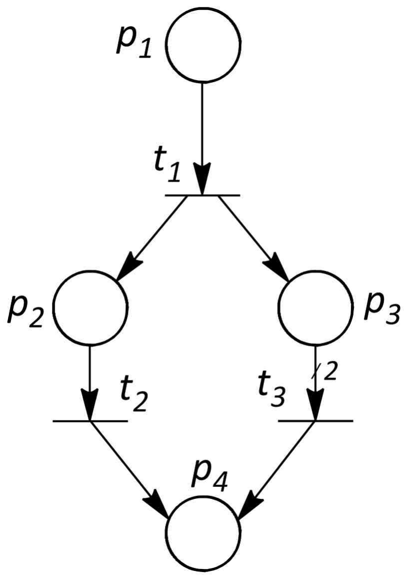


Fig. 2.1. An unmarked Petri net.

### 2.1.1 Marking of Petri Nets

Since a Petri net is a system modeling tool that supervises events, a mechanism is needed in order to present any events or changes in the system or the process. This is represented by adding tokens inside places of the Petri net, where the distribution of tokens is called a marking. A token is graphically presented as a black dot. Each place is marked with any natural number of tokens (i.e. any non-negative integer). However, at least one place must have a non-zero number of tokens in order to call the Petri net a marked Petri net. The number of tokens in a place $i$ is written as $m(p_i)$. Thus in addition to the previously mentioned four elements $(\mathbf{P}, \mathbf{T}, B^-, B^+)$ that define an unmarked Petri net, another element is needed to define a marked Petri net. This is the initial marking of the Petri net, which is written as $m_0$. Therefore the marked Petri net $(MPN)$ is defined as $MPN = (\mathbf{P}, \mathbf{T}, B^-, B^+, m_0)$ where $m_0$ is a column vector that contains the initial marking of all places in the net.

An example of a marked Petri net is shown in Fig. 2.2. This is the same Petri net displayed in Fig. 2.1 but with the following marking applied, $p_3$ with two tokens and both $p_1$ and $p_4$ with one token each. Therefore, the initial marking vector is:

$$m_0 = \begin{bmatrix} 1 & 0 & 2 & 1 \end{bmatrix}^T = \begin{bmatrix} 1 \\ 0 \\ 2 \\ 1 \end{bmatrix}$$

### 2.1.2 Dynamics of Petri Nets

After introducing tokens and marked Petri nets, it is possible to investigate the dynamics of Petri nets, which is the changing in the nets marking through the firing of at least one transition. A transition is said to be enabled if each of its input places contains a number of tokens that is greater than or equal to the arc weight that connects that specific place to the transition. In other words transition $t_j$ is enabled at a specific marking $m_k$ if and only if the condition of Eqn. 2.1 is satisified.

$$m_k(p_i) \geq B^-(p_i, t_j) \tag{2.1}$$

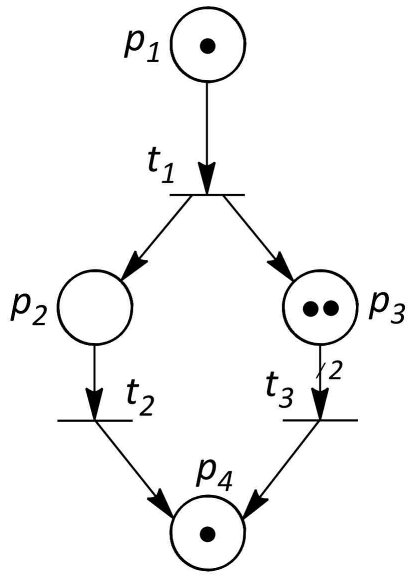


Fig. 2.2. A marked Petri net.

Based on this condition, from Fig. 2.2, it is clear that transitions $t_1$ and $t_3$ are enabled while the transition $t_2$ is not.

Generally, there is a difference between the terms enabled transition and fireable transition. While the enabling condition is defined in the inequality above, firing conditions can be more general; for a transition to be fireable, it should be enabled and it might need to satisfy other conditions such as from external events or a time condition.

For autonomous discrete Petri nets, those Petri nets which their dynamics are not affected by external events not time, as soon as a transition gets enabled, it fires. The dynamics of firing occurs in two steps, but it is an immediate process assumed to not have any time duration. First, a specific number of tokens are moved to the transitions from each input place to that transition. The number of tokens moved is equal to the weight of the arc that connects the input place to the transition. Second, tokens are transferred from the transition to all its output places. The number of tokens transferred to each output place is equal to the weight of the arc connecting the transition to the destination place [19].

To give an example of the firing of a transition, go back to Fig. 2.2. Transitions $t_1$ and $t_3$ are enabled while transition $t_2$ is not. So $t_1$ fires and a token will be removed from place $p_1$ and then a token will be added each to places $p_2$ and $p_3$. Therefore, the marking vector is changed from the initial marking $m_0 = \begin{bmatrix}1 & 0 & 2 & 1\end{bmatrix}^T$ to the marking $m_0 = \begin{bmatrix}0 & 1 & 3 & 1\end{bmatrix}^T$. This simple firing process can be written as $m_0 \xrightarrow{t_1} m_1$.

Starting from the initial marking $m_0$ , consider the firing sequence $\mathbf{S} = \{t_1, t_3, t_2\}$. As mentioned above, by firing the transition $t_1$ we will get the marking $m_1$. And then starting from $m_1$, firing the transition $t_3$ will generate the marking $m_2 = \begin{bmatrix}0 & 1 & 1 & 2\end{bmatrix}^T$ (two tokens are removed from $p_3$ and one token is added to $p_4$), which is written as $m_1 \xrightarrow{t_3} m_2$. Finally, the transition $t_2$ fires generating the marking $m_2 = \begin{bmatrix}0 & 0 & 1 & 3\end{bmatrix}^T$ , which is written as $m_2 \xrightarrow{t_2} m_3$. Thus, starting from the initial marking $m_0$ the firing sequence $\mathbf{S} = \{t_1, t_3, t_2\}$ generates the marking $m_3$, which is written as $m_0 \xrightarrow{t_1, t_3, t_2} m_3$ or $m_0 \xrightarrow{S} m_3$.

To assist the analysis of a Petri nets dynamics, there is a chart showing vectors of possible markings connected through arcs representing the transitions between each marking. This is a reachability graph, in which all possible markings are shown as well as their generated sequence. Fig. 2.3 shows the reachability graph for the Petri net in Fig. 2.1.

$$\begin{tikzcd}
 & {[1\ 0\ 2\ 1]} \arrow[dl, "t_1"'] \arrow[dr, "t_3"] & \\
{[0\ 1\ 3\ 1]} \arrow[d, "t_2"'] \arrow[drr, "t_3"] & & {[1\ 0\ 0\ 2]} \arrow[d, "t_1"] \\
{[0\ 0\ 3\ 2]} \arrow[drr, "t_3"] & & {[0\ 1\ 1\ 2]} \arrow[d, "t_2"] \\
 & & {[0\ 0\ 1\ 3]}
\end{tikzcd}$$

Fig. 2.3. The reachability graph of a discrete Petri net.

In some Petri nets and under specific initial markings, the number of the reachable markings is infinite. For the Petri net in Fig. 2.2, if a small modification is made by adding an arc from the transition $t_2$ to the place $p_1$, the number of reachable markings will be infinite. In other words, the possible number of tokens in some places is unbounded; such Petri nets are known as unbounded Petri nets. From the previous definition, it is clearly impossible to analyze unbounded Petri nets through the reachability graph. Instead, converability root tree technique is used, which is described in detail by David and Alla [10, 19].

### 2.1.3 State Equation of Petri Nets

With the previous Petri net in Fig. 2.2, it has been easy to study all possible firing sequences and figure out all reachable markings. However, the number of possible markings grows exponentially with the number of places in a Petri net and so in analyzing larger and more complicated systems, this method will take much more time and may become unfeasible to use after a certain point. Fortunately, the bipartite structure of Petri nets and the tokens dynamics offers a mathematical representation through linear algebra. This representation makes it easy to figure out the final marking generated from an initial marking through a given firing sequence and so it becomes much easier to analyze the dynamics of large systems by solving this equation through a program.

As previously mentioned, the marked Petri net is defined through the five elements $(\mathbf{P}, \mathbf{T}, B^-, B^+, m_0)$. Both matrices $B^-$ and $B^+$ are $N \times M$ dimensions. Where $N$ is the number of places and $M$ is the number of transitions. It is necessary now to introduce the incidence matrix $B$ which is:

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

So for the Petri net in Fig. 2.2, let $m_k$ be the marking of this Petri net before the firing sequence $\mathbf{S}$ and $m'_k$ be the marking after the occurrence of the firing sequence $\mathbf{S}$, so $m_k \xrightarrow{\mathbf{S}} m'_k$.

The relation between these two markings is known as the state equation of the Petri net, which is given by Eqn. 2.2.

$$m'_k = m_k + B.s \tag{2.2}$$

Where $m_k$ and $m'_k$ are $N \times 1$ marking vectors, $B$ is a $N \times M$ matrix, and $s$ is the characteristic vector of sequence $\mathbf{S}$ with a dimension of $M \times 1$. The component $s_j$ is the number of times the transition $t_j$ is fired in the sequence $\mathbf{S}$. Therefore, for the Petri net in Fig. 2.2 the firing sequence $\mathbf{S} = \{t_1, t_3, t_2\}$ is represented in the state equation through the vector $s = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$. Sticking with the given initial marking $m_0 = \begin{bmatrix} 1 & 0 & 2 & 1 \end{bmatrix}^T$ and the firing sequence $s = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$, it is possible to find final state, which is:

$$\begin{bmatrix} 1 \\ 0 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & -2 \\ 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 3 \end{bmatrix}$$

## 2.2 Continuous Petri Nets

Not all physical systems can be represented by discrete Petri nets. Parameters such as water flow and smoke movement cannot be represented through discrete models. Instead, continuous Petri nets can be used. The structure of the continuous Petri nets is the same as for discrete Petri nets, except in two points:

- The weights of the arcs connecting places to transitions or transitions to places can be any finite positive real number, where it was restricted to integers for discrete Petri nets.
- Instead of having integral number of tokens, places can be marked by any finite non-negative real number [10, 21, 22].

In the graphical representation of continuous Petri nets, a place is represented by a double circle while a transition is represented by a rectangular box. Fig. 2.4 shows an example of a continuous Petri net consisting of three places and two transitions.

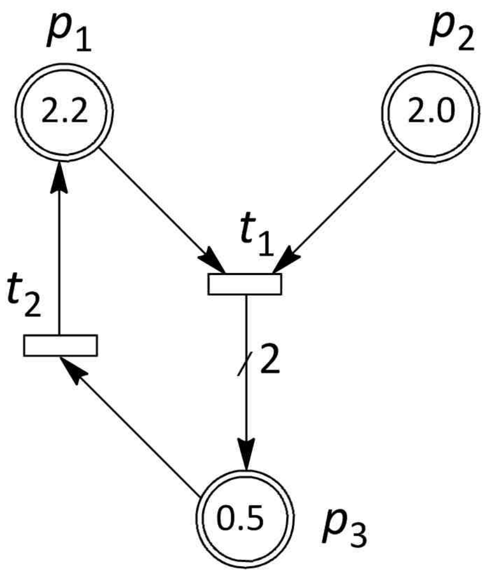


Fig. 2.4. A continuous Petri net.

For continuous Petri nets, a transition $t_j$ is said to be $q$-enabled in a specific marking $m_k$ if the enabling degree $q$ is greater than 0 and is given by Eqn 2.3.

$$q = min_{(i:p_i t_j^{in})} \frac{((m_k(p_i))}{(B^-(p_i, t_j))} \tag{2.3}$$

This means that the enabling degree $q$ of the transition $t_j$ at the marking $m_k$ is the minimum value of division each marking of the transition $t_j$ input places on the weight of the arc connects that place to the transition. Note that q is a finite positive real number [10].

Going back to Fig. 2.4 and applying the definition of enabling degree, it is clear that transition $t_1$ is 2-enabled and the transition $t_2$ is 0.5-enabled.

The dynamics of continuous Petri nets is different from discrete Petri nets in the enabling conditions, which is shown in Fig. 2.4. In order to investigate the firing of a transition and the firing sequence of continuous Petri nets, a new notation is introduced: $\alpha$. The term $[t_j]^\alpha$ means that the transition $t_j$ is fired by the value $\alpha$ at one time. To clarify this notation more, examine the continuous Petri net in Fig. 2.4. The term $[t_2]^{0.3}$ means that the transition $t_2$ firings by the amount of 0.3, so 0.3 tokens are removed from the continuous place $p_3$ and are added to the continuous place $p_1$. Therefore, the marking of the Petri net is changed from $s = \begin{bmatrix} 2.2 & 2.0 & 0.5 \end{bmatrix}^T$ to $s = \begin{bmatrix} 2.5 & 2.0 & 0.2 \end{bmatrix}^T$. The firing sequence and the state equation mentioned for the discrete Petri nets are also applicable for continuous Petri nets, with the only exception being that the components of sequence vector s are real non-negative numbers corresponding to each $[t_j]^\alpha$.

For continuous Petri nets, the number of reachable markings is infinite even when dealing with bounded Petri nets. Therefore, it is impossible to consider all the reachable markings in the reachability graph. Instead, the reachability graph consists only of general markings, called macro-markings, that show whether the places are marked or not (i.e. has tokens or empty). For example, the Petri net given in Fig. 2.4 has three places, so it could have maximum of eight macro-markings: $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$, $\begin{bmatrix} m_1 & 0 & 0 \end{bmatrix}^T$, $\begin{bmatrix} 0 & m_2 & 0 \end{bmatrix}^T$, $\begin{bmatrix} 0 & 0 & m_3 \end{bmatrix}^T$, $\begin{bmatrix} m_1 & m_2 & 0 \end{bmatrix}^T$, $\begin{bmatrix} m_1 & 0 & m_3 \end{bmatrix}^T$, $\begin{bmatrix} 0 & m_2 & m_3 \end{bmatrix}^T$, $\begin{bmatrix} m_1 & m_2 & m_3 \end{bmatrix}^T$, where the values $m_1$, $m_2$, and $m_3$ are positive real numbers. Clearly, the maximum number of macro-markings for a Petri net consisting of $N$ places is $2^N$. Even though the initial markings of the Petri net in Fig. 2.4 is considered in the macro-marking $\begin{bmatrix} m_1 & m_2 & m_3 \end{bmatrix}^T$, it can be presented as a separate macro-marking in the reachability graph. Fig. 2.5 illustrates the concept of macro-marking through the reachability graph of the given Petri net. It should be noted that the reachability graph consists only of four macro-markings, which is because the other possible four markings in which the marking $m_1$ is zero are not reachable since it is impossible for the marking $m_1$ to have less than 0.2 tokens.

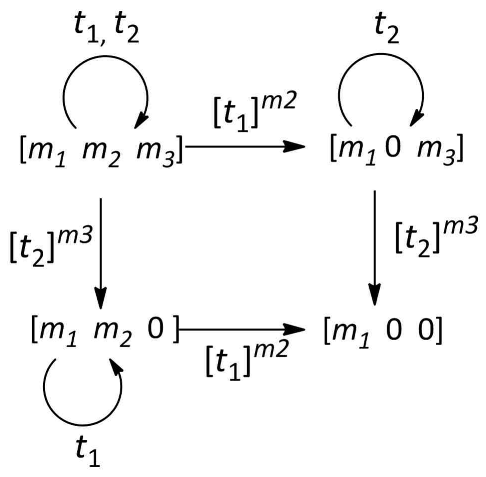


Fig. 2.5. The reachability graph for a continuous Petri net.

