TIMED-ARC PETRI NETS MODELING AND FORBIDDEN STATE CONTROL APPROACH

Doctor of Philosophy (Ph.D.) Dissertation

Alpaslan YUFKA

Eskişehir 2019

TIMED-ARC PETRI NETS MODELING AND FORBIDDEN STATE CONTROL APPROACH

Alpaslan YUFKA

DOCTOR OF PHILOSOPHY (Ph.D.) DISSERTATION

Department of Electrical-Electronics Engineering
Supervisor: Prof. Dr. Aydın AYBAR
(Co-Supervisor: Assoc. Prof. Dr. Hanife Apaydın ÖZKAN)

Eskişehir
Eskişehir Technical University
Graduate School of Science
April 2019

This thesis has been supported under the project number 1610F665, which is accepted by BAP (Scientific Research Projects) Commission.

FINAL APPROVAL FOR THESIS

This thesis titled "**Timed-Arc Petri Nets Modeling and Forbidden State Control Approach**" has been prepared and submitted by **Alpaslan YUFKA** in partial fulfillment of the requirements in "Eskişehir Technical University Directive on Graduate Education and Examination" for the Degree of Doctor of Philosophy (Ph.D.) in **Electrical-Electronics Engineering** Department has been examined and approved on …./…./**2019**.

Committee Members	**Title Name Surname**	**Signature**
Member (Supervisor)	: Prof. Dr. Aydın AYBAR	.....
Member	: Prof. Dr. Altuğ İFTAR	.....
Member	: Prof. Dr. Hüseyin AKÇAY	.....
Member	: Prof. Dr. Osman PARLAKTUNA	.....
Member	: Assist. Prof. Dr. Nihat ADAR	.....

Prof. Dr. Ersin YÜCEL

Director of Graduate School of Sciences

ÖZET

BAĞLANTILARI ZAMANLANDIRILMIŞ PETRİ AĞLARI MODELLENMESİ VE YASAKLANMIŞ DURUM KONTROLÜ YAKLAŞIMI

Alpaslan YUFKA

Elektrik-Elektronik Mühendisliği Bölümü

Eskişehir Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Nisan 2019

Danışman: Prof. Dr. Aydın AYBAR

(İkinci Danışman: Doç. Dr. Hanife Apaydın ÖZKAN)

Bu tezde, zamanlandırılmış Petri Ağları için zaman gecikmelerinin bağlantılara atandığı yeni bir matematiksel ve grafiksel modelleme yöntemi sunulmuştur. Zamanlandırılmış Petri Ağlarında, durumlar belirtilerin akışını zamanla değiştirmektedir. Bu zamana bağlı hareket, ateşleme süreçleri sırasında ağın matematiksel ve grafiksel değerlendirmesini izlemekte yetersiz kalmaktadır. Önerilen modelde, geçiş halindeki belirtilerin izlenmesini sağlayan zaman elemanı olarak adlandırılan bir üçgen gösterimi sunulmuştur. Ayrıca, sistemin herhangi bir zamandaki durumu, yerlerdeki belirtilerin durumunu bu zamanda gösteren işaretleme vektörü ile geçiş halindeki belirtilerin kalan süresini gösteren kalan zaman vektörünü içermektedir. Sistem durumunun işaretleme ve kalan zaman vektörleri cinsinden ifade edilmesi, ulaşılabilirlik kümesinin elde edilmesini ve zamanlanmış bir ulaşılabilirlik ağacının oluşturulmasını mümkün kılmıştır. Önerilen matematiksel model için kontrolör tasarımı da dikkate alınmıştır. Hem kontrol tasarımı hem de ulaşılabilirlik kümesinin elde edilmesi için ilgili algoritmalar geliştirilmiş ve MATLAB ile benzetimi yapılmıştır. Sonuçlar, üretim, demiryolu ve otomotiv sistemleri gibi gerçek zamanlı ve gerçek dünya uygulamaları ile sunulmuştur. Önerilen yaklaşımın performansını değerlendirmek için, bu yaklaşım Zamanlandırılmış Petri Ağları için başka bir modelleme yöntemi olan Uzatılmış (Streç) Petri Ağları ile karşılaştırılmıştır.

Anahtar Sözcükler: Zaman öğesi, Kalan zaman vektörü, Zamanlandırılmış ulaşılabilirlik grafiği (ağacı), Yasaklı durum denetleyicisi, Otomotiv/üretim/raylı sistemler.

ABSTRACT

TIMED-ARC PETRI NETS MODELING AND
FORBIDDEN STATE CONTROL APPROACH

Alpaslan YUFKA

Department of Electrical-Electronics Engineering

Eskişehir Technical University, Graduate School of Science, April 2019

Supervisor: Prof. Dr. Aydın AYBAR
(Co-Supervisor: Assoc. Prof. Dr. Hanife Apaydın ÖZKAN)

In this thesis, a new mathematical and graphical modeling method where time delays are assigned to arcs is presented for Timed Petri Nets. In Timed Petri Nets, states change by the flow of tokens with the time. This time-dependent movement causes an inability to track mathematical and graphical evaluation of the net during firing processes. In the proposed model, a triangular representation, called time element, that allows monitoring of tokens in transitions is introduced. Additionally, the state of the system at any time contains the marking vector representing the status of tokens in places and the remaining time vector representing the remaining time of tokens in transitions at that time. Expressing the state in terms of the marking and remaining time vectors makes it possible to obtain the reachability set and generate a timed-reachability tree. The controller design is also considered for the proposed mathematical model. Corresponding algorithms are developed for the construction of the reachability set and the controller design, and simulated with MATLAB. Results are presented through real-time and real-world case studies, such as manufacturing, railway, and automotive systems. In order to evaluate the performance of the proposed approach, it is compared with Stretched Petri Nets that is another modeling method for Timed Petri Nets.

Keywords: Time element, Remaining Time Vector, Timed-reachability graph (tree), Forbidden state controller, Automotive/manufacturing/railway systems.

ACKNOWLEDGEMENT

First, I would like to thank my dear supervisors Prof. Dr. Aydın AYBAR and Assoc. Dr. Hanife Apaydın ÖZKAN for their guidance and patience during this thesis. They were more than a supervisor, and they have been always supportive.

I would like to thank Prof. Dr. Altuğ İFTAR and Asst. Prof. Dr. Nihat ADAR for their wise and precious contributions, and the rest of the members of the jury, Prof. Dr. Hüseyin AKÇAY, and Prof. Dr. Osman PARLAKTUNA, with respect.

I'm grateful to my wife, Burcu YUFKA, my mother, Nebahat YUFKA, my father, Mustafa YUFKA, my brother, Muhittin YUFKA, my father-in-law, Mehmet ALPAY, and mother-in-law, Döndü ALPAY, for their patience and support during my doctorate and research. I'm also grateful to my naughty son, Ata Yağız YUFKA because of his scribbles on my papers during my research.

I would like to thank my company, Otokar Automotive and Defense Industry Inc. and Koç Holding, for their permissions to maintain my education. I'm very grateful to my manager Uğur TURHAN for his patience and permissions during my thesis.

I would like to appreciate my friends, Bora TARIMTÖRÜ, Gökhan AÇAR, and other instructors who contributed to and supported my research.

The research in this thesis, which is BAP-1610F665, namely "*Dynamic Control Approaches for Timed Discrete-Event Systems*", was supported by Anadolu University/Eskişehir Technical University. I would like to thank the BAP (Scientific Research Projects) Commission for their support.

..../..../2019

STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

I hereby truthfully declare that this thesis is an original work prepared by me; that I have behaved in accordance with the scientific ethical principles and rules throughout the stages of preparation, data collection, analysis and presentation of my work; that I have cited the sources of all the data and information that could be obtained within the scope of this study, and included these sources in the references section; and that this study has been scanned for plagiarism with "scientific plagiarism detection program" used by Eskişehir Technical University, and that "it does not have any plagiarism" whatsoever. I also declare that, if a case contrary to my declaration is detected in my work at any time, I hereby express my consent to all the ethical and legal consequences that are involved.

.....

Alpaslan YUFKA

CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ALGORITHMS

Page

SYMBOLS AND ABBREVIATIONS

$\boldsymbol{B}$	: The bound vector for the token(s) at places
$B(p)$	: The upper limit of available tokens at the place p at the vector $\boldsymbol{B}$
$\mathcal{C}(S, \phi)$	: The controller function for the forbidden state controller
D	**:** The time delay matrix for outgoing arcs in TdAPN
$D(p, t)$	**:** The time delay of an outgoing arc from t to p in matrix D in TdAPN
$\mathfrak{D}$	: The set of time delays in Timed PN
d_t	**:** The total duration of a firing process t^{λ} in TdAPN
$\mathfrak{d}_t$	: The time delay of the transition t in Timed PN, where $\mathfrak{d}_t \in \mathfrak{D}$
$E(G, k)$	: The set of enabled transitions
$E(G, \boldsymbol{M})$	: The set of enabled transitions at the marking vector $\boldsymbol{M}$
$\hat{E}(G, k)$	: The set of sets of simultaneously-enabled transitions
$\hat{E}(G, \boldsymbol{M})$	: The set of sets of simultaneously-enabled transitions at the marking vector $\boldsymbol{M}$
$F(k)$	: The set of all activated firing processes of G_A at time k, where $F(k)$ is $F_{pre}(k) \cup F_{start}(k)$
$F_{pre}(k)$	: The set of previously activated firing processes of G_A at time k
$F_{start}(k)$	: The set of newly activated firing processes of G_A at time k
$F(S)$	: The set of transitions whose firing processes have been previously and currently activated. It is used for the forbidden state controller. $F(S)$ is $F_{pre}(S) \cup \phi$.
$F_{pre}(S)$	: The set of transitions whose firing processes have been previously activated. It is used for the forbidden state controller.
G	: The tuple of Petri Net (e.g. G_A for TdAPN, G_U for untimed PN).
h_p^t	: The symbol of a time element attached to an outgoing arc from t to p, where $h_p^t \in \nabla$
ϕ	: The set of simultaneously-enabled transitions (it can also be a singleton)
$\hat{\nabla}$	: The expanded set of time elements in TdAPN
∇	: The set of time elements in TdAPN, $\nabla \subseteq \hat{\nabla}$
$\boldsymbol{\nabla^R}$	: The remaining time vector assigned to time elements in TdAPN

$\boldsymbol{\nabla^R}(k)$: The remaining time vector assigned to time elements at time k in TdAPN

$\nabla^R\left(k, h_p^t\right)$: The remaining time of flowing tokens at the time element h_p^t at time k in TdAPN

k : Discrete-time variable, where time is discretized into time slots by using an appropriate sampling period

$\tilde{\mathcal{L}}_0$: The set of deadlock states, $\tilde{\mathcal{L}}_0 \subseteq \tilde{\mathcal{L}}$

$\mathcal{L}_0$: The initial set of undesired states, $\mathcal{L}_0 \subseteq \hat{\mathcal{L}}$

$\mathcal{L}_i$: The i 'th set of undesired states, which are obtained in the l'th iteration and leads the system to $\hat{\mathcal{L}}$, $\mathcal{L}_i \subseteq \hat{\mathcal{L}}$

$\hat{\mathcal{L}}$: The expanded set of undesired states, $\hat{\mathcal{L}} \subseteq R_S(G,S)$

$\tilde{\mathcal{L}}$: The expanded set of deadlock states, $\tilde{\mathcal{L}} \subseteq R_S(G,S)$

$R_S(G,S)$: The reachability set of G from S

$R_R(G,S_0)$: The irreversible set of G, where $R_R(G,S_0) \subset R_S(G,S_0)$

S : The state of G, where $S = \{\boldsymbol{M}, \boldsymbol{\nabla^R}\}$ for TdAPN

$S(k)$: The state of G at time k, where $S(k) = \{\boldsymbol{M}(k), \boldsymbol{\nabla^R}(k)\}$ for TdAPN

S_0 : The initial state of G at the initial time k_0, i.e., $S(k_0)$

t^λ : The symbol of a firing process of t started at time $k = \lambda$ in TdAPN

λ : The starting time-instant of t^λ in TdAPN

$\rho\left(S, F(S)\right)$: The function that gives the next state of S according to $F(S)$ for the forbidden state controller, where $F(S) = F_{pre}(S) \cup \phi$

$\gamma_1(S)$: The function that gives the marking-vector part $\boldsymbol{M}$ of S

$\gamma_2(S)$: The function that gives the remaining-time vector part $\boldsymbol{\nabla^R}$ of S

k_0 : The initial time instant

$k+1$: The next time instant

$\boldsymbol{M}$: The marking vector

$\boldsymbol{M}(k)$: The marking vector at time k

$M(k,p)$: The number of available tokens at the place p at time k in $\boldsymbol{M}(k)$

$\mathbb{N}$: The set of natural numbers

N : The input matrix (It specifies weights for ingoing arcs.)

$N(p,t)$: The weight of an ingoing arc from p to t in matrix N

O	: The output matrix (It specifies weights for outgoing arcs.)
$O(p,t)$	: The weight of an outgoing arc from t to p in matrix O
P	: The finite set of places
$p\bullet$	: The post set of transitions connected to the place $p \in P$
T	: The finite set of transitions
$t\bullet$	: The set of output places connected to the transition $t \in T$
$\bullet t$	: The set of input places connected to the transition $t \in T$
$[.]'$	: The transpose of $[.]$
$[\ \ ]$	: The null vector (an empty vector with no dimension)
$\lvert\{.\}\rvert$	: The cardinality of $\{.\}$
$2^{\{.\}}$	: The power set of $\{.\}$
$\mathbf{0}^{\lvert.\rvert\times 1}$	: The $\lvert.\rvert$ by 1 sized zeros vector.
$\mathbf{1}^{\lvert.\rvert\times 1}$	: The $\lvert.\rvert$ by 1 sized ones vector.
$\emptyset$ or $\varnothing$	: The empty set
◊	: The symbol that denotes the end of example in this study.
GPS	**: G**lobal **P**ositioning **S**ystem
MATLAB	**: MAT**rix **LAB**oratory (the software by MathWorks Inc.)
MECU	**: M**otor **E**lectronic **C**ontrol **U**nit
PNs	**: P**etri **N**et(s)
SPNs	**: S**tretched **P**etri **N**et(s)
TC	**: T**racking **c**ircuit (in railway systems)
TdAPNs	**: T**ime**d-A**rc **P**etri **N**et(s)
ts	: **T**ime **s**lot
VECU	**: V**ehicle **E**lectronic **C**ontrol **U**nit

1. INTRODUCTION

Today's large-scale and complex systems often comprise subsystems which are composed of configuration items including many components (systems of systems). All of these serve a common purpose and work in conjunction with each other to satisfy the current need for automation and data exchange in manufacturing technologies. The current industrial revolution, Industry 4.0, is based on connectivity, big data, and event-based operational technologies. This futuristic innovation includes many large-scale systems, and their infrastructure is constructed using the concept of Systems of Systems. Moreover, safer and better transportation is another popular issue to develop intelligent and autonomous solutions in the field of land, marine, air, and railway systems. Railway systems are more interesting than other fields of transportation due to their safety record [1, 2]. The growing population requires more technological designs in this century. The above systems are sophisticated, large-scale, and event-driven. These can best be expressed by sequential events in the course of time. Discrete-Event Systems are used for this purpose in order to describe such systems using the occurrence of events [3-6]. The state evolution of such systems depends on these occurrences. *Petri Nets* (PNs) are used for, formally and graphically, representing this dependency between events [3-9].

PN is a useful methodology to model and verify Discrete-Event Systems; for instance, event-driven systems, manufacturing systems, transportation systems, automotive systems, etc. [2, 3, 8]. It is also used to analyze such systems so that it allows designing a controller for such systems. PN was basically introduced without the notion of time as Place/Transition Nets or basic PNs [3-6, 8, 9]. Event-based systems consist of activities that include time delays; however, these delays are not expressed using basic PNs at first [10]. Thus, a basic PN is insufficient to express the dynamics of time-delayed systems. In other words, the dynamics of the system were insufficiently expressed using such basic PNs. In such time-delayed systems, the time that is used for transferring the dynamics into the model plays an important role [3, 6, 10-14]. The time information is associated with places, transitions, arcs, or tokens (as age or clock) of basic PNs [4-6, 10-13, 15-21]. In order to specify time delays in the net, *Time PNs* and *Timed PNs* were developed. Then, the basic PN has been called *untimed PNs* after the time extension of basic PNs was introduced. Time delays are exact (deterministic) in Timed PNs while time intervals are used in Time PNs. In this thesis, the deterministic approach is chosen, and Timed PN is considered.

Timed PN is a useful tool to accurately express timed and dynamic systems whose time delays are deterministic. Timed PN is essentially considered in three classes [5, 6]: *Timed-Transition PNs*, where transitions are labeled with time delays; *Timed-Place PNs*, where places are labeled with time delays; and *Timed-Arc PNs*, where arcs (or tokens restricted by arcs) are labeled with time delays. In addition, the methodology of modeling in Timed PNs is formed according to the interpretation of time delays, such as enabling durations, holding durations, and firing durations [20]. Events in real cases have time durations and result in after a certain time is elapsed. In Timed PNs, time delays are mostly associated with transitions because transitions represent events [12]. However, this approach is insufficient to express distinct outputs, which are related to the same event and have different time labels, and it is also inadequate to express dynamic operations, such as transportation, motion, etc. These dynamics can be easily represented and transferred into the model by associating time delays with arcs using Timed-Arc PNs. Furthermore, due to the nature of Timed-Arc PN, arcs that are connected to the same transition (event) can be labeled with different time delays while Timed PN, transitions are labeled with time delays, have no ability in that way. This thesis presents a new mathematical modeling method and a graphical representation for Timed-Arc Petri Nets.

Many studies have been developed for Timed-Arc PNs [6, 15-20, 22-24]. Timed-Arc PNs can be thought of as falling into two groups as follows:

- In the first group, time information is related to arcs and tokens, where tokens have age (clock), and time-intervals attached to arcs limit the transition of tokens according to the age of tokens [15-19, 23]. Time delays are interpreted as enabling durations in this group. The number of applications has been performed using the approach in [25-27].
- In the second group, time information is related to only arcs. This group consists of Timed-Arc PNs, where arcs are only labeled with exact (deterministic) time durations [6, 20, 22, 24]. Time delays are interpreted as enabling or enabling and holding durations in this group.

In the first group, Time PN, where arcs are labeled with time delays (durations), was firstly introduced by Walter [15], and it was called *Time-Arc PNs* (namely, *Aging Token PNs*) by Bolognesi et.al. [16]. Time intervals that are bounded by two non-negative integers were attached to ingoing arcs in order to restrict tokens who have age (clock). The age of the token was incremented by one at each ‘tick’, and the token leaves its input

place when it matures. Other studies on Timed-Arc PNs [17-19, 23, 25-27] were based on this approach. Time delays in this type of Time-Arc PNs in the first group have been interpreted as enabling durations [15-19, 23]. However, the time is elapsed at tokens, where arcs with time-intervals are used to restrict the flow of tokens. On the other hand, in the second group, deterministic Timed-Arc PN was developed [6, 20, 22, 24], where time delays were associated with only ingoing and outgoing arcs in terms of deterministic time durations rather than time intervals. Zhu and Denton firstly introduced deterministic Timed-Arc PNs, where they assumed that only one enabled transition was allowed to be fired at any given time. Next, Bowden developed *Timed PN Superclass* that is another model for deterministic Timed-Arc PNs [20, 24]. In this approach, ingoing and outgoing arcs were attached with deterministic time delays, and both enabling and holding durations were considered in the model of Timed-Arc PNs [20, 24]. Time delays in this type of Time-Arc PNs in the second group have been interpreted as enabling durations or both enabling and holding durations. However, in Timed PNs with enabling durations, tokens at places before a transition can be used by another active firing process related to these tokens. Time delays were not interpreted as firing durations/delays (i.e., time delays were not associated with any continuing transition-firing and flowing process in the transition) neither in the first group nor in the second group. In this thesis, the proposed approach introduces a new mathematical modeling method and a graphical representation for the second group of Timed-Arc PNs, where time delays are interpreted as firing durations. Yufka et.al. have recently presented this novel model of Timed-Arc PNs in [1, 28-31].

In Timed PNs, states of the system are defined by the change of tokens in Petri Nets and movements (flow) of tokens. These time-dependent movements cause an inability to track mathematically and graphically over PNs during the firing process. This causes temporary disappearance of tokens in the marking vector during the firing process such that tokens in transition (namely *flowing tokens*) are not observed mathematically and graphically [6, 11-14]; in addition, the marking vector does not include any information about flowing tokens. The marking vector shows the status information of the system using the presence/absence/number of token in the places. In addition, Timed PN shows the elapsed time of the firing process of a transition instead of indicating the state. The main drawback of Timed PNs is the inability of calculating the next state of Timed PNs and mathematically and graphically observing tokens in transition at each state. This

property of Timed PNs complicates finding all states that the system is able to reach, and designing a forbidden state controller for Timed PNs compared to basic PNs [12, 32]. In the proposed model, a triangular representation, called *time element*, that allows monitoring of flowing tokens is defined. The proposed Timed-Arc PN overcomes the main drawback of Timed PNs by including time elements and transforms any Timed PN into a tripartite structure. The state of the system and the remaining time of the work/operations are shown in terms of vectors. The proposed approach makes possible to obtain the reachability set and generate a timed-reachability graph (tree) enhanced by the time information as long as the marking and remaining time vectors are used to express the state of the system. All situations of tokens, i.e., all situations of states can be computed by using these vectors. In addition, the controller design is also presented for time-delayed systems that are modeled by the proposed mathematical model of Timed-Arc PNs. Basic behavioral properties of the proposed Timed-Arc PN are defined by using the reachability set in order to permit analysis of the proposed approach. A forbidden state controller for time-delayed systems is designed using the reachability set and behavioral properties of the proposed Timed-Arc PN in order to make the system avoid undesired states. Algorithms for constructing the reachability set and constructing a forbidden state controller for the proposed Timed-Arc PN are developed and simulated with MATLAB. Results are presented through real-time and real-world case studies, such as manufacturing, railway, and automotive systems. Furthermore, in order to evaluate the performance of the proposed Timed-Arc PN, the proposed methodology is compared with Stretched PNs that were developed by Aybar and İftar [11-14, 46, 47] in this thesis. The methodology of Stretched PN is a novel model to overcome the same drawback of Timed PNs [11-14]. Stretched PN uses the methodology of transition-stretching, namely *Transition-Stretched PNs* [11-14], by adding a pair of place-transition into the timed net, or the methodology of place-stretching, namely *Place-Stretched PNs* [46, 47], by adding a pair of transition-place into the timed net; as a result, Stretched PN transforms Timed PN into a stretched version of this Timed PN. This makes Timed PN as if it is an equivalent untimed PN. Thus, Stretched PN allows analyzing Timed PNs and designing a supervisory controller for Timed PNs.

In this thesis, it is aimed to develop a new mathematical modeling method and graphical representation, which represents a new notion and model for Timed PNs. It is also aimed to develop control approaches for the proposed Timed-Arc PN based on the

reachability set of time-delayed systems. Then, real-world case studies, such as manufacturing systems, railway systems, or automotive systems, etc. are modeled and controlled by using the proposed approach.

2. PETRI NETS

In this chapter, first of all, basic definitions are presented for basic (untimed) PNs [3-8] with an example. Then, the time extension of such basic PNs is introduced as Timed PNs [4-6, 11-14, 20, 22, 46-47] in terms of interpretation of time delays with a comprehensive study. Next, Stretched PNs [11-14, 46, 47] that offer a stretched net of Timed PNs are described in order to compare the proposed methodology in this thesis with Stretched PNs.

2.1. Untimed Petri Nets

PN is a modeling paradigm. PN provides verification and formal modeling for Discrete Event Systems. PN is used for modeling and analyzing event-based systems. It is also used to design a controller for such systems. Basic PNs are Place/Transition Nets, where there is no time delay (duration) in the model of basic PNs; thus, this type of PNs is also called *untimed PNs* in the literature [3-6]. The net of basic PNs is composed of places, transitions, and arcs. In the graphical representation of basic PNs, circles show places, bars show transitions, and arrows show arcs between places and transitions or vice versa. In addition to these elements, tokens are used to denote the status of places, and they are shown by filled dots "●". Moreover, a place that is connected to a transition as a premise is called an input place of this transition. Similarly, a place that is connected to a transition as a successor is called an output place of this transition. Arcs from places to transitions are called *ingoing arcs*, and arcs from transitions to places are called *outgoing arcs* [5-8]. An example PN with these elements is shown in Figure 2.1.

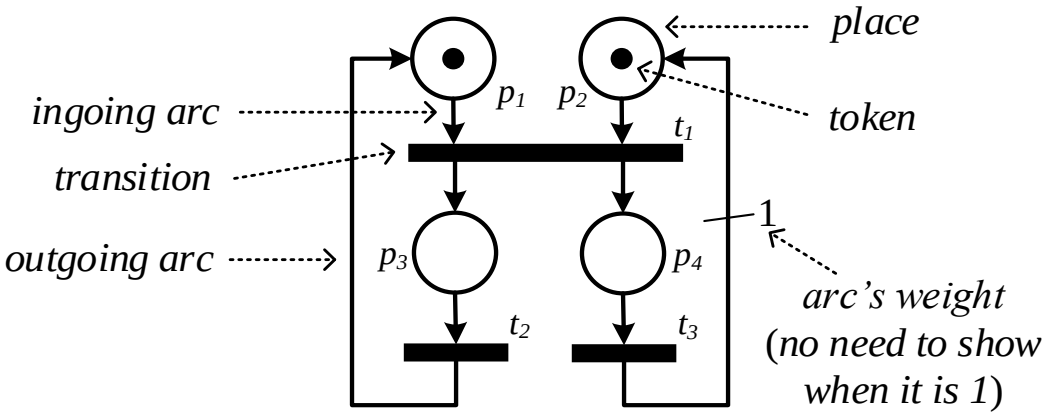


Figure 2.1. *Example Petri net*

A place may denote a condition, a resource, a signal, a buffer, or a datum; and a transition may denote an event, a process, a task/job, a logical clause, or a computational step [7]. A resource can be a machine, a robot, a person, etc. Arcs are used to construct a logical relation between places and transitions. Thus, prerequisites of an event are determined by the weights of arcs, and an event happens when all prerequisites are satisfied. The adequacy of the prerequisites is determined by tokens at related places. Moreover, the state of the system is represented by a marking vector that denotes the number of tokens at all places.

A tuple of an untimed PN is represented by $G_U(P, T, N, O, \boldsymbol{M}_0)$. Here, P denotes the set of places. T denotes the set of transitions, where P and T are disjoint ($P \cap T = \emptyset$). An element in the set P is denoted by a place $p \in P$, and an element in the set T is denoted by a transition $t \in T$. $N: PxT \rightarrow \mathbb{N}$ denotes the input matrix that specifies weights of ingoing arcs. $\mathbb{N}$ is the set of natural numbers. If a connection exists from $p \in P$ to $t \in T$, then $N(p,t) \neq 0$. Otherwise, $N(p,t)$ is equal to zero. Similarly, $O: PxT \rightarrow \mathbb{N}$ denotes the output matrix that specifies weights of outgoing arcs. If a connection exists from $t \in T$ to $p \in P$, then $O(p,t) \neq 0$. Otherwise, $O(p,t)$ is equal to zero. The state of the system is represented by the marking vector $\boldsymbol{M}: P \rightarrow \mathbb{N}$, and the initial marking vector is denoted by $\boldsymbol{M}_0$.

In G_U, a transition $t \in T$ is enabled at $\boldsymbol{M}$ if and only if $M(p) \geq N(p,t) \geq 1$, $\forall p \in P$, where $M(p)$ represents the number of tokens at $p \in P$. This condition is called ***enabledness*** (or ***firing rule***). The set of enabled transitions at $\boldsymbol{M}$ is represented by $E(G_U, \boldsymbol{M})$. For an enabled transition $t \in E(G_U, \boldsymbol{M})$ that fires at $\boldsymbol{M}$, the new marking vector is computed as follows:

$$\widehat{M}(p) = M(p) + O(p,t) - N(p,t)\,, \qquad p \in P,\ \ t \in E(G_U, \boldsymbol{M}) \tag{2.1}$$

Here, the new marking vector is represented by $\widehat{\boldsymbol{M}}: P \rightarrow \mathbb{N}$, and the number of available tokens at the place $p \in P$ at $\widehat{\boldsymbol{M}}$ is represented by $\widehat{M}(p)$.