## 2.3 Hybrid Petri Nets

As mentioned previously, continuous Petri nets are suitable for modeling systems such as water flow. Nevertheless, there can be systems too complex to be modeled through continuous Petri nets. Consider the example of water flow between two tanks where a pump is used to transfer the water from tank A to tank B while an open/close valve is used for water transfer in the other direction. Although the process of moving the water between the two tanks is continuous, the actions of starting, stopping, and switching the direction of this process is discrete (open/close the valve or turn on/off the pump) [11]. These discrete actions are taken depending on other continuous parameters such as if the level of water in the tank is at a specific point, the temperature of the pump has reached a critical value, etc. Therefore, the optimum representation of this system using Petri nets cannot be achieved except by using both continuous and discrete nodes, in a hybrid Petri net [23–26]. Marked hybrid Petri nets are structured based on these six elements:

$$HPN = (\mathbf{P}, \mathbf{T}, B^-, B^+, m_0, h)$$

where $h$ is the hybrid function, which is used for both places and transitions to indicate whether the node is discrete or continuous:

$$h : \mathbf{P} \cup \mathbf{T} \to \{C, D\}$$

The other five elements $\mathbf{P}$, $\mathbf{T}$, $B^-$, $B^+$, $m_0$ have the same definitions as before. However, in HPN, the set of places $\mathbf{P}$ consists of both continuous and discrete places and the set of transitions T consists of both discrete and continuous transitions. If $\mathbf{P}^C$ is defined as a set that consists of all continuous places, $\mathbf{P}^D$ is defined as a set that consists of all discrete places in a HPN, then $\mathbf{P} = \mathbf{P}^C \cup \mathbf{P}^D$. Similarly, $\mathbf{T} = \mathbf{T}^C \cup \mathbf{T}^D$. Where $\mathbf{T}^C$ is the set of continuous transitions and $\mathbf{T}^D$ is the set of discrete transitions. Furthermore, the initial markings $m_0$ is restricted to finite non-negative integer values for the places in the set $\mathbf{P}^D$ while it could be any finite non-negative real number for the rest of the places (i.e. the set $\mathbf{P}^C$). Usually, the marking $m$ of hybrid Petri nets is described in order depending on the place type; discrete or continuous. If $m^D$ is the marking of discrete Petri nets in a specific hybrid Petri net, $m^C$ is the marking of continuous places of this Petri nets, $md$ is the number of discrete places, and $mc$ is the number of continuous places in this Petri net, then the marking of the HPN is written in the form $m = (m^D, m^C)$ while the marking vector can be described as $\begin{bmatrix} p_1 & p_2 & \cdots & p_{md} & p_{md+1} & \cdots & p_{md+mc} \end{bmatrix}^T$.Finally, the arcs connecting a discrete place and a continuous transition have to be equal in their weights. In other words, if the notations $p_c$, $p_d$, $t_c$, and $t_d$ represent a continuous place, a discrete place, a continuous transition, and a discrete transition, respectively, the relation of Eqn. 2.4 has to be satisfied for all places and transitions.

$$B^+(p_d, t_c) = B^-(p_d, t_c) \tag{2.4}$$

Since hybrid Petri nets have both continuous and discrete nodes, they should have different enabling conditions for the continuous and discrete transitions as well as different token movements to/from continuous and discrete transitions. For discrete transitions, the enabling condition used for discrete Petri nets is still applicable; it does not matter whether the input places are discrete or continuous. In a specific marking $m_k$ a discrete transition is enabled if and only if the condition of Eqn. 2.5 is satisified.

$$m_k(p_i) \geq B^-(p_i, t_j^D) \tag{2.5}$$

Alternatively, the firing condition for a continuous place depends on the type of input place. In a specific marking $m_k$ a continuous transition is enabled if these two conditions are satisfied:

$$m_k(p_i^D) \geq B^-(p_i^D, t_j^C) \tag{2.6}$$

for all input discrete places to the transition $t_j$

$$m_k(p_i^C) > 0 \tag{2.7}$$

for all input continuous places to the transition $t_j$

The evolution behavior of hybrid Petri nets is followed the state equation given in Eqn. 2.2. As for the marking vector, firing sequence vector s is also placed in order that the firing times of discrete transitions in are ordered first, and then those for the continuous once. If $s^D$ gathers the number of firings of discrete transitions in a specific hybrid Petri net, $s^C$ gathers the number of firings of continuous transitions, $nd$ is the number of discrete transitions, and $nc$ is the number of continuous places in this Petri net, then the firing sequence vector of the HPN is written as $s = \begin{bmatrix} t_1 & t_2 & \cdots & t_{nd} & t_{nd+1} & \cdots & t_{nd+nc} \end{bmatrix}^T$. Where the firing times for the transitions in $s^D$ can take only nonnegative integers, while for transitions in $s^C$ the firing times can be any real positive value (or zero). David and Alla have examples [10, 11] that analyze the marking changes of hybrid Petri nets depending on the graphical analysis and using the state equation also to demonstrate the results.

## 2.4 Timed Hybrid Petri Nets

In the previous sections the discrete, continuous, and hybrid Petri nets which were introduced and analyzed are known as autonomous Petri nets in which Petri nets model and simulate only event evolution of physical systems. Obviously, for some physical systems and processes time modeling is an essential issue that has to be taken in account together with the event evolution. These processes can be modeled and simulated through non-autonomous Petri nets, which are generally defined as Petri nets whose dynamics are affected by external events; if these external events are only change in time, then these Petri nets are known also as timed Petri nets. Since in this research, no other external events are modeled, the term, non- autonomous Petri nets, is equivalent to the term, timed Petri net (TPN). Similar to the autonomous Petri nets, TPN are also classified as discrete, continuous, and hybrid. Since in this research only the hybrid non-autonomous Petri nets are used, their structure and dynamics are described in details in this section while other publications [10, 27] can be used to know more about discrete and continuous timed Petri nets. The structure of the marked timed hybrid Petri nets is exactly as the structure of the marked hybrid Petri net with adding one element $\lambda$. So the marked THPN is defined as

$$THPN = (\mathbf{P}, \mathbf{T}, B^{-}, B^{+}, m_0, h, \lambda)$$

The first six elements $\mathbf{P}, \mathbf{T}, B^{-}, B^{+}, m_0, h$ have the same definitions as in the previous section. $\lambda$ is a function that assigns a nonnegative integer number symbolized as $d_i$ to each discrete transition (each element in the set $T^D$) and it assigns a non-negative real number symbolized as $U_i$ to each continuous transition (each element in the set $T^C$). $d_i$ is a time associated with the discrete transition $t_i$ and means that this transition is fired after $d_i$ time units since it becomes enabled. If the time associated with a discrete transition is zero, this transition is fired as soon as it becomes enabled, which is the case of all discrete transitions in non-autonomous Petri nets. One the other hand, $U_i$ is the maximum flow rate associated with the continuous transition $t_i$ which can be used to calculate the maximum firing speed for this transition [10].

After introducing the structure of THPN, transition enabling and the dynamics are analyzed briefly. In THPN, the enabling condition for discrete transitions is the same for those transitions in HPN, which is previously mentioned in Eqn 2.5. Thus, a discrete transition is said to be enabled if each of its input places (discrete and continuous) contains a number of tokens that is greater than or equal to the arc weight that connects that specific place to the transition. For continuous transitions, the enabling concept is a little bit complicated compare to the HPN. The continuous transition $t_i$ is said to be strongly enabled if the two conditions in Eqn. 2.6 and Eqn. 2.7 are satisfied. If only the first condition is met (the input discrete places condition), the continuous transition $t_i$ is said to be weekly enabled if all of its input continuous places that are zero marked have at least an input transition which is firing. In THPN, each continuous transition fires in a deterministic speed, known as the firing speed. This firing speed can be constant or variable. In the model presented in this research, only constant firing speed transitions are used. However, the term constant firing speed does not mean that the transition fires in only one speed all the time, but it implies that for a given marking $m_k^D$ of discrete places each transition has a known maximum firing speed that it does not exceed. Maximum firing speed is given by Eqn. 2.8.

$$V_i = U_i \times D(t_i, m_k^D) \tag{2.8}$$

Where $U_i$ is the associated flow rate to the transition $t_i$ , and $D(t_i, m_k^D)$ is the enabling degree of the continuous transition $t_i$, considering only the discrete input places. If assumed that the continuous transition $t_i$ has k discrete input places, then its enabling degree is the least number of tokens that can be transferred in one shut from one of the $k$ discrete places to this transition. If a continuous transition is not connected to any discrete place, then it is known as an immediate transition that has an infinite flow rate. Transition $t_i$ fires in a speed $v_i$ that is greater than or equal to zero and less than or equal to $V_i$. In the THPN model presented in this research, since all the discrete places that connected to continuous transitions are marked with no more than one token, the maximum firing speed is the same as the maximum flow rate assigned to the transition. Usually in THPN, each continuous transition is connected to at least one discrete place. Fig. 2.6 shows a THPN that consists of six places and four transitions; $p_1$, $p_2$, $p_3$, $t_1$, and $t_2$ are discrete nodes while $p_4$, $p_5$, $p_6$, $t_3$, and $t_4$ are continuous nodes.

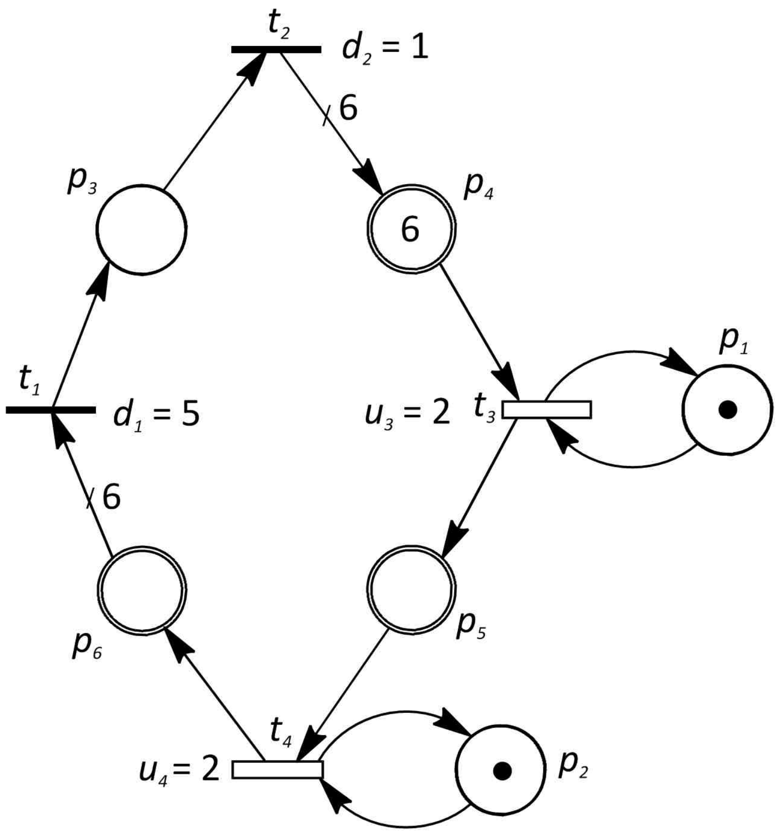


Fig. 2.6. A timed hybrid Petri net.

As shown at time zero, the marking of the Petri net is $m_0 = m(t = 0) = (1, 1, 0, 6, 0, 0)$ while $\lambda = (5, 1, 2, 2)$. To investigate the dynamics of the given Petri net, the first step is to determine the enabled transition at time $t = 0$. It is obvious that transition $t_3$ is strongly enabled and transition $t_4$ is weakly enabled while both discrete transitions $t_1$ and $t_2$ are not enabled. Therefore, at time $t = 0$, transition $t_3$ starts to fire with a firing speed of 2 tokens per time unit. As soon as $t_3$

starts firing, tokens are transferring continuously from the continuous input place $p_4$ to the continuous output place $p_5$ with a rate of 2 tokens per time unit. Thus, at time $\epsilon$ which is a very small real number $\epsilon \rightarrow 0$, transition $t_4$ starts to fire in the same firing speed for $t_3$ (2 tokens per time unit). At time $\epsilon$ the marking of the Petri net becomes $m(t = \epsilon) = (1, 1, 0, 6 - 2\epsilon, 2\epsilon, 0)$. After another small time interval $\epsilon$ passes, the Petri net marking becomes $m(t = 2\epsilon) = (1, 1, 0, 6 - 4\epsilon, 2\epsilon, 2\epsilon)$. During the second infinity small time interval $\epsilon$ another infinity small amount of tokens $(2\epsilon)$ transferred from the place $p_4$ to the place $p_5$ while the amount $2\epsilon$ which already was in the place $p_5$ is transferred to the place $p_6$. Hence, during the time interval (0 ¡t ¡3) the marking of the Petri net is written as $m(t) = (1, 1, 0, 6 - 2\epsilon - 2t, 2\epsilon, 2t)$. Following the same steps, the marking of the Petri net at the exact time $t = 3$ can be estimated to be $m(3) = (1, 1, 0, 0, 2\epsilon, 6 - 2\epsilon)$, and in the time $t = 3 + \epsilon$, the marking $m(3 + \epsilon) = (1, 1, 0, 0, 0, 6)$ is reached. However, since $\epsilon \rightarrow 0$, the approximation $m(3) = m(3 + \epsilon)$ can be used to write the marking of the Petri net after 3 time units as $m(3) = (1, 1, 0, 0, 0, 6)$. It is obvious that the marking of $p_4$ takes an infinity small value all the time during this interval because the firing speeds of the upstream and downstream transitions are equal. Similar scenario is expected if the firing speed of the downstream transition is greater than the firing speed of upstream transition. On the other hand, if the firing speed of the upstream transition is greater than the firing speed of the downstream transition, the place in-between will have a linearly increasing marking until the firing of the upstream transition ends. Coming back to the exampled Petri net, at time $t = 3$, 6 tokens have been transferred from the place $p_4$ to the place $p_6$ through the place $p_5$. Therefore, the discrete transition $t_1$ becomes enabled but it cannot be fired until after passing of another 5 time unites (because of its associated time $d_1 = 5$). At time $t = 8$, the transition $t_1$ fires and 6 tokens are removed from the continuous place $p_6$, and one token is deposited in the discrete place $p_3$. We come up with the marking $m(8) = (1, 1, 1, 0, 0, 0)$. One time unit later (since $d_2 = 1$), the transition $t_2$ fires, so the token at the discrete place $p_3$, is removed and 6 tokens are added to the continuous place $p_4$, to come up with the initial marking again at $t = 9$, $m(9) = m(0) = (1, 1, 0, 6, 0, 0)$. Then the same firing sequence takes place again and so on.

Similar to the discrete, continuous, and autonomous HPN, the dynamics of THPN can be observed and the marking evolution can be calculated using the state equation. However, the state equation for THPN is different from what was introduced before since the time factor is taken into consideration. Assuming the number of discrete places, continuous places, discrete transitions, and continuous transitions in a THPN are $md$, $mc$, $nd$, and $nc$ respectively. Also the total number of places is m where $m = md+mc$ and the total number of transitions is $n$ where $n = nd+nc$. Furthermore, let $v$ be the firing speed vector. The size of $v$ vector is n where the first nd elements are zero (since the discrete transitions do not have a firing speed), while the other nc elements represent the firing speeds of the continuous transitions. Thefore, $\mathbf{v}$ is written in the form $\mathbf{v} = \begin{bmatrix} 0 & 0 & \cdots & 0 & v_{nd+1} & \cdots & v_{nd+nc} \end{bmatrix}^T$ where the element $v_i$ is the firing speed of the continuous transition $t_i$. On the other hand, let $d(t)$ be the vector representing the number of firing times of transitions in the time interval (0 to t). The size of $d(t)$ vector is n where the last nc elements are zero (those which represent the continuous transition), while the first $mc$ elements represent the firing times of discrete transitions in the time interval (0 to t). Thefore, $d(t)$ is written in the form $d(t) = \begin{bmatrix} d_1 & d_2 \cdots & d_i & \cdots & d_{nd} & 0 & \cdots & 0 \end{bmatrix}^T$ where the element $d_i$ is an integer nonnegative number and represents the number of firings of the discrete transitions $t_i$ between 0 and t. Finally, it is possible now to introduce the vector $s(t)$ which is equivalent to the vector s in autonomous Petri nets [10].

$$s(t) = d(t) + \int_0^t v(u).du \tag{2.9}$$

Where $u$ represents the firing rate of the continuous transitions. In the similar form of the one of autonomous Petri nets, the state equation of timed hybrid Petri nets is written as

$$m(t) = m(0) + B.s(t) = m(0) + B.(d(t) + \int_0^t v(u).du) \tag{2.10}$$

Eqn. 2.10 represents the marking evolution of THPN starting at time $t = 0$. It is also possible to write the marking of the THPN at any time $t_2$, if its marking is known at any time $t_1$, such that $0 \leq t_1 < t_2$

$$m(t_2) = m(t_1) + B.(d(t_2) - d(t_1) + \int_{t_1}^{t_2} v(u).du) \tag{2.11}$$

When considering the two vectors $d(t)$ and $v(u)$, the discrete transitions are ordered in front of the continuous ones. Therefore, the vector form of $m(t_i)$ is written as $\begin{bmatrix} m^D(t_i) & m^C(t_i) \end{bmatrix}^T$ or $\begin{bmatrix} p_1(t_i) & p_2(t_i) & \cdots & p_{md}(t_i) & p_{md+1}(t_i) & \cdots & p_{md+mc}(t_i) \end{bmatrix}^T$. Furthermore, the matrices $B^+$ and $B^-$ are written as:

$$B^+ = \begin{bmatrix} B^+_{DD} & B^+_{DC} \\ B^+_{CD} & B^+_{CC} \end{bmatrix}, B^- = \begin{bmatrix} B^-_{DD} & B^-_{DC} \\ B^-_{CD} & B^-_{CC} \end{bmatrix}$$

where $B^+_{DD}$ and $B^-_{DD}$ represent the arc weights between the discrete places and the discrete transitions. $B^+_{CC}$ and $B^-_{CC}$ represent the weights between the continuous nodes. $B^+_{CD}$, $B^-_{CD}$ represent the connections between continuous places and discrete transitions.$B^+_{DC}$, $B^-_{DC}$ represent the connections between discrete places and continuous transitions. Since $B^+_{DC} = B^-_{DC}$, the incident matrix $B$ can be written as:

$$B = \begin{bmatrix} B_{DD} & 0 \\ B_{CD} & B_{CC} \end{bmatrix}$$

Now, it is easy to write Eqn. 2.11 in the matrix form

$$\begin{bmatrix} m^D(t_2) \\ m^C(t_2) \end{bmatrix} = \begin{bmatrix} m^D(t_1) \\ m^C(t_1) \end{bmatrix} + \begin{bmatrix} B_{DD} & 0 \\ B_{CD} & B_{CC} \end{bmatrix} \times \left( \begin{bmatrix} d_1(t_2) - d_1(t_1) \\ d_2(t_2) - d_2(t_1) \\ \vdots \\ d_{nd}(t_2) - d_{nd}(t_1) \\ 0 \end{bmatrix} + \int_{t_1}^{t_2} \begin{bmatrix} 0 \\ v_{nd+1}(u) \\ v_{nd+2}(u) \\ \vdots \\ v_{nd+nc}(u) \end{bmatrix} \mathrm{d}u \right)$$

## 2.5 Conflict in Petri Nets

A conflict is a property of Petri nets that depends on the structure and the marking. In a specific marking $m_k$, when a place $p_i$ is connected as an input place to more than one transition and the number of tokens at $p_i$ is not enough to enable and fire all the possible enabled output transitions, conflict appears between two or more transitions. In Fig. 2.7-a, a conflict in a discrete Petri net is shown. At the shown marking (1,1), two possible firings conflict, either $t_1$ fires or $t_2$ fires. However, if the marking of the net was (0,1), no conflict would exist since transition $t_1$ is not enabled because no tokens exist at its input place $p_1$, so $t_2$ is enabled and it fires. Conflicts occur in continuous and hybrid Petri nets as well. In Fig. 2.7-b, a conflict between a discrete transition and a continuous transition is shown, such a conflict exists in the THPN model introduced in Chapter 3. It is obvious that transitions $t_1$ and $t_2$ cannot be fired at the same time. This conflict is a design aspect that has to be determined (which transition will take the priority) depending on the physical system or process that is modeled [10].