Let us consider the untimed Petri Net shown in Figure 2.1. This untimed PN model is described as $G_U(P, T, N, O, \boldsymbol{M}_0)$. Here, the set of places is $P = \{p_1, p_2, p_3, p_4\}$. The set of transitions is $T = \{t_1, t_2, t_3\}$. $N = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$ and $O = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ are input and output matrices, respectively. The initial marking vector is $\boldsymbol{M}_0$=[1 1 0 0]' ([.]' represents the

transpose of [.]). The set of enabled transitions at $\boldsymbol{M}_0$ is $E(G_U, \boldsymbol{M}_0) = \{t_1\}$. Reachable states from the initial marking $\boldsymbol{M}_0$ are represented as $R_S(G_U, \boldsymbol{M}_0)$, which denotes *the reachability set* of G_U. For this example, the reachability set is $R_S(G_U, \boldsymbol{M}_0)$ = {$\boldsymbol{M}_0$=[1 1 0 0]', $\boldsymbol{M}_1$=[0 0 1 1]', $\boldsymbol{M}_2$=[1 0 0 1]', $\boldsymbol{M}_3$=[0 1 1 0]'}. Moreover, relations among these states are depicted as shown in Figure 2.2. This figure is generally called *reachability tree/graph.* ◊

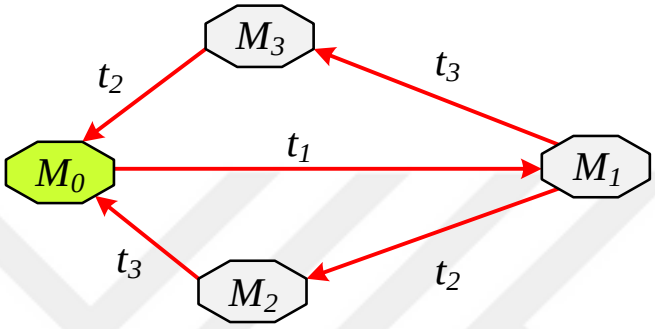


Figure 2.2. *Reachability tree of Petri Net in Figure 2.1*

2.2. Timed Petri Nets

A basic PN is insufficient to express the dynamics of time-delayed systems. Such systems consist of activities that include time delays; however, these delays are not expressed using basic PNs [10]. The time is necessary to express in time-delayed systems [3, 10]. In order to specify time delays in the net, Time PNs and Timed PNs were developed. Time delays are considered as [5, 6, 20]: deterministic durations, where time values are exact and selected from a subset of ℕ; stochastic durations, where time values are expressed by a probabilistic function; and time intervals, where time durations have lower and upper bounds. Time delays are exact (deterministic) in Timed PNs while time intervals are used in Time PNs. In this thesis, the deterministic approach is chosen. In Timed PNs, time delays are associated with basic components of basic PNs that are transitions, places, arcs, and also tokens [5, 6, 10-13, 15-19]. Timed PN is named according to the association-type of time delays. If the time is attached to transitions, then Timed PN is named Timed-Transition PN. If the time is attached to places, then Timed PN is named Timed-Place PN. If the time is attached to arcs, then Timed PN is named Timed-Arc PN. This thesis introduces a new mathematical modeling method and a new representation for Timed-Arc PNs.

The methodology of modeling in Timed PNs is formed according to the interpretation of time delays, such as firing durations, holding durations, and enabling durations [20].

Firing durations relate to transition-firing processes. The time delay is associated with transitions. The interpretation of firing durations is illustrated in Figure 2.3 [20]. In this figure, the analysis of the Timed PN is given in the time interval $[k_0, k_a]$, where the discrete time variable is denoted by $k \in \mathbb{N}$. This time variable is denoted in terms of time slots. In this net, t_1 has a time delay of $k_a - k_0 = a$ time slots while time delays of t_2 and t_3 are considered one time slot. The transition t_1 fires at time $k = k_0$. Tokens at places p_1 and p_2 disappear at time $k > k_0$ while the transition t_1 holds tokens at the time interval $k \in (k_0, k_a)$ during the firing process as shown in Figure 2.3.(b). This disappearance is called *flowing token* in this thesis. Flowing tokens are not monitored through the marking vector. The firing process of the transition t_1 ends at time $k = k_a$, and Tokens appear at places p_3 and p_4 at time $k = k_a$ as shown in Figure 2.3.(c).

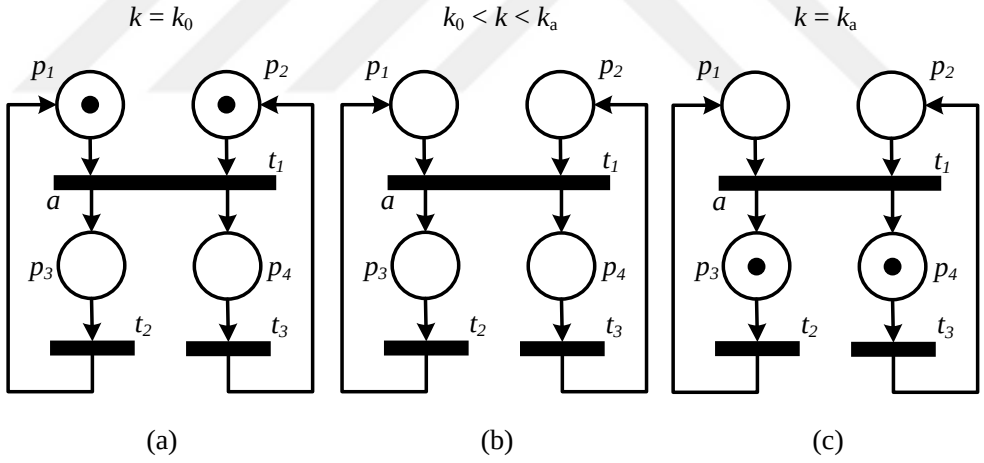


Figure 2.3. *Firing process of t_1 for Timed PN with firing durations*

A Timed PN with firing delays, namely Timed-Transition PN, is represented by a 6-tuple $G_T(P, T, N, O, \mathfrak{D}, \boldsymbol{M}_0)$ [6, 11-13]. The time delay of a transition $t \in T$ is represented by $\mathfrak{d}_t \in \mathbb{N} \setminus \{0\}$, and the set of time delays is denoted by $\mathfrak{D} := \{\mathfrak{d}_{t_1}, \mathfrak{d}_{t_2}, \dots, \mathfrak{d}_{t_n}\}$. $\boldsymbol{M}_0 = \boldsymbol{M}(k_0)$ is the initial marking vector at the initial time k_0. The state of G_T includes the marking vector and the pair of firing transitions and their elapsed time.

Holding durations relate to firing processes evaluated at output places (places after the transition). The time delay can be associated with places, transitions or outgoing arcs. Let us consider the case of associating time-delays with places. The interpretation of holding durations is illustrated in Figure 2.4 [20]. In this figure, the analysis of the Timed PN is given in the time interval $[k_0, k_a]$, where $k_b < k_a$. In this net, p_3 has a holding time delay of $k_a - k_0 = a$ time slots; and p_4 has a holding time delay of $k_b - k_0 = b$ time slots while holding time delays of p_1 and p_2 are considered one time slot. The transition t_1 fires at time $k = k_0$. Tokens at places p_1 and p_2 disappear at time $k > k_0$ while the output place p_3 holds the token at the time interval $k \in (k_0, k_a)$ and the output place p_4 holds the token at the time interval $k \in (k_0, k_b)$ during the firing process as shown in Figure 2.4.(b). Tokens held at places p_3 and p_4 are denoted in the marking vector during the firing process of t_1; however, these tokens are not available as long as output places p_3 and p_4 hold these tokens. The firing process of the transition t_1 ends at time $k = k_a$, and tokens become available at the place p_3 at time $k = k_a$ and at the place p_4 at time $k = k_b$ as shown in Figure 2.4.(c).

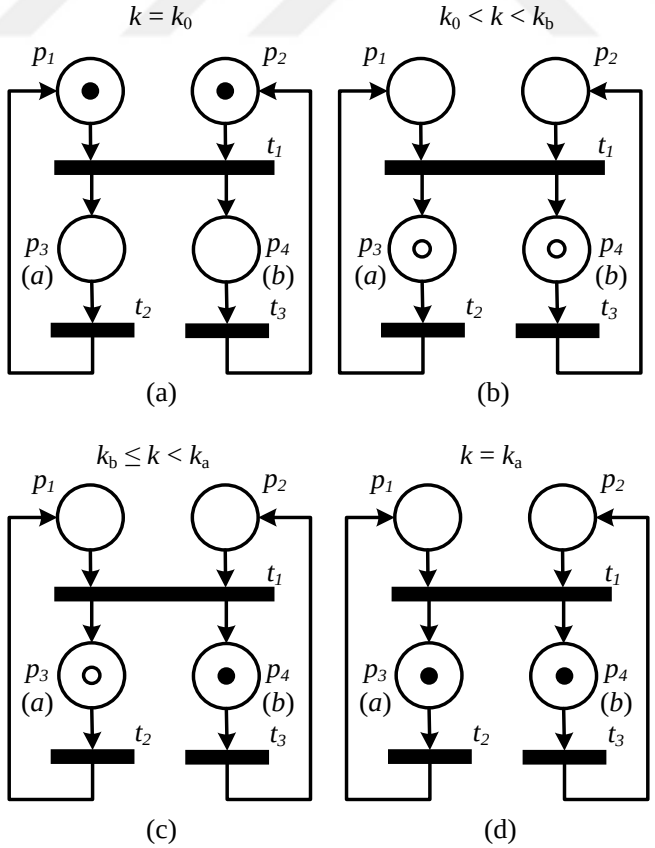


Figure 2.4. *Firing process of t_1 for Timed PN with holding durations*

A Timed PN with holding delays, namely Timed-Place PN, is represented by a 6-tuple $G_P(P,T,N,O,\mathcal{D},S_0)$ [6, 46, 47]. Here, $\mathcal{D}: P \rightarrow \mathbb{N}\backslash\{0\}$ represents holding time delays of places. The time delay of a place $p \in P$ is represented by $\mathcal{D}(p) \in \mathbb{N} \setminus \{0\}$. $S_0 = S(k_0)$ is the initial state of G_P at the initial time k_0. The state of G_P includes the marking vector and the pair of tokens (shown by unfilled circles) held in the output place during the firing process and the remaining time of tokens to be available.

Enabling durations relate to firing processes evaluated at input places (places before the transition). The time delay can be associated with places, transitions or ingoing arcs. Let us consider the case of associating time-delays with transitions. The interpretation of enabling durations is illustrated in Figure 2.5 [20]. In this figure, the analysis of the Timed PN is given in the time interval $[k_0, k_\mathrm{a}]$. In this net, t_1 has an enabling time delay of $k_\mathrm{a} - k_0 = a$ time slots while time delays of t_2 and t_3 are considered one time slot. The firing process of the transition t_1 starts at time $k = k_0$. Input places p_1 and p_2 hold tokens at the time interval $k \in [k_0, k_a)$ during the firing process as shown in Figure 2.5.(b). During this time interval, tokens at input places p_1 and p_2 are not removed from these places. These tokens are denoted in the marking vector during the firing process; however, these tokens may be used for another firing process whose enabling duration is shorter if the place p_1 or p_2 were a place in common with another transition. The transition t_1 fires at time $k = k_a$ such that the firing process of the transition t_1 also ends at this time. Tokens appear at output places p_3 and p_4 at time $k = k_a$ as shown in Figure 2.5.(c).

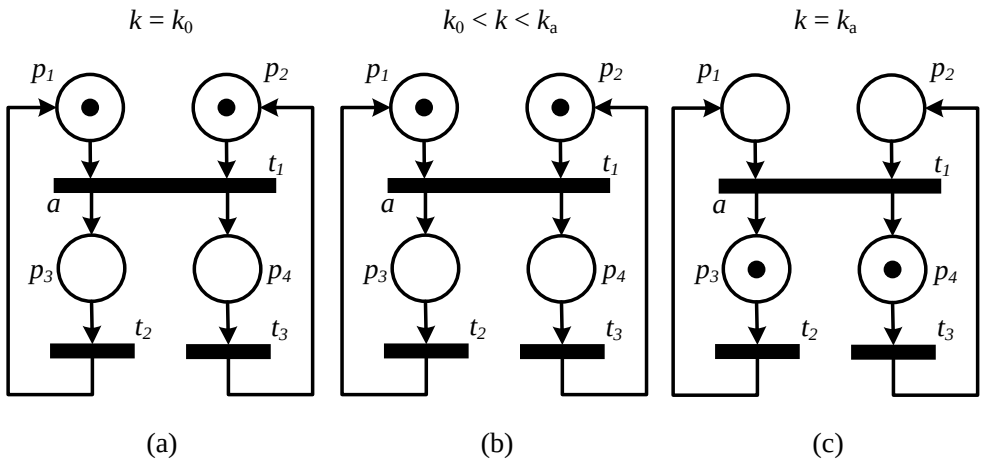


Figure 2.5. *Firing process of t_1 for Timed PN with enabling durations*

Figure 2.3, Figure 2.4, and Figure 2.5 show basic examples of Timed PNs, where time delays are interpreted as firing, holding and enabling durations. A time-delayed system should be modeled using the appropriate interpretation of time delays, such that the modeling strategy changes according to this.

Interpreting time delays as firing durations rather than holding and enabling is more realistic and closer to real-world applications and prevents the use of tokens used for a firing process from being used by another firing process. In addition, distinct time delays are labeled with arc so that distinct operations related to the same event can be easily defined. In this thesis, the proposed methodology interprets time delays as firing durations and associates time delays with arcs.

2.3. Stretched Petri Nets

In Timed PNs with firing durations, transitions hold tokens during the firing process. Unfortunately, the marking vector alone is insufficient to represent the complete state of the system; in addition, the state of such Timed PNs indicates the elapsed time of the firing process related to the transition instead of representing the status of flowing tokens. In Timed PNs with holding durations, tokens reside in output places [6, 20, 46, 47]. These tokens become available at the corresponding output place after a certain time delay is elapsed. These properties of Timed PNs with firing or holding durations complicate the design of a forbidden state controller for Timed PNs compared to untimed PNs [12, 32]. For this purpose, Stretched PNs was recently developed by Aybar and İftar [11-14, 46, 47]. Stretched PN transforms the structure of Timed PNs into a structure of the basic (untimed) PNs in order to analyze Timed PNs and design a supervisory controller for Timed PNs.

Stretched PN is obtained by using two approaches, such as a transition-stretching and a place-stretching. For the transition-stretching, a pair of place-transition is added into the model of Timed PNs with firing durations for a unit time delay. When the methodology of the transition-stretching is used, Stretched PN is called *Transition-Stretched PNs* [11-14]. Similarly, for the place-stretching, a pair of transition-place is added into the model of Timed PNs with holding durations for a unit time delay. When the methodology of the place-stretching is used, Stretched PN is called *Place-Stretched PNs* [46, 47].

A Transition-Stretched PN is defined as a 5-tuple $G_{TS}(P_s, T_s, N_s, O_s, \boldsymbol{M_{s_0}})$ [11-14], and any Timed PN with firing durations is stretched by using the transition-stretching procedure defined in [11-14]. Transition-Stretched PN uses discrete time variable k and unit time delays that are associated with transitions. In G_{TS}, $P_s := P \cup \overline{P}_s$ denotes the set of places after the transition-stretching procedure, where $\overline{P}_s$ includes newly generated places as $p_i^t \in \overline{P}_s$ for $1 \leq i \leq \mathfrak{d}_t - 1$. $T_s := T \cup \overline{T}_s$ denotes the set of transitions after the transition-stretching procedure, where $\overline{T}_s$ includes newly generated transitions as $t_i^t \in \overline{T}_s$ for $1 \leq i \leq \mathfrak{d}_t - 1$. $N_s: P_s \times T_s \to \mathbb{N}$ is the input matrix. $O_s: P_s \times T_s \to \mathbb{N}$ is the output matrix. $\boldsymbol{M_{s_0}}: P_s \to \mathbb{N}$ is the initial marking vector at the initial time k_0. The state of G_{TS} includes the marking vector after the transition-stretching procedure, and it is obtained from the marking vector of the Timed PN with firing durations by using a transformation matrix. The state of G_{TS} includes the marking vector of the Timed PN with firing durations and information of flowing tokens related to the continuing firing process. An example of the equivalent Transition-Stretched PN for the Timed PN with firing durations in Figure 2.3 is given in Figure 2.6. Let us consider the time delay of t_1 in Figure 2.3 as $k_a - k_0 = 3$ time slots.

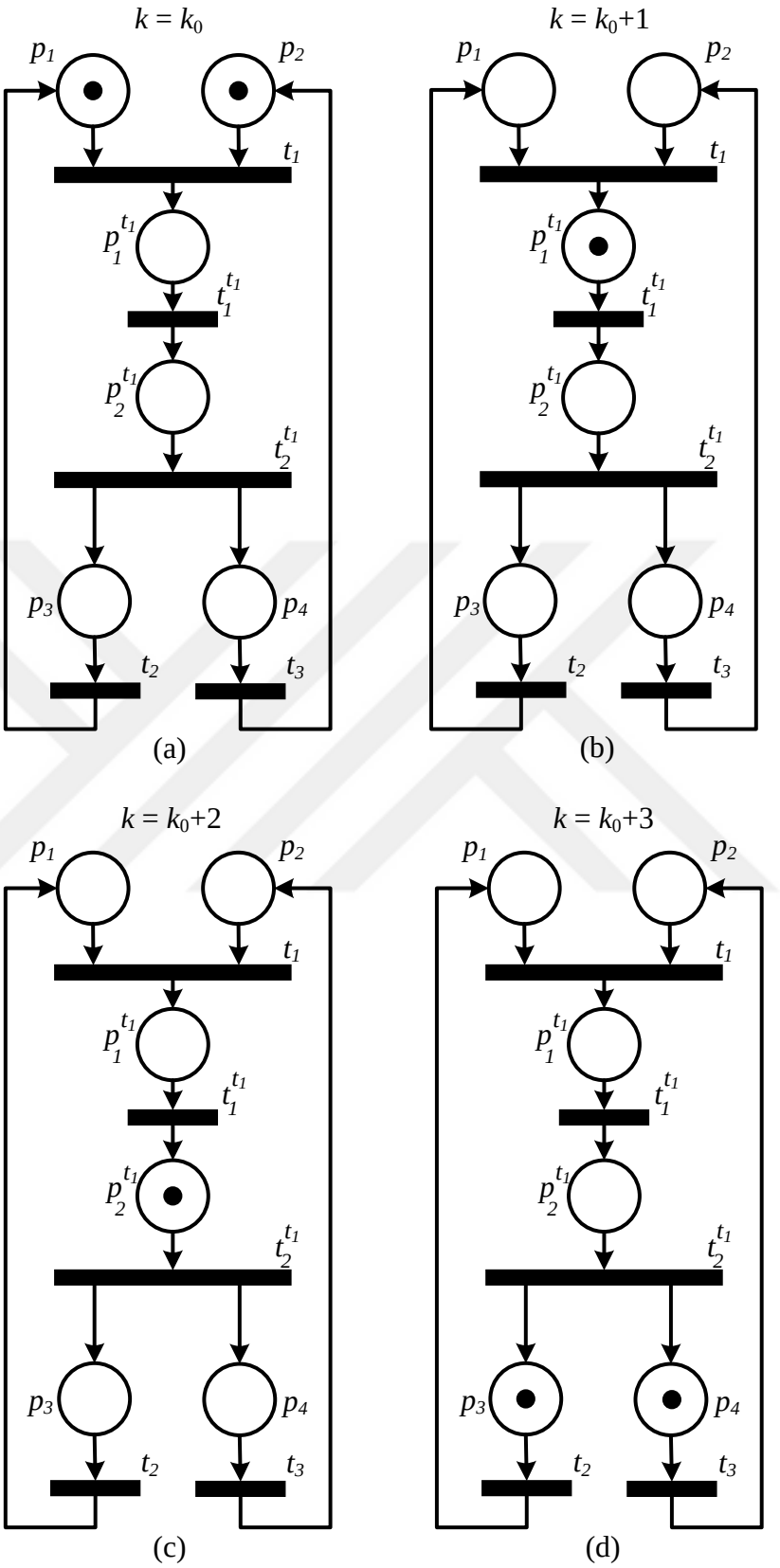


Figure 2.6. *Transition-Stretched PN Equivalent for Timed PN in Figure 2.3*

Let us consider an example of Transition-Stretched PN in Figure 2.6. The transition t_1 has a time delay of three time slots. Thus, places $p_1^{t_1}$, $p_2^{t_1}$ and transitions $t_1^{t_1}$, $t_2^{t_1}$ are added in order to stretch the transition t_1. The transition t_1 fires at time $k = k_0$, and the transition $t_1^{t_1}$ immediately fires at time $k = k_0 + 1$. The flow of the token related to the firing process of t_1 for the Timed PN with firing durations is observed through places $p_1^{t_1}$ and $p_2^{t_1}$ as shown in Figure 2.6.(b), (c). The firing process related to the transition t_1 ends after three time slots are elapsed, and tokens appear in places p_3 and p_4 at time $k_0 + 3$ as shown in Figure 2.6.(d). ◊

A Place-Stretched PN is defined as a 5-tuple $G_{PS}(P_s, T_s, N_s, O_s, \boldsymbol{M_{s_0}})$ [46, 47], and any Timed PN with holding durations is stretched by using the place-stretching procedure defined in [46, 47]. Place-Stretched PN uses discrete time variable k and unit time delays that are associated with places. In G_{PS}, $P_s := P \cup \overline{P}_s$ denotes the set of places after the place-stretching procedure, where $\overline{P}_s$ includes newly generated places as $p_i^p \in \overline{P}_s$ for $1 \leq i \leq \mathcal{D}(p) - 1$. $T_s := T \cup \overline{T}_s$ denotes the set of transitions after the place-stretching procedure, where $\overline{T}_s$ includes newly generated transitions as $t_i^p \in \overline{T}_s$ for $1 \leq i \leq \mathcal{D}(p) - 1$. $N_s: P_s \times T_s \to \mathbb{N}$ is the input matrix. $O_s: P_s \times T_s \to \mathbb{N}$ is the output matrix. $\boldsymbol{M_{s_0}}: P_s \to \mathbb{N}$ is the initial marking vector at the initial time k_0. The state of G_{PS} includes the marking vector after the place-stretching procedure. The state of G_{PS} includes the marking vector of the Timed PN with holding durations and information of flowing tokens related to the continuing firing process. An example of the equivalent Place-Stretched PN for the Timed PN with holding durations in Figure 2.4 is given in Figure 2.7. Let us consider time delays of p_3 and p_4 in Figure 2.4 as $k_\mathrm{a} - k_0 = 3$ time slots and $k_\mathrm{b} - k_0 = 2$ time slots.

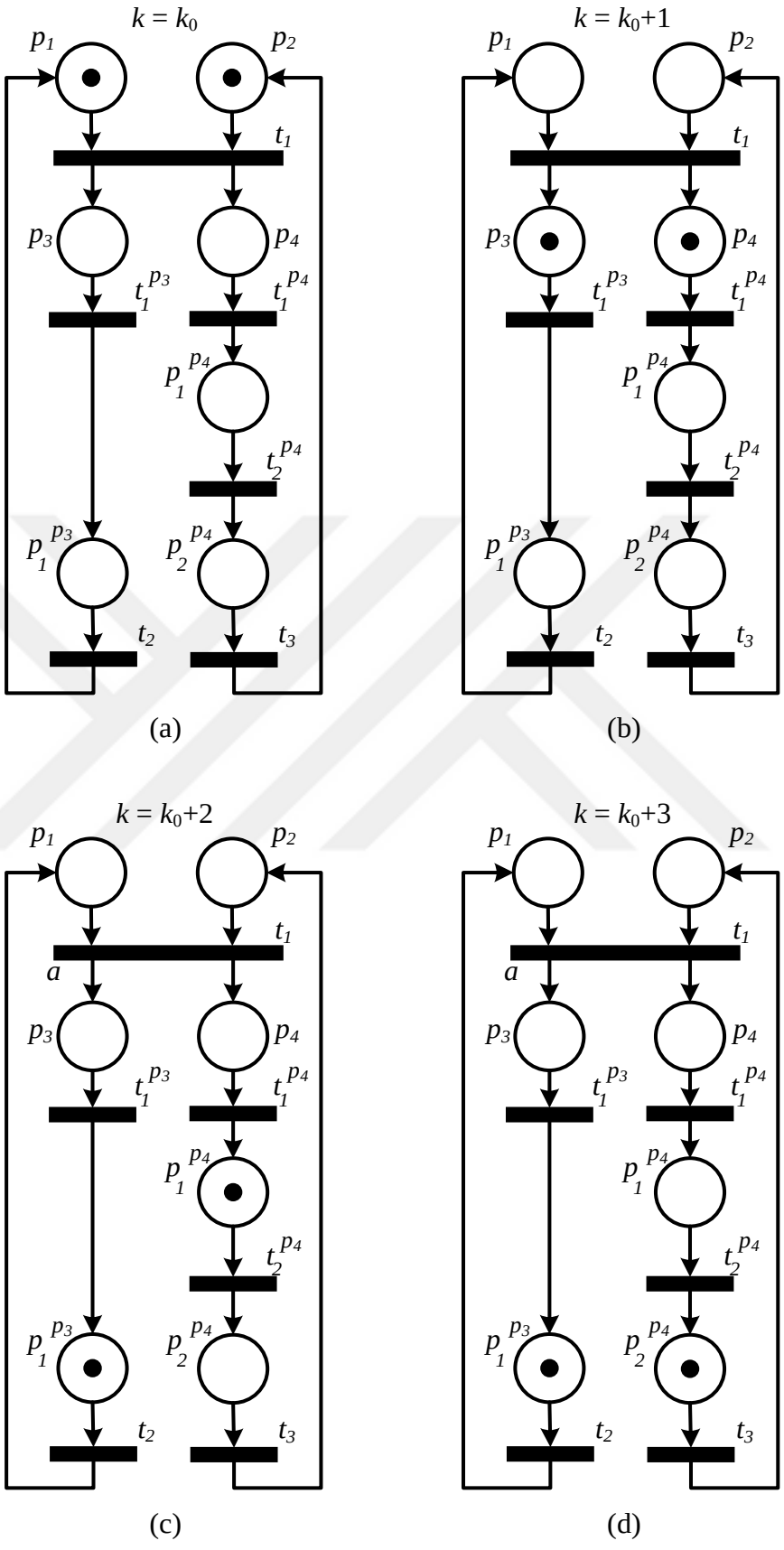


Figure 2.7. *Place-Stretched PN Equivalent for Timed PN in Figure 2.4*

Let us consider an example of Place-Stretched PN in Figure 2.7. The place p_3 has a time delay of 3 time slots, and the place p_4 has a time delay of 2 time slots. Thus, transitions $t_1^{p_3}$, $t_1^{p_4}$, $t_2^{p_4}$ and places $p_1^{p_3}$, $p_1^{p_4}$, $p_2^{p_4}$ are added in order to stretch places p_3 and p_4. The transition t_1 fires at time $k = k_0$, transitions $t_1^{p_3}$ and $t_1^{p_4}$ immediately fire at time $k = k_0 + 1$, and the transition $t_2^{p_4}$ immediately fires at time $k = k_0 + 2$. The flow of tokens related to the firing process of t_1 for the Timed PN with holding durations is observed through places p_3, p_4, $p_1^{p_3}$, $p_1^{p_4}$, $p_2^{p_4}$ as shown in Figure 2.7.(b)-(d). The firing process related to the transition t_1 ends after three time slots are elapsed. In addition, the flowing token related to p_3 appears in the place $p_1^{p_3}$ at time $k = k_0 + 2$, and the flowing token related to p_4 appears in the place $p_2^{p_4}$ at time $k = k_0 + 3$ as shown in Figure 2.6.(d).

◊

In this thesis, Stretched PNs are used for comparing the proposed Timed-Arc PN with Stretched PNs [11-14, 46, 47] in order to evaluate the performance of the proposed methodology. Next chapter introduces the proposed Timed-Arc PNs for Timed PNs.

3. TIMED-ARC PETRI NETS

In Timed PN, where time is interpreted as firing durations, transitions hold tokens during the firing process. Therefore, tokens are not monitored over the net of Timed PN during the firing process. This causes temporary disappearance of tokens in the marking vector of Timed PN during the firing process such that tokens in transition are not mathematically and graphically observed [6, 11-14]. In addition, the marking vector does not include any information about tokens in transition. However, there may be instances where it is necessary to monitor and determine the state, besides the marking vector. In addition, Timed PN shows the elapsed time of the firing process of a transition instead of indicating the state. The main drawback of Timed PN is the inability to calculate the next state of Timed PN and to observe tokens in transition, mathematically and graphically, at each state. This property of Timed PN complicates finding all states that the system is able to reach and designing a forbidden state controller for Timed PNs compared to basic Petri Nets [11-14].

In order to represent temporal dynamics that become invisible during the firing process of such Timed PNs, mathematically and graphically, this thesis presents a new model of Timed-Arc PNs. In the proposed Timed-Arc PNs, the next state is formally computed using marking and remaining time vectors, which allows computing all situations of states. Thus, the reachability set is readily constructed. Based on the reachability set, behavioral properties of the proposed Timed-Arc PNs are defined in order to permit analysis of the proposed approach. Using the reachability set and behavioral properties of the proposed Timed-Arc PNs, a forbidden state controller for Timed-Arc Petri Nets is designed so as to make the system avoid undesired states (see, Chapter 5). Algorithms for obtaining the reachability set and designing a forbidden state controller for Timed-Arc PNs are developed and simulated using MATLAB (see, Chapter 7). In addition; examples of modeling and designing for real-world systems, such as manufacturing, railway, and automotive are carried out using the proposed approach (see, Chapter 6). Furthermore, the proposed Timed-Arc PN is a new model for Timed PNs. In order to evaluate performance of the proposed model, the methodology of Stretched PNs [11-14, 46, 47] is considered. Stretched PN is another type of Timed PNs, which is used to overcome the same problem of Timed PNs (see, Chapter 4). The following subsections present the proposed method of Timed-Arc PNs. Yufka et.al. have presented this novel

type of Timed PNs, where time delays are assigned to arcs and interpreted as firing durations, in [1, 28-31].

This chapter presents a new mathematical modeling method for Timed-Arc PNs with firing durations. The proposed model can also be used for obtaining the reachability set of Timed PNs with firing durations. When a Timed PN with firing durations is used in examples, this is called the original model of the Timed PN before transforming it into the proposed model of the Timed-Arc PN with firing durations. The following sections present the mathematical model and the graphical representation of the proposed Timed-Arc PNs with its behavioral properties.

3.1. Mathematical Model and Definition of Timed-Arc Petri Nets

The proposed Timed-Arc Petri Net (TdAPN) is defined as $G_A(P, T, N, O, D, S_0)$. Here; the set of places is represented by P; the set of transitions is represented by T; the input matrix is denoted by N; the output matrix is denoted by O; the time delay matrix is denoted by D; and the initial state of TdAPN is denoted by S_0. D and S_0 will be explained in detail in the following paragraphs. Time delays in G_A are considered exact (deterministic) and are expressed in terms of *time slots* (*ts*). These are associated with arcs rather than transitions or places. Time delays are interpreted as the firing durations that is related to ***flowing tokens*** of a firing process.

Assumption 3.1: *Time Delay of the Ingoing Arcs* - In TdAPN, an event occurred at the present time affects the net at the next time (after one *ts* is elapsed); therefore, in this thesis, time delays for all ingoing arcs are equal to one *ts*.

Based on *Assumption 3.1*, if the time delay of an event is considered to be equal to $\mathfrak{d}_t$ *ts*, then $\mathfrak{d}_t - 1$ *ts* time delay can be associated with the outgoing arcs in the representation of TdAPNs while one *ts* is assigned to ingoing arcs. Thus, the time delays of the outgoing arcs is expressed in terms of time slots and in a matrix form as $D: P \times T \rightarrow \mathbb{N}$, namely the time delay matrix. An element $D(p, t)$ of this matrix denotes the time delay of an outgoing arc from the transition $t \in T$ to the place $p \in P$. The time delay of an outgoing arc is greater than or equal to zero *ts*. Moreover, the set of input places connected to the transition $t \in T$ is represented by $\bullet t = \{p \in P | N(p, t) \neq 0\}$, and

the set of output places connected to the transition $t \in T$ is represented by $t\bullet = \{p \in P | O(p,t) \neq 0\}$.

In G_A, time elements are used to describe flowing tokens, and they are attached to outgoing arcs. A time element attached to an outgoing arc from the transition $t \in T$ to the corresponding place $p \in t\bullet$ is denoted by h_p^t. The cardinality of time elements is equal to the number of nonzero elements of the time delay matrix D. Each time element is associated with its outgoing arc. All these elements are shown as $\nabla := \{h_p^t | O(p,t) \geq 1 \text{ and } D(p,t) \geq 1\}$, namely the set of time elements. However, some outgoing arcs may have zero-time delay even if their weights are greater than one; as a result, time elements for zero-time delayed outgoing arcs are not used in the state of TdAPN in order to avoid operation confusion. The set ∇ is disjoint from both the set of places and transitions. The state of TdAPN is represented by $S(k) := \{\boldsymbol{M}(k), \boldsymbol{\nabla^R}(k)\}$. Here, $k \in \mathbb{N}$ represents the discrete time variable that is discretized into *time slots* (*ts*) using an appropriate sampling period. The status of tokens at places at time k is represented by the marking vector $\boldsymbol{M}(k): P \to \mathbb{N}$, and the status of flowing tokens at time elements at time k is represented by the remaining time vector $\boldsymbol{\nabla^R}(k): \nabla \to \mathbb{N}$ in terms of time slots. Note that its initial state is indicated as $S(k_0) := \{\boldsymbol{M}(k_0), \boldsymbol{\nabla^R}(k_0)\}$, i.e., $S_0 := \{\boldsymbol{M}_0, \boldsymbol{\nabla^{R_0}}\}$, where $\boldsymbol{M}(k_0) = \boldsymbol{M}_0$ and $\boldsymbol{\nabla^R}(k_0) = \boldsymbol{\nabla^{R_0}}$. In addition, in this thesis, $M(k,p) \in \mathbb{N}$ of $\boldsymbol{M}(k)$ is used to represent the number of tokens at the place $p \in P$ at time k, and $\nabla^R(k, h_p^t) \in \mathbb{N}$ of $\boldsymbol{\nabla^R}(k)$ is used to represent the remaining time of flowing tokens at the time element $h_p^t \in \nabla$ at time k.

On the representation of Timed PN with firing durations, the time delay $\eth_t$ is shown below the bar of the transition t as illustrated in Figure 3.1.(a); in addition, flowing tokens are not monitored during the firing process of the transition t. In TdAPN, $\boldsymbol{\nabla^R}(k)$ is used as a mathematical representation for flowing tokens. In order to depict this mathematical representation of the time element in the **graphical representation** of PNs, a triangular-formed component is introduced in the representation of TdAPNs. Thus, in addition to mathematical sense, flowing tokens are monitored over this new component in the graphical sense. A time element h_p^t is graphically depicted as in Figure 3.1.(b), which is the representation of TdAPN for the representation of Timed PN in Figure 3.1.(a).

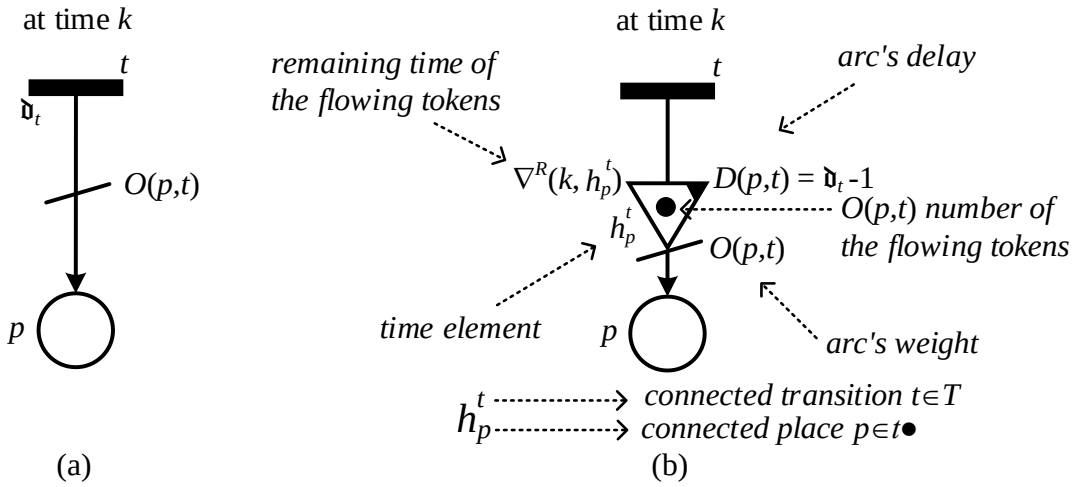


Figure 3.1. *Representations of (a) Timed PN with firing durations and (b) TdAPN*

Figure 3.1.(b) explains parts of the time element h_p^t with its external connections and gives mathematical entities related to this time element. The time element consists of four parts as follows:

- *The right corner with filled indicator* indicates the time delay of the outgoing arc that is $D(p,t)$. A filled triangular indicator is placed at the inner corner of the time element h_p^t, so as to prevent any confusion.

  In order to emphasize zero-time delayed time elements, the inside of the time element is filled with gray color. This gray-representation is only used for indicating the time delay and weight of an outgoing arc, and is not used for showing flowing tokens.
- *The left corner without filled indicator* indicates the remaining time of flowing tokens at the time element h_p^t at time k that is $\nabla^R\left(k, h_p^t\right)$. This remaining time is indicated next to the corner as long as $\nabla^R\left(k, h_p^t\right)$ is greater than zero *ts*.
- *The dots inside of the triangle* indicates flowing tokens (shown as dots in the time element h_p^t in Figure 3.1.(b)). Flowing tokens reside in the time element h_p^t as long as the remaining time is in the range of time interval as $1\ ts \leq \nabla^R\left(k, h_p^t\right) \leq D(p,t)\ ts$. The number of flowing tokens is equal to the weight of the outgoing arc, i.e., $O(p,t)$.
- *The line at the end of the triangle* indicates the weight of the outgoing arc that is $O(p,t)$. There is no need to indicate the middle line as long as $O(p,t)$ is equal to one.

Let us consider the time element, such as $h_{p_1}^{t_1}$ attached onto the outgoing arc from t_1 to p_1. The graphical representation of $h_{p_1}^{t_1}$ is shown in Figure 3.2.

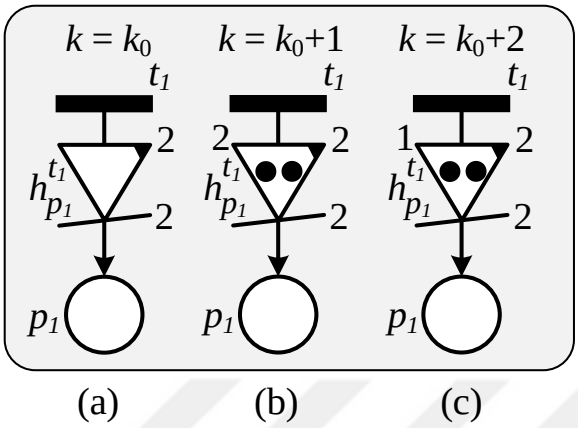


Figure 3.2. *Representations of the time element*

For Figure 3.2.(a)-(c); the time element $h_{p_1}^{t_1} \in \nabla$ has a time delay of $D(p_1, t_1) = 2\ ts$ and is weighted by $O(p_1, t_1) = 2$. Figure 3.2.(a) indicates that no flowing token is associated with $h_{p_1}^{t_1}$. However, in Figure 3.2.(b) and (c), two flowing tokens appear in $h_{p_1}^{t_1}$, due to $1\ ts \leq \nabla^R\left(k, h_{p_1}^{t_1}\right) \leq 2\ ts$. ◊

Let us consider some variations of graphical representations for time elements, such as $h_{p_2}^{t_2}$, $h_{p_3}^{t_3}$ and a time element attached onto a zero-time delayed outgoing arc. These are shown in Figure 3.3.

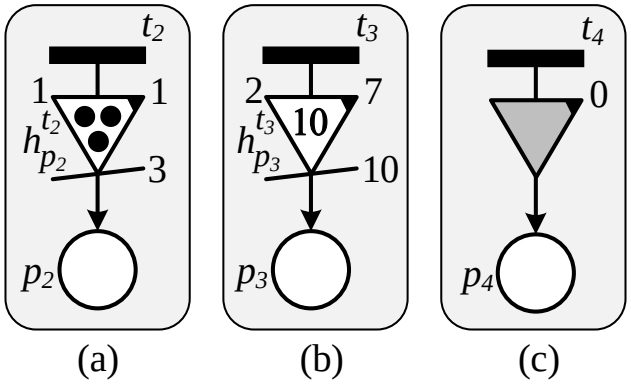


Figure 3.3. *Representation of time elements*

Here in Figure 3.3.(a) and (b), time elements $h_{p_2}^{t_2}$ and $h_{p_3}^{t_3}$ are members of the set ∇ due to $D(p,t) \neq 0\ ts$. The time element in Figure 3.3.(c) is not considered in the set ∇ and the state of G_A due to $O(p,t) \neq 0$ and $D(p,t) = 0\ ts$. Let us explain these graphical representations as follows:

- For Figure 3.3.(a);the time element $h_{p_2}^{t_2} \in \nabla$ has a time delay of $D(p_2, t_2)$ = 1 ts and is weighted by $O(p_2, t_2)$ = 3. Three flowing tokens reside in $h_{p_2}^{t_2}$. These appear in the output place $p_2 \in t_2\bullet$ after $\nabla^R\left(k, h_{p_2}^{t_2}\right) = 1\ ts$ is elapsed.
- For Figure 3.3.(b); similarly, the time element $h_{p_3}^{t_3} \in \nabla$ has a time delay of $D(p_3, t_3) = 7\ ts$ and is weighted by $O(p_3, t_3) = 10$. Ten flowing tokens reside in $h_{p_3}^{t_3}$. These appear in the output place $p_3 \in t_3\bullet$ after $\nabla^R\left(k, h_{p_3}^{t_3}\right) = 2\ ts$ is elapsed. Here, during the time element $h_{p_3}^{t_3}$, the number of flowing tokens is illustrated as a text instead of dots because of the limited area.
- For Figure 3.3.(c); the time element attached onto a zero-time delayed outgoing arc has a zero-time delay as $D(p_4, t_4) = 0\ ts$ and is weighted by $O(p_4, t_4)$=1. This type of time element is only used to indicate the time delay and weight of the outgoing arc. It is neutral and not a member of the set ∇; as a result, its status is not presented in the remaining time vector $\boldsymbol{\nabla^R}(k)$. ◊

It is important to monitor flowing tokens in Timed PNs because a complete picture of reachable states of time-delayed systems is required in many practical systems. The proposed graphical representation of TdAPN can transform any original Timed PN into a tripartite graph including places, transitions, and time elements. The tripartite structure of TdAPN allows the net to start at any state and any initial-time instant. This also allows determining all situations of tokens (some of them are stationary at places, and the rest of them flow over time elements), which indicates all situations of states for a deterministic timed-delay system. A time element physically indicates a continuing operation related to an event in practice.

3.2. Enabledness and Next State

This section presents the enabledness rule of TdAPN based on the firing process of TdAPN and the computation of the next state of TdAPN.

In TdAPN, a firing process related to an enabled transition t fired at time $k = \lambda$ is represented by t^{λ}. Firing processes are considered in two groups based on their starting time as *previously activated firing process* and *newly activated firing process*:

- The set of firing processes of TdAPN activated before time k and not finished yet, namely *previously activated firing process*, is represented by $F_{pre}(k) \coloneqq \{t^{\lambda} | \lambda < k < \lambda + d_t\}$.
- The set of firing processes of TdAPN started at time $k = \lambda$, namely *newly activated firing process*, is represented as $F_{start}(k) \coloneqq \{t^{\lambda} | k = \lambda\}$.

The set of all activated firing processes, i.e., $F_{pre}(k) \cup F_{start}(k)$, of TdAPN at time k is represented by $F(k) := \{t^{\lambda} | \lambda \leq k < \lambda + d_t, t \in T\}$. Here, $d_t \in \mathbb{N}\backslash\{0\}$ is called total duration of the firing process t^{λ} in terms of *ts*. It is determined by $d_t := \max_{p \in t\bullet}\{1 + D(p,t)\}$, which is the sum of the maximum time delay among ingoing arcs that is one *ts* and the maximum time delay among outgoing arcs.

Assumption 3.2: *Only One Continuing Firing Process Related to the Same Transition* - In TdAPN, the same transition is allowed to fire once at the same time because a physical system, such as a machine, is mostly unavailable while an event related to this system is in progress. Thus, in this thesis, it is assumed that the transition t is not enabled at time k while its firing process $t^{\lambda} \in F_{pre}(k)$ continues.

In TdAPN, a firing process t^{λ} is expressed in three parts in terms of starting time-instant, ending time-instant for an output place $p \in t\bullet$ and ending time-instant for the firing process t^{λ} as follows:

- <u>*Starting time-instant*</u> is a time instant represented by λ, where the transition t fires and its firing process begins at time $k = \lambda$.

  At time $k = \lambda$; the firing process t^{λ} is added into the set of newly activated firing processes, i.e., $F_{start}(\lambda) \subseteq F(\lambda)$ if d_t is greater than one *ts*. Otherwise (d_t is equal

to one *ts*), t^{λ} is not considered in the set $F_{start}(\lambda)$ because the time delay of all outgoing arcs connected to the transition t is equal to zero *ts*.

At time $k = \lambda + 1$; $N(p,t)$ number of tokens leave all input places $p \in \bullet t$, and $O(p,t)$ number of flowing tokens appear in corresponding time elements of the set ∇ that are connected to this transition. Otherwise, these tokens directly appear in the corresponding output place $p \in t\bullet$.

- *Ending time-instant for an output place $p \in t\bullet$* is a time instant that is indicated by $\lambda + \big(1 + D(p,t)\big)$. Note that one *ts* comes from the maximum time delay among ingoing arcs. At this time instant; flowing tokens, which transit via the time element $h_p^t \in \nabla$, finishes their transition, and $O(p,t)$ number of flowing tokens appear in the output place $p \in t\bullet$.
- *Ending time-instant for the firing process t^{λ}* is a time instant, when the firing process t^{λ} ends completely at time $k = \lambda + d_t$; as a result, t^{λ} is removed from $F(\lambda + d_t)$.

In TdAPN, a transition $t \in T$ is enabled at time k at the marking vector $\boldsymbol{M}(k)$ if and only if t satisfies the condition in (3.1) at time $k = \lambda$, $t^{\lambda} \notin F_{pre}(k)$ (see, *Assumption 3.2*).

$$M(k,p) \geq N(p,t) \geq 1, \quad \forall p \in \bullet t \tag{3.1}$$

The set of enabled transitions at time k is denoted by $E(G_A,k)$. More than one transition $t \in E(G_A,k)$ can be simultaneously enabled at time k. A set of transitions $\phi \subseteq E(G_A,k)$ is simultaneously enabled at time k at the marking vector $\boldsymbol{M}(k)$ if and only if the following condition is satisfied as:

$$M(k,p) \geq \sum_{t \in \phi} N(p,t), \qquad \forall p \in \mathrm{P} \tag{3.2}$$

Here, the set of simultaneously-enabled transitions, ϕ, contains more than one transition, but it can also contain a single transition. Any set ϕ, which satisfies the condition in (3.2), is added into the set $\hat{E}(G_A,k) \subset 2^{E(G_A,k)} \backslash \emptyset$ that represents the set of sets of simultaneously-enabled transitions at time k at the marking vector $\boldsymbol{M}(k)$. Here, $2^{\{.\}}$ denotes the power set of $\{.\}$. The set $\phi \in \hat{E}(G_A,k)$ can be selected and simultaneously fired at any time k as long as it is enabled at time k at $\boldsymbol{M}(k)$. In this thesis, in order to

obtain a state in a minimum time, a transition in the selected set ϕ fires at time k as soon as it is enabled at time k at $\boldsymbol{M}(k)$.

The next state of TdAPN is represented by $S(k+1) = \{\boldsymbol{M}(k+1), \boldsymbol{\nabla^R}(k+1)\}$:

$$M(k+1,p) \coloneqq M(k,p) + \sum_{t^{\lambda} \in F(k)} \left(\delta\left[k-\left(\lambda + D(p,t)\right)\right].O(p,t) - \delta[k-\lambda].N(p,t)\right) \quad (3.3)$$

$$\nabla^R\left(k+1, h_p^t\right) \coloneqq \nabla^R\left(k, h_p^t\right) + \sum_{t^{\lambda} \in F(k)} \left(\delta[k-\lambda].D(p,t) - \sum_{l=\lambda+1}^{\lambda+D(p,t)} \delta[k-l]\right) \quad (3.4)$$

, where $M(k+1,p)$ is the next marking vector at time $k+1$ for the place $p \in P$, and $\nabla^R\left(k+1, h_p^t\right)$ is the next remaining time vector at time $k+1$ for the time element $h_p^t \in \nabla$. Here in the above equations, $\delta[k] \in \{0,1\}$, $k \in \mathbb{N}$, denotes the discrete-time unit impulse function. When enabled transitions of the set $\phi \in \hat{E}(G_A, k)$ fire at time k, firing processes of these transitions start at time k and are added into the set $F_{start}(k)$. When all newly and previously activated firing processes in the set $F(k)$ are considered at time k, $S(k+1)$ is computed using (3.3) and (3.4).

The tripartite structure of TdAPN allows the net to start at any state and any initial-time instant. Using equations (3.3) and (3.4) for computing the next state of TdAPN, it is possible to find next state including both next marking and remaining time vectors for time $k_0 \neq 0$. As a result, it is possible to find all reachable states of TdAPN starting at time $k = k_0 \neq 0$.

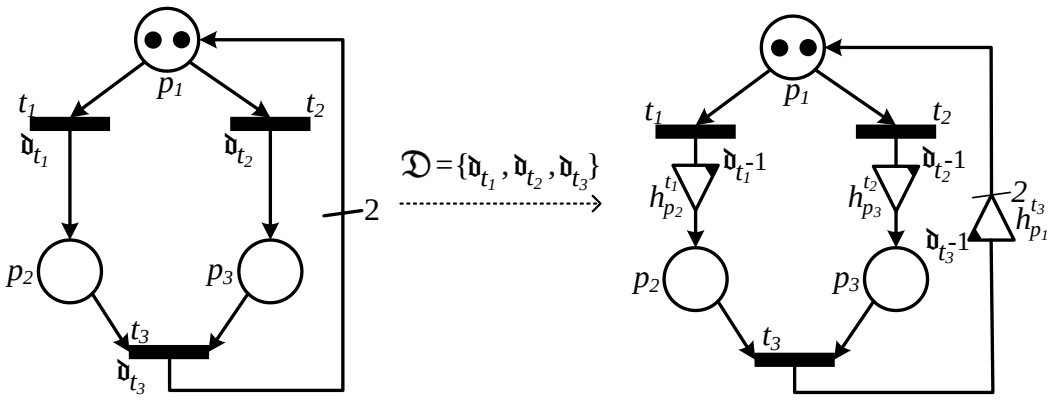


Figure 3.4. *Representation of TdAPN for the original Timed PN*

Let us consider the original Timed PN in Figure 3.4.(a). The set of places is $P = \{p_1, p_2, p_3\}$. The set of transitions is $T = \{t_1, t_2, t_3\}$. Input and output matrices are $N = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$ and $O = \begin{bmatrix} 0 & 0 & 2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$, respectively. Time delays associated with transitions are shown by the set of time delays, i.e., $\mathfrak{D} = \{\mathfrak{d}_{t_1}, \mathfrak{d}_{t_2}, \mathfrak{d}_{t_3}\}$. These time delays are arbitrarily chosen as $\mathfrak{d}_{t_1} = 3$ *ts*, $\mathfrak{d}_{t_2} = 2$ *ts* and $\mathfrak{d}_{t_3} = 4$ *ts* for this example. Let us convert the original Timed PN in Figure 3.4.(a) into a tripartite graph of TdAPN including places, transitions, and time elements. This tripartite structure allows monitoring all situations of tokens, especially flowing tokens. We obtain the representation of TdAPN for this original Timed PN as shown in Figure 3.4.(b). The tuple of TdAPN is defined as $G_A(P, T, N, O, D, S_0)$ including time elements. Here, the time-delay matrix is expressed as follows:

$$D = \begin{bmatrix} 0 & 0 & \mathfrak{d}_{t_3} - 1 \\ \mathfrak{d}_{t_1} - 1 & 0 & 0 \\ 0 & \mathfrak{d}_{t_2} - 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 3 \\ 2 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

The initial state of TdAPN is $S(k_0) = \{\boldsymbol{M}(k_0), \boldsymbol{\nabla^R}(k_0)\} = \{[2\ 0\ 0]', [0\ 0\ 0]'\}$. In G_A, the set of time elements is $\nabla = \left\{h_{p_2}^{t_1}, h_{p_3}^{t_2}, h_{p_1}^{t_3}\right\}$, where $O(p,t) \neq 0$ and $D(p,t) \neq 0$. In $S(k_0)$, $\boldsymbol{\nabla^R}(k_0)$ is a zeros vector, such that there is no previously activated firing process t^λ. This yields $F_{pre}(k_0) = \emptyset$. Using conditions of enabledness in (3.1) and (3.2), the set of enabled transitions is determined as $E(G_A, k_0) = \{t_1, t_2\}$, and the set of sets of simultaneously-enabled transitions is determined as $\hat{E}(G_A, k_0) = \left\{\{t_1\}, \{t_2\}, \{t_1, t_2\}\right\}$. The set of simultaneously-enabled transitions can be selected as $\phi = \{t_1\}$, $\phi = \{t_2\}$ or $\phi = \{t_1, t_2\}$ starting at time $k = k_0$. It is also possible not to select any ϕ at time $k = k_0$; therefore, time variable k will increase by one *ts* such that k will be equal to $k_0 + 1$. Moreover, $F_{start}(k_0)$ is an empty set as long as any enabled set in $\hat{E}(G_A, k_0)$ is selected at $S(k_0)$. In this thesis, a transition in the selected set $\phi \in \hat{E}(G_A, k_0)$ fires at time k_0 as soon as it is enabled at time k_0. If the enabled set $\{t_1\} \in \hat{E}(G_A, k_0)$ is selected and fired at time k_0, the set of active-firing transitions at $S(k_0)$ is $F(k_0) = \left\{t_1^\lambda\right\}$, where $F_{start}(k_0) = \left\{t_1^\lambda\right\}$ and $\lambda = k_0$. Similarly, for the set $\{t_2\} \in \hat{E}(G_A, k_0)$, the set of active-firing transitions is $F(k_0) = \left\{t_2^\lambda\right\}$. For the set $\{t_1, t_2\} \in \hat{E}(G_A, k_0)$, the set of active-firing transitions is $F(k_0) = \left\{t_1^\lambda, t_2^\lambda\right\}$. ◊

Suppose that the transition $t_1 \in \hat{E}(G_A, \lambda)$ is selected to fire at time $k = \lambda$ such that the firing process t_1^λ starts at time λ, and it is added into the set $F_{start}(\lambda) = \{t_1^\lambda\}$, $F_{start}(\lambda) \subseteq F(\lambda)$. Let us monitor calculations of discrete-time unit impulse functions and state evolutions both in places and time elements for the firing process t_1^λ. Time instants for the firing process t_1^λ are as follows:

- The firing time of the transition t_1 is $k = \lambda$, i.e., the starting time is λ for the firing process t_1^λ.
- The ending time for $p_2 \in t_1\bullet$ is $\lambda + \big(1 + D(p_2, t_1)\big) = \lambda + 3$, where $t_1\bullet = \{p_2\}$ and $D(p_2, t_1) = 2\ ts$. This is also the ending time for t_1^λ that is $\lambda + d_{t_1} = \lambda + 3$.

As a result of t_1^λ, next marking vectors are computed using (3.3) and discrete-time unit impulse functions as given in Table 3.1. The first column denotes the discrete-time k. The second column gives the present marking vector $\boldsymbol{M}(k)$, and its next marking vector $\boldsymbol{M}(k+1)$ is given in the last column. The third column indicates the activated firing process $t_1^\lambda \in F(k)$. The fourth column represents calculations of discrete-time impulse functions for the output side, while the fifth column denotes calculations of discrete-time impulse functions for the input side.

Table 3.1. *Computing* $\boldsymbol{M}(k+1)$ *using impulse functions when* t_1 *fires at* $k=\lambda$

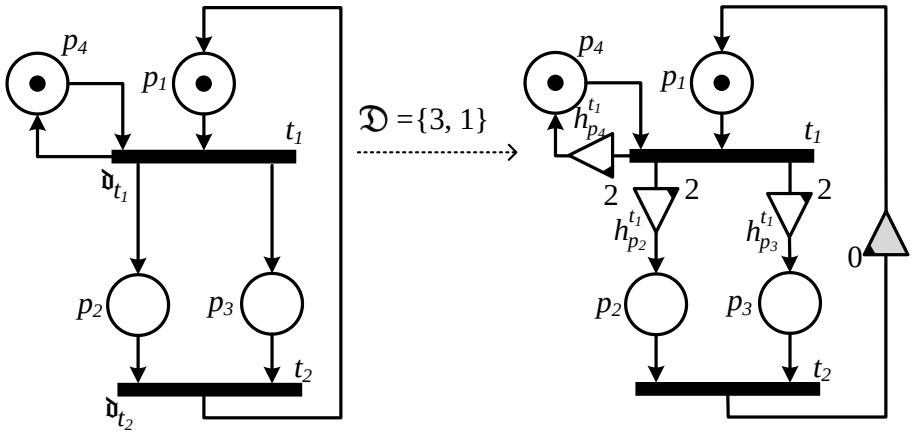
k	$\boldsymbol{M}(k)$	$F(k)$	**Output Side** $\delta\big[k-\big(\lambda+D(p,t)\big)\big].O(p,t)$	**Input Side** $-\delta[k-\lambda].N(p,t)$	**Next Marking** $\boldsymbol{M}(k+1)$
-	$\begin{bmatrix}2\\0\\0\end{bmatrix}$	$\emptyset$	-	-	$\begin{bmatrix}2\\0\\0\end{bmatrix}$
λ	$\begin{bmatrix}2\\0\\0\end{bmatrix}$	$\{t_1^\lambda\}$	$\begin{bmatrix}\delta[k-(\lambda+3)].0\\ \delta[k-(\lambda+3)].1\\ \delta[k-(\lambda+3)].0\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$	$-\begin{bmatrix}\delta[k-\lambda].1\\ \delta[k-\lambda].0\\ \delta[k-\lambda].0\end{bmatrix} = -\begin{bmatrix}1\\0\\0\end{bmatrix}$	$\begin{bmatrix}1\\0\\0\end{bmatrix}$
$\lambda+1$	$\begin{bmatrix}1\\0\\0\end{bmatrix}$	$\{t_1^\lambda\}$	$\begin{bmatrix}\delta[k-(\lambda+3)].0\\ \delta[k-(\lambda+3)].1\\ \delta[k-(\lambda+3)].0\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$	$-\begin{bmatrix}\delta[k-\lambda].1\\ \delta[k-\lambda].0\\ \delta[k-\lambda].0\end{bmatrix} = -\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}1\\0\\0\end{bmatrix}$
$\lambda+2$	$\begin{bmatrix}1\\0\\0\end{bmatrix}$	$\{t_1^\lambda\}$	$\begin{bmatrix}\delta[k-(\lambda+3)].0\\ \delta[k-(\lambda+3)].1\\ \delta[k-(\lambda+3)].0\end{bmatrix} = \begin{bmatrix}0\\1\\0\end{bmatrix}$	$-\begin{bmatrix}\delta[k-\lambda].1\\ \delta[k-\lambda].0\\ \delta[k-\lambda].0\end{bmatrix} = -\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}1\\1\\0\end{bmatrix}$
$\lambda+3$	$\begin{bmatrix}1\\1\\0\end{bmatrix}$	$\emptyset$	-	-	$\begin{bmatrix}1\\1\\0\end{bmatrix}$

In addition, as a result of t_1^λ, next remaining time vectors are computed using (3.4) and discrete-time unit impulse functions as given in Table 3.2. The first column denotes the discrete-time k. The second column gives the present remaining time vector $\boldsymbol{\nabla^R}(k)$ and its next remaining time vector $\boldsymbol{\nabla^R}(k+1)$ is given in the last column. The third column indicates the activated firing process $t_1^\lambda \in F(k)$. The fourth and fifth columns represent calculations of discrete-time impulse functions for the remaining time. Remember that this vector is defined as $\boldsymbol{\nabla^R}(k): \nabla \rightarrow \mathbb{N}$ and the set of time elements is $\nabla = \{h_{p_2}^{t_1}, h_{p_3}^{t_2}, h_{p_1}^{t_3}\}$. ◊

Table 3.2. *Computing $\nabla^R(k+1)$ using impulse functions when t_1 fires at $k=\lambda$*

k	$\boldsymbol{\nabla^R}(k)$	$F(k)$	**Delay Assignment** $\delta[k-\lambda].D(p,t)$	**Summation** $-\sum_{l=\lambda+1}^{\lambda+D(p,t)} \delta[k-l]$	**Next Remaining** $\boldsymbol{\nabla^R}(k+1)$
-	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\emptyset$	-	-	$\begin{bmatrix}0\\0\\0\end{bmatrix}$
λ	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\{t_1^\lambda\}$	$\begin{bmatrix}\delta[k-\lambda].2\\ \delta[k-\lambda].0\\ \delta[k-\lambda].0\end{bmatrix} = \begin{bmatrix}2\\0\\0\end{bmatrix}$	$-\begin{bmatrix}\sum_{l=\lambda+1}^{\lambda+2}\delta[k-l]\\ \sum_{l=\lambda+1}^{\lambda}\delta[k-l]\\ \sum_{l=\lambda+1}^{\lambda}\delta[k-l]\end{bmatrix} = -\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}2\\0\\0\end{bmatrix}$
$\lambda+1$	$\begin{bmatrix}2\\0\\0\end{bmatrix}$	$\{t_1^\lambda\}$	$\begin{bmatrix}\delta[k-\lambda].2\\ \delta[k-\lambda].0\\ \delta[k-\lambda].0\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$	$-\begin{bmatrix}\sum_{l=\lambda+1}^{\lambda+2}\delta[k-l]\\ \sum_{l=\lambda+1}^{\lambda}\delta[k-l]\\ \sum_{l=\lambda+1}^{\lambda}\delta[k-l]\end{bmatrix} = -\begin{bmatrix}1\\0\\0\end{bmatrix}$	$\begin{bmatrix}1\\0\\0\end{bmatrix}$
$\lambda+2$	$\begin{bmatrix}1\\0\\0\end{bmatrix}$	$\{t_1^\lambda\}$	$\begin{bmatrix}\delta[k-\lambda].2\\ \delta[k-\lambda].0\\ \delta[k-\lambda].0\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$	$-\begin{bmatrix}\sum_{l=\lambda+1}^{\lambda+2}\delta[k-l]\\ \sum_{l=\lambda+1}^{\lambda}\delta[k-l]\\ \sum_{l=\lambda+1}^{\lambda}\delta[k-l]\end{bmatrix} = -\begin{bmatrix}1\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$
$\lambda+3$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\{\emptyset\}$	-	-	$\begin{bmatrix}0\\0\\0\end{bmatrix}$

Let us consider another original Timed PN in Figure 3.5.(a). Its representation of TdAPN is given in Figure 3.5.(b), where time delays of the original Timed PN are arbitrarily chosen as $\mathfrak{d}_{t_1} = 3$ *ts* and $\mathfrak{d}_{t_2} = 1$ *ts*.