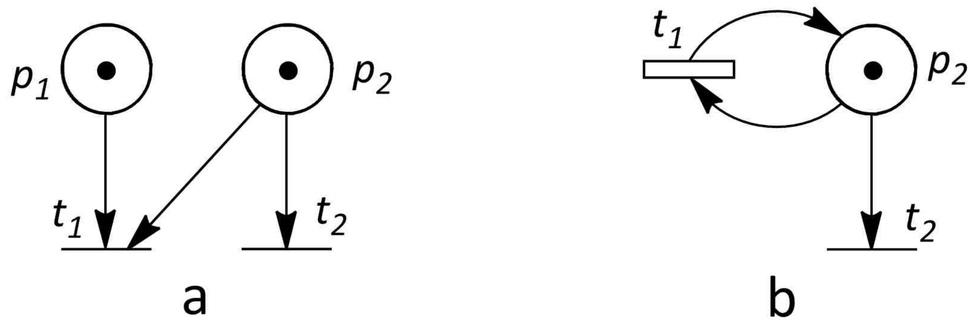


Fig. 2.7. Conflicts in Petri nets.

# 3. HYBRID PETRI NET MODELING OF TWO CONNECTED INTERSECTIONS

In order to design a Petri net model for traffic intersections that capture vehicle flow in different locations and the change of signal phases, discrete and continuous nodes have to be used together. The general modeling concept is to use continuous nodes for vehicle flow and the discrete nodes for the traffic light signal (phase changing). In this thesis, vehicle flows are represented through the firing speeds of continuous transitions. Arc weights represent split and cross-direction factors (e.g., the percentage of vehicles that take left turns, go straight, and take right turns at intersections). Queues of vehicles at intersections are modeled through continuous places. The conflict between continuous and discrete transitions is used to represent decisions of enabling/disabling vehicle movements. Timed transitions play a significant role to capture the time for vehicles to cross the intersection. Furthermore, they are used to build a delay circuit in the modeling of a road connecting two intersections. In this chapter, the structure of the traffic network is reviewed first, and then a hybrid Petri net model for a single intersection is presented, followed by a simple traffic network model that consists of two connected signalized intersections. Finally, a Petri net simulator is used to simulate the traffic flow for these two connected intersections, detailed results are analyzed to show the effectiveness of the proposed modeling approach.

## 3.1 Urban Traffic Network

Generally speaking, an urban traffic network consists of two major elements: Roads and Intersections. As mentioned by Febbraro et al. [3], an urban traffic network can be represented as the set $\mathbf{U}$:

$$\mathbf{U} = \{\mathbf{R}, \mathbf{I}\}$$

.

where $\mathbf{R}$ is a finite set consists of $M$ roads such that $\mathbf{R} = \{R_1, R_2, \cdots, R_m, \cdots, R_M\}$ and $\mathbf{I}$ is a finite set consists of $N$ intersections such that $\mathbf{I} = \{I_1, I_2, \cdots, I_n, \cdots, I_N\}$.

Since this thesis focuses only on signalized intersection, the term intersection is used instead for signalized intersection. The intersection $I_n$ is defined as "an urban traffic network element which connects two or more roads and consists physically of the area which is occupied by vehicles crossing the intersection as well as a part of the adjacent roads from which vehicle flows come and to which vehicle flows go" [3]. The function of the traffic signals is to controls vehicle flows through the intersection. The road $R_m$ is an urban traffic network element which connects two successive intersections and permits only one way travelling direction.

## 3.2 Single Intersection Structure

In Fig. 3.1, an eight-roads intersection model is shown with parts from its accessible roads. Direction symbols $\{n, s, e, w\}$ are used to define the eight roads. They represent the four directions north, south, east, and west, respectively:

In Fig. 3.1, $R_i^{in}, R_i^{out}$ are the road which connected to the intersection from the direction $i$. Traffic in $R_i^{in}$ flows towards the intersection while traffic in $R_i^{out}$ flows out from the intersection. Therefore, the road set $\{R_w^{in}, R_e^{in}, R_n^{in}, R_s^{in}, R_w^{out}, R_e^{out}, R_n^{out}, R_s^{out}\}$ represents the roads connecting to the intersection; the subset $\{R_w^{in}, R_e^{in}, R_n^{in}, R_s^{in}\}$ represents the incoming roads, while the subset $\{R_w^{out}, R_e^{out}, R_n^{out}, R_s^{out}\}$ represents the outgoing roads. Considering vehicle flow, similarly, the notation $f_i^{in}$ represents the flow of the vehicles that travel toward the intersection from the direction $i$, while the notation $f_i^{out}$ represents the flow of vehicles leaving the intersection toward the direction $i$. Therefore, the vehicle flow entering the intersection from the four directions is gathered by the set $\{f_w^{in}, f_e^{in}, f_n^{in}, f_s^{in}\}$

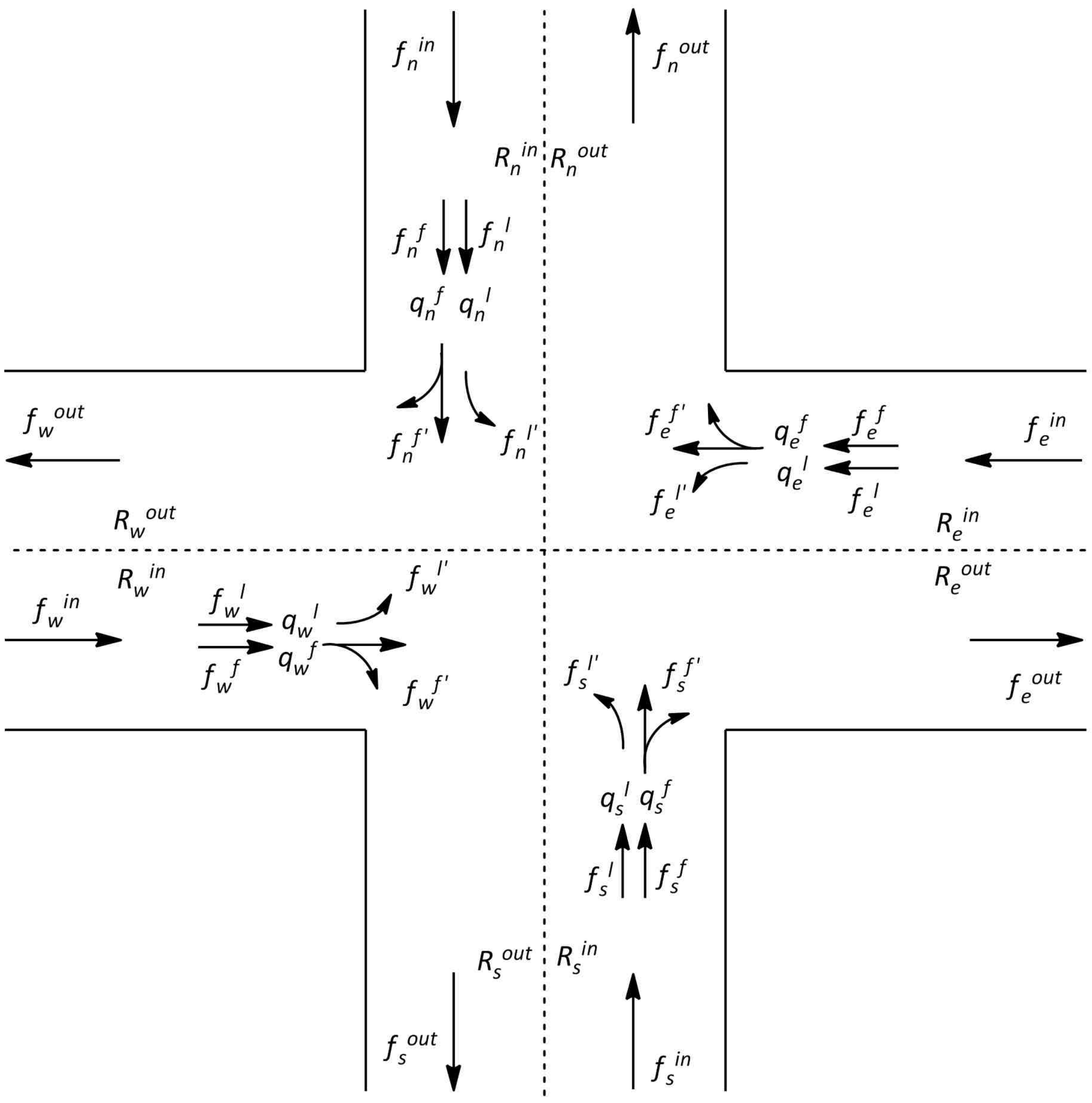


Fig. 3.1. The structure of a single intersection.

while the set $\{f_w^{out}, f_e^{out}, f_n^{out}, f_s^{out}\}$ captures the vehicle flow that is moving out of the intersection in the four directions.

Assuming that, before reaching the intersection within a certain distance, any vehicle has three options: being in the left lane, which means turning left at the intersection is obligatory; being in the middle lane, which means the vehicle will go straight at the intersection; or being on the right lane, which gives the vehicle the option to keep going forward or turning right when it reaches the intersection.

Consequently, vehicle flow $f_i^{in}$ splits into two flows: $f_i^l$ which represents the flow of vehicles that turning left at the intersection; while $f_i^f$ represents the flow of vehicles that keep going forward or turning right at the intersection. Thus, the two sets $\{f_w^l, f_e^l, f_n^l, f_s^l\}$ and $\{f_w^f, f_e^f, f_n^f, f_s^f\}$ represent these two flow divisions for the four incoming directions.

Furthermore, the notation $f_i^{l'}$ represents the flow of vehicles entering the intersection from direction $i$ and turning left direction, while the notation $f_i^{f'}$ is defined as the flow of vehicles entering the intersection from direction i and traveling forward or turning right direction. Consequently, two sets gather the flow of vehicles entering the intersection from the four incoming directions: $\{f_w^{l'}, f_e^{l'}, f_n^{l'}, f_s^{l'}\}$ and $\{f_w^{f'}, f_e^{f'}, f_n^{f'}, f_s^{f'}\}$. Finally, the notation $q_w^l$ represents the queue of vehicles that are waiting to cross the intersection toward the left coming from direction $i$. Similarly, the notation $q_w^f$ represents the queue of vehicles that are coming from direction i and waiting to cross the intersection and going forward or turning right direction. Therefore, two queue sets appeared $\{q_w^l, q_e^l, q_n^l, q_s^l\}$ and $\{q_w^f, q_e^f, q_n^f, q_s^f\}$.

When it comes to designing a model for an intersection in an urban traffic network, the percentages of vehicles that turn left, keep going forward, and turn right at the intersection are considered to be an essential piece of information that shall be known (or assumed). Therefore, it is necessary to define the parameter $\alpha_i^j$ as the split factor where i takes one of the four directions $\{n, s, e, w\}$ and $j$ stands for $f$ or $l$. Hence, $\alpha_e^l$ represents the factor of vehicles coming from the east and turn left (south) as they approach the intersection, while $\alpha_n^f$ is the percentage of vehicles coming from the north and avoiding the left lane as they approach the intersection (which includes both vehicles that keep going forward (south) or turning right (west) at the intersection). These two relations have to be satisfied:

$$0 \leq \alpha_i^j \leq 1 \tag{3.1}$$

$$\alpha_i^f + \alpha_i^l = 1 \tag{3.2}$$

As vehicles reach the intersection, another parameter becomes important which determines the percentages of vehicles that turn right at the intersection and those which keep going forward. This parameter is defined as the cross-direction factor and symbolized as $\beta_i^j$ where $i$ takes one of the four directions $\{n, s, e, w\}$ and $j$ stands for $ff$ or $fr$. Hence, $\beta_i^{ff}$ represents the percentage of vehicles moving forward through the intersection out of the total number of vehicles coming from the direction $i$ and split forward (not left) while they approach the intersection. Similarly, $\beta_i^{fr}$ is the percentage of vehicles turning right at the intersection out of the total number of vehicles that coming from the direction $i$ and split forward (not left) while they approach the intersection. As mentioned for the split factor, the cross-direction factors also satisfy the relations:

$$0 \leq \beta_i^j \leq 1 \tag{3.3}$$

$$\beta_i^{ff} + \beta_i^{fr} = 1 \tag{3.4}$$

Mathematically, the split factor and the cross-direction factors are described by the equations as follows:

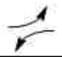
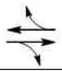
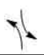
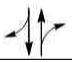
$$\alpha_i^j = \frac{f_i^j}{f_i^{in}} \tag{3.5}$$

$$\beta_i^j = \frac{f_i^j}{f_i^f} \tag{3.6}$$

Where $f_i^{ff}$ and $f_i^{fr}$ are the flows of vehicles that keep going forward and turn right at the intersection, respectively. Even though vehicle flows change with time, the split and cross-direction factors are assumed to be fixed for a given time interval. It is logical to assume that the split and cross-direction factors are constant for a known time interval. For a particular intersection, both of the split factors and the cross-direction factors can be measured. Depending on these measurements each day in the week is divided to time intervals. For example, in a given intersection it was noted that during the morning rush hour (7 AM to 8 AM) 1/3 of the vehicles coming from the east turn left while the 2/3 keep going forward as they approach the intersection. At the intersection, 1/2 of this 2/3 turn right while the other 1/2 keep going forward. While for all other times of that day, it is assumed that 55% of the vehicles coming from the east turn left while 45% keep going forward as they approach the intersection. At the intersection, 20% of vehicles either in the right late or in the middle lane will turn right and the 80% will keep going forward. As a result it is clear that for that day two time intervals exist; at the time interval (7 AM to 8 AM), the factors are: $\alpha_e^l = \frac{1}{3}$, $\alpha_e^f = \frac{2}{3}$, $\beta_e^{fr} = \frac{1}{2}$, and $\beta_e^{ff} = \frac{1}{2}$ while at other time interval the factors are: $\alpha_e^l = 0.55$, $\alpha_e^f = 0.45$, $\beta_e^{fr} = 0.2$, and $\beta_e^{ff} = 0.8$.

As shown in In Fig. 3.1, the intersection is represented physically by means of roads, vehicle flow, and queues. On the other hand, it is represented functionally by the means of the traffic light signals. The role of a traffic light signal is to safely control vehicle, and also optimize the traffic by shortening the queue lengths and equalizing them or maximizing the number of vehicles crossing the intersection during a specific time interval. Generally speaking, in intelligent transportation systems this process is implemented through controlling different parameters such as phase transition, lights durations, and the offset between cycles (the total time duration required for all phases to take place once) for a group of intersections. For traffic signals, a phase is defined as an event of giving particular permission to vehicles coming from specific directions to cross the physical area of the intersection toward specific directions. These permissions are given by the green and yellow lights, while disabling the movement of other vehicles through the red light. The number of phases in a specific signalized intersection depends primarily on the number of roads connected through this intersection and the expected vehicle flow in each direction. A specific intersection might be controlled through more than one plan, each of them is applied in a specific time in the day or specific day (days) of the week. Each plan has a specific number of phases, deterministic time durations for each phase, and

Table 3.1
Four phases of a signalized intersection.

| Phase | Enabled Directions | Allowed Flows | Phase Duration |
|---|---|---|---|
| 1 |  | $f^l_w, f^l_e$ | $\phi_1 = \phi^g_1 + \phi^y_1$ |
| 2 |  | $f^f_w, f^f_e$ | $\phi_2 = \phi^g_2 + \phi^y_2$ |
| 3 |  | $f^l_n, f^l_s$ | $\phi_3 = \phi^g_3 + \phi^y_3$ |
| 4 |  | $f^f_n, f^f_s$ | $\phi_4 = \phi^g_4 + \phi^y_4$ |

particular offsets between cycles of the group of controlled intersections. For a four-bidirectional road intersection, the number of phases is not more than eight as shown by List and Cetin [16]. In this modeled intersection, the following assumptions were made for the traffic signal:

- There are four possible phases such that: in phase 1 vehicle flow from east to south and from west to north are permitted, in phase 2 vehicle flow from east to west (and north) and from west to east (and south) are permitted, in phase 3 vehicle flow from north to east and from south to west are permitted, and in phase 4 vehicle flow from north to south (and west) and from south to north (and east) are permitted. It is clear that the directions between brackets are the right direction of the straight allowed flow, so they are too allowed at the same phase.

- The duration (length) of phase $i$ known as $\phi_i$ , which is the summation of the green light duration $\phi^g_i$ and the yellow light duration $\phi^y_i$ of that phase.

- Except for the enabled ones, all vehicle flows, are disabled and so cannot cross the intersection through a red light.

## 3.3 Timed Petri Net Model of a Single Intersection

Before introducing the Petri net model, some assumptions need to be taken in order to simplify the model and make it possible to be simulated. First, the flow rate of vehicles at a particular location during a specific time interval in a particular phase is assumed to be constant. This approximation is necessary for the model to be simulated. In reality, the flow rate of vehicles crossing an intersection is variable since it is a function of the average speed of vehicles, which can also vary. At the beginning of the phase, the vehicles that cross the intersection first are those that have zero initial speed, those that reduce their speed at the intersection, and finally those crossing the intersection without any changes in their speed. However, for the model to be simulated it is necessary to make this approximation, which does not mean that the flow rate will be fixed during a particular phase but it will be given in a step function form that usually has one or two values for its range. Furthermore, this approximation is accurate and reflects the reality of solving problems that consider the total outcome of each phase. It should be noted that this approximation does well if the phase durations are long. Second, it is assumed that the flow rate is the same during the green and yellow light durations; which does make sense in reality. Finally, it is assumed that a time interval exists between any two successive phases (between switching to the red light at the leading phase and the switching to the green light at the lagging phase). This time interval will be given a value that is greater than or equal to the time consumed by a vehicle to cross the intersection. This assumption reflects the safety condition in reality and is also considered an essential assumption in terms of Petri net structure to avoid some expected conflicts.

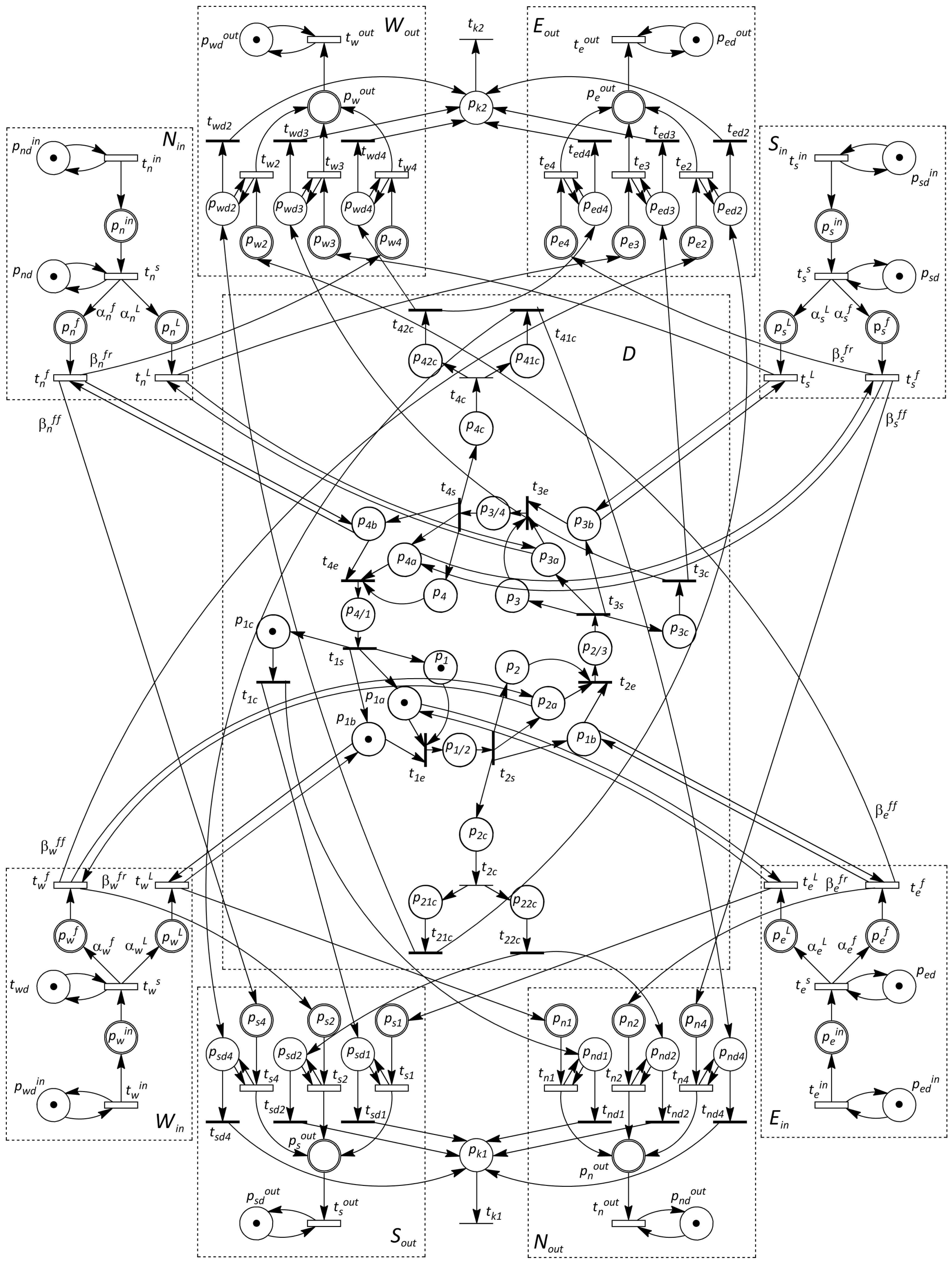


Fig. 3.2. Timed hybrid Petri net model of a single intersection.

Fig. 3.2 shows a timed hybrid Petri net model for the four bidirectional single intersection given in Fig. 3.1. The model is divided into nine sub Petri nets (modules) and each one is bounded by a dashed rectangular box. Four of these modules represent the movements of vehicles while entering the intersection ($N_{in}, S_{in}, E_{in}, W_{in}$) and another four represent vehicles while crossing and leaving the intersection ($N_{out}, S_{out}, E_{out}$, $W_{out}$). The remaining dashed box ($D$) in the middle of the graph represents the switching between the four different phases. The sub Petri net ($D$) is the only pure discrete net while the other eight consists of hybrid nodes. Starting with the pure discrete part, the process of phase changing is modeled through 16 places and 8 transitions. The eight transitions are timed transitions, so each of them has a specific non-negative integer value assigned ($d_{i/j}$) to it and represents the time units that it take this transition to be fired after it becomes enabled. Each transition represents either the beginning or the end of a specific phase. When transition $t_{ie}$ fires, phase $i$ ends; the firing of the transition $t_{js}$ means that phase $j$ starts. Since four phases exist, only eight transitions are required to represent the beginning and ending of these phases. Out of the 16 places mentioned, four of them are used only as an indicator to show which phase is currently active. A token in place $p_i$ means that phase i is active while losing this token indicates the end of the phase. The condition $p_1 + p_2 + p_3 + p_4 \leq 1$ has to be satisfied at all times (the reason that being less than is acceptable instead of just equivalence is because it is assumed that there is a short time duration between each two phases). Each place $p_i$ is attached with two other places $p_{ia}$ and $p_{ib}$ which are used to translate the meaning of enabling/disabling vehicles movement to the modeling language as will be shown in the model dynamics description. The transition $t_{i/j}$ represents the transition from phase $i$ to phase $j$, so four transitions are required to represent the whole cycle. In addition to these 16 places and 8 transitions, which are shown in a cycle form in the center of Fig. 3.2, another set of places and transitions are used to simulate the time consumed by vehicles when they are crossing the intersection. As soon as phase i starts, a token is added to the place $p_{ic}$, which in turn immediately charges the places $p_{i1c}$ and $p_{i2c}$ through the immediate transitions $t_{ic}$ (the last step takes place in phases 2 and 4 only). These places and transitions are responsible for delaying the flow of vehicles leaving the intersection from the flow of vehicles entering the intersection if required (if the time that takes a vehicle to cross the intersection is large enough to be simulated). Finally, the arcs weighs between all the discrete nodes mentioned above are one.

For an input module$i_{in}$ where i takes the four directions symbols $\{w, e, n, s\}$, the transition $t_i^{in}$ represents the flow rate of the vehicles entering the intersection from the $i^{th}$ direction. A real positive value is assigned to this transition; this value, which is the flow rate of the transition (tokens per time unit) in the Petri net language, represents the flow rate of vehicles approaching the intersection in the unit of vehicle per time unit. Since the discrete place $p_{id}^{in}$ that connected to the transition $T_i^{in}$ is marked with one token each, the firing speed of transition $t_i^{in}$ (which is the maximum speed in this case) is the same as the flow rate. The transition $t_i^{in}$ is connected to the place $P_i^{in}$ with arc weights of 1. At any time the number of tokens in place $p_i^{in}$ represents the number of vehicles approaching the intersection from the direction $i$. Downstream $p_i^{in}$, a split transition $t_i^s$ appears. This transition splits the incoming flow (symbolized $f_i^{in}$ in Fig. 3.1) into the flows $f_i^l$ and $f_i^f$, which respectively represents the flow of vehicles taking a left versus those that keep going forward or take a right at the intersection. The weight of the arc connected $p_i^{in}$ to the transition $t_i^s$ is 1 while the weights of arcs connected the transition $t_i^s$ to the places $p_i^f$ and $p_i^l$ are the split factors $\alpha_i^f$ and $\alpha_i^l$, respectively. As mentioned for transition $t_i^{in}$, real positive value is assigned to the transition $t_i^s$. These values represent the saturation (maximum allowed) flow of vehicles approaching the intersection from direction $i$. The markings of the two places $p_i^f$ and $p_i^l$ respectively represent the number of vehicles waiting at the queues to take left at the intersection, and the number of vehicles that will keep going forward or take a right. Places $p_i^f$ and $p_i^l$ are connected to transitions $t_i^f$ and $t_i^l$ respectively through arcs of weight 1. The firing of the transition $t_i^l$ represents vehicles that are coming from direction $i$ and taking a left, which are crossing the intersection with a flow rate that is equivalent to the firing speed of transition $t_i^l$. The assigned value to the transition $t_i^l$ represents the saturation (maximum allowed) flow of vehicles crossing the intersection while the actual flow rate (firing rate) can be this value or less depending on the firing rate of transitions upstream. Similarly, firing of transition $t_i^f$ represents that vehicles coming from direction i and keep going forward or take a right are crossing the intersection with a flow rate less than or equal to the saturation flow, which is the flow rate values that are assigned to the transition $t_i^f$ in the Petri net language. Since all vehicles that are taking a left at the intersection are going to one destination, only one output module is required to represent this destination. Transition $t_i^l$ is connected to the output module $j_{out}$ through an arc of weight 1, where $j$ takes the direction symbol that represents the destination of vehicles coming from the incoming direction $i$ and taking left at the intersection. On the other hand transition $t_i^f$ is connected to the two output modules $k_{out}$, $l_{out}$, where $k$ takes the direction symbol that represents the destination of vehicles coming from the incoming direction $i$ and keep going forward with the intersection while l takes the direction symbol that represents the destination of vehicles coming from the incoming direction i and taking right with the intersection. That the arc connecting the transition $t_i^f$ to the module $k_{out}$ equals $\beta_i^{ff}$ while the arc connecting transition $t_i^f$ to the module $l_out$ equals $\beta_i^{fr}$.

The output module $j_out$ represents the vehicles crossing and leaving the intersection toward direction $j$. The module stream starts with three continuous places $p_{jx}$ where $x$ takes three different values from the set $\{1, 2, 3, 4\}$ for each phase. The place $p_{jx}$ represents the vehicles while crossing the intersection during phase $x$ towards the direction $j$. The three nodes $p_{jdx}$, $t_{jdx}$, and $t_{jx}$ work together as a time delay circuit in order to represent the average time consumed by vehicles to cross the intersection during phase $x$ towards the direction $j$ as will be shown on the dynamics description of the model. The continuous place $p_j^{out}$ represents the vehicle leaving the intersection toward the direction $j$ while the transition $t_j^{out}$ exists downstream the mentioned output continuous place and reflects the flow rate of vehicles leaving the intersection.

Each continuous transition is connected to at least one discrete place. Some of these connections are necessary for the design purposes, primarily the enabling and disabling the transition, while others are added only to make sure that the model is theoretically suitable to be analyzed with any simulator. As mentioned in David and Alla's book [10], the continuous transition which is not connected to a discrete place will be fired at an infinite speed. Therefore, to avoid this, a discrete place marked with a token is added to each transition that does not connect to a discrete place previously.

## 3.4 The Dynamics of Timed Hybrid Petri Net Model

If it is assumed that time 0 is the beginning of the first phase, the markings of the discrete places shall be as shown in Fig. 3.2. As soon as the time interval between the fourth and the first phases ends, one token is removed from $p_{(4/1)}$ and one token is added to each place of $p_1$, $p_{1a}$, $p_{1b}$, and $p_{1c}$. Having a token at $p_{1a}$ and $p_{1b}$ means that the transitions $t_w^l$ and $t_e^l$ are enabled and ready to be fired, if the markings of $p_w^l$ and $p_e^l$ are greater than zero. In traffic flow language, this means permission is given to those vehicles that are waiting to go left from the west and east sides of the intersection to cross the intersection towards north and south direction, respectively. At the beginning of this phase, the flow rates of the transitions $t_w^l$ and $t_e^l$ depend on the markings of $p_w^l$ and $p_e^l$ at $t = 0$ respectively, which reflect the queues of vehicles waiting to cross the intersection from west to north and east to south. If the marking of $p_w^l$ is greater than or equal to the maximum firing speed of $t_w^l$, then $t_w^l$ will fire in its maximum firing speed (which is the maximum flow rate). Otherwise, $t_w^l$ will fire in a rate equal to the marking of $p_w^l$ at time $t = 0$ and is given in the unit of token per time unit. The flow rates of transitions $t_w^{in}$, $t_e^{in}$, $t_n^{in}$, and $t_s^{in}$ represent the flow of vehicles approaching the intersection from the four directions. Since these transitions do not have any continuous place as an input, their maximum speeds are their flow rate or firing rate. The continuous firing of the transition $t_i^{in}$ will transfer tokens to the places $p_i^l$ and $p_i^f$ through the place $p_i^{in}$ and the transition $t_i^s$, which represent the split point where vehicles take the left lane or keep using the other two lanes. Therefore, the places $p_i^l$ and $p_i^f$ are used as cumulative places (vehicles queues). Then, as mentioned above, permission is given to both transitions $t_w^l$ and $t_e^l$ to fire during the first phase. The tokens (vehicles) flows reach the places $p_{n1}$ and $p_{s1}$, respectively. All the events mentioned up to now happened at time t = 0, excluding the firing of transitions $t_{n1}$ and $t_{s1}$. These two transitions will not be enabled until a token is transferred from the place $p_{1c}$ to the discrete places $p_{nd1}$ and $p_{sd1}$ through firing the timed transition $t_{1c}$, which will be fired after a specific number of time units $\Delta t = d_{1c}$ since it becomes enabled at time t = 0. This value $d_{1c}$ is a design value that is assigned to the transition $t_{1c}$ and represents the average time that is consumed by a vehicle coming from the west or east direction to cross the intersection towards north or south direction in a left turn, respectively. Thus, transitions $t_{n1}$ and $t_{s1}$ will be fired at time $t = d_{1c}$. Of course, these two transitions will not be fired if there are no vehicles crossing the intersection during the first phase because their continuous input places are marked zero in this case. Firing of transitions $t_{n1}$ and $t_{s1}$ results in tokens at the output places $p_n^{out}$ and $p_s^{out}$, which in turn fire the output places $t_n^{out}$ and $t_s^{out}$ with a flow rate depending on the previous nodes and represent the flow rate of vehicles leaving the intersection toward north and south directions, respectively.

If the first phase duration is $\phi_1$ time units, the time associated with the timed discrete transition $t_{1e}(d_{1e})$ shall be $\phi_1$ time units too. Therefore, as soon as the first phase ends the transition $t_{1e}$ fires and a token is removed from $p_1$, $p_{1a}$, and $p_{1b}$ and one token is added to the place $P_{(1/2)}$; this token will spend some time here before firing the transition $t_{2s}$. This time is equal to the interval time between the two phases (first and second), and is assigned to the transition $t_{2s}$. Therefore, after $d_{2s}$ time units, a token is removed from the place $p_{(1/2)}$ and a token is added to each place of $p_2$, $p_{2a}$, $p_{2b}$, and $p_{2c}$.