(a) *The Original Timed PN*

(b) *The Representation of TdAPN after Time-Delay Assignment*

Figure 3.5. *Representation of TdAPN for the original Timed PN in (a)*

The representation of TdAPN in Figure 3.5.(b) is expressed as $G_A(P, T, N, O, D, S_0)$ including time elements. Here, the set of places is $P = \{p_1, p_2, p_3, p_4\}$ and the set of transitions is $T = \{t_1, t_2\}$. The input matrix, the output matrix, and the time delay matrix are as follows, respectively:

$$N = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, O = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}, D = \begin{bmatrix} 0 & \mathfrak{d}_{t_2} - 1 \\ \mathfrak{d}_{t_1} - 1 & 0 \\ \mathfrak{d}_{t_1} - 1 & 0 \\ \mathfrak{d}_{t_1} - 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 2 & 0 \\ 2 & 0 \\ 2 & 0 \end{bmatrix}.$$

The initial state is $S(k_0) = \{[0\ 1\ 0\ 1]', [0\ 0\ 0]'\}$. For TdAPN in Figure 3.5.(b), The set of time elements is $\nabla = \{h_{p_2}^{t_1}, h_{p_3}^{t_1}, h_{p_4}^{t_1}\}$. ◊

In Timed PNs, the time delay of a transition t is represented by $\mathfrak{d}_t$ *ts* as shown in Figure 3.5.(a), where $\mathfrak{d}_t$ denotes the total duration of an event related to t. In order to associate the time delay $\mathfrak{d}_t$ of the original Timed PN with the representation of TdAPN, one *ts* is assigned to all ingoing arcs of the transition t (see, *Assumption 3.1*) and $\mathfrak{d}_t - 1$ *ts* is directly assigned to all outgoing arcs of the transition t as shown in the representation of TdAPN in Figure 3.5.(b). However, TdAPN also allows assigning time delays distinctly as long as d_t of TdAPN is equal to $\mathfrak{d}_t$ of the original Timed PN. Time delays transferred to events can be differentiated at outgoing arcs of Timed-Arc PNs. This representation, where time delays are distinctly assigned to outgoing arcs that are connected to the same transition, is shown in Figure 3.6 as an alternative representation

of TdAPN. This representation is required in practice when different operations (in this case, output places together with corresponding outgoing arcs and time elements) connected to the same event (in this case, the transition) have different time delays.

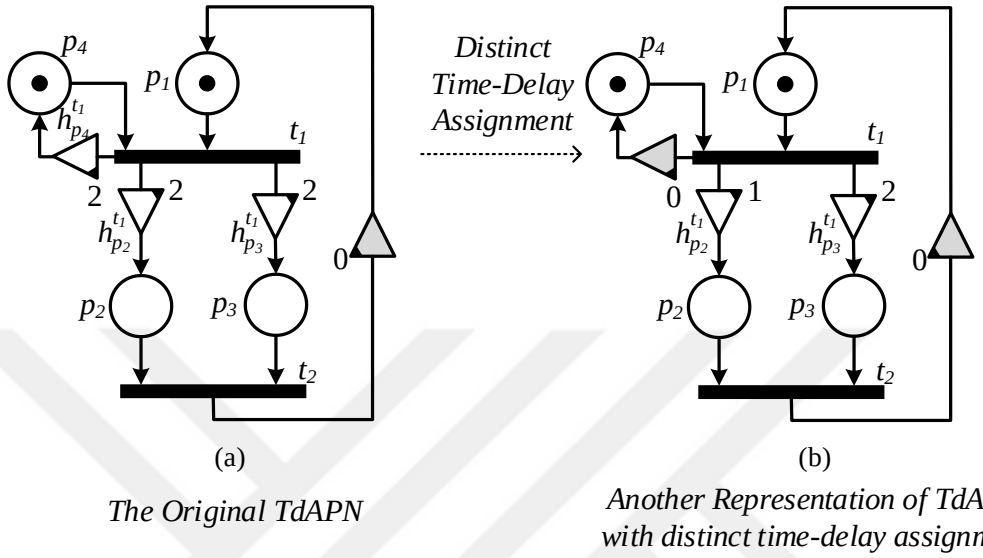


Figure 3.6. *Another representation of TdAPN with distinct time delays*

In order to indicate that time delays are also differently assigned to outgoing arcs in TdAPNs, let us change time delays of TdAPN in Figure 3.5.(b) that are distinctly assigned to outgoing arcs of the transition t_1. This representation of TdAPN is shown in Figure 3.6.(b). Here, $D(p, t_1)$ has different values for output places p_2 and p_4 of the set $t_1\bullet$, and the set of time elements is $\nabla = \{h_{p_2}^{t_1}, h_{p_3}^{t_1}\}$. Thus, the initial state of TdAPN in Figure 3.6.(b) becomes $S(k_0) = \{[0\ 1\ 0\ 1]', [0\ 0]'\}$, and the time delay matrix for this TdAPN is as follows:

$$D = \begin{bmatrix} 0 & \mathfrak{d}_{t_2} - 1 \\ \mathfrak{d}_{t_1} - 2 & 0 \\ \mathfrak{d}_{t_1} - 1 & 0 \\ \mathfrak{d}_{t_1} - 3 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 2 & 0 \\ 0 & 0 \end{bmatrix}.$$

The assignment of time delays to arcs instead of transitions allows differentiation of the token evolution at each output place $p \in t_1\bullet$. Therefore, faster outputs (p_2 and p_4 in this case) complete their token evolution without waiting for the slowest output (p_3 in this case). Moreover, it is also possible to set all outgoing arc's delays so that they are equal to a fixed duration of $\mathfrak{d}_{t_1} - 1\ ts$, where the condition of $\mathfrak{d}_{t_1} = d_{t_1}$ is satisfied (see, Figure

3.6.(a)). However, in this case, different time labels are selected by satisfying the condition of $d_{t_1} = \mathfrak{d}_{t_1} = 3\ ts$, where $d_{t_1} = \underset{p \in t_1 \bullet}{max}\{1 + D(p, t_1)\}$. ◊

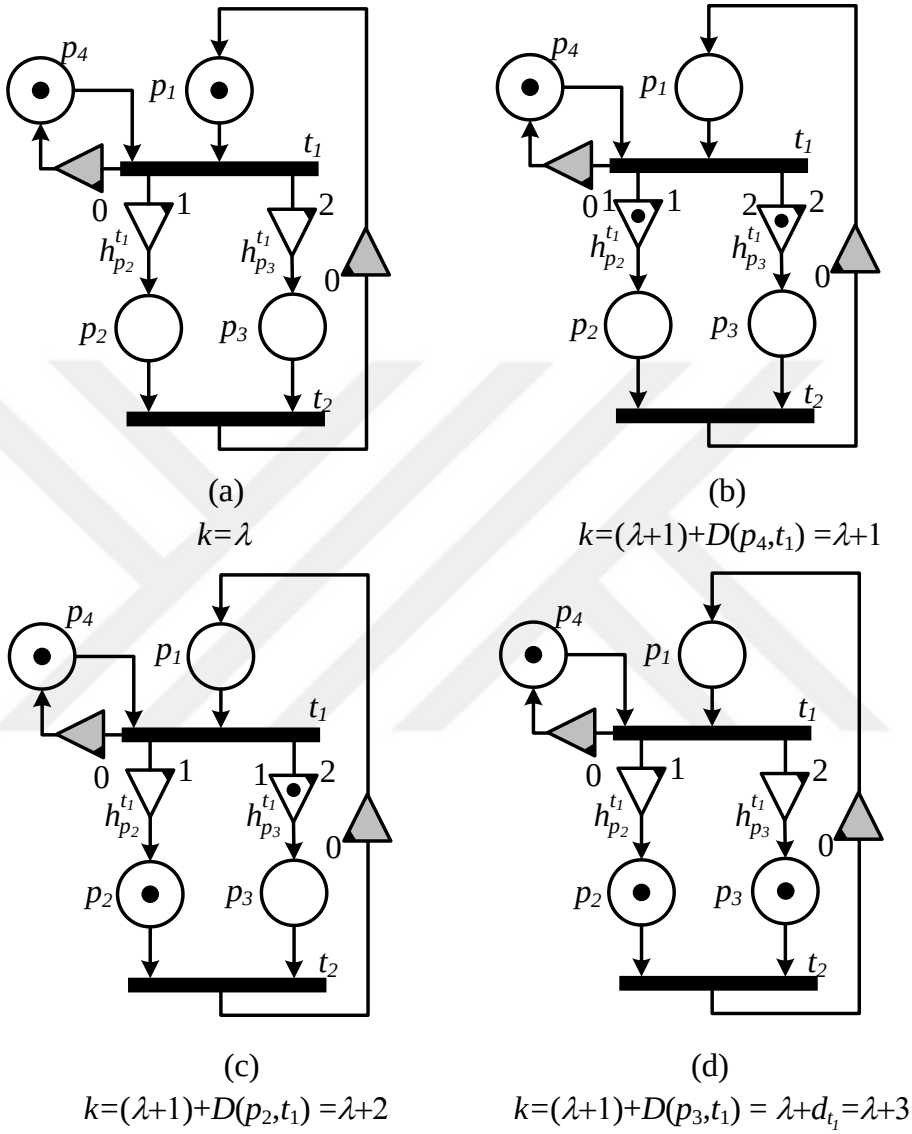


Figure 3.7. *Example firing process of t_1 for TdAPN in Figure 3.6.(b)*

Let us consider another representation of TdAPN in Figure 3.6.(b). This TdAPN is shown in Figure 3.7.(a). When the enabled transition t_1 of the set $\phi \in \hat{E}(G_A, k_0)$ fires at time $k = \lambda$,

- The firing process $t_1^{\lambda} \notin F_{pre}(\lambda)$ related to the transition t_0, where $F_{pre}(\lambda) = \emptyset$, starts at the starting time-instant $k = \lambda$, during which it is considered to be disabled; as a result, the transition t_1 is added into the set $F_{start}(\lambda) \subseteq F(\lambda)$, where $F_{start}(\lambda) = \{t_1^{\lambda}\}$ such that $F(\lambda)$ is equal to $F_{start}(\lambda)$. Note that the ending time-instant for t_1^{λ} is $\lambda + d_{t_1}$ that is $\lambda + 3$ (see, Figure 3.7.(a)).
- At time $k = (\lambda + 1) + D(p_4, t_1) = \lambda + 1$, $O(p_4, t_1) = 1$ number of tokens directly appear in $p_4 \in t_1\bullet$. In addition, $O(p_2, t_1) = 1$ number of tokens reside in the time element $h_{p_2}^{t_1}$. Similarly, $O(p_3, t_1) = 1$ number of tokens reside in the time element $h_{p_3}^{t_1}$. The firing process t_1^{λ} continues and it is in $F_{pre}(\lambda + 1) \subseteq F(\lambda + 1)$, where $F_{start}(\lambda + 1) = \emptyset$ and $F_{pre}(\lambda + 2) = \{t_1^{\lambda}\}$ such that $F(\lambda + 1)$ is equal to $F_{pre}(\lambda + 1)$ (see, Figure 3.7.(b)).
- At time $k = (\lambda + 1) + D(p_2, t_1) = \lambda + 2$, $O(p_2, t_1) = 1$ number of flowing tokens at $h_{p_2}^{t_1}$ disappear, and these tokens appear in $p_2 \in t_1\bullet$. The firing process t_1^{λ} continues and it is in $F_{pre}(\lambda + 2) \subseteq F(\lambda + 2)$, where $F_{start}(\lambda + 2) = \emptyset$ and $F_{pre}(\lambda + 2) = \{t_1^{\lambda}\}$ such that $F(\lambda + 2) = F_{pre}(\lambda + 2)$ (see, Figure 3.7.(c)).
- At time $k = (\lambda + 1) + D(p_3, t_1) = \lambda + 3$, $O(p_3, t_1) = 1$ number of flowing tokens at $h_{p_3}^{t_1}$ disappear, and these tokens appear in $p_3 \in t_1\bullet$. This time instant is also the ending time-instant for t_1. Thus, the firing process t_1^{λ} ends, and the transition t_1 can be reconsidered for enabledness rule. t_1^{λ} is not in $F_{pre}(\lambda + 3) \subseteq F(\lambda + 3)$ anymore, where $F_{pre}(\lambda + 3) = \emptyset$ (see, Figure 3.7.(d)). ◊

3.3. Behavioral Properties of Timed-Arc Petri Nets

In TdAPN, the state of G_A at time k is previously defined as $S(k) = \{\boldsymbol{M}(k), \boldsymbol{\nabla^R}(k)\}$. This notation is used in online computations of states for G_A. The set of all reachable states from the initial state S_0, namely ***reachability set***, is denoted by $R_S(G_A, S_0) = \{S_0, S_1, \ldots\}$ (e.g., see, Table 3.3). Here, each obtained state during computations is represented by $S_j := \{\boldsymbol{M}_j, \boldsymbol{\nabla^{R_j}}\}$ without any k notation, where $j = 0,1, \ldots, |R_S(G_A, S_0)| - 1$. In brief, the state of G_A without any k notation is represented as $S = \{\boldsymbol{M}, \boldsymbol{\nabla^R}\}$. For any $S \in R_S(G_A, S_0)$, it is possible to find parts, such as $\boldsymbol{M} = \gamma_1(S)$ and $\boldsymbol{\nabla^R} = \gamma_2(S)$. Here, the function $\gamma_i(S)$ gives the i 'th part of S.

Assumption 3.3: *Reachability set for the bounded TdAPNs* - In this thesis, it is assumed that all considered PNs are bounded (see, Definition 3.2 for the definition of boundedness).

The relation between elements of the set $R_S(G_A, S_0)$ can be represented by a graph, namely ***timed-reachability graph/tree***. Timed-reachability tree, which is enhanced by the time information, depicts the complete dynamic picture of time-delayed systems (see, Figure 3.8). Each arrow in the timed-reachability graph indicates one *ts* (see, Figure 3.8). Thus, it is possible to find the minimum time to reach from any state to any state. The reachability set is constructed for the given TdAPN during online computations by using the algorithms given in Chapter 7. Moreover, note that $\boldsymbol{\nabla^R} = \gamma_2(S)$ includes the remaining-time (duration) information about flowing tokens, which is independent of discrete-time notation k. States in $R_S(G_A, S_0)$ are used for offline computations without any k notation.

In TdAPN, the set of enabled transitions at $\gamma_1(S)$ is represented by $E\big(G_A, \gamma_1(S)\big)$. A transition $t \in T$ is enabled at $\gamma_1(S)$ if and only if it satisfies the following condition as:

$$\gamma_1(S)(p) \geq N(p,t) \geq 1, \quad \forall p \in {\bullet}t\,, t \notin F_{pre}(S) \tag{3.5}$$

Here, $\gamma_1(S)(p) \in \mathbb{N}$ ($\boldsymbol{M} = \gamma_1(S)$) denotes the number of tokens at the place $p \in P$, and $F_{pre}(S)$ is determined according to the time duration information in $\gamma_2(S)$ and is defined as follows:

$$F_{pre}(S) \coloneqq \big\{t | \boldsymbol{\nabla^R} = \gamma_2(S)\big(h_p^t\big) \neq 0\big\} \tag{3.6}$$

Here, $\gamma_2(S)\big(h_p^t\big) \in \mathbb{N}$ ($\boldsymbol{\nabla^R} = \gamma_2(S)$) denotes the remaining time of flowing tokens at the time element $h_p^t \in \nabla$. $F_{pre}(S)$ represents the set of transitions whose firing processes activated previously and not finished yet. An analogy between $F_{pre}(S)$ and $F_{pre}(k)$ can be established. While $F_{pre}(k)$ is the time-counterpart of $F_{pre}(S)$ and contains previously activated firing processes t^λ, $F_{pre}(S)$ shows previously activated transitions for the state $S \in R_S(G_A, S_0)$ according to the remaining time information in the remaining-time vector $\boldsymbol{\nabla^R} = \gamma_2(S)$. In addition, $\hat{E}\big(G_A, \gamma_1(S)\big) \subset 2^{E\left(G_A, \gamma_1(S)\right)} \backslash \emptyset$ is used for representing the set of sets of simultaneously-enabled transitions at $\gamma_1(S)$. The set of simultaneously enabled

transitions satisfies the condition $\gamma_1(S)(p) \geq \sum_{t\in\phi} N(p,t)$ for all $p \in$ P, where $\phi \subseteq E\big(G_A, \gamma_1(S)\big)$. This condition is similar to (3.2) without k notation.

In order to permit analysis of the proposed TdAPN, this section presents behavioral properties of TdAPN. Behavioral properties of the basic PN, which are boundedness and safeness, liveness, deadlock, and reversibility, are adapted to TdAPNs. Moreover, a new behavioral property, such as *dynamicity*, is defined for TdAPNs as given in following definitions.

Definition 3.1: *Dynamicity* - Typically, states of G_A are divided into two types, such as relaxed states and dynamic states. A state $S \in R_S(G_A, S_0)$ is called a ***relaxed state*** if $\gamma_2(S) = \mathbf{0}^{|\nabla|\times 1}$, where $\mathbf{0}^{|\nabla|\times 1}$ represents a $|\nabla|$ by 1 sized zeros vector. Relaxed states preserve their status and do not yield another state in $R_S(G_A, S_0)$ until any enabled set $\phi \in \hat{E}(G_A, \gamma_1(S))$ is selected (e.g., see, Table 3.3: S_0, S_6, S_9, S_{11}, S_{15}, and S_{18}). On the other hand, a state $S \in R_S(G_A, S_0)$ is called a ***dynamic state*** if $\gamma_2(S) \neq \mathbf{0}^{|\nabla|\times 1}$ (e.g., see, Table 3.3: S_1, S_2, S_3, S_4, S_5, S_7, S_8, S_{10}, S_{12}, S_{13}, S_{14}, S_{16}, S_{17}, and S_{19}). Dynamic states indicate that at least one firing process of a transition is active ($F_{pre}(S) \neq \emptyset$); as a result, an event related to this firing process is still in progress. Moreover, they indicate the status and existence of flowing tokens. Dynamic states do not preserve its status; in addition, they lead the system to another state in $R_S(G_A, S_0)$.

Definition 3.2: *Boundedness and Safeness* – The property of boundedness is defined through the marking vector $\gamma_1(S)$ and tokens at places. G_A is said to be $\boldsymbol{B}$ bounded if there exists a bound vector $\boldsymbol{B}: P \rightarrow \mathbb{N}$ such that:

$$\gamma_1(S)(p) \leq B(p), \forall p \in P, \forall S \in R_S(G_A, S_0) \tag{3.7}$$

Here, $B(p)$ is the p 'th element of the bound vector $\boldsymbol{B}$ and is determined by $B(p) := \max_{S\in R_S(G_A,S_0)} \{\gamma_1(S)(p)\}$. Moreover, G_A is said to be safe if $\boldsymbol{B} = \mathbf{1}^{|P|\times 1}$, where $\mathbf{1}^{|P|\times 1}$ represents a $|P|$ by 1 sized ones vector.

Definition 3.3: *Liveness* - A transition $t \in T$ is said to be live if, for all states $S_j \in R_S(G_A, S_0)$, there exists $S \in R_S(G_A, S_j)$ such that $t \in \phi$ while $\phi \in \hat{E}\big(G_A, \gamma_1(S)\big)$. All live transitions are considered in the set $\hat{T} \subseteq T$. Therefore, G_A is considered to be $\hat{T}$-live if $\hat{T} \subset T$, and G_A is considered to be live if $\hat{T} = T$, which is T-live.

Definition 3.4: *Deadlock* – A dynamic state yields a new state based on time durations in its remaining time vector so that dynamic states reach a relaxed state at the end. Thus, the deadlock property of G_A is examined over relaxed states. Any relaxed state $S \in R_S(G_A, S_0)$ is considered to be a deadlock state if $\gamma_2(S) = \mathbf{0}^{|\nabla|\times 1}$ and $\hat{E}\big(G_A, \gamma_1(S)\big) = \emptyset$ (e.g., see, Figure 3.8: S_{15} and S_{18} are deadlock states). Note that G_A is not live if it has any deadlock state. In TdAPN, the set of deadlock states is represented by $\tilde{\mathcal{L}}_0 \subseteq R_S(G_A, S_0)$ as follows:

$$\tilde{\mathcal{L}}_0 := \big\{ S = \{\gamma_1(S), \gamma_2(S)\} \mid \gamma_2(S) = \mathbf{0}^{|\nabla|\times 1} \text{ and } \hat{E}\big(G_A, \gamma_1(S)\big) = \emptyset \big\} \tag{3.8}$$

Definition 3.5: *Reversibility* - If the initial state $S_0 \in R_S(G_A, S)$ is reachable from all states $S \in R_S(G_A, S_0)$, then G_A is considered as reversible. Note that G_A is not reversible if it has any deadlock state.

Let us analyze behavioral properties of the TdAPN given in Figure 3.4.(b). For this TdAPN, the reachability set is obtained as $R_S(G_A, S_0) = \{S_0, S_1, \dots, S_{19}\}$. 20 states of TdAPN in Figure 3.4.(b) are obtained as given in Table 3.3. In this table, the minimum time to reach each state $S \in R_S(G_A, S_0)$ from the initial state S_0 is indicated in the first column. Remember that a transition in the selected set ϕ fires at time k as soon as it is enabled at time k at $\boldsymbol{M}(k)$; as a result, the minimum time to reach any state S from S_0 is obtained in terms of *ts*. S_0 is initially reachable such that the minimum time is considered zero *ts*. The state $S = \{\boldsymbol{M} = \gamma_1(S), \boldsymbol{\nabla^R} = \gamma_2(S)\}$ is indicated in the third column with its label S_j in the second column. The possible set of simultaneously-enabled transitions $\phi \in \hat{E}\big(G_A, \gamma_1(S)\big)$ is indicated in the fourth column. According to the selection of ϕ, the next state is indicated in the last column.

Table 3.3. *Reachability set for TdAPN in Figure 3.4.(b)*

k	**State-Label**	**State** $S \in R_S(G_A, S_0)$	**The Selected Set of Events**	**Next State**
	S_j	$\{\boldsymbol{M} = \gamma_1(S), \boldsymbol{\nabla^R} = \gamma_2(S)\}$	$\phi \in \hat{E}\left(G_A, \gamma_1(S_j)\right)$	
0	$^{\dagger}S_0$	{[2 0 0]', [0 0 0]'}	-	$^{\dagger}S_0$
			$\{t_1\}$	S_1
			$\{t_2\}$	S_2
			$\{t_1, t_2\}$	S_3
1	S_1	{[1 0 0]', [2 0 0]'}	-	S_4
			$\{t_2\}$	S_5
	S_2	{[1 0 0]', [0 1 0]'}	-	$^{\dagger}S_6$
			$\{t_1\}$	S_7
	S_3	{[0 0 0]', [2 1 0]'}	-	S_8
2	S_4	{[1 0 0]', [1 0 0]'}	-	$^{\dagger}S_9$
			$\{t_2\}$	S_{10}
	S_5	{[0 0 0]', [1 1 0]'}	-	$^{\dagger}S_{11}$
	$^{\dagger}S_6$	{[1 0 1]', [0 0 0]'}	-	$^{\dagger}S_6$
			$\{t_1\}$	S_7
			$\{t_2\}$	S_{12}
	S_7	{[0 0 1]', [2 0 0]'}	-	S_8
3	S_8	{[0 0 1]', [1 0 0]'}	-	$^{\dagger}S_{11}$
	$^{\dagger}S_9$	{[1 1 0]', [0 0 0]'}	-	$^{\dagger}S_9$
			$\{t_1\}$	S_{13}
			$\{t_2\}$	S_{10}
	S_{10}	{[0 1 0]', [0 1 0]'}	-	$^{\dagger}S_{11}$
	$^{\dagger}S_{11}$	{[0 1 1]', [0 0 0]'}	-	$^{\dagger}S_{11}$
			$\{t_3\}$	S_{14}
	S_{12}	{[0 0 1]', [0 1 0]'}	-	$^{*}S_{15}$
4	S_{13}	{[0 1 0]', [2 0 0]'}	-	S_{16}
	S_{14}	{[0 0 0]', [0 0 3]'}	-	S_{17}
	$^{*}S_{15}$	{[0 0 2]', [0 0 0]'}	-	$^{*}S_{15}$
5	S_{16}	{[0 1 0]', [1 0 0]'}	-	$^{*}S_{18}$
	S_{17}	{[0 0 0]', [0 0 2]'}	-	S_{19}
6	$^{*}S_{18}$	{[0 2 0]', [0 0 0]'}	-	$^{*}S_{18}$
	S_{19}	{[0 0 0]', [0 0 1]'}	-	$^{\dagger}S_0$

† *denotes relaxed states and * denotes relaxed and deadlock states. "-" means that there is no selection of* ϕ *(*ϕ *can be considered as an empty set).*

States S_0, S_6, S_9, S_{11}, S_{15}, and S_{18} are relaxed states, while rest of states are dynamic states. These relaxed states are similar to states of its equivalent untimed PN. For dynamic states, they have time durations in their remaining time vectors $\gamma_2(S)$ and do not preserve their statuses. Dynamic states yield another state in $R_S(G_A, S_0)$. For instance; note that, although $\hat{E}\left(G_A, \gamma_1(S_3)\right)$ is an empty set, the state S_3 reaches to the state S_8 after one *ts* is

elapsed. Here, S_3 is a dynamic state that yields S_8, and the status of flowing tokens is indicated in $\gamma_2(S_3)$. S_8 is also a dynamic state. The boundedness is examined through the marking vector $\gamma_1(S)$ and tokens at places. The bound vector is found as $\boldsymbol{B} = [2\ 2\ 2]'$ such that the given TdAPN is not safe due to $B(p) > 1$ for places p_1, p_2, and p_3. This vector is determined according to the information in $\boldsymbol{M} = \gamma_1(S)$ for all $S \in R_S(G_A, S_0)$. States S_{15} and S_{18} are deadlock states, where the set of deadlocks is $\tilde{\mathcal{L}}_0 = \{S_{15}, S_{18}\}$. As a result, the net is not live and is also not reversible due to $S_0 \notin R_S(G_A, S_{15})$ and $S_0 \notin R_S(G_A, S_{18})$. Furthermore, the timed-reachability graph, which is enhanced by the time information for the given TdAPN in Figure 3.4.(b), is shown in Figure 3.8.