Removing tokens from $p_{1a}$, and $p_{1b}$ means that transitions $t_w^l$ and $t_e^l$ are no longer enabled because, as mentioned in Chapter 2, the discrete transition has the firing priority over the continuous one if a conflict exist between them in a HPN. So vehicles coming from west (east) are not permitted anymore to cross the intersection towards north (south). Furthermore, it is essential to make sure that the two tokens at the discrete places $p_{nd1}$ and $p_{sd1}$ at the end of this phase. Otherwise, when the first phase takes place again, two tokens will show up in each place of $p_{nd1}$ and $p_{sd1}$, which will affect the flow rate of the output transitions and give unreal results and the case will be worse during the flowing cycles of this phase (since more tokens will be added). Therefore, the timed discrete transitions $t_{nd1}$ and $t_{sd1}$ are used and assigned with values $d_{nd1}$ and $d_{sd1}$ which both of them equal $d_{1e}$ time units, the same as the first phase duration. Nodes $p_{k1}$, $t_{k1}$, $p_{k2}$ and $t_{k2}$ are used as sinks for draining these tokens. On the other hand, adding a token to each place of $p_2$, $p_{2a}$, $p_{2b}$, and $p_{2c}$ means the starting of the second phase through given permissions to vehicles coming from west (east) to keep going forward toward east (west) or taking right toward south (north) and the process will be similar to the one described for the first phase. The only difference is that the place $p_{2c}$ is divided through the transition $t_{2c}$ into two streams because during this phase, there are two different movements are taking place; vehicles crossing the intersection forward and right, which in reality might take a different amount of time to complete. The timed discrete transition $t_{21c}$ is used to reflect the time taken by vehicles traveling forward (from west to east and east to west) to cross the intersection during the second phase while transition $t_{22c}$ models the time consumed by vehicles taking a right at the intersection during the second phase (from west to south and east to north). This movement takes less time than the forward movement so it is excepted that $d_{22c} < d_{21c}$. Furthermore, $d_{21c}$ can be assigned the value of zero since it is usually a very small amount, which means transition $t_{21c}$ can be an immediate transition. Furthermore, for intersections where the time consumed by vehicles to cross the intersection is very small, all of these nodes that represent this consumed time can be removed from the model. However, the model used in this research is the general one.

## 3.5 Structure of Two Successive Intersections

After introducing the single intersection model, it is easy to establish a Petri net model for a full traffic network, which is desired in this study. Actually, this simple model was chosen to be the simplest network possible in order to avoid very complex Petri net drawings that come from modeling large networks. However, through the mathematical representation of Petri nets, it will be easy to deal with larger traffic networks (i.e. with more intersections), which can be modeled using the concepts described in this chapter.

Now, consider a network consisting of two intersections: Intersection-1 on the left (west) and Intersection-2 on the right (east); Fig. 3.3 shows this traffic network. The same symbols that are introduced in the single intersection graph are used to represent roads, queues, inputs, and outputs flows with adding (1) or (2) at the end of the bottom text for each variable to distinguish whether it belongs to Intersection-1 or Intersection-2. Similarly, the phase durations, split and cross-direction parameters are written in the same way. As it is shown in the figure, $R_{(1/2)}$, and $R_{(2/1)}$ represent the roads from Intersection-1 to Intersection-2 and from Intersection-2 to Intersection-1, respectively. Furthermore, road $R_{(1/2)}$ parameters are represented through symbols that are attached with the notation (1/2) at the end of their bottom text.

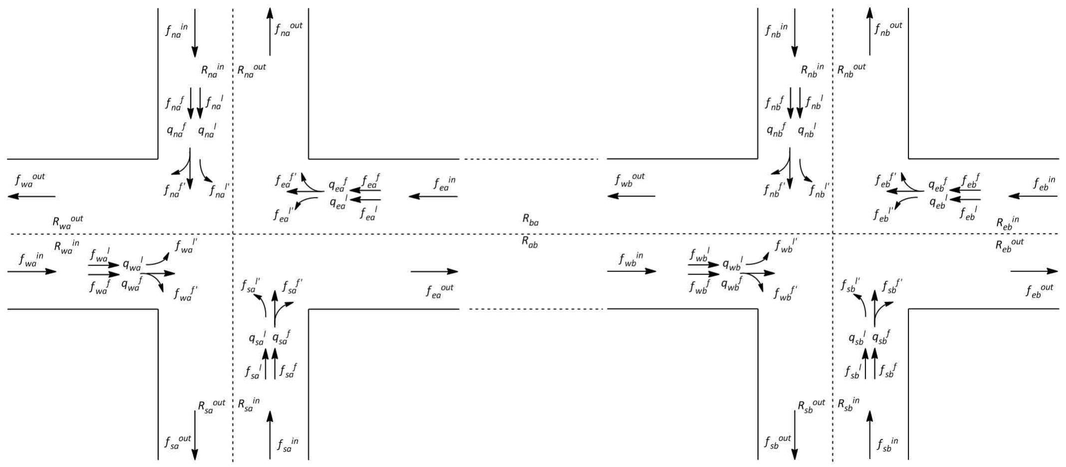


Fig. 3.3. Structure of two successive intersections.

Generally, when the vehicles flow is the desired objective to be studied, each road is divided to sections. These sections have different lengths and each section has its own parameters, such as vehicle density, input and output flows, and vehicles speed at that section. The relation between these parameters in divided into road models that can be described in equations as written by Papageorgiou and Payne [28, 29]. In this research, it is assumed that the two unidirectional roads connecting the two intersections are short enough that the road parameters are not changing so the roads need not be divided into sections. If the average time that takes a vehicle to travel from Intersection-1 to Intersection-2 is $t_{(1/2)}$ time units, then the coming flow from the east direction to Intersection-1 $f^{in}_{e(1)}$ is nothing but the outgoing flow directed to the west from Intersection-2 $f^{out}_{w(2)}$ delayed with $T_{(1/2)}$ time units. Similarly, the flow $f^{out}_{e(1)}$ is nothing but the flow $f^{in}_{w(2)}$ delayed with $T_{(2/1)}$ time units, which is the average time for vehicles to travel from Intersection-1 to Intersection-2.

## 3.6 Timed Petri Net Model for Two Successive Intersections

The Petri net model introduced previously for a single intersection will be used to represent each intersection of the given network in Fig. 3.4. Roads $R_{(1/2)}$, and $R_{(2/1)}$ are also modeled through hybrid Petri nets that connect the two intersections. Therefore, it is better to introduce the model of the road $R_{(1/2)}$, and $R_{(2/1)}$ first, then gather the four Petri nets (Intersection-1, Intersection-2, road $R_{(1/2)}$, and road $R_{(2/1)}$) and connect them to come up with the whole network model. Since the model introduced in this research can be used for both kind of problems, those which focus on the overall outcome during a specific time such as optimization problems and those which analyze and simulate the behavior of networks in detail and the macroscopic timing is considered as an essential parameter to be modeled. As it was done for the single intersection Petri net model when the time required for a vehicle to cross the intersection was modeled by using timed transitions and the priority firing concept, the time consumed by vehicles to cross a road is taken in consideration. However, since this time is respectively large compared to that which required for a vehicle to cross the intersection, using the same approach exactly will not be suitable. In other words, delaying the firing of a specific transition that represents the incoming flow to Intersection-2 from Intersection-1 a certain time, which is required for vehicles to cross the road through a single timed discrete transition, might be a suitable model for the overall outcome during a specific phase but it will not reflect suitable simulation results. Therefore, transition firing delaying will be used but after dividing the connection between Intersection-1 and Intersection-2 to $n$ sections. After that, the delaying approach will be applied to each section so vehicles traveling between the two intersections can be simulated. Fig. 3.4 represents a Petri net model for the road $R_{(1/2)}$.

As it is noted from Fig. 3.4, the road model consists mainly of three queues of Petri nets; each one represents vehicles travelling across the road in a specific phase of Intersection-1s traffic signal (no vehicles movement in $R_{(1/2)}$ during the first phase). The timed discrete transition $t_{21C(1)}$ is nothing but the transition in the single intersection model of Intersection-1 that is used to delay the flow of vehicles leaving the intersection forward (from west to east and from east to west) from the flow of vehicles entering the intersection during the second phase of Intersection-1 traffic light. As mentioned before, firing of this transition is considered permission for vehicles traveling from east to west and west to east to leave Intersection-1. However, from the network model perspective, firing of this transition will also initiate the part of Petri net road model that represents vehicles movement in road $R_{(1/2)}$ during the second phase to start a firing sequence simulating this movement. As soon as this transition fires a token is added to the discrete place $p^{in}_{d2(1/2)}$, which represents the input node to this queue.

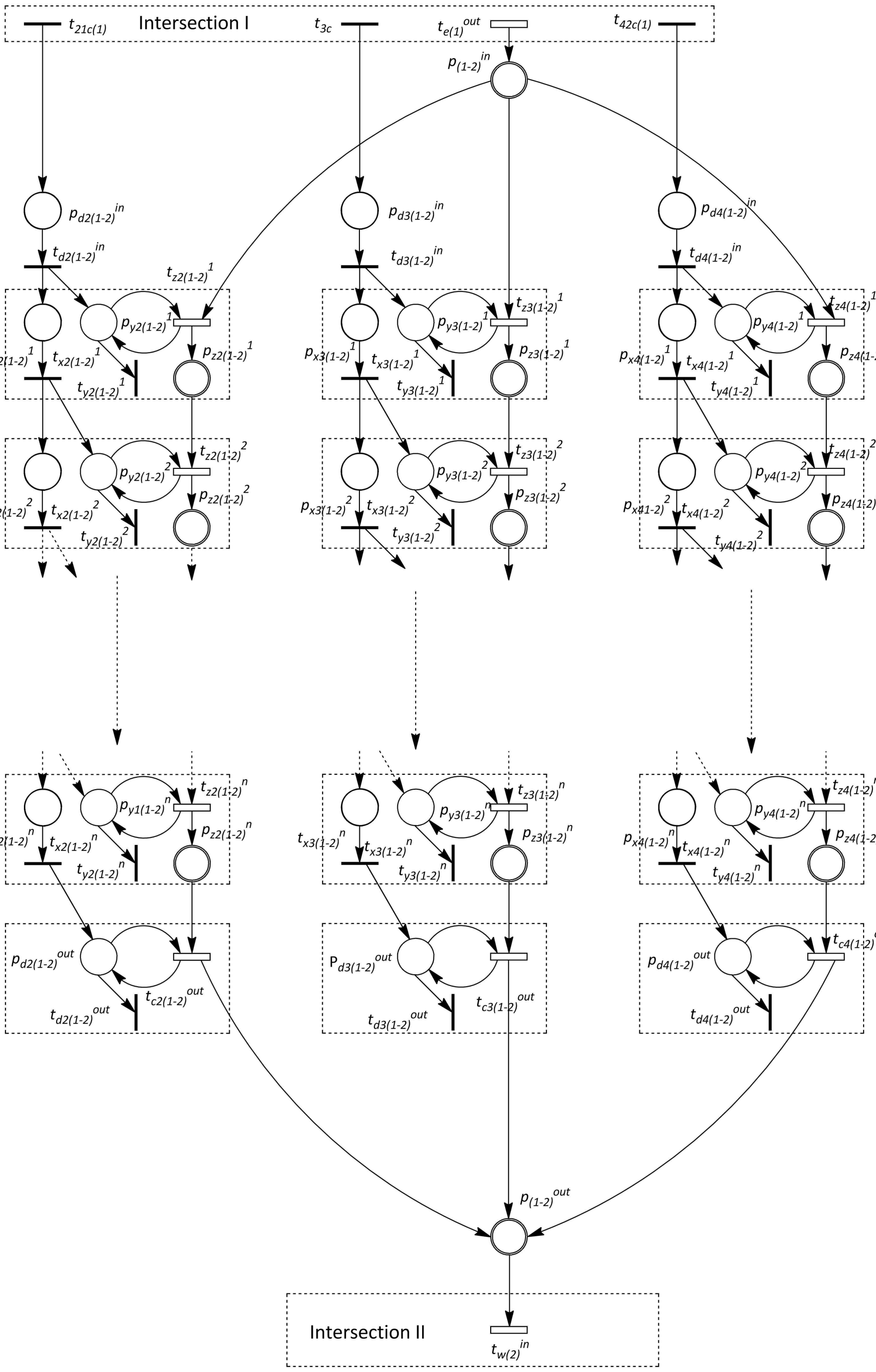


Fig. 3.4. Timed hybrid Petri net model of a road connecting two intersections.

In the subscript of each place, the number 2 indicates the second phase while the (1/2) indicate the road directed from Intersection-1 to Intersection-2. Having a token at $p^{in}_{d2(1/2)}$ implies that the transition $t^{in}_{d2(1/2)}$ will fire after a specific time $d^{in}_{d2(1/2)}$ , which is a design parameter that will be chosen depending on the number of divisions in this Petri net queue and the average traveling time from Intersection-1 to Intersection-2 $t_{(1/2)}$. As soon as this transition fires, a token is transferred from $p^{i}_{d2(1/2)}n$ to $p^{1}_{x2(1/2)}$ and $p^{1}_{y2(1/2)}$ which will give permission to the continuous transition $t^{1}_{z2(1/2)}$ to start firing and transferring tokens from place $p^{in}_{(1/2)}$ to $p^{1}_{z2(1/2)}$. Since $p^{i}_{(1/2)}n$ is downstream of transition $t^{out}_{e(1)}$ (which represents the flow of vehicles leaving Intersection-1 to the east side which is Intersection-2), it is obvious that this continuous nodes sequence will model vehicles movement in road $R_{(1/2)}$. Coming back to the discrete node $p^{1}_{y2(1/2)}$, it is easy to note that after a specific time units $d^{1}_{y2(1/2)}$ the token will be transferred to the sink place $p_{k(1/2)}$ through the firing of timed transition $t^{1}_{y2(1/2)}$. Since this process will take the firing permission away from the continuous transition $t^{1}_{z2(1/2)}$, $d^{1}_{y2(1/2)}$ shall be equal to $d_{2e(1)}$ which is the second phase duration in Intersection-1. For the token in place $p^{1}_{x2(1/2)}$ (the assigned time for transition $t^{1}_{x2(1/2)}$ to be fired) will remain only $d^{1}_{x2(1/2)}$ time units where $d^{1}_{x2(1/2)} = d^{in}_{d2(1/2)}$. Then the token will transfer to the following division places $p^{2}_{x2(1/2)}$ and $p^{2}_{y2(1/2)}$ and the same firing process will take place again. Finally the transition $t^{out}_{c2(1/2)}$ transfer tokens from the last continuous place in the divisions sequence $p^{n}_{z2(1/2)}$ to the place $p^{out}_{(1/2)}$, which will transfer those tokens (vehicles) to Intersection-2 input $t^{in}_{w(2)}$.

## 3.7 Simulation Results

With the objective of measuring the effectiveness of our modeling approach, some simulations were carried out for single signal intersection and connected intersections models. A demo version of SimHPN is used. SimHPN is a Matlab embedded software issued in 2012 that simulated hybrid Petri nets [30].

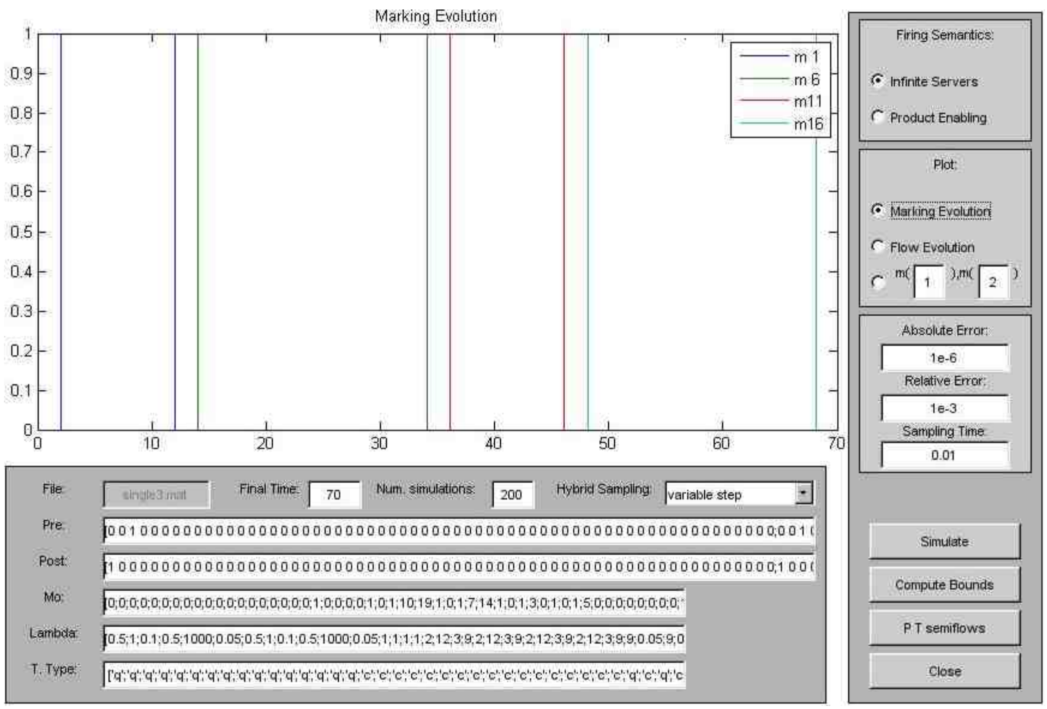
Starting with the single intersection model, Table 3.2 lists the parameters for the simulation. Some of these values are arc weights of input and output incident matrices and phase durations are the values that assigned to some timed discrete transitions.

Table 3.2
Initial values used in simulation.

| | |
|---|---|
| Each Input Flow | 2 vehicles per second |
| Phase 1 Duration | 10 seconds |
| Phase 2 Duration | 20 seconds |
| Phase 3 Duration | 10 seconds |
| Phase 4 Duration | 20 seconds |
| Time Duration Between Successive Phases | 2 seconds |
| Number of Vehicles at Queue $p_w^l$ | 10 |
| Number of Vehicles at Queue $p_w^f$ | 19 |
| Number of Vehicles at Queue $p_e^l$ | 7 |
| Number of Vehicles at Queue $p_e^f$ | 14 |
| Number of Vehicles at Queue $p_n^l$ | 5 |
| Number of Vehicles at Queue $p_n^f$ | 0 |
| Number of Vehicles at Queue $p_s^l$ | 3 |
| Number of Vehicles at Queue $p_s^f$ | 0 |
| $\alpha_i^l$ | 0.25 |
| $\alpha_i^f$ | 0.75 |
| $\beta_i^{ff}$ | 0.7 |
| $\beta_i^{fr}$ | 0.3 |

Finally, input vehicle flow is represented through the maximum firing speed of input continuous transitions.

Due to the large size of this simulation (the size of input and output incident matrices is $78 \times 62$), the simulation takes about 45 minutes. During our simulation, we save one figure per simulation process. Fig. 3.5 shows the durations of four phases and time intervals between each two successive phases. As it is mentioned in the Table 3.2, duration of the first and the third phase is 10 seconds while the second and the forth phases is 20 seconds. All of these values are design parameters that can be changed depending on the given problem; if not needed for a specific problem, the time intervals (2 seconds) can be taken off.



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: single3.mat

Final Time: 70

Num. simulations: 200

Hybrid Sampling: variable step

Pre: [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 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 (

Post: [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;1 0 0 (

Mo: [0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;0;0;1;0;1;10;19;1;0;1;7;14;1;0;1;3;0;1;0;1;5;0;0;0;0;0;0;0;0;

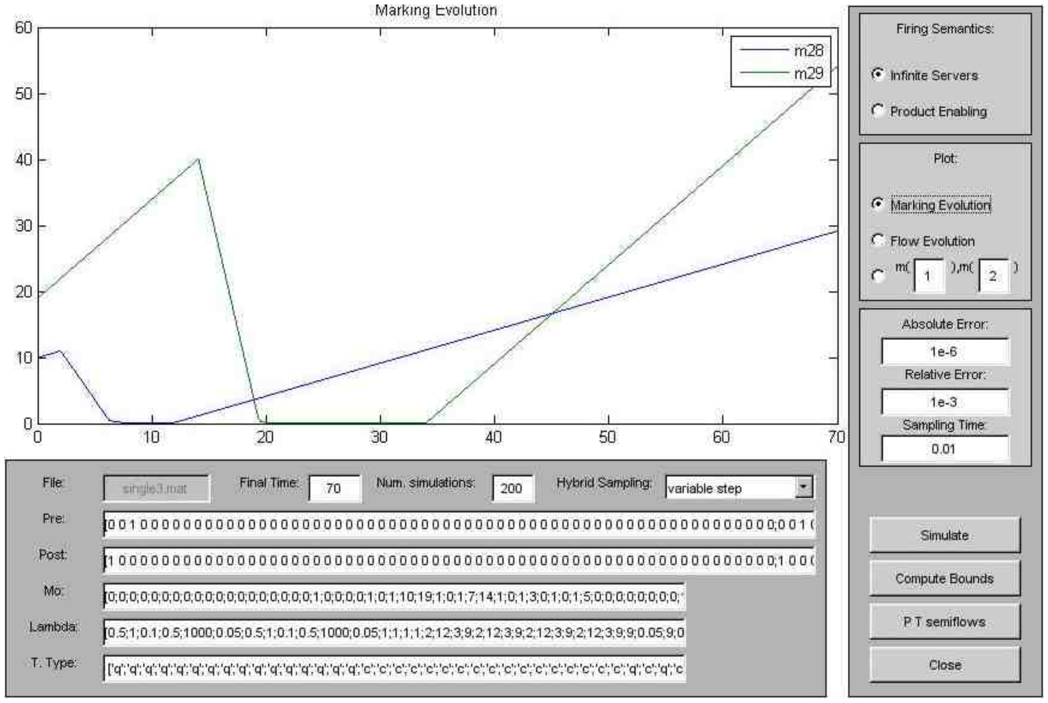
Lambda: [0.5;1;0.1;0.5;1000;0.05;0.5;1;0.1;0.5;1000;0.05;1;1;1;1;2;12;3;9;2;12;3;9;2;12;3;9;2;12;3;9;9;0.05;9;0

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

Fig. 3.5. Simulation result for durations of phases.

Fig. 3.6 shows vehicles queues at the western entrance of the intersection. The initial value of the vehicles queue that aims to take a left at the intersection (towards north) is 10 (plotted in blue). And since these vehicles are allowed to cross the intersection during the first phase (which starts at time of 2 seconds), the number of vehicles in the queue will keep decreasing until it becomes zero, which reflects the real case in traffic networks. As soon as the first phase ends (at time of 12 seconds), vehicles start to queue again and wait for the next time permission is given to cross the intersection.

For those vehicles queuing in the middle and right lanes to go east (forward) or south (right), they are initially 19 and they keep increasing up to the time of 12 seconds. Exactly at this point, the second phase starts and these vehicles cross the



File: single3.mat Final Time: 70 Num. simulations: 200 Hybrid Sampling: variable step

Pre: [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 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 (

Post: [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;1 0 0 (

Mo: [0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;0;0;1;0;1;10;19;1;0;1;7;14;1;0;1;3;0;1;0;1;5;0;0;0;0;0;0;0;0;'

Lambda: [0.5;1;0.1;0.5;1000;0.05;0.5;1;0.1;0.5;1000;0.05;1;1;1;1;2;12;3;9;2;12;3;9;2;12;3;9;2;12;3;9;9;0.05;9;0

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

Firing Semantics: [✓ Infinite Servers] Product Enabling

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

Absolute Error: 1e-6

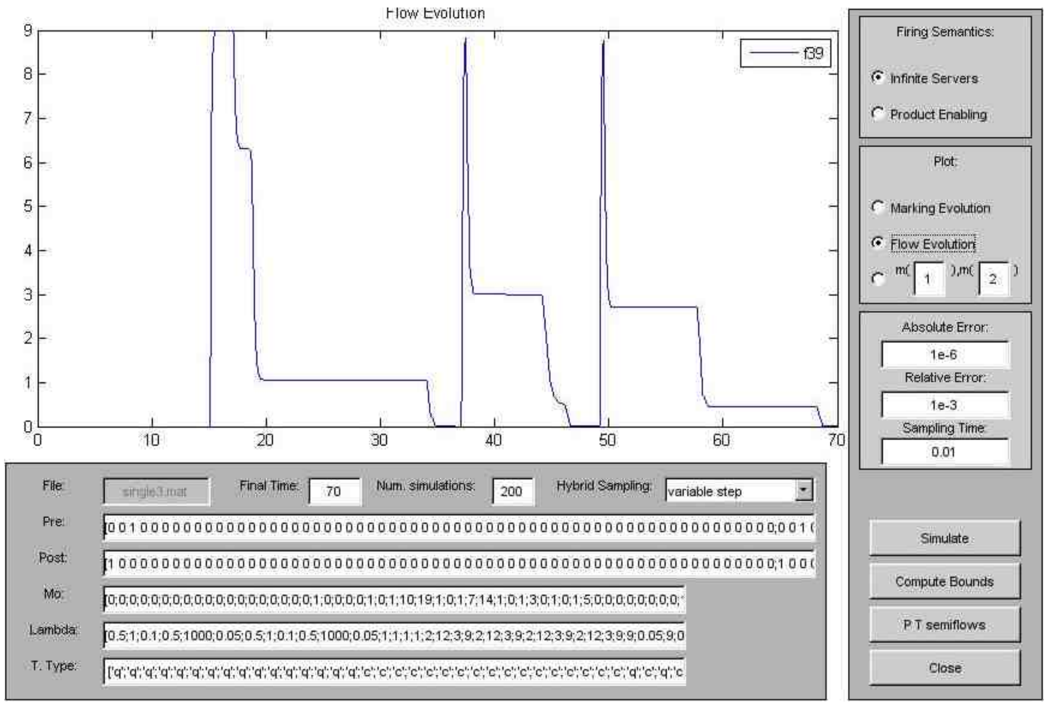
Relative Error: 1e-3

Sampling Time: 0.01

Fig. 3.6. Simulation results for vehicle queues at the western entrance of the intersection.

intersection. After 8 seconds, no vehicles will stop at the intersection up to the end of this phase and then they will accumulate again and so on.

As mentioned in the model description, vehicle flow is represented through continuous transitions. Fig. 3.7 shows the flow of vehicles leaving the intersection towards west. During the first phase, which ends at t = 12 seconds, no vehicle leaves the intersection towards west because, as shown in Table 3.1, during this phase the permitted movements are from west to north and east to south. After the end of the first phase, the interval time between two phases (2 seconds), and the simulated time for vehicles to cross the intersection from east to west, vehicles flow at the western exit of the intersection can be shown as a peak which corresponds to the queue of vehicles waiting on the east-side to cross the intersection. As these vehicles cross the intersection, vehicles flow decreases and take a constant value that is shown a little bit after the end of the second phase and is the result of simulating the required time to cross the intersection from east to west. In regards to the phases description in Table 3.1, it is expected to have output flow at the western exit during the two following phases as shown in Fig. 3.7.



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: single3.mat Final Time: 70 Num. simulations: 200 Hybrid Sampling: variable step

Pre: [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 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 (

Post: [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;1 0 0 (

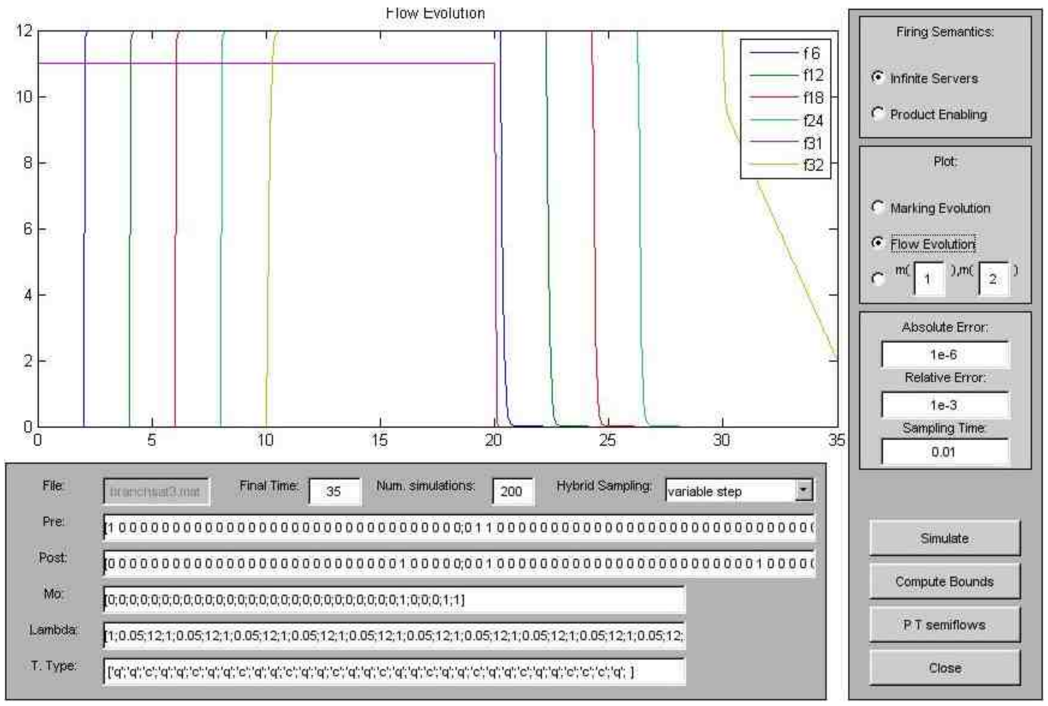
Mo: [0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;0;0;1;0;1;10;19;1;0;1;7;14;1;0;1;3;0;1;0;1;5;0;0;0;0;0;0;0;0;

Lambda: [0.5;1;0.1;0.5;1000;0.05;0.5;1;0.1;0.5;1000;0.05;1;1;1;1;2;12;3;9;2;12;3;9;2;12;3;9;2;12;3;9;9;0.05;9;0

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

Fig. 3.7. Simulation result for vehicle flow at the western exit.

Results of simulating the connections between the two intersections show some limitations. However, it gives good results if the input flow is not very small compared to the saturation flow of the road connecting the two intersections. In Fig. 3.8, vehicle flow (during the second phase of the first intersection) in the road connecting Intersection-1 (west side) to Intersection-2 (east side) is plotted. It is assumed that the out flow from Intersection-1 towards Intersection-2 is constant and the average travelling time between two intersections is assigned the value of 10 seconds (which is represented in terms of the Petri net with a value associated to some timed discrete transitions in the road model). The plotted flows are done for some successive locations (continuous transitions) in the road that are a fixed distance apart. The one which started at time zero belongs to the output flow of Intersection-1 while the last one that starts at time 10 represents the input flow to the Intersection-2 coming from Intersection-1.



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: branchsat3.mat Final Time: 35 Num. simulations: 200 Hybrid Sampling: variable step

```
Pre:     [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 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
Post:    [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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
Mo:      [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;1;1]
Lambda:  [1;0.05;12;1;0.05;12;1;0.05;12;1;0.05;12;1;0.05;12;1;0.05;12;1;0.05;12;1;0.05;12;1;0.05;12;1;0.05;12;
T. Type: ['q';'q';'c';'q';'q';'c';'q';'q';'c';'q';'q';'c';'q';'q';'c';'q';'q';'c';'q';'q';'c';'q';'q';'c';'q';'q';'c';'c';'c';'q'; ]
```

Fig. 3.8. Simulation result for vehicle flow in the connected road.

During the second phase of Intersection-1, flow was simulated in other branches (which represented the road model during phases 3 and 4) and the flow there is zero as expected. Due to the time-cost of simulating such large networks, simulations have been done by using equivalent nodes to represent the intersections with the assumption that the input flow is constant and is not small compared to the saturation flow.

# 4. DISCRETE PETRI NET MODELING OF TWO CONNECTED INTERSECTIONS

In the previous chapter, both discrete and continues nodes are used integrally to model vehicle flow which is affected and controlled by discrete events. Even though the gained information is valuable, the model was not easy to design and its dynamic mechanism was not simple to simulate and analyze, especially for large-scale traffic networks. In other traffic management problems, all gained information in the previous model may not be interesting. In this chapter, an event Petri net model is introduced. The objective of the proposed model is to represent the phase switching, enabled/disabled flows, queues, and other occurrences in term of event representation that does not consider time. In other words, the occurrence of the event that vehicles flow in a certain road in the network during a certain phase can be observed through this model without focusing on the time interval for this event. Also phases are represented as events, that each event of them is followed by other events such as enabling/disabling flows, vehicles crossing the intersection, vehicles leaving the intersection toward a specified destination, a queue exists, and so on.

In order to avoid the complexity, again a single intersection model is introduced in detail first. Then it will be extended to be applied for two successive intersections. However, this two intersections model by traditional Petri nets may lead to unbounded places which cannot accurately reflect the dynamics of the traffic. Attempting to control the unbounded places through Petri nets, control method results in undesired transition conflicts. Hence, the Modified Binary Petri net (MBPN) is used to overcome this limitation and resolve problems of conflicted transitions and unbounded places. This MBPN model is a powerful tool and can be useful for the modeling and analysis of many other similar applications.

## 4.1 Petri Net Model for a Single Intersection

The same structures for single and double intersections, which were introduced in the previous chapter through Fig. 3.1 and Fig. 3.3 respectively, are used as the desired modeling problem for this chapter. Furthermore, the same phase plan which is mentioned in Table 3.1 is applied.

The discrete Petri net model for the four-bidirectional road (as shown in Fig. 3.1) is depicted in Fig. 4.1. It is easy to note that the model is divided into 13 sub Petri nets (modules), where each one is bounded by a dashed rectangular box. Eight of these modules represent the movement of vehicles while entering and crossing the intersection considering their intention to cross it directed left $(N_l^{in}, S_l^{in}, E_l^{in}, W_l^{in})$ or directed forward/right $(N_f^{in}, S_f^{in}, E_f^{in}, W_f^{in})$. Four of the sub Petri nets $(N^{out}, S^{out}, E^{out}, W^{out})$ represent vehicles while leaving the intersection toward the four directions. Finally, the sub Petri net $(T)$ represents the phase change of the traffic light signal.

The sub Petri net $\mathbf{T}$ consists mainly of four places $(p_{p1}, p_{p2}, p_{p3}, p_{p4})$ and four transitions $(t_{p1}^s, t_{p2}^s, t_{p3}^s, t_{p4}^s)$ representing the four different phases mentioned in Table 3.1. The other nodes in the sub net $\mathbf{T}$ exist only to make sure that the model offers a safe operation (i.e. no directional conflicts between vehicles while crossing the intersection). Having a token at the place $p_{pk}$ indicates that the $k^{th}$ phase is taking place while losing this token means the end of this phase. Furthermore, firing of transition $t_{pk}^s$ reflects the starting of the $k^th$ phase. It should be noted that the condition $p_{p1} + p_{p2} + p_{p3} + p_{p4} = 1$ has to be satisfied all the time.