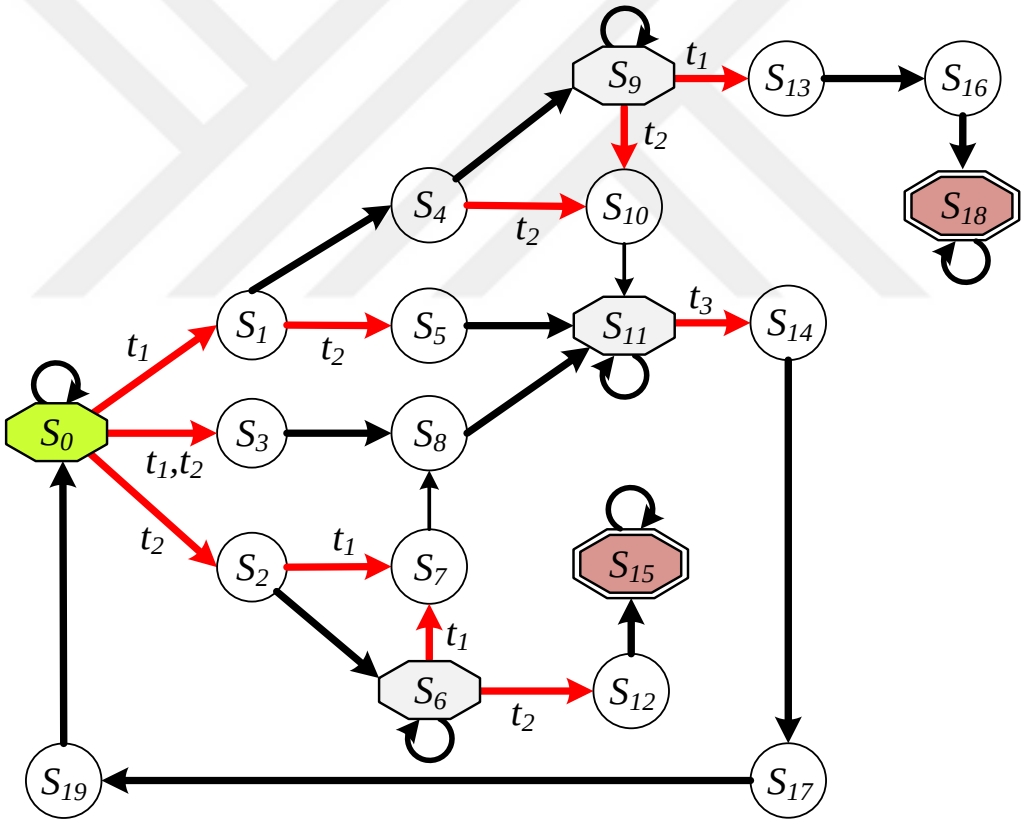


Figure 3.8. *Timed-reachability tree for TdAPN in Figure 3.4.(b)*

Figure 3.8 indicates all reachable states and the relation between them. In this figure, each arrow has an elapsed one *ts* time delay, and the choice of enabled transitions with regard to the enabled set $\phi \in \hat{E}\big(G_A, \gamma_1(S)\big)$ is also shown next to the arrows. Using this time information, it is possible to find how many time slots required for reaching

from any state to any state. For example, the state S_{11} is reachable from the initial state S_0 by the path ($S_0 \rightarrow S_3 \rightarrow S_8 \rightarrow S_{11}$) of $\{t_1, t_2\}$ after three *ts* has elapsed. This is the minimum time duration to reach S_{11}. However, it is also reachable using the path ($S_0 \rightarrow S_1 \rightarrow S_4 \rightarrow S_{10} \rightarrow S_{11}$) of $\{t_1, t_2\}$ after four *ts* has elapsed. Similarly, it is also reachable using the path ($S_0 \rightarrow S_1 \rightarrow S_4 \rightarrow S_9 \rightarrow S_{10} \rightarrow S_{11}$) of $\{t_1, t_2\}$ after five *ts* has elapsed. In Figure 3.8, relaxed states are illustrated by octagonal boxes while dynamic states are illustrated in circular boxes; and double octagonal boxes within the red color indicate deadlocks. The initial state is indicated in the green color. Remember that when the system is in a dynamic state, it means that flowing tokens are in the process of being transmitted. In addition, when the system is in a relaxed state, it means that there is no flowing token and this relaxed state is identical to that of its equivalent untimed PN. ◊

4. COMPARISONS

The proposed TdAPN is a new model for Timed PNs in order to represent, mathematically and graphically, temporal dynamics that become invisible during the firing process of Timed PNs with firing delays. In order to evaluate the performance of the proposed methodology, Stretched PNs [11-14, 46, 47], such as Transition-Stretched PN and Place-Stretched PN, are considered. This chapter presents the performance of the proposed TdAPN compared to Transition-Stretched PN [11-14] and Place-Stretched PN [46, 47] for the same original Timed PN. The performance is performed via case-studies in three ways, such as state-representation, computational complexity and computation time.

The following subsections present comparisons of TdAPN with Stretched PNs in terms of state-representation, computational complexity and computational times.

4.1. State-Representations

In this chapter, a case study of starting an engine and other case studies given in Chapter 6 are considered in evaluating the performance of TdAPN compared to Stretched PNs. A simple representative sketch of starting the engine is illustrated in Figure 4.1.

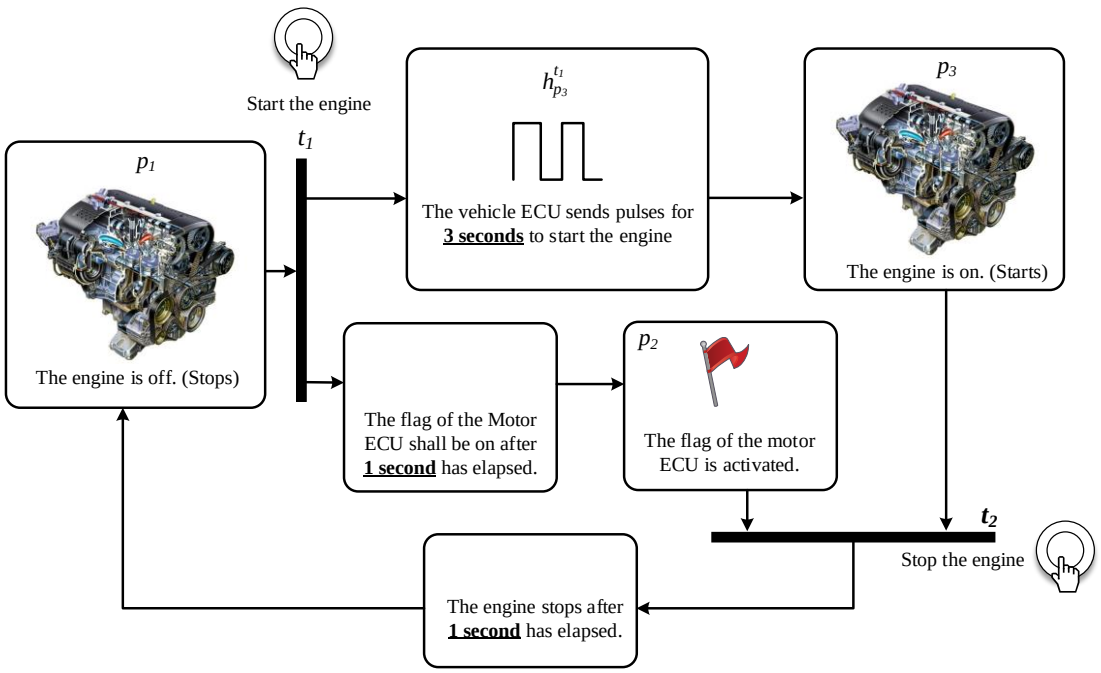
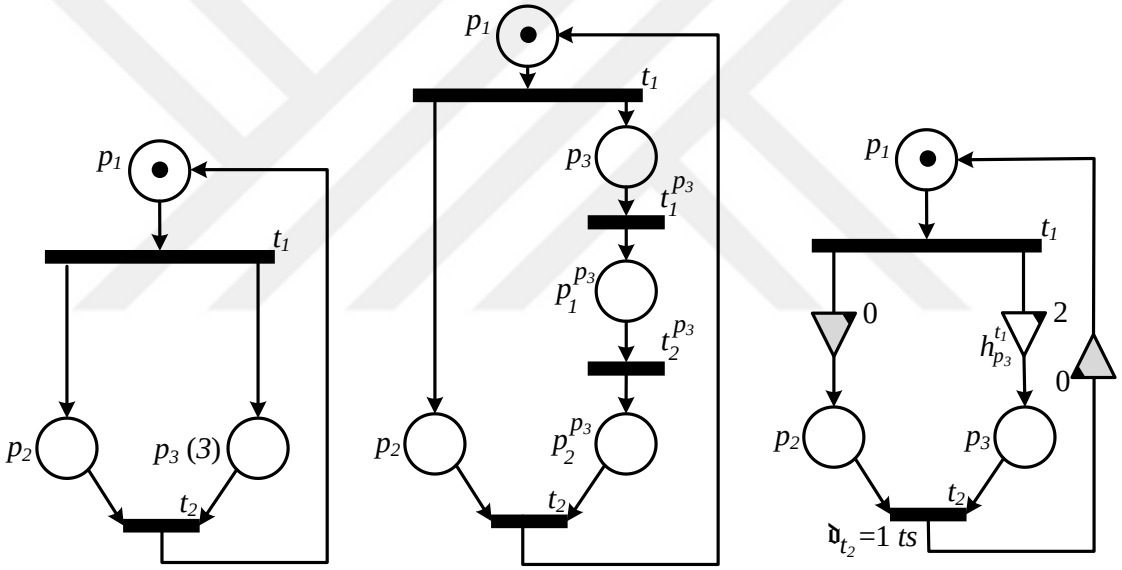


Figure 4.1. *Simple representative example of starting an engine*

In Figure 4.1, the engine is initially off. When a button for starting the engine is pressed (event t_1), the Motor Electronic Control Unit (MECU) is activated immediately (considered as a flag in p_2), and the Vehicle Electronic Control Unit (VECU) sends pulses for three seconds (considered in the time element $h_{p_3}^{t_1}$) in order to start the engine. Then, the engine starts (considered as p_3) after three seconds have elapsed. When a button for stopping the engine is pressed (event t_2), the engine stops after one second has elapsed. The Greatest Common Divisor of three and one seconds is one second; as a result, the appropriate sampling period can be chosen as one *ts* is equal to one second that is 1000 milliseconds. This example is modeled using Timed PN with holding durations, Place-Stretched PN, and the proposed TdAPN as illustrated in Figure 4.2, respectively.



(a) *The Original Timed PN with Holding Durations*

(b) *The Representation of Place-Stretched PN*

(c) *The Representation of TdAPN*

Figure 4.2. *Another representation of the engine example in Figure 4.1*

Physical meanings of places, transitions and time elements in these models are as follows:

- p_1 denotes the off-state of the engine. If a token exists, it means the engine stops.
- p_2 denotes the status of the flag of MECU. If a token exists, it means MECU is active.
- p_3 denotes the on-state of the engine. If a token exists, it means the engine runs.

- t_1 denotes the event of pressing the button to start the engine.
- t_2 denotes the event of pressing the button to stop the engine.
- The pair of $p_1^{p_3}$ and $t_1^{p_3}$ and the pair of $p_2^{p_3}$ and $t_2^{p_3}$ are additional elements to stretch the place p_2 of the original Timed PN with holding durations in Figure 4.2.(a). In Place-Stretched PN, the meaning of the place p_3 is transferred to the place $p_2^{p_3}$.
- $h_{p_3}^{t_1}$ in Figure 4.2.(c) denotes that VECU is sending pulses to start the engine. This operation takes two *ts*. Note that it requires three seconds in Figure 4.1, but the assignment of this time delay in TdAPN is equal to one minus of this time delay.

The representation of Place-Stretched PN for Timed PN in Figure 4.2.(a) is given in Figure 4.2.(b). However, Stretched PN brings with extra additional elements ($p_1^{p_3}$, $t_1^{p_3}$, $p_2^{p_3}$, $t_2^{p_3}$) compared to the original Timed PN model. The pair of transition-place is added into Place-Stretched PN in Figure 4.2.(b) to represent Timed PN in Figure 4.2.(a). On the other hand, the system in Figure 4.1 is modeled using the proposed TdAPN as shown in Figure 4.2.(c). In order to represent the time delay of the place p_3 in Figure 4.2.(a), only one time element $h_{p_3}^{t_1}$ is used in TdAPN in Figure 4.2. (c). The model of TdAPN in Figure 4.2.(c) is sufficient to represent the engine-example in Figure 4.1 with a minimal number of elements compared to the representation of Stretched PN in Figure 4.2.(b). The time element $h_{p_3}^{t_1}$ is a useful element that denotes the model completely and provides monitoring the status of the continuing operation related to an event; for this example, the physical meaning of $h_{p_3}^{t_1}$ is the continuing operation of sending pulses to the engine for a certain time. When the delay is updated from 3 *ts* to 5 *ts* (for instance, making the resolution better or updating the time delay according to measurements), there is no need to add new elements to TdAPNs compared to Stretched PN. The time element $h_{p_3}^{t_1}$ can solely represent the time delay of this operation.

Let us consider representations of Place-Stretched PN in Figure 4.2.(b), and TdAPN in Figure 4.2.(c). Suppose that the transition t_1 fires at time $k = \lambda$ and its firing process of t_1 starts. The illustrated comparison of these representations for the firing process of t_1 is given in Figure 4.3.(a) and (b).

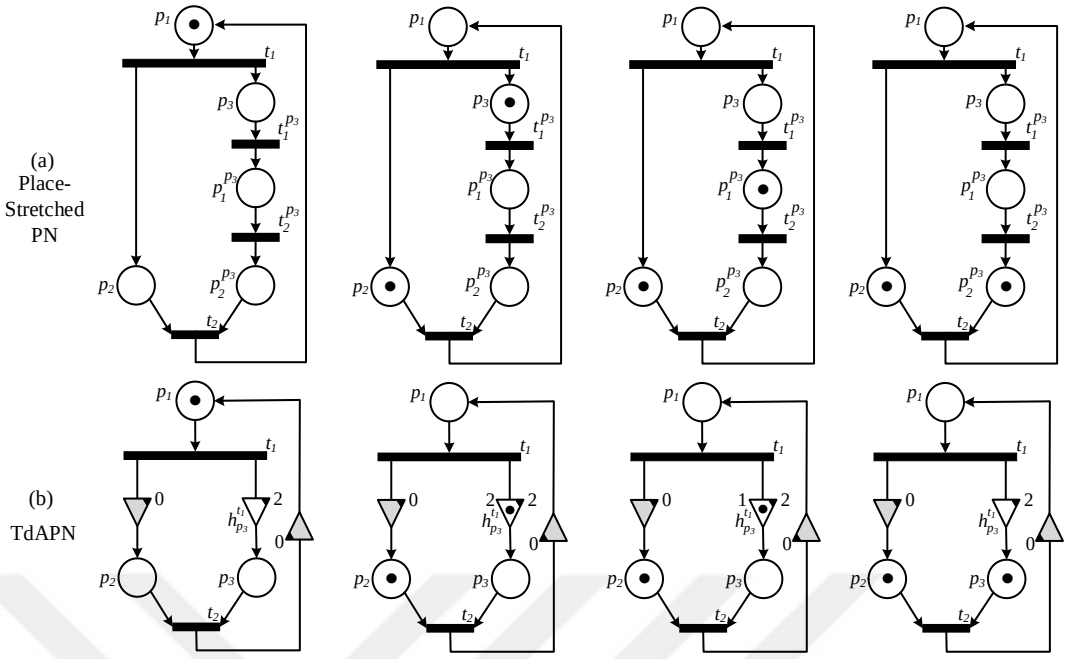


Figure 4.3. *Firing process of the transition t_1*

The firing process of t_1 starts at time $k = \lambda$ and ends at time $k = \lambda + d_{t_1}$. During the time $k \in (\lambda, \lambda + d_{t_1})$, flowing tokens are observed in representations of TdAPN and Place-Stretched PN. One time element $h_{p_3}^{t_1}$ in the representation of TdAPN is sufficient to represent the entire firing process of t_1, while Place-Stretched PN needs additional new elements, $p_1^{p_3}$, $t_1^{p_3}$, $p_2^{p_3}$, and $t_2^{p_3}$ for place-stretching, to represent this. However, Stretched PN transforms the original Timed PN into a stretched version of this Timed PN. This is a strong feature which makes the Timed PN as if it is an equivalent untimed PN. Thus, the representation of Place-Stretched PN is able to represent the state of the system and the firing process at each time. On the other hand, the proposed representation of TdAPN is also good at representing the firing process of t_1, where TdAPN uses only one time element $h_{p_3}^{t_1}$. Moreover, reachability sets of TdAPN in Figure 4.2.(c), and Place-Stretched PN in Figure 4.2.(b) are given in Table 4.1. All of them have the same number of states (four states).

Table 4.1. *Comparison for state-representations of TdAPN and Place-Stretched PN*

Time k	State-Label	The State of TdAPN	The State of Stretched PN
0	S_0	{[1 0 0]', [0]'}	[1 0 0 0 0]'
1	S_1	{[0 1 0]', [2]'}	[0 1 1 0 0]'
2	S_2	{[0 1 0]', [1]'}	[0 1 0 1 0]'
3	S_3	{[0 1 1]', [0]'}	[0 1 0 0 1]'

In Table 4.1, the state of Place-Stretched PN represents the state of the system completely by using a stretched version of the marking vector at time k. States S_0 and S_3 are relaxed states of TdAPN that are equal to the total number of states in the equivalent untimed PN. The rest of states show the significance of the time in time-delayed systems. For the state-representation, both TdAPN and Stretched PN use a vector form to indicate continuing firing processes and flowing tokens. TdAPN uses a fixed number of time elements in its vector form while Stretched PN uses newly generated places in its vector form, where the length of the vector is changed according to non-unity time delays defined in the original Timed PN. For this example, Place-Stretched PN in Figure 4.2.(b) uses five elements in its state-representation, such as p_1, p_2, p_3, $p_1^{p_3}$, and $p_2^{p_3}$, to represent the state of the system completely while TdAPN in Figure 4.2.(c) uses four elements, such as p_1, p_2, p_3 for $\boldsymbol{M}(k)$ and $h_{p_3}^{t_1}$ for $\boldsymbol{\nabla^R}(k)$. ◊

State-representation comparisons between the proposed TdAPN and Place-Stretched PN are given under this section. The representation of Transition-Stretched PN is also similar to Place-Stretched PN. In conclusion, models of TdAPN and Stretched PN for the same original Timed PN have the same number of states (under Assumption 3.2); however, their representations are different. Both models use a vector form to represent the state of the system completely. Stretched PN includes a stretched version of the marking vector together with new additional elements; however, the size of marking vector changes according to time delays described in the original Timed PN. On the other hand, the model of TdAPN uses an original marking vector of Timed PN and a remaining vector that is special to TdAPN. The advantage of TdAPN compared to Stretched PN is that the length of the remaining time vector is fixed by the number of time elements in the net. All situations of tokens are represented in the state-representations of TdAPN and Stretched PN.

4.2. Computational Complexity and Times

The complexity for several classes of PNs was conducted by Jones et.al. in [49]. Computational complexity for the construction of the reachability set for classical PNs is DSCAPE (exponential) hard [49]. Computational complexity is studied for the developed algorithms in this thesis. Using "for" loops in Algorithm 7.1, the computation complexity of Algorithm 7.1 is related to the number of elements of the reachability set, and the number of sets of places and transitions. This complexity can be related to $\mathfrak{M}2^n n^2 m$ in numerical terms. Here, $\mathfrak{M} = |R_S\,(G_A, S_0)|$ is the cardinality of the reachability set; $m = |P|$ is the cardinality of the set of places, and $n = |T|$ is the cardinality of the set of transitions. Similarly, the complexity of the algorithm for Stretched PN used in this thesis can be related to $\mathfrak{M}2^{\bar{n}}\bar{n}^3\bar{m}$. This complexity is related to the number of elements of the reachability set, and the number of sets of places and transitions after stretching procedure in [11-14, 46, 47]. Here, $\bar{n} = n + n_s = |T_s|$ is the cardinality of the set of transitions, where n_s is the cardinality of the set of newly generated transitions; and $\bar{m} = m + m_s = |P_s|$ is the cardinality of the set of places, where m_s is the cardinality of the set of newly generated places. Moreover, in order to compare TdAPN with original works of Stretched Petri Nets, the complexity of the original algorithm of Transition-Stretched PN [48] and Place-Stretched PN [32] can also be related to $\mathfrak{M}.\,(2^{\bar{n}} + \bar{n}^2.\,\bar{m})$ and $\mathfrak{M}2^{\bar{n}}\bar{n}\bar{m}$ by using "for" loops in their main algorithms, respectively.

Let us analyze the complexity of TdAPN for the automotive case-study in Figure 4.1. For the TdAPN and Place-Stretched PN, the complexities are obtained 192 and 1280, respectively, where $\mathcal{D}(p_3) = 3$ *ts*. For the case-study, using algorithms of TdAPN and Place-Stretched PN in MATLAB, the computational times are 1.382 seconds for TdAPN (0.046 second for the construction of the net and 1.336 for the construction of the reachability set) and 1.860 seconds for Place-Stretched PN (0.075 second for the construction of the net and 1.785 for the construction of the reachability set) for the automotive case-study in Figure 4.1. ◊

Let us analyze the complexity of TdAPN for the manufacturing case-study in Figure 6.1. For the TdAPN and Transition-Stretched PN, the complexities are obtained $4.6771*10^6$ and $5.4666*10^7$, respectively, where $\mathfrak{d}_{t_6} = 10$ *ts*. For the case-study, using algorithms of TdAPN and Transition-Stretched PN in MATLAB, the computational times are 3.6274 seconds for TdAPN (1.7714 second for the construction of the net and 1.8560

for the construction of the reachability set) and 8.3250 seconds for Transition-Stretched PN (1.9399 second for the construction of the net and 6.3851 for the construction of the reachability set) for the manufacturing case-study in Figure 6.1. ◊

In this section, the computational complexity of TdAPN and Stretched PNs [32, 48] are compared. For the complexity analysis, the construction of the reachability set is considered. Their complexities are related to the size of the reachability set and the size of sets of places and transitions. In addition, constructing the reachability set is DSCAPE (exponential) hard [49]. Results show that the complexity of Stretched PN is increased exponentially because of newly created pairs of place-transition/transition-place when the time delay of the transition/place increases. On the other hand, the time element is a useful element, where the number of elements of the set of time elements is not affected by this increase. The complexity of TdAPN is increased by the first order polynomial when the time delay of the transition/place increases. Moreover, computational times are measured using tic and toc functions of MATLAB. These computational times include the construction of the net and the construction of the reachability set. Computational times are obtained by a personal computer, which has the following features:

- The software environment is MATLAB v8.3.0.532 (R2014a) and Windows 7 Ultimate SP1 64-bit.
- The hardware environment is 240GB SSD, Core2 Duo 2.10 GHz CPU T6500 64 bits and 4060 MB DDR2 RAM.

In this thesis, it is not aimed to optimize or improve algorithms of TdAPN given in Section 7.1. It is just aimed to construct the reachability set of TdAPN for time-delayed systems and to show the usefulness of the time element compared to Stretched PNs.

5. CONTROLLER DESIGN

The controller design is one of the essential topics in PNs. In order to guarantee the desired property of PN, such as liveness, deadlock-free, reversibility, and boundedness, it is required to control the net. In the control-literature of PNs, two types of controllers have been presented, such as behavioral controllers and structural controllers [9, 11-14, 32-35, 37-41].

In the method of behavioral-controller design, supervisory controllers for untimed and timed PNs have been presented to enforce the system to ensure some basic behavioral properties, such as deadlock-free, reversibility, etc. [9, 11-14, 32, 36-40]. Deadlock-free is the most desired property among behavioral properties of PNs. After the construction of the reachability set, a supervisory controller is designed for enforcing the system to behave in a desired manner. This type of controller that prohibits undesired states, namely ***forbidden states***, and allows the desired states is called ***Forbidden State Controller*** in general. A Forbidden State Controller is based on the reachability set of PN, and rules of a control policy that are implemented through this set [12]. In this control policy, the considered enabled transition(s) is/are disabled by the designed controller, if the forbidden state(s) is/are reachable by using this/these transition(s). In addition to the method of the behavioral controller, the method of structural-controller design that adds new additional places, namely monitor places, into the original net have also been presented [33, 34, 41]. This type of controller uses the structural properties of PNs [33-35, 41].

This chapter presents a Forbidden State Controller for the proposed TdAPN that is based on the approach developed in [13] by Aybar et.al.

5.1. Forbidden State Controller Design for Timed-Arc Petri Nets

Behavioral properties of G_A, such as liveness, deadlock, reversibility, and boundedness, are analyzed by using the reachability set $R_S(G_A, S_0)$. According to the analysis, a supervisory controller that avoids the occurrence of forbidden states can be designed.

The state of G_A at time k is $S(k) = \{\boldsymbol{M}(k), \boldsymbol{\nabla^R}(k)\}$. After the construction of the reachability set $R_S(G_A, S_0)$, any state can be represented by $S = \{\boldsymbol{M}, \boldsymbol{\nabla^R}\}$. $\boldsymbol{\nabla^R}$ includes the remaining-time (duration) information about flowing tokens, which is independent of discrete-time notation k. Remember that $S = \{\gamma_1(S), \gamma_2(S)\}$, $S \in R_S(G_A, S_0)$, where the marking-vector part of S is $\boldsymbol{M} = \gamma_1(S)$, and the remaining-time vector part of S is $\boldsymbol{\nabla^R} = \gamma_2(S)$.

The next state that is denoted by $\tilde{S} = \{\widetilde{\boldsymbol{M}}, \widetilde{\boldsymbol{\nabla}}^{\boldsymbol{R}}\}$ is computed as given in (5.1) and (5.2) by using the present state $S = \{\boldsymbol{M}, \boldsymbol{\nabla^R}\}$ and enabled transition(s) in the set $\phi \subseteq F(S)$, where $\phi \in \hat{E}\big(G_A, \gamma_1(S)\big)$ and $F(S) = F_{pre}(S) \cup \phi$ (see, definition (3.6)).

$$\widetilde{M}(p) \coloneqq \gamma_1(S)(p) + \sum_{t \in F(S)} \big(O(p,t) - N(p,t)\big) \tag{5.1}$$

$$\widetilde{\nabla}^R\big(h_p^t\big) \coloneqq \gamma_2(S)\big(h_p^t\big) + \sum_{t \in \phi} D(p,t) - \sum_{t \in F_{pre}(S) \text{ and } \gamma_2(S)(h_p^t) > 0} 1 \tag{5.2}$$

This computation is represented by a function $\rho\big(S, F(S)\big)$, i.e., $\tilde{S} = \rho\big(S, F(S)\big)$.

According to the selection of $\phi \in \hat{E}\big(G_A, \gamma_1(S)\big)$, while $S \in R_S(G_A, S_0)$, any state S may cause the system to be directed to an undesired state; as a result, starting a choice of a certain event at S violates a desired behavioral property of G_A. For instance, deadlock-free is the desired property, and $\rho\big(S, F(S)\big)$ leads the system to a deadlock state, where $F(S)$ includes an event that must be avoided. When the system enters in such an undesired state, there is no chance to avoid the deadlock. When faced with such a situation, the system has to be initialized or brought to a known safe state. This will probably result in a cost. Therefore, a controller design that avoids forbidden states is an absolute necessity.

In order to design a Forbidden State Controller for TdAPNs that prevents the system from entering into undesired states, first of all, these undesired states should be determined. The set of undesired states is represented by $\mathcal{L}_0$, where the subscript of zero stands for indicating the initial set and $\mathcal{L}_0 \subset R_S(G_A, S_0)$. This set is determined by the user or by behavioral properties of G_A that must be enforced. $\mathcal{L}_0$ is an initial set such that any state S may result in an undesired state in $\mathcal{L}_0$ by the selection of an event; therefore, the set of undesired states $\mathcal{L}_0$ is enlarged to an expanded set of undesired states, $\hat{\mathcal{L}} \coloneqq \bigcup_{i=0}^{n} \mathcal{L}_i$,

where $\mathcal{L}_i$ is defined as in (5.3) [13]. The set of $\mathcal{L}_i \subset \hat{\mathcal{L}}$ is constructed for $i = 1,2,\dots,n+1$, where $n \in \mathbb{N}$ is such that $\mathcal{L}_n \neq \emptyset$ and $\mathcal{L}_{n+1} = \emptyset$ [13].

$$\mathcal{L}_i \coloneqq \left\{ S \in R_S(G_A, S_0) \,\middle|\, \rho\big(S, F(S)\big) \in \bigcup_{l=0}^{i-1} \mathcal{L}_l \,, \forall \phi \in \hat{E}\big(G_A, \gamma_1(S)\big) \right\} \tag{5.3}$$

States in $\hat{\mathcal{L}}$ lead the system to the undesired domain, such that it must be avoided from entering in such states. Thus, the forbidden state controller is able to prohibit and disable any enabled transition in $\phi \in \hat{E}\big(G_A, \gamma_1(S)\big)$, where the next state $\tilde{S} = \rho\big(S, F(S)\big)$ is in $\hat{\mathcal{L}}$ after the selection of $\phi \in \hat{E}\big(G_A, \gamma_1(S)\big)$. For this purpose, the controller has a controller function that is represented by $\mathcal{C}(S, \phi)$, where it allows an event to be enabled by one and to be disabled by zero as defined in (5.4) [13].

$$\mathcal{C}(S, \phi) \coloneqq \begin{cases} 0 \,, if\ \rho\big(S, F_{pre}(S) \cup \phi\big) \in \hat{\mathcal{L}} \\ 1 \,, otherwise \end{cases} \tag{5.4}$$

Here, $F(S) = F_{pre}(S) \cup \phi$.

$\mathcal{C}(S, \phi) = 0$ means that the controller disables ϕ at the state S. $\mathcal{C}(S, \phi) = 1$ means that the controller allows ϕ at the state S. Moreover, if $S_0 \in \hat{\mathcal{L}}$, then $R_S(G_A, S_0) = \{S_0\}$ and $\hat{\mathcal{L}} = \mathcal{L}_0$.