Events related to vehicles moving through the intersection are described by the other twelve sub Petri nets; eight sub Petri nets are used to model vehicles entering and crossing the intersection such as $I_j^{in}$ where $I$ takes the notations $(N, S, E, W)$, $i$ stands for $(n, s, e, w)$ and $j$ takes either $l$ (left) or $f$ (forward and right). The sub net $I_j^{in}$ consists of five places and four transitions; the appearance of a token in the place $p_{ij}^i n$ implies that vehicles are entering the intersection from the $i^{th}$ incoming direction with the intention to take the direction of $j$. Otherwise, having no tokens

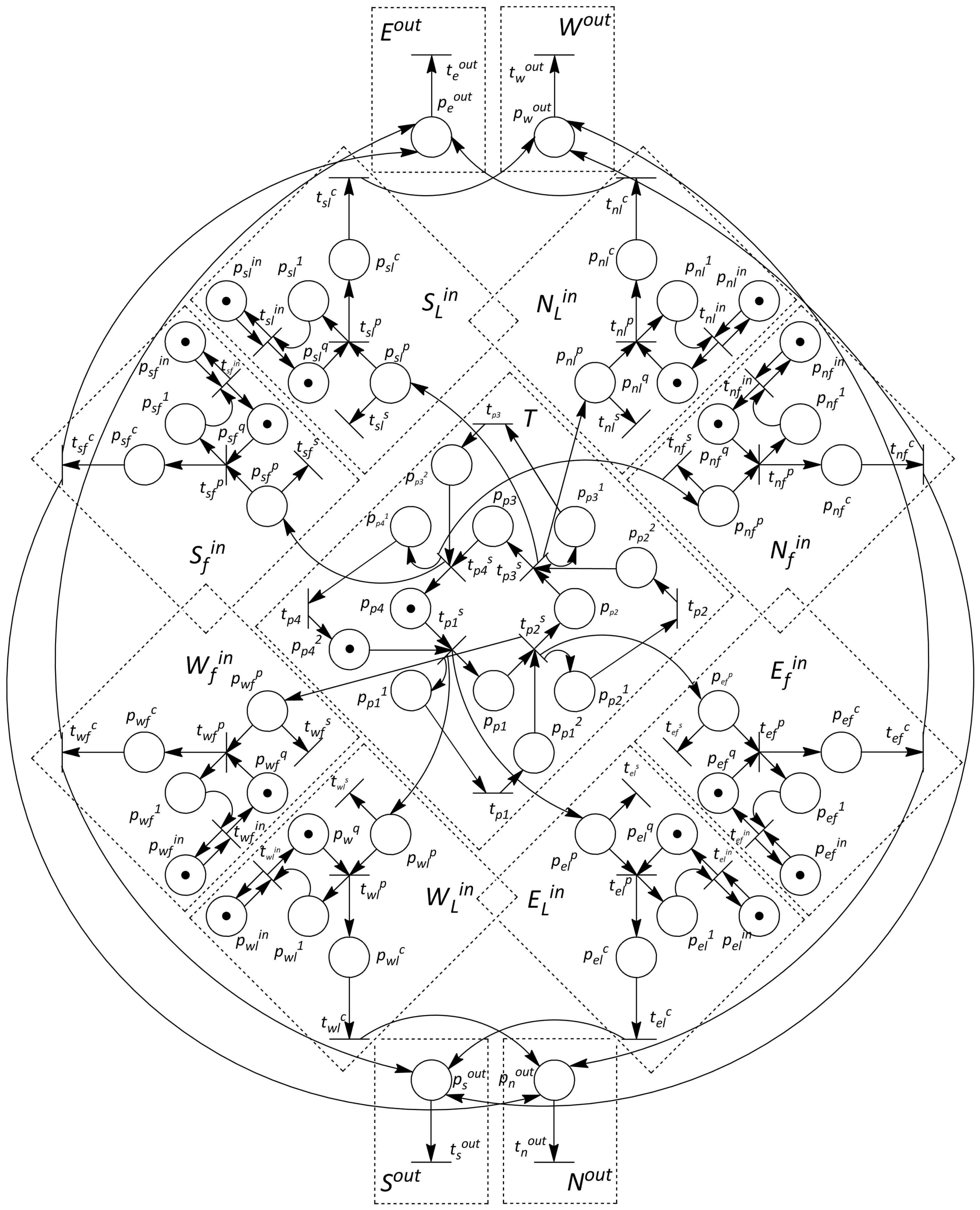


Fig. 4.1. Petri net model for a single signalized intersection.

in this place implies no vehicles are in such a case. Similarly, having a token in the place $p_{ij}^q$ indicates the event that there is a queue of vehicles waiting in the incoming direction $i$ with the intention to cross the intersection directed $j$ (left or forward/right). Furthermore, a token at the places $p^p_{ij}$ means permission is given to those vehicles in the queue to cross the intersection while a token existing in $p^c_{ij}$ represents the event that those vehicles are crossing the intersection. Finally, the four sub Petri nets ($N^{out}, S^{out}, E^{out}, W^{out}$) are quite simple. Each of them consists of a place $p^{out}_i$ and a transition $t^{out}_i$. A token in the aforementioned places describes the event that vehicles are leaving the intersection in the outgoing direction.

The execution of the Petri net model in Fig. 4.1 is based mainly on the state equation given in Eqn. 2.2 and on the assumption that whenever a transition is enabled it fires immediately. The fact that permission is given to vehicles entering the intersection from a specific source towards a specific destination during a specific phase results a modeling consideration that this permission has to be canceled at the end of this specific phase whether some vehicles cross the intersection or not. Consequently, eight conflicts appear in the model between each coupled transitions $(t^p_{ij}, t^s_{ij})$. The proposed algorithm specifies a priority execution for each conflict. If the initial marking of the Petri net model is the one given in Fig. 4.1, the only enabled transition is $t^s_{p1}$. Therefore, it fires immediately through removing a token from each input place to this transition $(p_{p4}, p^2_{p4})$ and adding a token to each output place of this transition $(p_{p1}, p^1_{p1}, p^p_{wl}, p^p_{el})$. It simply means that the first phase starts (indicated through the token at $p^{in}_{p1}$) and permission is given to vehicles entering the intersection from east and west towards left to north and south respectively to cross the intersection.

To clarify the firing sequence, focus only on those vehicles that enter from west towards south. As shown in Fig. 4.1, initially a token exists at $p^q_{wl}$, so vehicles are waiting in the west entrances to take a left with the intersection. However, a conflict appears at this step between transitions $(t^p_{wl}, t^s_{wl})$ since both of them are enabled and only one token marked their common input place $p^p_{wl}$. At this point the algorithm will give the priority to transition $t^p_{el}$, whose firing results removing tokens from the permission and queuing places $(p^p_{wl}, p^q_{wl})$ and adding a token to the place $p^c_{wl}$ which indicates the event of vehicles crossing the intersection from west to south that has taken place simultaneously with the occurrence of the event that vehicles crossing the intersection from east to north.

In the following step, three transitions are fired together ($t^c_{wl}, t^c_{el}, t^s_{p2}$). Firing the transitions $t^c_{wl}, t^c_{el}$ implies that vehicles are leaving the intersection towards south and north through removing tokens from $p^c_{wl}, p^c_{el}$ and adding a token to each one of $p^{out}_s, p^{out}_n$ which indicates the event that vehicles are leaving intersection towards south and north directions. Simultaneously, in the sub Petri net $\mathbf{T}$, the transition $t^s_{p2}$ fires indicates the starting of the second phase and so on.

The necessity of having the sink transition $t^s_{wl}$ is to make sure that vehicles cross the intersection safely. If we assume no vehicles are waiting in the queue to cross the intersection from west to south during the first phase, the permission will still be available through the existence of a token in $p^p_{wl}$ , which will lead vehicles coming from west to cross the intersection even after the ending of the first phase and is unsafe. Therefore, the role of the sink transition $t^s_{wl}$ is to take away this permission if it has not been used during the first phase.

An algorithm was developed to identify the enabled transitions at each time step and fire the enabled transitions immediately based on the state equation and the mentioned priority role. Given the initial marking and input and output incident matrices, the algorithm will firstly construct the column matrix A, which initially is equal to the initial marking and then it will find the following marking using the state equation given in Eqn. 2.2 and compare it to the existing columns. If it is not equal to any of them, it will be considered a new marking, will be saved as a new column in matrix A, and will be used to find the subsequent marking. Otherwise, if the new calculated marking is equal to any of the previous ones (any column in A), the algorithm will stop and matrix A at this point consists of all possible markings in their order of occurrences. The block diagram of the algorithm (Algorithm 1) for capturing the state evolution of the Petri net is shown in Fig. 4.2.

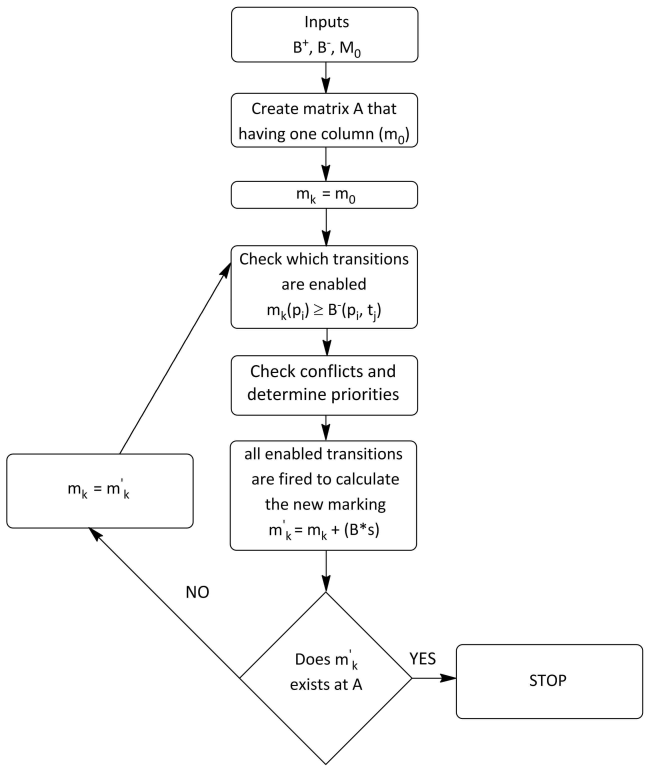

Inputs
B+, B-, M0
Create matrix A that having one column (m0)
mk = m0
Check which transitions are enabled
mk(pi) ≥ B-(pi, tj)
Check conflicts and determine priorities
all enabled transitions are fired to calculate the new marking
m'k = mk + (B*s)
Does m'k exists at A
YES
NO
STOP
mk = m'k

Fig. 4.2. Block diagram of Algorithm 1 for capturing the state evolution of the Petri net shown in Fig. 4.1.

In Table 4.1, we capture the state (marking) evolution of the Petri net model shown in Fig. 4.1 related to each step of transition firing. The first column in the table represents the initial marking of the Petri net. When a specific event occurs at a specific firing step, it will be represented through one at the table cell, which belongs to this event/firing step. Similarly, losing one to be zero in a specific table cell implies the end of that specific event. Since the model consists of 56 places, it is not possible to include all of them in the table. We focused only on the places with essential meanings. It is to be noted that at each firing step, one and only one of the four phase places $(p_{p1}, p_{p2}, p_{p3}, p_{p4})$ will take the value 1 which is expected as previously mentioned. Some places have identical marking through all firing steps, which implies they are identical and can be modeled through one row only.

Table 4.1
The state evolution of the Petri net model for a single intersection.

| | $m_0$ | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th | 9th |
|---|---|---|---|---|---|---|---|---|---|---|
| $p_{p1}$ | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| $p_{p2}$ | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| $p_{p3}$ | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| $p_{p4}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| $p_{ij}^{in}$ | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $p_{wl}^{c}, p_{el}^{c}$ | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $p_{wf}^{c}, p_{ef}^{c}$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| $p_{nl}^{c}, p_{sl}^{c}$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $p_{nf}^{c}, p_{sf}^{c}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| $p_{w}^{out}$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| $p_{e}^{out}$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| $p_{n}^{out}$ | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| $p_{s}^{out}$ | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |

If another marking is used initially, such as a marking that represents the case where no vehicles enter the intersection from north, south, and east, the value 1 will never exist at the place $p_w^{out}$ during any firing step, which means that the event vehicles leave the intersection towards west will not take place. This is an expected result for having avoided the occurrence of other events.

## 4.2 Petri Net Model for Two Connected Intersections

After modeling events in a single intersection through a discrete Petri net, the next step is to extend the model to serve the simple traffic network shown in Figure 3-3, which consists of two successive intersections and the roads connecting them. If we call the intersection on the left side Intersection-1 and the intersection on the right side Intersection-2, the east output of Intersection-1 is the west input to Intersection-2 while the west output of Intersection-2 is the east input to Intersection-1.

For this case, the Petri net model for traffic network shown in Figure 3-3 consist of two single intersection Petri nets models connected to each other. However, minor modifications are done to each one of these two models. For Intersection-1 model, the arcs connecting $t_{el}^{in}$ to $p_{el}^{in}$ and $t_{ef}^{in}$ to $p_{ef}^{in}$ are removed while for Intersection-2 model, arcs connecting $t_{wl}^{in}$ to $p_{wl}^{in}$ and $t_{wf}^{in}$ to $p_{wf}^{in}$ are removed. To distinguish between the Petri net nodes belonging to each intersection, we added (1) as the subscript for each node symbol in Intersection-1 similarly adding (2) for Intersection-2. The two single intersection models are connected to each other through their east and west inputs/outputs nodes shown in Fig. 4.3.

The markings of phase indication places are considered the same for both intersections. However, it does not matter whether they are synchronized or not because the model is an event representation of the logical relations of events (not considering the detailed timings). If this synchronization is changed, there will be a different state evolution but the result will be the same in that the same events will occur.

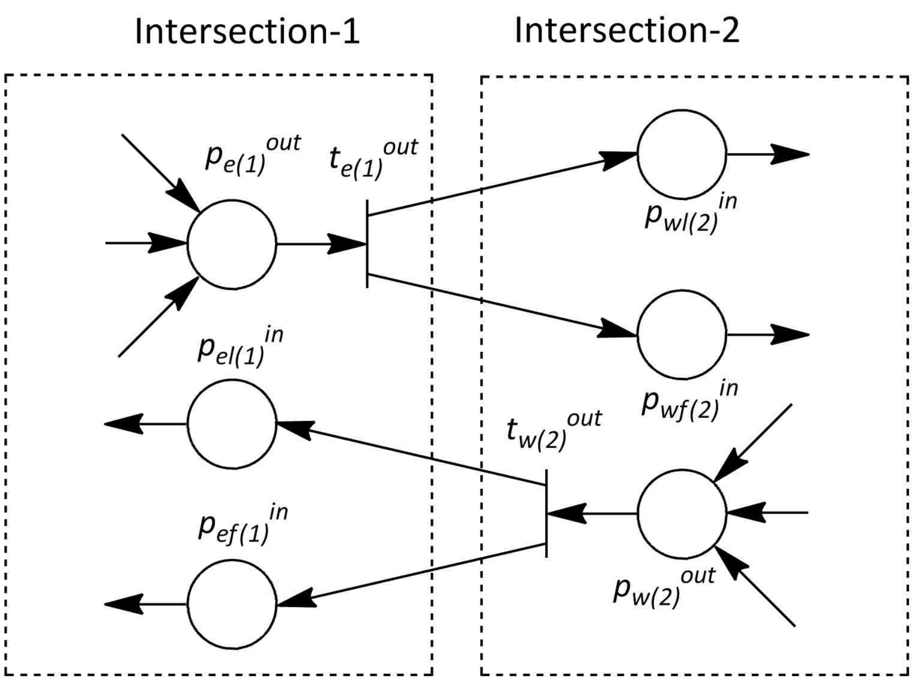


Fig. 4.3. Petri net model of two connected intersections.

During the event that vehicles are leaving Intersection-1 (east directed toward Intersection-2), a token at $p_{e(1)}^{out}$ will result in the firing of transition $t_{e(1)}^{out}$, which removes the token from $p_{e(1)}^{out}$ and adds a token to both places $p_{wf(2)}^{in}$ and $p_{wl(2)}^{in}$ representing the occurrence of events that vehicles are entering Intersection-2 from the west. However, this step does not happen smoothly when we execute the double intersection model using Algorithm 1. The case is that during one phase cycle (the four phases take a place once successively) transitions $t_{e(1)}^{out}$ and $t_{w(2)}^{out}$ fires three times because places $p_{e(1)}^{out}$ and $p_{w(2)}^{out}$ are charged three times per cycle since there are vehicles leave Intersection-1 east directed and Intersection-2 west directed during three phases in a cycle. On the other the four the input places, $p_{ef(1)}^{in}$, $p_{el(1)}^{in}$, $p_{wf(2)}^{in}$ and $p_{wl(2)}^{in}$, gain three tokens each per cycle and release only one of these tokens since the transitions downstream these places fire once each cycle. It is obvious that this firing mechanism will lead the Petri net to be unbounded. Physically, the model will not be considered a good representation of event occurrence anymore because at some point if no vehicles enter to the network through one of the three entrance of Intersection-1, west, north, and south, there will still be vehicles traveling from Intersection-1 to Intersection-2 which is not a real case. That is because places $p^{in}_{ef(1)}$, $p^{in}_{el(1)}$ will still have tokens to feed the net of Intersection-2. The same case of unreal representation will exist for vehicles travel from Intersection-2 towards Intersection-1.

There are several approaches to tackle this modeling issue. First, a Petri net controller was designed based on work by Moody et al. [31] in order to keep the number of tokens in each one of the four mentioned places does not exceed 1. This controller did so but it transferred the unbounded problem to the upstream places $p^{out}_{e(1)}$ and $p^{out}_{w(2)}$. Adding another controller places to control these two nodes will not solve the problem because it causes conflicts, which stops the execution of Algorithm 1 when it is used for the new model (with the controller). Designing a controller for the six places in one step will give the same result. Other solutions such as changing the weights of arcs connecting $p^{out}_{e(1)}$ to $t^{out}_{e(1)}$ and $p^{out}_{w(2)}$ to $t^{out}_{w(2)}$ to be three instead of one will solve the problem only by avoiding having any place in the network be unbounded but the system will not be a good representation of the event sequence. Therefore, a modified binary Petri net was used, which will be presented in detail in the next section.

## 4.3 Modified Binary Petri Nets

As it can be seen from previous discussions, Petri nets can be used to determine whether an event has happened or not based on the occurrence of other events. Modeling in such a way can be easily done through representing the occurrence of a specific event by associating a token in a specified place that describes this event. When the event does not take place, no tokens are assigned to the place. However, with traditional Petri nets modeling mechanisms, there is a problem of having more than one token in some places, which will produce other states that no longer accurately represent event occurrences/non-occurrences. Thus, the idea of using the traditional

Petri net firing mechanisms, but enhanced by adding a restriction that no places can be marked with more than one token, will be a good solution for this problem. Therefore, the modified binary Petri net is defined similarly as the traditional Petri net but with three additional rules as follows.

- The weights of all arcs in the Petri net are ones; which indicates the nets are ordinary and all elements in the input and output incidents matrices can only be zeros or ones.
- The initial marking of the modified binary Petri net can only be zero or one.
- State evolution from one marking to another is performed through the following two steps:
  1. The marking $m'_k$ is calculated according to the traditional state equation of Petri nets given in Eqn. 2.2 ($m'_k = m_k + B.s$);
  2. The marking $m'_k$ is updated according to the number of tokens as follows.

$$m'_k(p) = \begin{cases} 0 & \text{for places } p \text{ such that } m'_k(p) = 0 \\ 1 & \text{for places } p \text{ such that } m'_k(p) \geq 1 \end{cases} \tag{4.1}$$

Based on this definition it is easy to see that, for modified binary Petri nets, we have:

- Elements of incident matrixes take only three values -1, 0, or 1.
- Any modified binary Petri net is bounded since the number of states is less than or equal to $2^n$ where $n$ is the number of places.

To illustrate the differences between the traditional and modified binary Petri nets, lets study the Petri net exampled provided in Fig. 4.4.

If Algorithm 1 is applied when the net is traditional, the marking evolution will be given by:

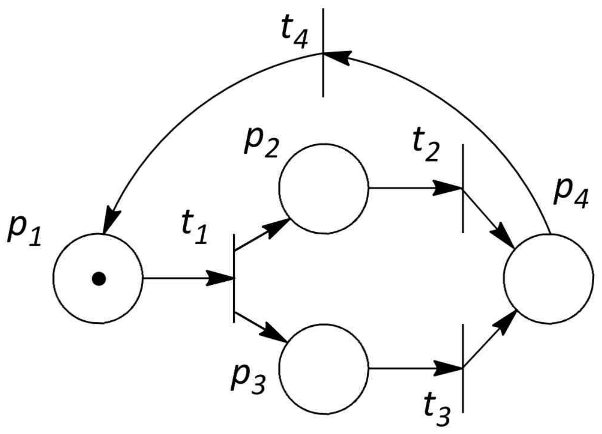


Fig. 4.4. A simple modified binary Petri net model.

$$\begin{bmatrix}1\\0\\0\\0\end{bmatrix} \xrightarrow{t_1} \begin{bmatrix}0\\1\\1\\0\end{bmatrix} \xrightarrow{t_2,t_3} \begin{bmatrix}0\\0\\0\\2\end{bmatrix} \xrightarrow{t_4} \begin{bmatrix}1\\0\\0\\1\end{bmatrix} \xrightarrow{t_1,t_4} \begin{bmatrix}1\\1\\1\\0\end{bmatrix} \xrightarrow{t_1,t_2,t_3} \begin{bmatrix}0\\1\\1\\2\end{bmatrix} \xrightarrow{t_2,t_3,t_4} \begin{bmatrix}1\\0\\0\\3\end{bmatrix} \xrightarrow{t_1,t_4} \begin{bmatrix}1\\1\\1\\2\end{bmatrix}$$

$$\xrightarrow{t_1,t_2,t_3,t_4} \begin{bmatrix}1\\1\\1\\3\end{bmatrix} \xrightarrow{t_1,t_2,t_3,t_4} \begin{bmatrix}1\\1\\1\\4\end{bmatrix} \cdots \begin{bmatrix}1\\1\\1\\r-1\end{bmatrix} \xrightarrow{t_1,t_2,t_3,t_4} \begin{bmatrix}1\\1\\1\\r\end{bmatrix} \xrightarrow{t_1,t_2,t_3,t_4} \begin{bmatrix}1\\1\\1\\r+1\end{bmatrix}$$

As it is shown from marking evolution, the Petri net is unbounded due to place $p_4$, where the number of tokens in it will keep increasing after the marking $\begin{bmatrix}1 & 1 & 1 & 2\end{bmatrix}^T$.

Assume that the Petri net mechanism is restricted to the binary modified net. Even before calculating the marking evolution using MBPN, it is obvious that the total number of achieved markings cannot exceed $2^4$ or 16. The marking evolution for the exampled Petri net will be given by:

$$\begin{bmatrix}1\\0\\0\\0\end{bmatrix} \xrightarrow{t_1} \begin{bmatrix}0\\1\\1\\0\end{bmatrix} \xrightarrow{t_2,t_3} \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \xrightarrow{t_4} \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

From this simple example, it is easy to get this result from analyzing the graph. Using a modified algorithm that support the binary marking feature gives the same result. Only three markings exist and they will keep repeating.

It is not difficult to see that the modified binary Petri nets have bounded markings and are more suitable for the modeling and analysis of connected traffic intersections. Generally, for applications that can be represented through binary-state nodes (0 or 1), the number of states is finite no matter how many nodes the net has. Representing these binary systems through the traditional Petri net might not be a good idea since the number of the reachable states (markings) might be infinite which does not reflect the real evolution of these binary systems. Modified binary Petri net will guarantee smooth representation for event occurrences for those binary systems through its property of bounded markings.

When investigating the Petri net model in Fig. 4.3, Algorithm 1 is still used but with a small modification that takes into account Eqn. 4.1 mentioned above. The algorithm performs 15 steps before it stops due to the condition of having repeated markings. Because of the size of the used Petri net model ($P = 112, T = 88$), Table 4.2 shows only the markings of those places which represent the connections between the two intersections. The modified binary Petri net framework resolves the issue of unbounded states and is a more realistic tool for the modeling and analysis of certain traffic applications other than traditional Petri nets.

Table 4.2
Marking evolution of Petri net model for two connected intersections.

| | $m_0$ | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th | 9th | 10th | 11th | 12th | 13th | 14th | 15th |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p_{p1(1)}, p_{p1(2)}$ | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| $p_{p2(1)}, p_{p2(2)}$ | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| $p_{p3(1)}, p_{p3(2)}$ | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| $p_{p4(1)}, p_{p4(2)}$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| $p_{e(1)}^{out}$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| $p_{wl(2)}^{in}$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| $p_{wf(2)}^{in}$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| $p_{w(2)}^{out}$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| $p_{el(2)}^{in}$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| $p_{ef(2)}^{in}$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

# 5. CONCLUSION

## 5.1 Summary

In Chapter 1, the problem of traffic congestions and the motivation to carry out the proposed research in this thesis was introduced. In addition, related work to the approaches that are used in this thesis (using Petri nets in traffic networks) was presented.

Chapter 2 provided the background knowledge on Petri net models and hybrid Petri net models. Examples including discrete and hybrid Petri nets were described in details in order to simplify the theoretical definitions and dynamics of Petri nets.

Chapter 3 described in detail the structure and parameters of a signalized intersection. Then the hybrid Petri net model for this single intersection where combinations of discrete and continuous nodes are used to represent events and vehicle flow in the intersection was introduced. This model was extended later to model two successive intersections by introducing a hybrid Petri net model for roads that are connecting these two intersections. Simulations were carried out and the results were presented and discussed in details. The simulation results showed the effectiveness of this modeling approach.

Chapter 4 developed a discrete Petri net model for a single traffic intersection and then extended it to represent a traffic network consists of two connected intersections. This model was based on traditional Petri nets in order to analyze the effect of occurrence/non-occurrence of specific events on other network nodes without considering the detailed dynamics. An algorithm was designed to capture transition firings based on the state equation, the priority assignment, and the role of immediate firing for any enabled transitions. It was noted that the traditional Petri net model for two connected intersections lead to unbounded places in the model, which are not realistic for monitoring and control. Thus, a modification to the traditional Petri nets was introduced to resolve this issue. We called this new type of Petri net the Modified Binary Petri nets, which is very good representation for traffic network in terms of event occurrences and traffic dynamic correlations.

## 5.2 Conclusions

Hybrid Petri nets are a very useful modeling and analysis tool for the study of traffic networks. Timed discrete transitions also play a significant role in modeling flow delays in the network, which allowed representing the time required for a vehicle to cross the intersection or travel between two intersections. The firing speed feature of continuous transitions and the flexibility to choosing non-integer values for the arc weights between continuous nodes were the two keys for represent vehicle flow. Furthermore, the role of priority in conflicts between discrete and continuous transitions makes it easy to represent enabling/disabling vehicles movements. Even though the simulator used was a demo-version that was issued this year and was still under development, it was very helpful in many ways, such as the simulation of vehicle flows and queues.

Although discrete Petri nets do not provide the same amount of detail that can be captured through hybrid Petri nets, their simple dynamics can be used smoothly to model some traffic problems including those that need many nodes to be modeled. Modified binary Petri nets were finally introduced in this thesis to overcome some design limitations of traditional discrete Petri nets. It will be a very helpful tool in the study of other applications. For instance, it is smoothly used to model events occurrence/non-occurrences of systems consisting of 112 nodes (places).

## 5.3 Future Work

### 5.3.1 Modified Binary Petri Nets

Starting from the two connected intersection discrete model, it is possible to expand the approach developed in this thesis to study traffic network with much more complex dynamics and behaviors. To see this implication, assuming that there is a complex traffic network consisting of n intersections, it is possible to derive its input incident matrix $B$ composed from input incident block matrices as

$$B^- = \begin{bmatrix} B_1^- & 0 & \cdots & 0 \\ 0 & B_2^- & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_n^- \end{bmatrix}$$

where $B_1^-, B_2^-, B_n^-$ are the input incident matrices for intersections $I_1$, $I_2$, through $I_n$, respectively. The size of these block matrices may be different because it depends on the intersection specifications such as the number of incoming and outgoing roads connected to the intersection and the number of phases. Note that the off-diagonal entries are zeros because intersections are connected through transitions to places.

For the output incident matrix, the diagonal block matrices will be the output incident matrixes for the intersections $B_1^+, B_2^+, B_n^+$. Each one of other block matrices will represent the connection between two intersections. While $B_{1-2}^+$ represents the connections from intersection $I_1$ to intersection $I_2, B_{2-1}^+$ represents the other way connection from $I_2$ to $I_1$. The general form of the output incident matrix for a complex traffic network can be written as:

$$B^+ = \begin{bmatrix} B_1^+ & B_{2-1}^+ & \cdots & B_{n-1}^+ \\ B_{1-2}^+ & B_2^+ & \cdots & B_{n-2}^+ \\ \vdots & \vdots & \ddots & \vdots \\ B_{1-n}^+ & B_{2-n}^+ & \cdots & B_n^+ \end{bmatrix}$$

It is easy to expand the two-intersection discrete model to a larger traffic network with a much greater number of intersections. In addition, the modified binary Petri net model can guarantee the boundedness of places in the net, and thus are suitable for capturing the event occurrences of the overall traffic network.

One direction for future work is to enhance the model with timing information by using timed transitions. By incorporating time information, the new model will be suitable for event simulations in large traffic networks. Another interesting research direction is to identify a wider range of applications (e.g., routing, scheduling, etc.) of modified binary Petri nets in traffic networks.

### 5.3.2 Hybrid Petri Nets

It is also possible to build a hybrid Petri net model for large traffic networks. However, it will take lots of effort. One future direction is to develop a simulation software on hybrid Petri nets and investigate some other models such as variable speed hybrid Petri net [10] (which will be more suitable tool to represent vehicles flow) to improve our simulation and analysis of the traffic network with complex nature and dynamics.

# LIST OF REFERENCES

# LIST OF REFERENCES

[1] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang, "Review of road traffic control strategies," *Proceedings of the IEEE*, vol. 91, pp. 2043–2067, December 2003.

[2] J. Lei and U. Ozguner, "Decentralized hybrid intersection control," in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, vol. 2, pp. 1237–1242, 2001.

[3] A. Di Febbraro, D. Giglio, and N. Sacco, "Urban traffic control structure based on hybrid petri nets," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 5, pp. 224–237, December 2004.

[4] G. Improta and G. E. Cantarella, "Control systems design for an individual signalized junction," *Transp. Res. B*, vol. 18, pp. 147–167, 1984.

[5] D. I. Robertson, "Transyt method for area traffic control," *Traffic Eng. Control*, vol. 10, pp. 276–281, 1969.

[6] M. T. Li and A. C. Gan, "Signal timing optimization for oversaturated networks using transyt-7f," *Transportation Research Board*, pp. 118–126, 1999.

[7] N. H. Gartner, "Development and testing of a demand-responsive strategy for traffic signal control," in *American Control Conference, 1982*, pp. 578–583, June 1982.

[8] D. I. Robertson, "The scoot on-line traffic signal optimization technique," *Traffic Eng. Control*, vol. 23, pp. 190–192, 1982.

[9] J. Y. K. Luk, "Two traffic-responsive area traffic control methods: Scat and scoot," *Traffic Eng. Control*, vol. 25, pp. 14–22, 1984.

[10] R. David and H. Alla, *Discrete, Continuous, and Hybrid Petri Nets.* Springer, first ed., 2005.

[11] R. David and H. Alla, "On hybrid petri net," 2001.

[12] D. Ling-xun, D. Li-hua, Y. Hong-ju, and L. Hang, "Hybrid modeling of control system based on hybrid petri nets," in *Control and Automation, 2007. ICCA 2007. IEEE International Conference on*, pp. 2158–2162, June 2007.

[13] A. Di Febbraro and S. Sacone, "Hybrid modelling of transportation systems by means of petri nets," in *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, vol. 1, pp. 131–135, October 1998.

[14] A. Di Febbraro, D. Giglio, and N. Sacco, "Modular representation of urban traffic systems based on hybrid petri nets," in *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pp. 866–871, 2001.

[15] C. Va andzquez, H. Sutarto, R. Boel, and M. Silva, "Hybrid petri net model of a traffic intersection in an urban network," in *Control Applications (CCA), 2010 IEEE International Conference on*, pp. 658–664, September 2010.

[16] G. List and M. Cetin, "Modeling traffic signal control using petri nets," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 5, pp. 177–187, September 2004.

[17] Y. Qu, L. Li, Y. Liu, Y. Chen, and Y. Dai, "Travel routes estimation in transportation systems modeled by petri nets," in *Vehicular Electronics and Safety (ICVES), 2010 IEEE International Conference on*, pp. 73–77, July 2010.

[18] J. Wu, A. Abbas-Turki, and A. El Moudni, "Discrete methods for urban intersection traffic controlling," in *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pp. 1–5, April 2009.

[19] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems.* Springer, second ed., 2008.

[20] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, pp. 541–580, April 1989.

[21] R. David and H. Alla, "Continuous petri nets," *in Proc. 8th European Workshop on Application and Theory of Petri Nets*, June 1987.

[22] R. David and H. Alla, "Autonomous and timed continuous petri nets," *11th International Conference on Application and Theory of Petri Nets*, pp. 367–386, June 1990.

[23] R. David, "Modeling of hybrid systems using continuous and hybrid petri nets," in *Petri Nets and Performance Models, 1997., Proceedings of the Seventh International Workshop on*, pp. 47–58, June 1997.

[24] M. ALLAM and H. ALLA, "Modeling production systems by hybrid automata and hybrid petri nets," *Conf on Control of Industrial Systems*, May 1997.

[25] Z. Huiqin, G. Jun, X. Youbao, and L. Wei, "Modeling and analysis of a testing system using hybrid petri net," in *Electronic Measurement and Instruments, 2007. ICEMI '07. 8th International Conference on*, vol. 1, pp. 465–470, July 2007.

[26] M. Dotoli, M. Fanti, and A. Mangini, "Fault monitoring of automated manufacturing systems by first order hybrid petri nets," in *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, pp. 181–186, August 2008.

[27] M. Kloetzer, C. Mahulea, C. Belta, L. Recalde, and M. Silva, "Formal analysis of timed continuous petri nets," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 245–250, December 2008.

[28] H. J. Payne, “Models of freeway traffic and contro,” *in Mathematical Models of Public Systems (Simulation Council Proc)*, vol. 1, pp. 51–61, 1971.

[29] M. Papageorgiou, *Applications of Automatic Control Concepts to Traffic Flow Modeling and Control*. Berlin, Germany: Springer-Verlag, first ed., 1983.

[30] J. Julves and C. Mahulea, “Simhpn: A matlab toolbox for hybrid petri nets  user manual,” vol. 1, January 2012.

[31] J. Moody, K. Yamalidou, M. Lemmon, and P. Antsaklis, “Feedback control of petri nets based on place invariants,” in *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, vol. 3, pp. 3104–3109, December 1994.