A forbidden state controller for the proposed TdAPN can be designed to avoid any undesired state by using the approach in this subsection. This controller will be named according to its functionality as follows:

- If it is desired to make the system avoid deadlock states, then these deadlock states are considered as forbidden states and the designed controller is called the ***deadlock avoidance controller***.
- If it is desired to make the system avoid loops or irreversible states, then these irreversible states are considered as forbidden states and the designed controller is called the ***reversibility enforcement controller***.

5.2. Controller Examples for Timed-Arc Petri Nets

Let us design a forbidden state controller for TdAPN in Figure 3.4.(b), which offers a *deadlock avoidance controller*. Based on the description of the deadlock in Definition 3.4 and (3.8), the set of undesired states $\mathcal{L}_0$ is equal to $\tilde{\mathcal{L}}_0$ for the *deadlock avoidance controller*. The reachability set of TdAPN in Figure 3.4.(b) is $R_S(G_A, S_0) = \{S_0, S_1, ..., S_{19}\}$ as given in Table 3.3, and its timed-reachability tree is shown in Figure 3.8. Using $R_S(G_A, S_0)$ and (3.8), the set of undesired states including deadlocks is determined as $\mathcal{L}_0 = \tilde{\mathcal{L}}_0 = \{S_{15}, S_{18}\}$. There are initially two undesired states. The expanded set of undesired states, which refers to states that lead the system to $\mathcal{L}_0$, is denoted by $\hat{\mathcal{L}} = \bigcup_{i=0}^{n} \mathcal{L}_i$, where $\mathcal{L}_i$ as defined in (5.3). Using $R_S(G_A, S_0)$, $\mathcal{L}_0$ and (5.3), sub-sets of $\hat{\mathcal{L}}$ are iteratively found as: $\mathcal{L}_0$, $\mathcal{L}_1 = \{S_{12}, S_{16}\}$, $\mathcal{L}_2 = \{S_{13}\}$, $\mathcal{L}_3 = \emptyset$; as a result, the expanded set of undesired states, which must be avoided, is found as: $\hat{\mathcal{L}} = \{S_{12}, S_{13}, S_{15}, S_{16}, S_{18}\}$. In order to avoid a state $S \in \hat{\mathcal{L}}$, the controller disables the set $\{t_2\} \in \hat{E}\big(G_A, \gamma_1(S_6)\big)$ at the state S_6, where $F(S_6)$ is equal to $F_{pre}(S_6) \cup \{t_2\}$, and $\rho\big(S_6, F(S_6)\big)$ gives $S_{12} \in \hat{\mathcal{L}}$; and the set $\{t_1\} \in \hat{E}\big(G_A, \gamma_1(S_9)\big)$ at the state S_9, where $F(S_9)$ is equal to $F_{pre}(S_9) \cup \{t_2\}$, and $\rho\big(S_9, F(S_9)\big)$ gives $S_{13} \in \hat{\mathcal{L}}$. Values of the controller function are determined as follows:

$$\mathcal{C}(S, \phi) = \begin{cases} 0\ , S = S_6\ and\ \phi = \{t_2\} \\ 0\ , S = S_9\ and\ \phi = \{t_1\} \\ 1\ , otherwise \end{cases}.$$

The timed-reachability tree in Figure 3.8 is re-illustrated as in Figure 5.1 to show the effect of controller values. The purple color indicates the disabled transitions and undesired states that must be avoided; in addition, both straight and dashed lines in the purple color represent the path that leads the system into an undesired state. Moreover, states in the yellow color represent the states, where certain events must be disabled in order to guarantee deadlock-free. ◊

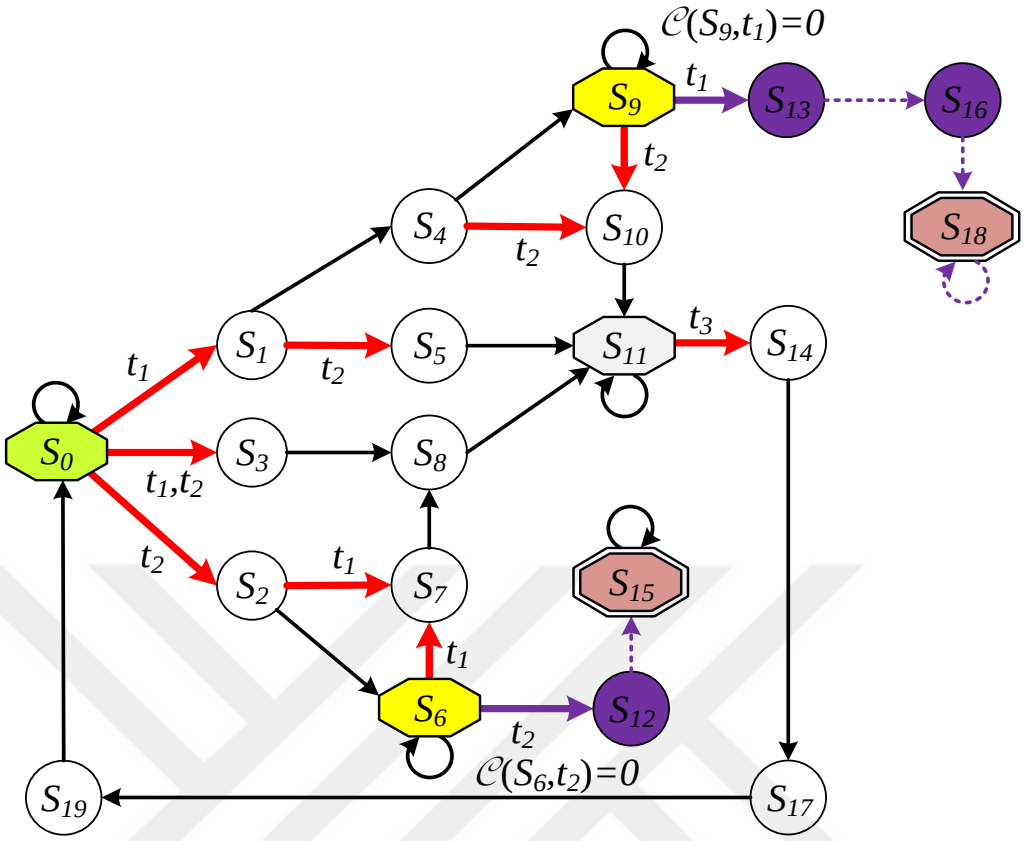


Figure 5.1. *Timed-reachability tree of TdAPN in Figure 3.4 with the controller*

Let us design a forbidden state controller for TdAPN in Figure 5.2, which offers a *reversibility enforcement controller*.

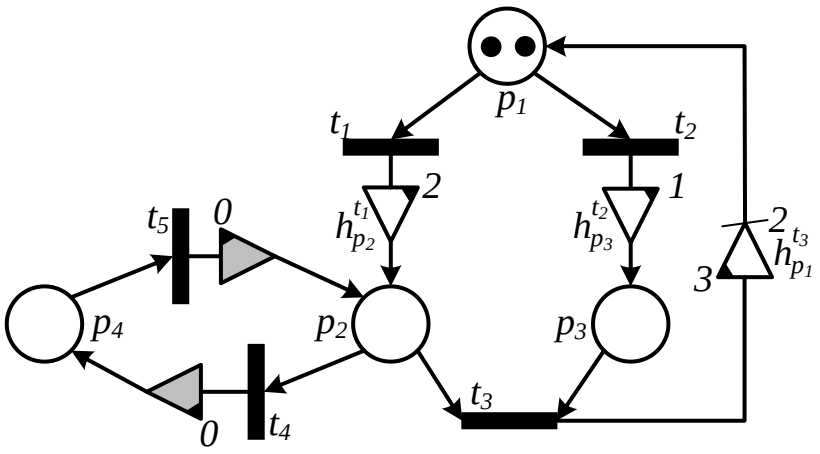


Figure 5.2. *Example of TdAPN includes a deadlock state and loop*

TdAPN in Figure 5.2 is defined as $G_A(P, T, N, O, D, S_0)$. Here, the set of places is $P = \{p_1, p_2, p_3, p_4\}$ and the set of transitions is $T = \{t_1, t_2, t_3, t_4, t_5\}$. Input, output and time delay matrices are respectively:

$$N = \begin{bmatrix} 1\,1\,0\,0\,0 \\ 0\,0\,1\,1\,0 \\ 0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1 \end{bmatrix}, O = \begin{bmatrix} 0\,0\,2\,0\,0 \\ 1\,0\,0\,0\,1 \\ 0\,1\,0\,0\,0 \\ 0\,0\,0\,1\,0 \end{bmatrix}, D = \begin{bmatrix} 0\,0\,3\,0\,0 \\ 2\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0 \end{bmatrix}.$$

The set of time elements is $\nabla = \{h_{p_2}^{t_1}, h_{p_3}^{t_2}, h_{p_1}^{t_3}\}$, where $D(p,t) \neq 0$ and $O(p,t) \neq 0$. The state of the TdAPN at the initial time k_0 is $S(k_0) = \{[2\ 0\ 0\ 0]', [0\ 0\ 0]'\}$.

The *reversibility enforcement controller* is used to avoid states, which violate the property of reversibility described in Definition 3.5 (Reversibility Property). Remember that G_A is considered as reversible if the initial state $S_0 \in R_S(G_A, S)$ is reachable from all states $S \in R_S(G_A, S_0)$ according to Definition 3.5. If G_A does not satisfy this condition, then the overall net is considered as irreversible throughout the net. This situation generally occurs when the net has any deadlock state or any loop. On the other hand, a subset of the reachability set $R_S(G_A, S_0)$, where all states in this subset ensure reversibility, can be obtained. If the initial state $S_0 \in R_S(G_A, S)$ is not reachable from all states $S \in R_S(G_A, S_0)$, then a subset of $R_S(G_A, S_0)$ is constructed in a set form as $R_R(G_A, S_0)$ while $R_R(G_A, S_0) \subset R_S(G_A, S_0)$. Here, $R_R(G_A, S_0)$ represents the irreversible set of G_A. For the *reversibility enforcement controller*, the set of undesired states is $\mathcal{L}_0 = R_R(G_A, S_0)$, and $\hat{\mathcal{L}}$ is equal to $\mathcal{L}_0$. In order to determine $\mathcal{L}_0$, the reversibility analysis should be performed for all states in $R_S(G_A, S_0)$ such that this analysis also covers states in $R_R(G_A, S_0)$.

For TdAPN in Figure 5.2, 27 states of TdAPN are obtained as given in Appendix-1. Its reachability set is obtained as $R_S(G_A, S_0) = \{S_0, S_1, \dots, S_{26}\}$. Moreover, the timed-reachability tree for this TdAPN, which indicates all reachable states and the relation between them, is shown in Figure 5.3. Descriptions of elements in this figure are similar to descriptions for Figure 3.8.

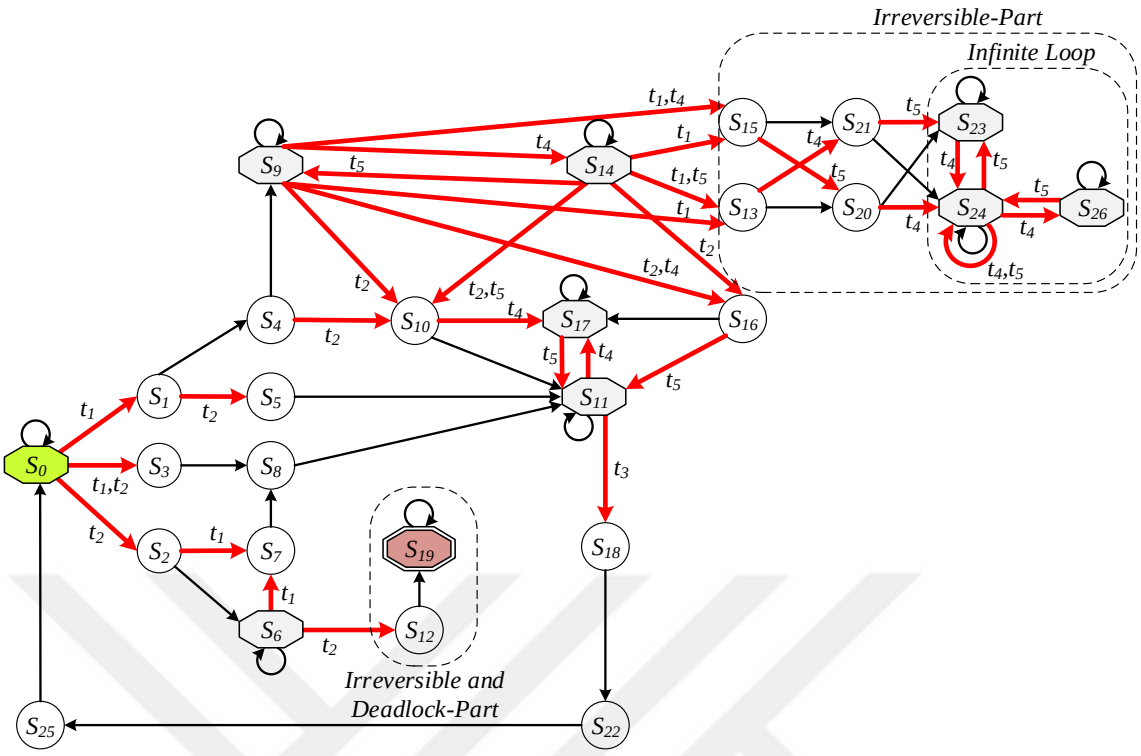


Figure 5.3. *Timed-reachability tree of TdAPN in Figure 5.2*

In order to determine undesired states, including irreversible states, the reversibility analysis is performed for all states S in $R_S(G_A, S_0)$ and the irreversible set $R_R(G_A, S_0) \subset R_S(G_A, S_0)$ is obtained as $R_R(G_A, S_0)$ = {S_{12}, S_{13}, S_{15}, S_{19}, S_{20}, S_{21}, S_{23}, S_{24}, S_{26}}. These states in $R_R(G_A, S_0)$ are forbidden states for the *reversibility enforcement controller*. The set of undesired states is $\mathcal{L}_0 = R_R(G_A, S_0)$ such that $\mathcal{L}_0$ = {S_{12}, S_{13}, S_{15}, S_{19}, S_{20}, S_{21}, S_{23}, S_{24}, S_{26}}, where the expanded set of deadlock states is $\hat{\mathcal{L}} = \mathcal{L}_0$. The values of the controller function $\mathcal{C}(S, \phi)$ are similarly obtained as discussed in the example of the *deadlock avoidance controller*. These are determined as follows (see, Figure 5.4): $\mathcal{C}(S_6, \{t_2\}) = 0$, $\mathcal{C}(S_9, \{t_1\}) = 0$, $\mathcal{C}(S_9, \{t_1, t_4\}) = 0$, $\mathcal{C}(S_{13}, \{t_4\}) = 0$, $\mathcal{C}(S_{14}, \{t_1\}) = 0$, $\mathcal{C}(S_{14}, \{t_1, t_5\}) = 0$, $\mathcal{C}(S_{15}, \{t_5\}) = 0$, $\mathcal{C}(S_{20}, \{t_4\}) = 0$, $\mathcal{C}(S_{21}, \{t_5\}) = 0$, $\mathcal{C}(S_{23}, \{t_4\}) = 0$, $\mathcal{C}(S_{24}, \{t_4\}) = 0$, $\mathcal{C}(S_{26}, \{t_5\}) = 0$, where $\phi \in \hat{E}\big(G_A, \gamma_1(S)\big)$. Otherwise, $\mathcal{C}(S, \phi) = 1$, where $S \in R_S(G_A, S_0) \backslash \hat{\mathcal{L}}$.

In order to illustrate the effect of controller values, the timed-reachability tree in Figure 5.3 is re-illustrated as in Figure 5.4. Descriptions of colors and shapes are similar as given in Figure 5.1. Here in Figure 5.4, states in the yellow color represent the states, where certain transitions must be disabled in order to enforce the system reversibility. ◊

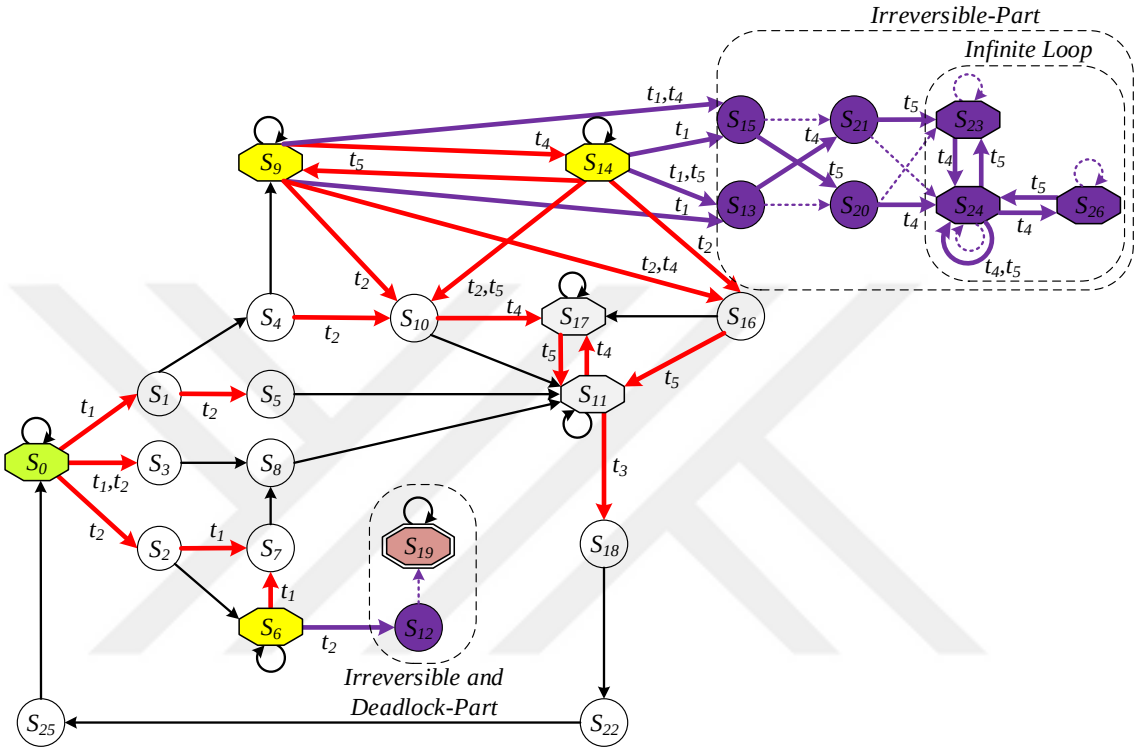


Figure 5.4. *Timed-reachability tree of TdAPN in Figure 5.3 with the controller*

6. MODELING AND DESIGN FOR REAL WORLD SYSTEMS

This chapter presents special applications of TdAPN that can be applied on manufacturing systems, railway systems and automotive systems as case studies; in addition, corresponding results aided by the software of TdAPN are presented (see, algorithms in Chapter 7 for the software of the proposed TdAPN).

6.1. Manufacturing Systems

The current industrial revolution, Industry 4.0, is based on connectivity, big data, and event-based operational technologies. This futuristic innovation includes many large scale systems, and its infrastructure is constructed using the concept of Systems of Systems. Such systems are best described by the occurrence of events. In order to model these, PN is a nice modeling paradigm. However, time delays have a significant role in such systems. Thus, Timed PN is a useful tool to accurately express them. For this purpose, this section includes a practical manufacturing example, studied in [12] as a case study, including an industrial robot, a machine, storages that are modeled by TdAPN. The system, illustrated in Figure 6.1, comprises a machine; a buffer whose capacity is limited to storing a single part; one main store whose storage capacity is limited to two parts; two flat pallets to transfer parts; and an industrial robot. Each pallet is able to transport only one part in one go.

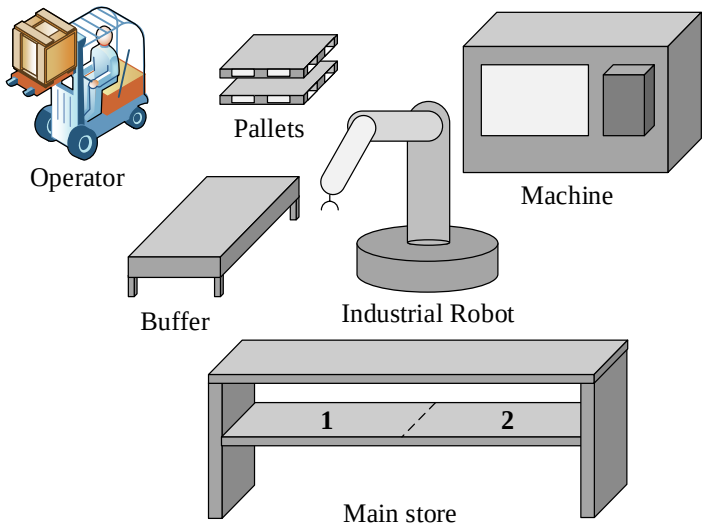
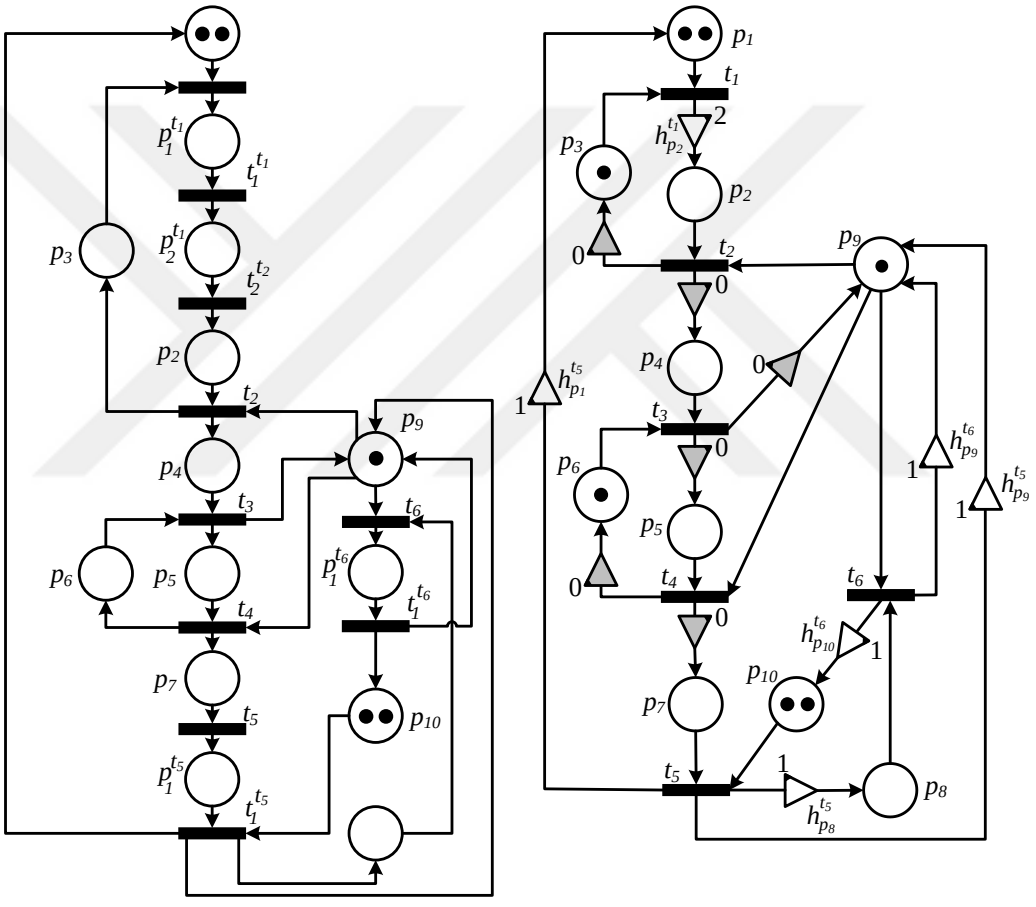


Figure 6.1. *Representative manufacturing example*

6.1.1. Modeling manufacturing system using TdAPN

The representation of Transition-Stretched PN for the manufacturing example in Figure 6.1, which is the transition-stretched equivalent of Timed PN in [12], is shown in Figure 6.2.(a). For Timed PN in [12], the set of time delays is $\mathfrak{D} = \{\mathfrak{d}_{t_1}, \mathfrak{d}_{t_2}, \mathfrak{d}_{t_3}, \mathfrak{d}_{t_4}, \mathfrak{d}_{t_5}, \mathfrak{d}_{t_6}\}$, where $\mathfrak{d}_{t_1} = 3$ *ts*, $\mathfrak{d}_{t_2} = 1$ *ts*, $\mathfrak{d}_{t_3} = 1$ *ts*, $\mathfrak{d}_{t_4} = 1$ *ts*, $\mathfrak{d}_{t_5} = 2$ *ts*, and $\mathfrak{d}_{t_6} = 2$ *ts*. This manufacturing example is modeled by the representation of TdAPN as illustrated in Figure 6.2.(b).



(a) *The Representation of Transition-Stretched PN*

(b) *The Representation of TdAPN*

Figure 6.2. *Model of (a) Transition-Stretched PN* [12] *and (b)TdAPN*

The model of TdAPN in Figure 6.2 is described as $G_A(P, T, N, O, D, S_0)$. The input matrix N, the output matrix O, and the time delay matrix D are as follows:

$$N = \begin{bmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 1&0&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \\ 0&1&0&1&0&1 \\ 0&0&0&0&1&0 \end{bmatrix}, O = \begin{bmatrix} 0&0&0&0&1&0 \\ 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&1&0&1&1 \\ 0&0&0&0&0&1 \end{bmatrix}, D = \begin{bmatrix} 0&0&0&0&1&0 \\ 2&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&1&1 \\ 0&0&0&0&0&1 \end{bmatrix}.$$

The set of time elements is $\nabla = \{h_{p_2}^{t_1}, h_{p_1}^{t_5}, h_{p_8}^{t_5}, h_{p_9}^{t_5}, h_{p_9}^{t_6}, h_{p_{10}}^{t_6}\}$, where $D(p,t)$ and $O(p,t)$ are not equal to zero. $S_0 = \{\boldsymbol{M}_0, \boldsymbol{\nabla}^{\boldsymbol{R_0}}\}$ is the initial state of G_A, where $\boldsymbol{M}_0 = [2\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 2]'$ and $\boldsymbol{\nabla}^{\boldsymbol{R_0}} = [0\ 0\ 0\ 0\ 0\ 0]'$. Here, Places denote conditions, transitions denote events, and time elements denote continuing operations related to the events as described in Table 6.1. According to descriptions in this table, $M(k_0, p_1)$=2 indicates the availability of two pallets at the initial time. $M(k_0, p_3)$=1 denotes the machine is available. $M(k_0, p_6)$=1 shows the buffer is available. $M(k_0, p_9) = 1$ denotes the robot is available. And, $M(k_0, p_{10})$=2 shows that there are two unoccupied rooms for storage in the main store, where its maximum storage capacity is 2 parts. $\boldsymbol{\nabla}^{\boldsymbol{R_0}} = \mathbf{0}^{|\nabla|\times 1}$ indicates that there is currently no dynamic operation (no previously activated firing process, so $F_{pre}(k_0) = \emptyset$).

Table 6.1. *Physical meanings for elements of TdAPN in Figure 6.2.(b)*

Element	Explanation	Status
p_1	The number of available pallets.	**#T:** the number of pallets.
p_2	The operation of the machine.	**NT:** uncompleted, **T:** completed.
p_3	The availability of the machine.	**NT:** unavailable (no), **T:** available (yes).
p_4	The robot is assigned to unload the machine.	**NT:** no assignment, **T:** assigned.
p_5	The fullness of the buffer.	**NT:** empty, **T:** full.
p_6	The availability of the buffer.	**NT:** unavailable, **T:** available.
p_7	The robot's task is transferring the produced part to an unoccupied room in the main store.	**NT:** no assignment, **T:** the task is assigned.
p_8	The number of occupied rooms for storage in the main store.	**#T:** the number of occupied rooms.
p_9	The robot's status.	**NT:** busy, **T:** free.
p_{10}	The number of unoccupied rooms for storage in the main store.	**#T:** the number of unoccupied rooms.

***NT**: No token exists. **T**: Token exists. **#T**: The existing number of tokens.*

Table 6.1. (Continue) *Physical meanings for elements of TdAPN in Figure 6.2.(b)*

Element	Explanation	Status
t_1	Start the machine to produce a part.	
t_2	Make the robot unload the produced part.	
t_3	Make the robot load the produced part to the buffer.	
t_4	Make the robot unload the buffer.	
t_5	Set a pallet free and make the robot transfer the produced part from buffer to an unoccupied room of main store.	
t_6	Make the robot unload one part from the main store.	
$h_{p_2}^{t_1}$	The machine is producing a part. This operation takes 2 *ts*.	
$h_{p_1}^{t_5}$	The pallet is going to be free. This operation takes 1 *ts*.	
$h_{p_8}^{t_5}$	The robot is transferring the produced part to an unoccupied room of main store. This operation takes 1 *ts*.	
$h_{p_9}^{t_5}$	The robot is going to be available after it transfers the produced part. This operation takes 1 *ts*.	
$h_{p_9}^{t_6}$	The robot is going to be available after it unloads one part. This operation takes 1 *ts*.	
$h_{p_{10}}^{t_6}$	The robot is unloading one part from the main store. This operation takes 1 *ts*.	

Algorithms for TdAPN (Algorithm 7.1) found 75 states of the reachability set $R_S(G_A, S_0)$ for the manufacturing example as given in Appendix-2. Here, the number of states of TdAPN is equal to the number of states for representations of Timed PN in [12], Transition-Stretched PN in [12], and the equivalent representation of Place-Stretched PN. The reachability set was obtained as $R_S(G_A, S_0) = \{S_0, S_1, \dots, S_{74}\}$. Based on $R_S(G_A, S_0)$, the net is not live and not reversible due to deadlocks S_{20}, S_{49}, S_{72}, and S_{74}. The software also generated a timed-reachability tree by considering the relation between states in the set $R_S(G_A, S_0)$ as shown in Figure 6.3. The A3-page (zoomed in) version of Figure 6.3 is reachable from Appendix-5. Descriptions of this graph are similar to explanations for Figure 3.8. 1.8560 seconds were required to construct $R_S(G_A, S_0)$ of TdAPN and 1.7714 seconds were required for the construction of TdAPN (total is 3.6274 seconds), while 6.3851 seconds were required to construct the reachability set for the equivalent representation of Transition-Stretched PN and 1.9399 seconds were required for the construction of Transition-Stretched PN (total is 8.3250 seconds). The construction time of the reachability set of TdAPN is shorter than the Transition-Stretched PN.

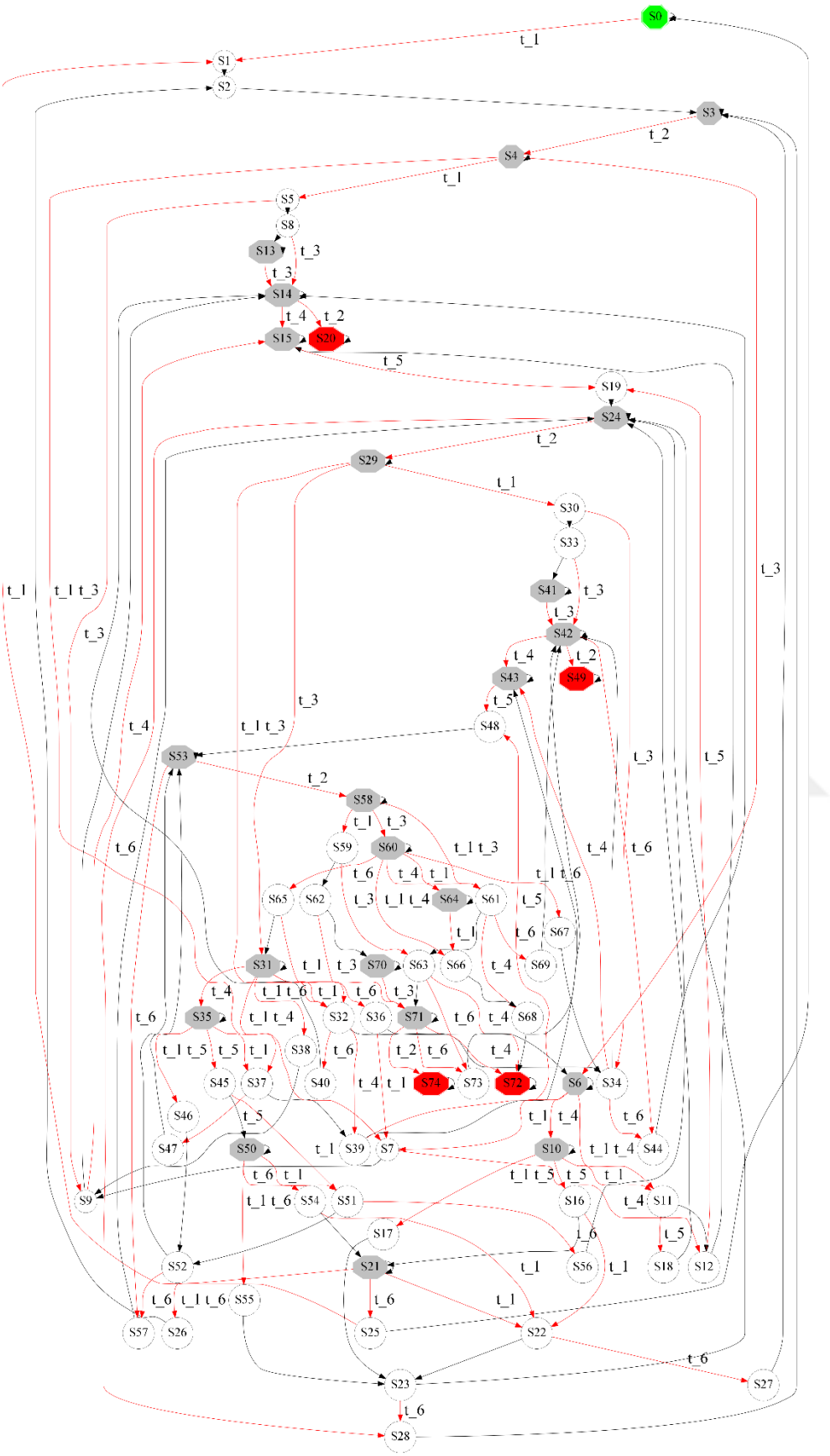


Figure 6.3. *Timed-reachability tree for TdAPN in Figure 6.2.(b)*

6.1.2. Controller-design

Let us design a forbidden state controller for TdAPN in Figure 6.2.(b), which enforces the system to be reversible and deadlock-free. The sub-algorithm *findExpandedSet* (Algorithm 7.8) finds the expanded set of deadlocks as {S_{20}, S_{49}, S_{64}, S_{66}, S_{68}, S_{72}, S_{74}}. The net is analyzed by the sub-algorithm *findIrreversibleSet* (Algorithm 7.9) so as to obtain the irreversible set as $R_R(G_A, S_0)$ = {S_{20}, S_{49}, S_{64}, S_{66}, S_{68}, S_{72}, S_{74}} while the expanded set of deadlocks is a subset of $R_{\mathrm{R}}(G_A, S_0)$. The expanded set of undesired states is $\hat{\mathcal{L}} = R_R(G_A, S_0)$. Using the sub-algorithm *controlForbiddenState* (Algorithm 7.10), values of the controller function are determined as follows:

$$\mathcal{C}(S,\phi) = \begin{cases} 0, \ S = S_{64} \ and \ \phi = \{t_1\} \\ 0, \ S \in \{S_{14}, S_{42}, S_{71}\} \ and \ \phi = \{t_2\} \\ 0, \ S \in \{S_{60}, S_{61}, S_{63}, S_{71}\} \ and \ \phi = \{t_4\} \\ 0, \ S = S_{60} \ and \ \phi = \{t_1, t_4\} \\ 1, \ otherwise \end{cases}.$$

6.2. Railway Systems

Safer and better transportation is a popular issue to develop intelligent and autonomous solutions in the field of land, marine, air, and railway systems. The growing population requires more technological designs in this century. Railway systems are more interesting than other fields of transportation due to their safety record [1, 2]. They are sophisticated, large-scale, and event-driven. Thus, they comprise subsystems which are composed of configuration items including many components (systems of systems). There are formal techniques to model and verify Railway Systems in the "Table A.17" of EN50128:2011, where a high level of safety is required in railway automation systems [25, 42, 43]. One of these techniques is using Petri Nets.

Since the railway system is concurrent and dynamic, Timed PN is a useful tool for railway systems. Therefore, due to distinctly associating time delay onto outgoing arcs that are connected to the same transition, Timed-Arc PN is more useful than Timed PN in order to represent the dynamics of system activities (e.g. motion, movement, etc.) into the model. In Timed-Arc PNs, all situations of the dynamic system are considered at any time without loss of information. In this section, a transition of a train between adjacent blocks is basically modeled by using TdAPN. This case study has been presented by Yufka et.al. in [1]. In railway systems, tracks of the railway network comprises blocks.

Each block section includes tracking circuits (TCs) to detect trains. Hence, the railway automation becomes aware of the block section whether it is occupied by a train. The railway network has block sections and their corresponding TCs. To denote these block sections in applications, the set of blocks in a specific route on a track is represented by $Set_{Blocks} := \{Block_i | i = 1,2,\dots,\mathfrak{B}\}$, where $\mathfrak{B} \in \mathbb{N}\backslash\{0,\infty\}$ is the total number of block sections on this route. In addition, the set of TCs on its corresponding block section $Block_i$ is represented by $Set_{Circuits} := \{TC_i | i = 1,2,\dots,\mathfrak{B}\}$. Train transitions between two adjacent blocks, such as $Block_i$ and the post block $Block_{i+1}$, can be modeled considering time delays.

Let us consider two adjacent blocks as shown in Figure 6.4.(a); for instance, $Block_{current}$ is used for the current block $Block_i$, and $Block_{post}$ is used for its adjacent (post) block $Block_{i+1}$. Here, $Block_{current}$ is the i 'th block, where the train is currently on. $Block_{post}$ is the $i + 1$ 'th adjacent block, where the train moves on after the current block. This transition between adjacent blocks is sketched as in Figure 6.4.(b). Note that, after the train transits from its current block to the post block, this post block becomes a current block in the new case. The transition between adjacent blocks and moving in blocks cause time delays. These delays are considered exact durations in this study.

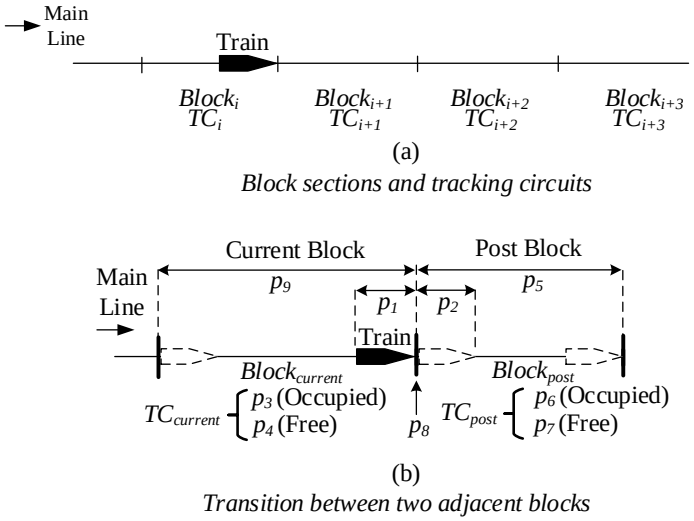


Figure 6.4. *Blocks and tracking circuits on a railway network*

In Figure 6.4.(b), a tracking circuit related to $Block_{current}$ is denoted by $TC_{current}$, and a tracking circuit related to $Block_{post}$ is denoted by TC_{post}. When a *TC* detects a train on its corresponding block section, this block is considered as occupied; in addition, in the case of not detecting a train, this block is considered as free. Let us model this block transition between adjacent block using TdAPN as given in Figure 6.5.

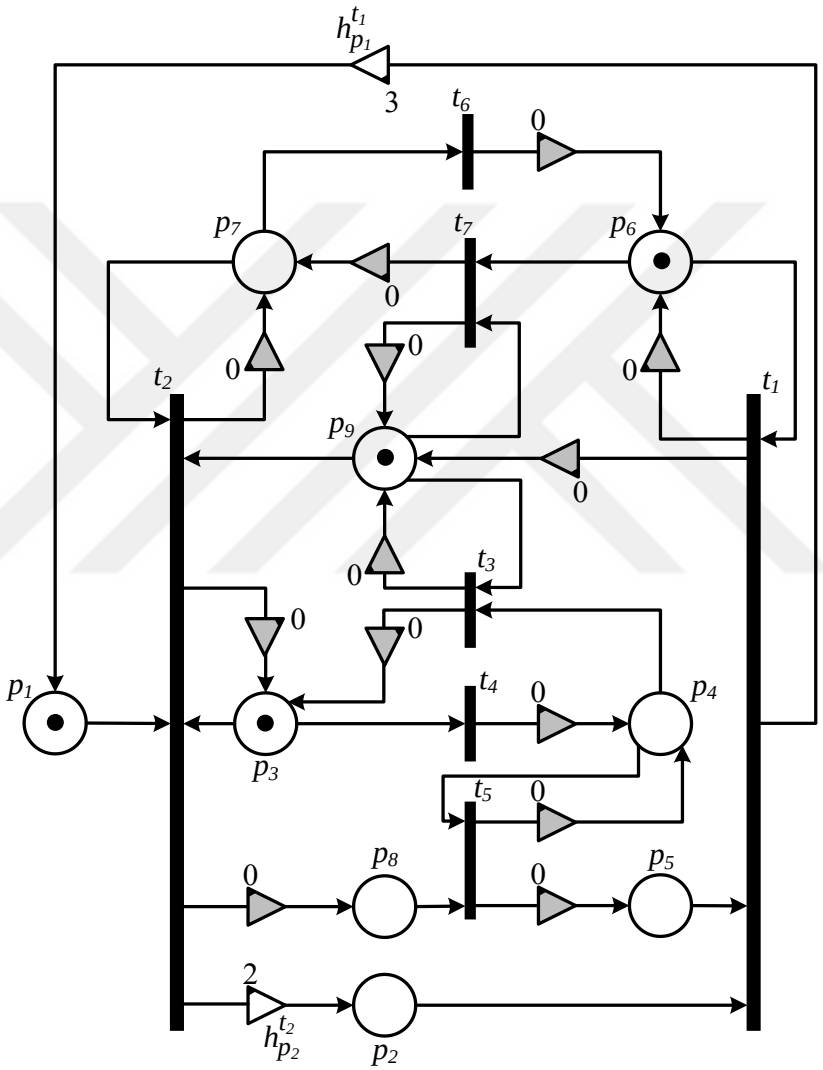


Figure 6.5. *Block transition between adjacent blocks modeled using TdAPN*

The Place-Stretched PN Equivalent of TdAPN in Figure 6.5 is also given in Figure 6.6. Here in Figure 6.6, there are 5 pair of transitions and places because of time delays.

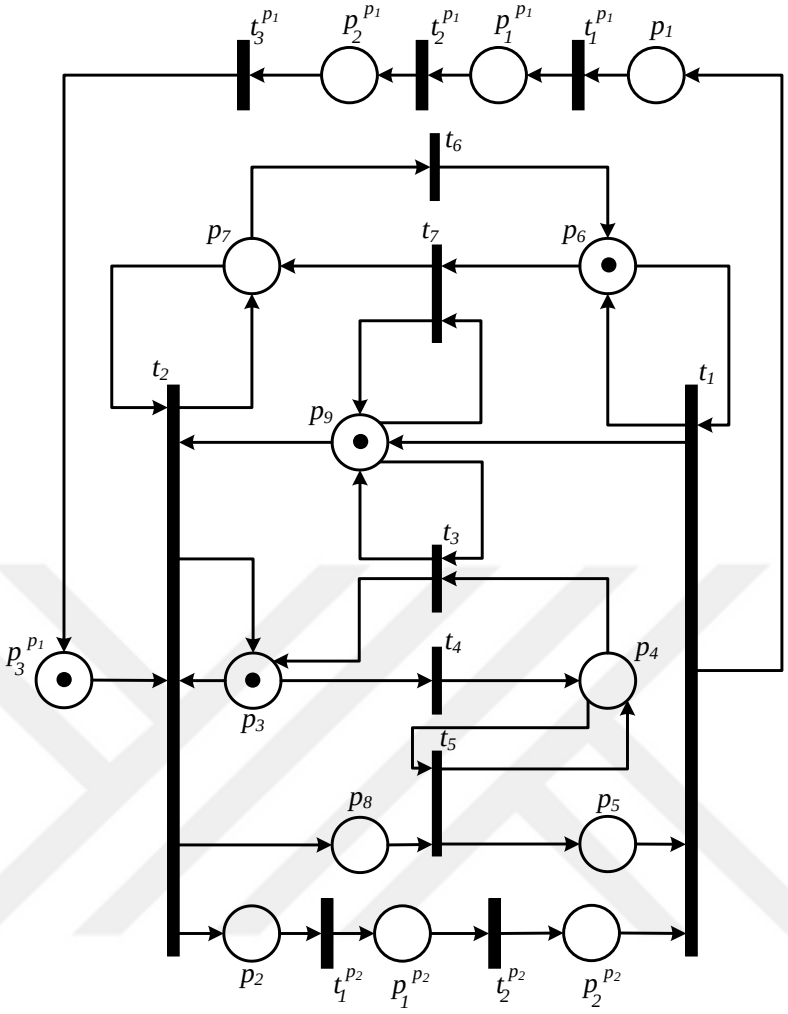


Figure 6.6. *Place-Stretched PN Equivalent of TdAPN in Figure 6.5*

For TdAPN in Figure 6.5, it is defined by $G_A(P, T, N, O, D, S_0)$. Here, the set of places is $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$, and the set of transitions is $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$. Input, output and time delay matrices are as follows:

$$N = \begin{bmatrix} 0&1&0&0&0&0&0 \\ 1&0&0&0&0&0&0 \\ 0&1&0&1&0&0&0 \\ 0&0&1&0&1&0&0 \\ 1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&1 \\ 0&1&0&0&0&1&0 \\ 0&0&0&0&1&0&0 \\ 0&1&1&0&0&0&1 \end{bmatrix}, O = \begin{bmatrix} 1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0 \\ 0&1&1&0&0&0&0 \\ 0&0&0&1&1&0&0 \\ 0&0&0&0&1&0&0 \\ 1&0&0&0&0&1&0 \\ 0&1&0&0&0&0&1 \\ 0&1&0&0&0&0&0 \\ 1&0&1&0&0&0&1 \end{bmatrix}, D = \begin{bmatrix} 3&0&0&0&0&0&0 \\ 0&2&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \end{bmatrix}.$$

The initial state is $S_0 = \{\boldsymbol{M}_0, \boldsymbol{\nabla^{R_0}}\}$, where $\boldsymbol{M}_0$=[1 0 1 0 0 1 0 0 1]' and $\boldsymbol{\nabla^{R_0}}$=[0 0]'. The set of time elements used in $\boldsymbol{\nabla^{R}}$ is $\nabla = \{h_{p_1}^{t_1}, h_{p_2}^{t_2}\}$. Physical meanings of places, transitions and time elements in these models are as follows:

- p_1 denotes that train is at the end of the current block.
- p_2 denotes that train is at the beginning of the post block.
- p_3 and p_4 denote the status of TC_i (the i^{th} block section is occupied and free, respectively).
- p_5 denotes that train is completely in the post block (completely means that no part of the train is in the current block).
- p_6 and p_7 denote the status of TC_{i+1} (the $(i+1)^{th}$ block section is occupied and free, respectively).
- p_8 denotes the start of the train's transition between adjacent blocks.
- p_9 denotes that train is completely in the current block (completely means that no part of the train is in the post block).
- t_1 denotes the event of assigning the post block as a new current block.
- t_2 denotes the event to start transition from the current block to its adjacent post block.
- t_3 and t_4 are sensory events to detect the train on the i^{th} block section whether it is occupied or not ,respectively. They are triggered by the sensor of the TC_i.
- t_5 is an event to verify the train's transition is completed.
- t_6 and t_7 are sensory events to detect the train on the $(i+1)^{th}$ block section whether it is occupied or not ,respectively. They are triggered by the sensor of the TC_{i+1}.
- $h_{p_1}^{t_1}$ denotes that train is moving in the current block and it will be at the end of the current block after its remaining time is elapsed. This operation takes 3 *ts*.
- $h_{p_2}^{t_2}$ denotes that train is transiting from the current block to the post block and it will complete its transition after its remaining time is elapsed. This operation takes 2 *ts*.

Here, time elements $h_{p_1}^{t_1}$ and $h_{p_2}^{t_2}$ represent dynamic situations. Time delays for time elements were arbitrarily chosen. These can be modified.

Algorithms of TdAPN (Algorithm 7.1) found 23 states of the reachability set $R_S(G_A, S_0)$ for the railway system as given in Appendix-3. Here, the number of states of TdAPN is equal to the number of states of the equivalent representation of Place-Stretched PN. The given TdAPN has 10 relaxed states. The reachability set was obtained as $R_S(G_A, S_0) = \{S_0, S_1, \dots, S_{22}\}$. 0.6396 second was required to construct the reachability set for TdAPN and 1.3610 seconds were required for the construction of TdAPN (total is 2.0006 seconds), while 29.9878 seconds were required to construct the reachability set for Place-Stretched PN and 2.4923 seconds were required for the construction of Place-Stretched PN (total is 32.4801 seconds). The construction time of the reachability set of TdAPN is shorter than the Place-Stretched PN. In this application, there is no deadlock state. The net is live and reversible. Moreover, the software also generated a timed-reachability tree by considering the relation between states in the set $R_S(G_A, S_0)$ as shown in Figure 6.7. Descriptions of this timed-reachability tree are similar to explanations for Figure 3.8.

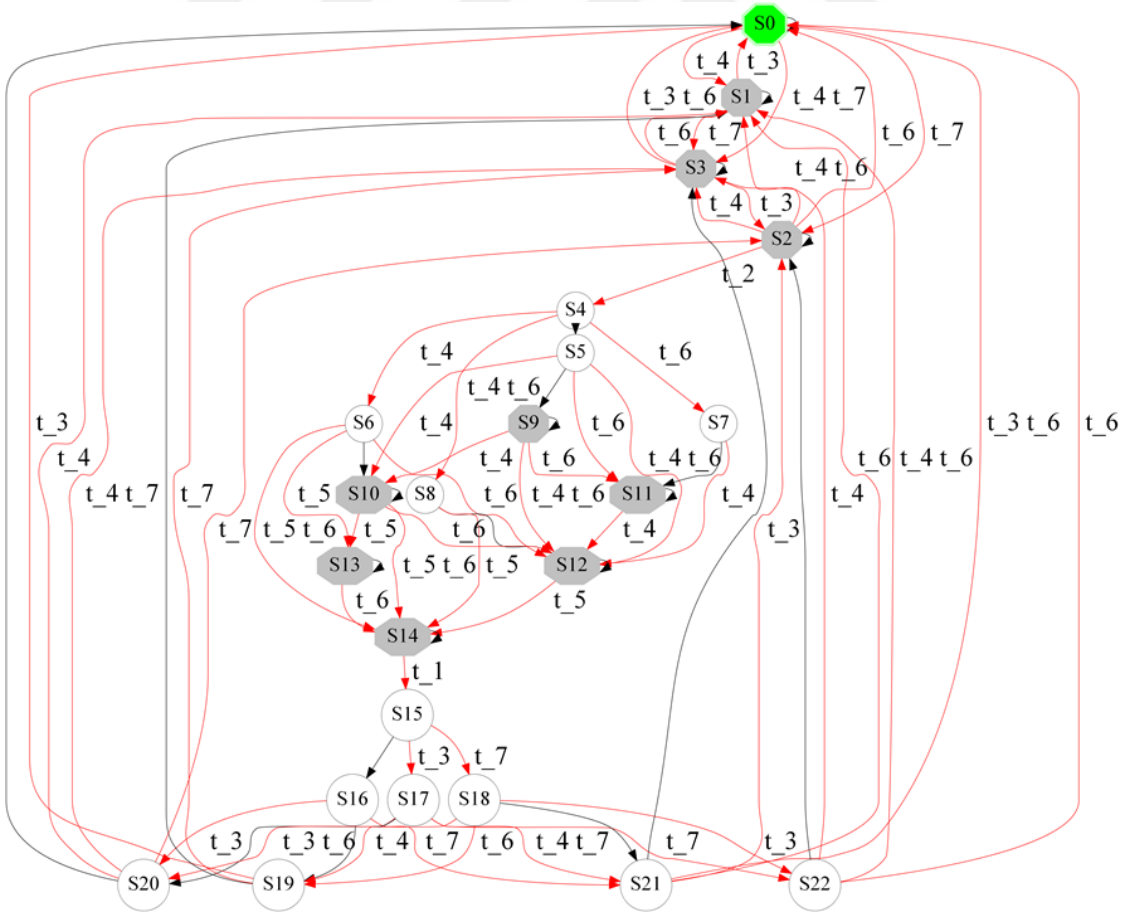


Figure 6.7. *Timed-reachability tree for TdAPN in Figure 6.5*

6.3. Automotive Systems

In this subsection, a cruise control system is considered as a case study of an automotive application [44, 45], and this is modeled using the proposed TdAPN modeling methodology. This system is an optional part of our modern cars today and its improved version that is the adaptive cruise control system will be indispensable for futuristic self-driving (autonomous) cars. This system is an embedded real-time system, and is composed of sensors and actuators to keep the speed of the vehicle at a certain (desired) speed. This system is activated or deactivated by user inputs via the buttons on the steering wheel. It is fact that the engine must be running to activate the system when the user presses the activation switch. This system records the speed of the vehicle and maintains the desired speed that the user sets up during the active period. This system is mostly deactivated when the accelerator pedal, the brake pedal or the button of deactivation is pressed.

The design schema of the cruise control system is given in [44, 45]. The cruise control system monitors the user inputs (buttons), the accelerator pedal, the brake pedal, the engine's status and *Global Positioning System* (GPS), which are called sensor-scan processes; it measures the current speed of vehicle from the wheel rotation; it computes necessary control values for the speed-adjustment; and based on these computed values, it updates related parameters and sends adjusted values (e.g. decrease/increase/maintain speed) to throttle actuator. Here, it needs a significant computational time to calculate control values and to update parameters [44]. In [44], the cruise control is implemented in two processors in parallel to execute operations the above, and the design is realized considering time delays of parallel tasks as shown in Figure 6.8; however, this is modeled using the basic untimed PN [44]. These operational tasks of the cruise control system is modeled by using TdAPN as illustrated in Figure 6.9.

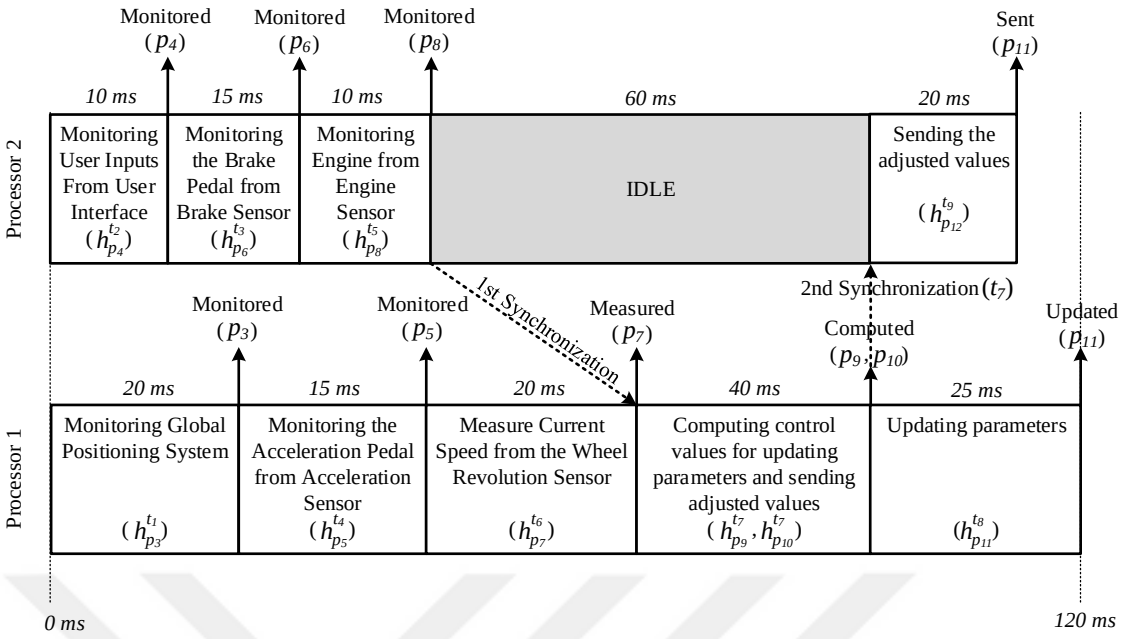


Figure 6.8. *Schedule of tasks and assignment to processors* [44]

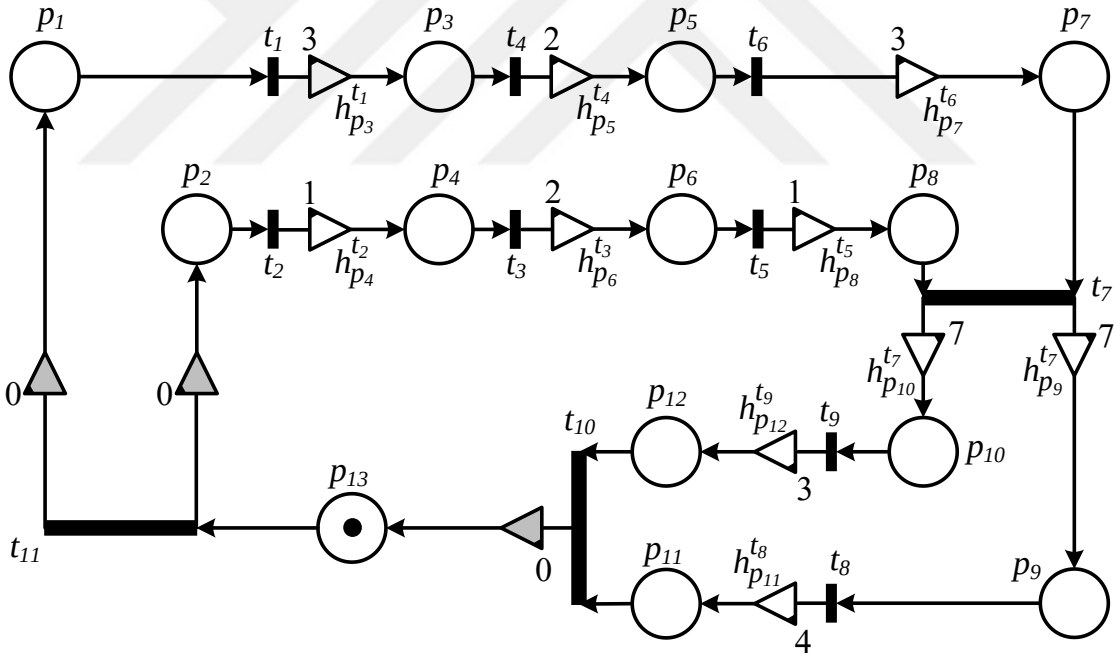


Figure 6.9. *Model of TdAPN for the cruise control in Figure 6.8*

The Transition-Stretched PN Equivalent of TdAPN in Figure 6.9 is also given in Figure 6.10. Here in this figure, there are 26 pair of places and transitions because of time delays.

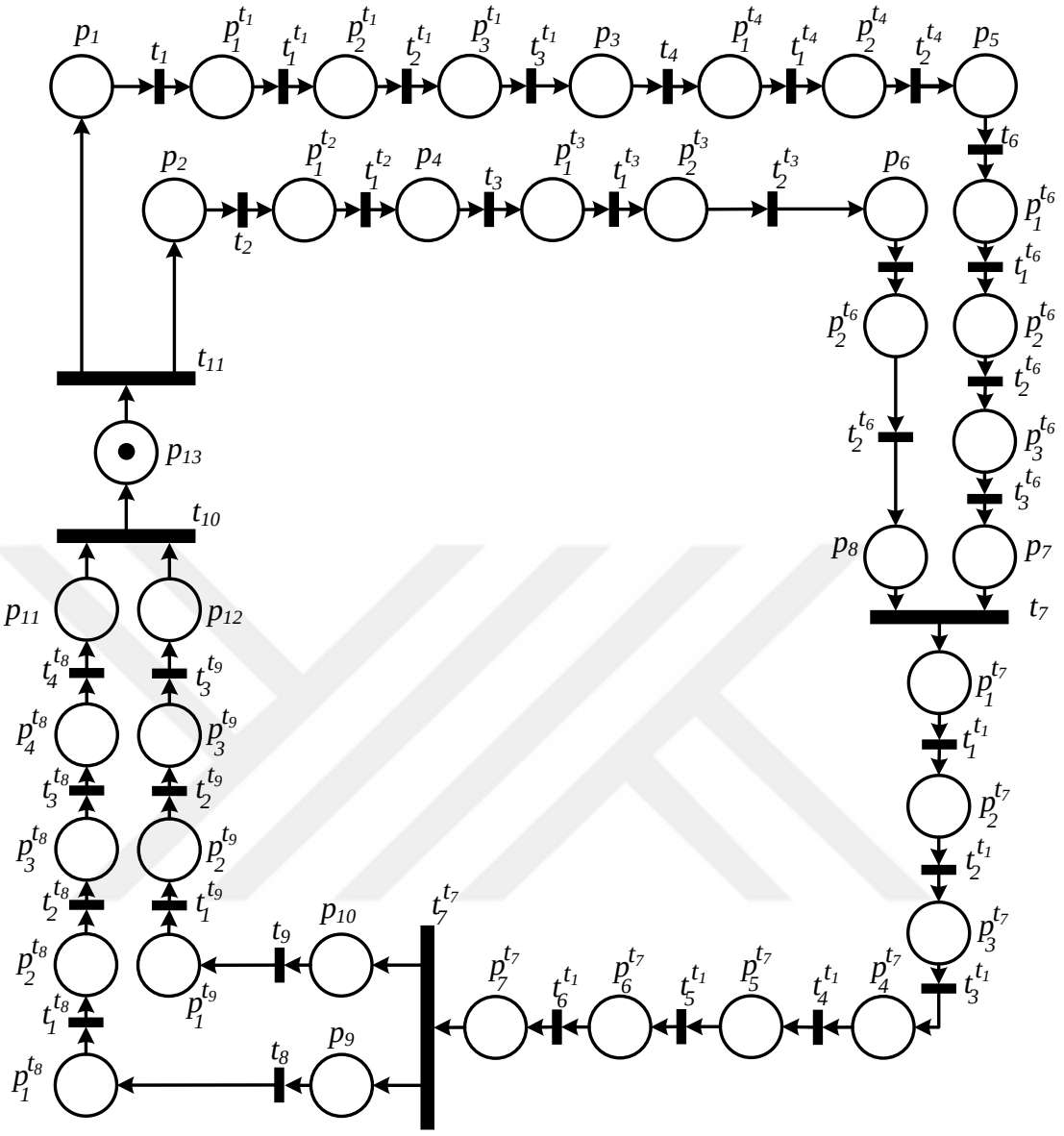


Figure 6.10. *Transition-Stretched PN Equivalent of TdAPN in Figure 6.9*

Note that time delays are in terms of milliseconds in Figure 6.8. These must be discretized using an appropriate sampling period in order to apply the proposed deterministic TdAPN. This sampling period can be five *ms* for one *ts* as long as all time-delays in Figure 6.8 are the factor of five; as a result, one *ts* is considered as five *ms*. The TdAPN model in Figure 6.9 is described by $G_A(P, T, N, O, D, S_0)$. Here, places are in a set form as $P = \{p_i | i = 1,2, \dots ,13\}$; $T = \{t_i | i = 1,2, \dots ,11\}$ is the set of transitions; and the input matrix N, output matrix O, and time delay matrix D in terms of *ts* are as follows:

$$N = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&1 \end{bmatrix}, O = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&0&0&0&1\\ 1&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&1&0 \end{bmatrix}, D = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 3&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&2&0&0&0&0&0&0&0&0\\ 0&0&2&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&3&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&7&0&0&0&0&0\\ 0&0&0&0&0&0&7&0&0&0&0&0\\ 0&0&0&0&0&0&0&4&0&0&0&0\\ 0&0&0&0&0&0&0&0&3&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix}.$$

The set of time elements is $\nabla = \{h_{p_3}^{t_1}, h_{p_4}^{t_2}, h_{p_6}^{t_3}, h_{p_5}^{t_4}, h_{p_8}^{t_5}, h_{p_7}^{t_6}, h_{p_9}^{t_7}, h_{p_{10}}^{t_7}, h_{p_{11}}^{t_8}, h_{p_{12}}^{t_9}\}$, where $D(p,t) \neq 0$ and $O(p,t) \neq 0$. $S_0 = \{\boldsymbol{M}_0, \boldsymbol{\nabla^{R_0}}\}$ is the initial state of G_A at $k_0 = 0$, where $\boldsymbol{M}_0$=[0 0 0 0 0 0 0 0 0 0 0 0 1]' and $\boldsymbol{\nabla^{R_0}}$=[0 0 0 0 0 0 0 0 0 0]'. Physical meanings for elements of TdAPN in Figure 6.9 are given in Table 6.2. According to this table, $M(k_0, p_{13}) = 1$ indicates that the source is ready to execute parallel tasks. $\boldsymbol{\nabla^{R_0}} = \mathbf{0}^{|\nabla|\times 1}$ indicates that there is currently no dynamic operation ($F_{pre}(k_0) = \emptyset$)).

Table 6.2. *Physical meanings for elements of TdAPN in Figure 6.9*

Element	Explanation	Status
p_1	Source for the processor 1 is available. (Dummy place)	**NT:** unavailable (no), **T:** available (yes).
p_2	Source for the processor 2 is available (Dummy place)	**NT:** unavailable (no), **T:** available (yes).
p_3	Global Positioning System is monitored.	**NT:** not monitored (no), **T:** monitored (yes).
p_4	User inputs are monitored.	**NT:** not monitored (no), **T:** monitored (yes).
p_5	Acceleration is monitored.	**NT:** not monitored (no), **T:** monitored (yes).
p_6	Brake is monitored.	**NT:** not monitored (no), **T:** monitored (yes).
p_7	Current speed is measured.	**NT:** not monitored (no), **T:** monitored (yes).
p_8	Engine is monitored.	**NT:** not monitored (no), **T:** monitored (yes).
p_9	Control values for updating parameters are computed.	**NT:** not computed (no), **T:** computed (yes).
p_{10}	Control values for adjusted speed values are computed.	**NT:** not computed (no), **T:** computed (yes).
p_{11}	Parameters are updated.	**NT:** not updated (no), **T:** updated (yes).
p_{12}	Adjusted values are sent.	**NT:** not sent (no), **T:** sent (yes).
p_{13}	Source is ready. (Dummy place)	**NT:** not ready (no), **T:** ready (yes).

***NT**: No token exists. **T**: Token exists.*

Table 6.2. (Continue) *Physical meanings for elements of TdAPN in Figure 6.9*

Element	Explanation	Status
t_1	Monitor Global Positioning System.	
t_2	Monitor user inputs.	
t_3	Monitor the brake pedal.	
t_4	Monitor the acceleration pedal.	
t_5	Monitor the engine.	
t_6	Measure the current speed.	
t_7	Compute control values.	
t_8	Update parameters.	
t_9	Send adjusted values.	
t_{10}	Sink (dummy transition).	
t_{11}	Source (dummy transition).	
$h_{p_3}^{t_1}$	Monitoring Global Positioning System. This operation takes 3 *ts*.	
$h_{p_4}^{t_2}$	Monitoring user inputs from the user interface. This operation takes 1 *ts*.	
$h_{p_6}^{t_3}$	Monitoring the brake pedal. This operation takes 2 *ts*.	
$h_{p_5}^{t_4}$	Monitoring the acceleration pedal. This operation takes 2 *ts*.	
$h_{p_8}^{t_5}$	Monitoring the engine. This operation takes 1 *ts*.	
$h_{p_7}^{t_6}$	Measuring current speed from the wheel revolution sensor. This operation takes 3 *ts*.	
$h_{p_9}^{t_7}$	Computing control values for updating parameters. This operation takes 7 *ts*.	
$h_{p_{10}}^{t_7}$	Computing control values for sending adjusted speed values. This operation takes 7 *ts*.	
$h_{p_{11}}^{t_8}$	Updating parameters. This operation takes 4 *ts*.	
$h_{p_{12}}^{t_9}$	Sending adjusted values. This operation takes 3 *ts*.	

Algorithms of TdAPN (Algorithm 7.1) found 134 states of the reachability set $R_S(G_A, S_0)$ for the automotive example shown in Appendix-4, where 21 states of the basic (untimed) PN were found in [44]. These are called relaxed states in TdAPN. The reachability set was obtained as $R_S(G_A, S_0) = \{S_0, S_1, \dots, S_{133}\}$. For this example, there is no deadlock state. The net is live and reversible. 22.8500 seconds were required to construct the reachability set for TdAPN and 1.6696 seconds were required for the construction of TdAPN (total is 24.5196 seconds). The software also generated a timed-reachability tree for the reachability set by considering the relation between states. The generated timed-reachability tree, which concerns $R_S(G_A, S_0)$, is shown in Figure 6.11. Descriptions of the timed-reachability tree are also similar to explanations for Figure 3.8. Results show that the design for the cruise control system makes it operate correctly as long as its TdAPN model is live and reversible.

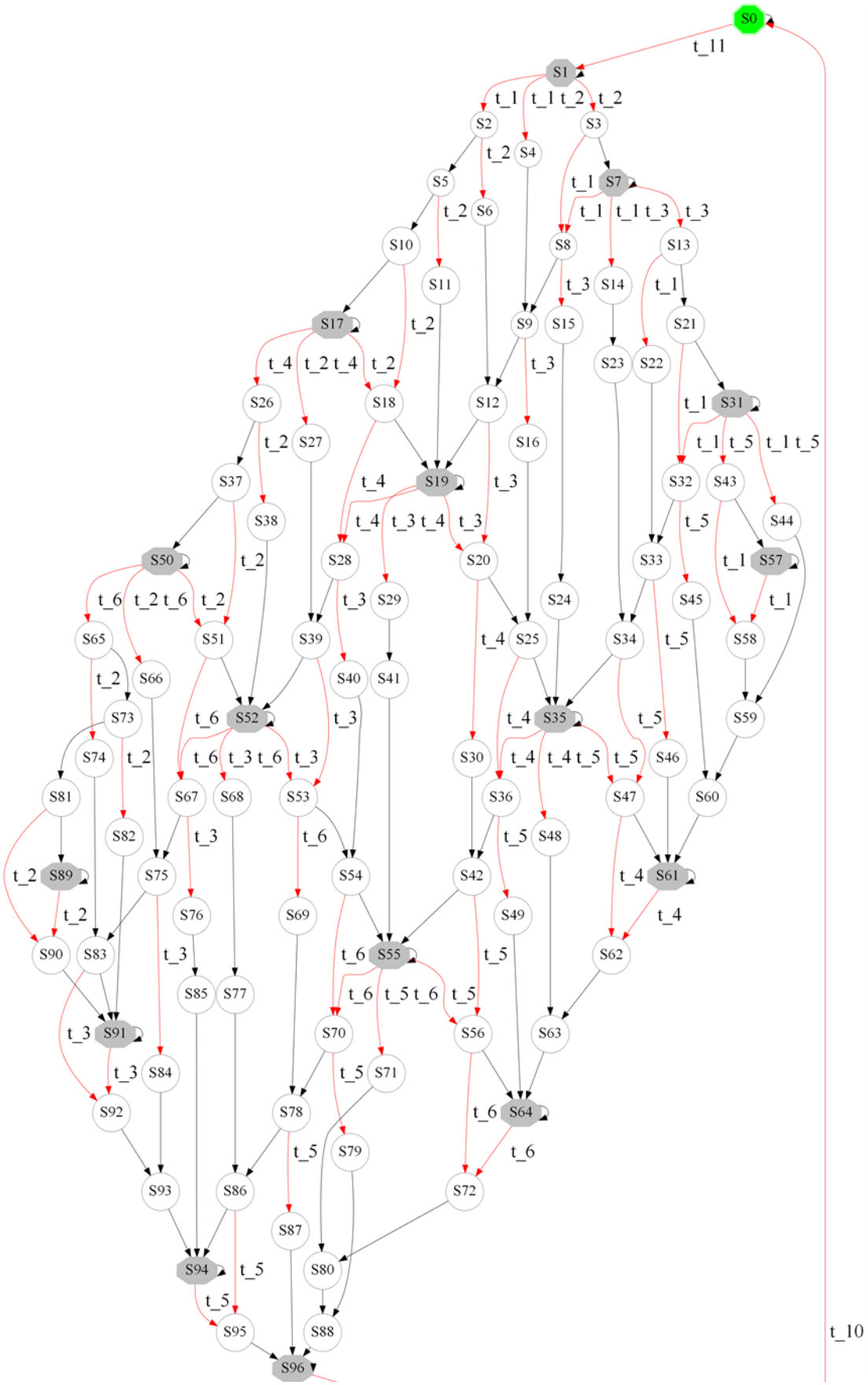


Figure 6.11. *Timed-reachability tree of TdAPN in Figure 6.9*

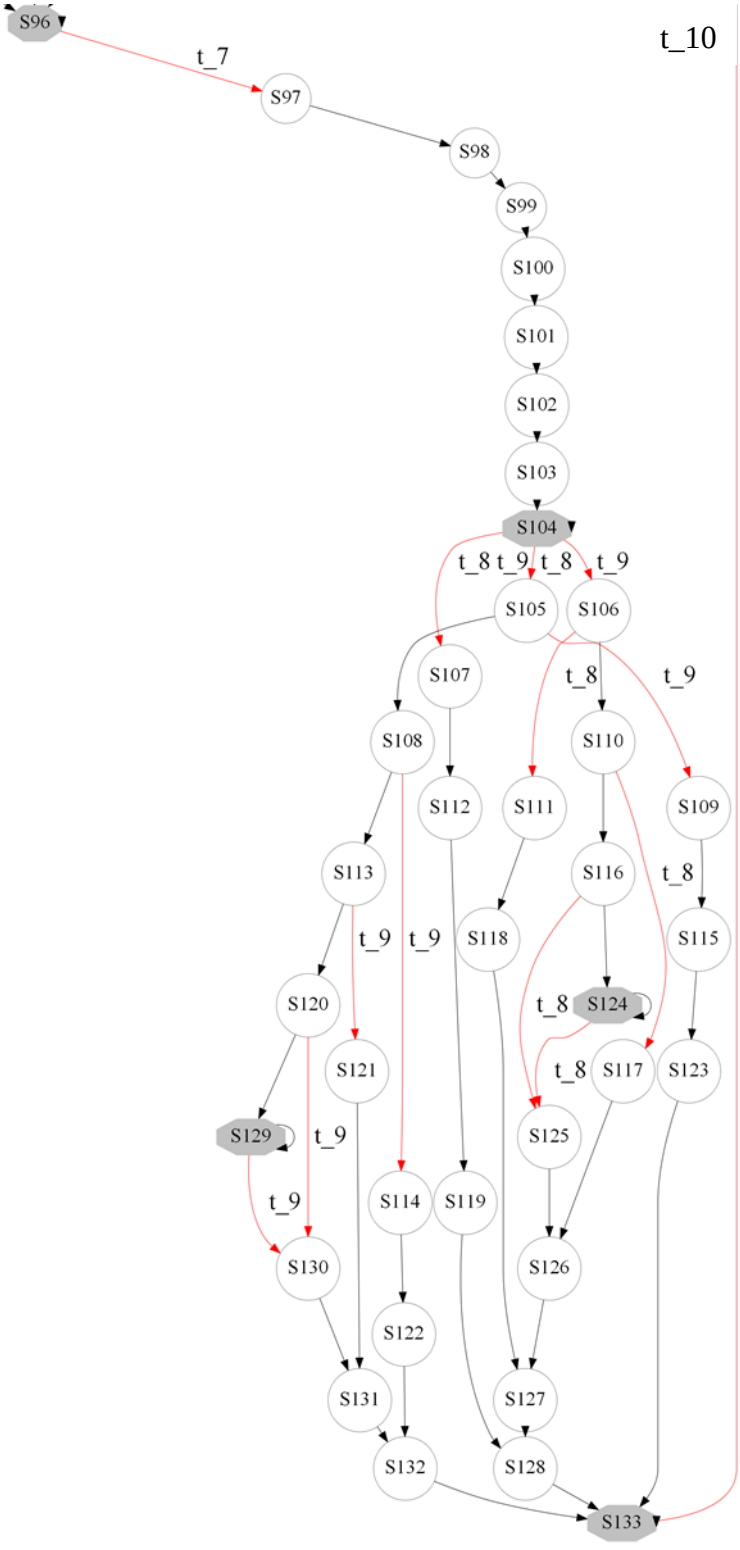


Figure 6.11.(Continue) *Timed-reachability tree of TdAPN in Figure 6.9*

7. ALGORITHMS FOR TIMED-ARC PETRI NETS

This chapter presents algorithms for the proposed TdAPN that construct the reachability set $R_S(G_A, S_0)$ of G_A and find values of the forbidden state controller's function based on the set of undesired states, i.e., $\mathcal{L}_0$ and $R_S(G_A, S_0)$. These are also implemented and simulated in the MATLAB; in addition, corresponding results are presented. Algorithms of constructing the reachability set for TdAPN have been presented in [31] by Yufka et.al.

7.1. Algorithms to Construct Reachability Set

In this section, algorithms calculating the next state $S(k+1)$ and constructing the reachability $R_S(G_A, S_0)$ set for TdAPNs are presented. In order to perform these, there are two parts in the software of the proposed TdAPN. These are called *Prepare-Initials Part* and *Main-Function Part* as shown in Figure 7.1. These are explained in the following subsections. *Prepare-Initials Part* is used for preparing initial inputs for *Main-Function Part*, while *Main-Function Part* is used for constructing $R_S(G_A, S_0)$; in addition it also constructs the set of deadlock states $\tilde{\mathcal{L}}_0$ and the set of next states $R_{next}(G_A, S_0)$ that includes the pair of arguments of the function $\rho\left(S, F_{pre}(S) \cup \phi\right)$ together with its resulting state $\tilde{S}$, where states S and $\tilde{S}$ are the member of the reachability set $R_S(G_A, S_0)$.

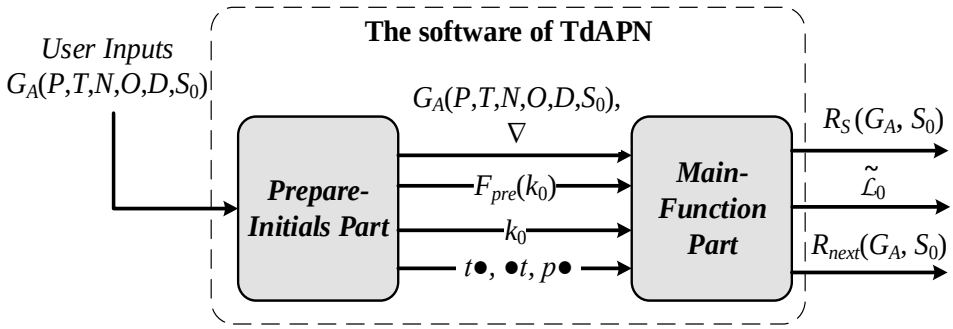


Figure 7.1. *Parts of the software of TdAPN to obtain the reachability set*

7.1.1. Prepare-initials part

Prepare-Initials Part is used to prepare initials for *Main-Function Part* using $G_A(P, T, N, O, D, S_0)$, which must be given to *Prepare-Initials Part* by the user. This part is used only at the initial time k_0. *Prepare-Initials Part* produces the following outputs:

- TdAPN's definition that is $G_A(P, T, N, O, D, S_0)$, and the set of time elements ∇,

- The set of previously activated firing processes at the initial state S_0 at time k_0 that is $F_{pre}(k_0)$,
- The initial time instant k_0 (k_0 is equal to zero as default if $F_{pre}(k_0)$ is an empty set. Otherwise, it means $k_0 > 0$),
- For the transition $t \in T$, the set of input places $\bullet t$ and the set of output place $t\bullet$; and for the place the $p \in P$, post set of transitions $p\bullet$.

Then, these outputs of *Prepare-Initials Part* are used by *Main-Function Part* as input information.

7.1.2. Main-function part

Main-Function Part is responsible from the state evaluation thus calculating next state $S(k+1) = \{\boldsymbol{M}(k+1), \boldsymbol{\nabla}^R(k+1)\}$ via discrete-time unit impulse functions as given in (3.3) and (3.4); as a result, it generates the reachability set $R_S(G_A, S_0)$; in addition, $\tilde{\mathcal{L}}_0$ and $R_{next}(G_A, S_0)$. These are outputs of *Main-Function Part*. $R_S(G_A, S_0)$, $\tilde{\mathcal{L}}_0$ and $R_{next}(G_A, S_0)$ will be used for the forbidden state controller in the next subsection. The detailed diagram of *Main-Function Part* is given in Figure 7.2.

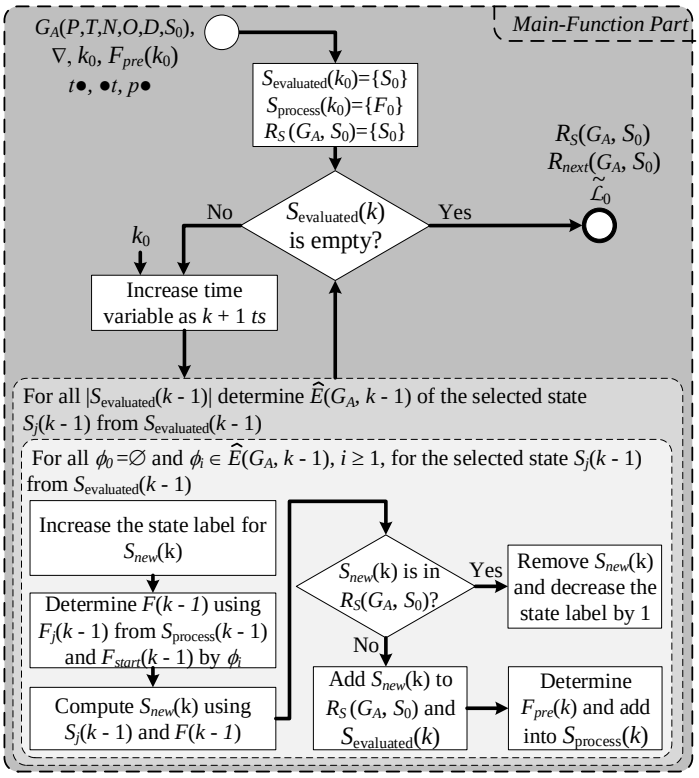


Figure 7.2. *Detailed diagram of Main-Function Part*

Here in Figure 7.2, the entities that belong to the initial time k_0 come from *Prepare-Initials Part*. The state evolution for $k > k_0$ is realized in *Main-Function Part*. Here, k is the discrete time variable, and states in $R_S(G_A, S_0)$ are obtained in the minimum time; as a result, k is stopped to increase by one *ts* when the last state of $R_S(G_A, S_0)$ is reached. Moreover, remember that each state in $R_S(G_A, S_0)$ is denoted and labeled as $S_j = \{\gamma_1(S_j) = \boldsymbol{M}_j, \gamma_2(S_j) = \boldsymbol{\nabla}^{\boldsymbol{R_j}}\}$ where $F(S_j)$ is $F_{pre}(S_j) \cup \phi$, while $\phi \in \hat{E}\left(G_A, \gamma_1(S_j)\right)$. In algorithms, F_j is used for denoting the set $F_{pre}(S_j)$ related to the state $S_j \in R_S(G_A, S_0)$. F_j is the set of transitions at state S_j whose firing processes are previously activated and not finished yet. The state of S_j at time k is represented as $S_j(k) = \{\boldsymbol{M}_j(k), \boldsymbol{\nabla}^{\boldsymbol{R_j}}(k)\}$ and the set of F_j at time k is represented as $F_j(k)$ in *Main-Function Part*; in addition, the set of sets of simultaneously-enabled transitions at $\boldsymbol{M}_j(k)$ at time k is $\hat{E}(G_A, k)$. In order to perform algorithmic computations, elements of $\hat{E}(G_A, k)$ are represented with their sub-indices as $\phi_i \in \hat{E}(G_A, k)$, where $i \in 1,2, \ldots, \left|\hat{E}(G_A, k)\right|$, and ϕ_i points out the i 'th element of $\hat{E}(G_A, k)$. Here, $|\{.\}|$ indicates the cardinality of the set $\{.\}$. It is also the zero sub-index is used for denoting no selection of $\phi \in \hat{E}(G_A, k)$; thus ϕ_0 stands for an empty set. The selected transitions in the set ϕ is added into the set $F_{start}(k)$ at time k such that $F(k)$ is constructed by $F_j(k) \cup F_{start}(k)$ at the state $S_j(k)$ according to the selection of $\phi \in \hat{E}(G_A, k)$.

In *Main-Function Part* in Figure 7.2, there is a main *while* loop, and there is a *for*-in-*for* loop in *while* loop. At time k_0, initials are k_0, $S(k_0)$ and $F_{pre}(k_0)$; furthermore, there is a counter for the state-label that is initially equal to zero. At each k, this counter is equal to the state-label of the last element in $R_S(G_A, S_0)$ that is initially $j = 0$ for S_0. For instance; at the beginning, the reachability set is $R_S(G_A, S_0) = \{S_0\}$ so that the counter for state-label is zero at time $k = k_0$. Moreover, in algorithms, in order to compute the next state using information about the present state, states that is not evaluated yet at time k is represented by $Set_{evaluated}(k)$, namely the set of sets of not yet evaluated states. $Set_{evaluated}(k)$ includes the set of states at time k, which will be used for the evaluation of the next state. This set is $Set_{evaluated}(k_0) = \{S_0\}$ at time k_0. In addition, $Set_{evaluated}(k+1)$ is constructed by next states of present states in $Set_{evaluated}(k)$. On the other hand, the set of sets of previously firing processes related to states in $Set_{evaluated}(k)$ at time k is represented by $Set_{process}(k)$. This set is initially

$Set_{process}(k_0) = \{F_0\}$, where F_0 is $F_{pre}(k_0)$. Subscripts of $F_{pre}(k)$ in $Set_{process}(k)$is suited to indices of $Set_{evaluated}(k)$. Based on the above information, the following steps are iteratively followed in *Main-Function Part* as:

- *While* loop checks the condition whether $Set_{evaluated}(k) = \emptyset$. If $Set_{evaluated}(k) \neq \emptyset$, then time variable k is increased by one *ts*, and the code enters the outer *for* loop. If $Set_{evaluated}(k) = \emptyset$, then $R_S(G_A, S_0)$ is obtained.
- Outer *for* loop counts all states $S_j(k-1) \in Set_{evaluated}(k-1)$. Note that k is previously increased by one *ts* in the *while* loop. The main evaluated state is currently $S_j(k-1)$. Then, the code enters inner *for* loop. In addition, when the code leaves outer *for* loop, $Set_{evaluated}(k)$, which includes newly created states added into $R_S(G_A, S_0)$ at time k, is generated and $Set_{process}(k)$ is also generated.
- Inner *for* Loop counts all $\phi \in \hat{E}(G_A, k-1)$ of $S_j(k-1)$. Initially, the counter for the state-label is increased by 1 and a temporary new state is created as $S_{new}(k)$. $F(k-1)$ is generated using previously activated firing processes in $F_j(k-1) \in Set_{process}(k-1)$ of $S_j(k-1)$ and $F_{start}(k-1)$ by ϕ; as a result, $F(k-1)$ is created including $F_j(k-1)$ and newly started firing processes of $t \in \phi$ in $F_{start}(k-1)$. The new next state $S_{new}(k)$ is computed by using the function of $\rho\left(S_j(k-1), F(k-1)\right)$. Then, the duplication is checked for $S_{new}(k)$ whether it is in the reachability set $R_S(G_A, S_0)$. If $S_{new}(k) \in R_S(G_A, S_0)$, then $S_{new}(k)$ is deleted and the counter for the state-label is decreased by one. If $S_{new}(k) \notin R_S(G_A, S_0)$, then it is added into $R_S(G_A, S_0)$ at time k. It is also that if $S_{new}(k)$ is a deadlock state, then it is added into $\tilde{\mathcal{L}}_0$. Completed firing processes are also checked for $F_{pre}(k)$ that is $F_{new}(k)$ related to the state $S_{new}(k)$, and $F_{new}(k)$ is added into $Set_{process}(k)$. The pair of arguments of the function $\rho\left(S, F_{pre}(S) \cup \phi\right)$ together with its resulting state $\tilde{S}$ is added into $R_{next}(G_A, S_0)$.

Main-Function Part has a *main algorithm* (given in Algorithm 7.1) and sub-algorithms as given in Figure 7.3.

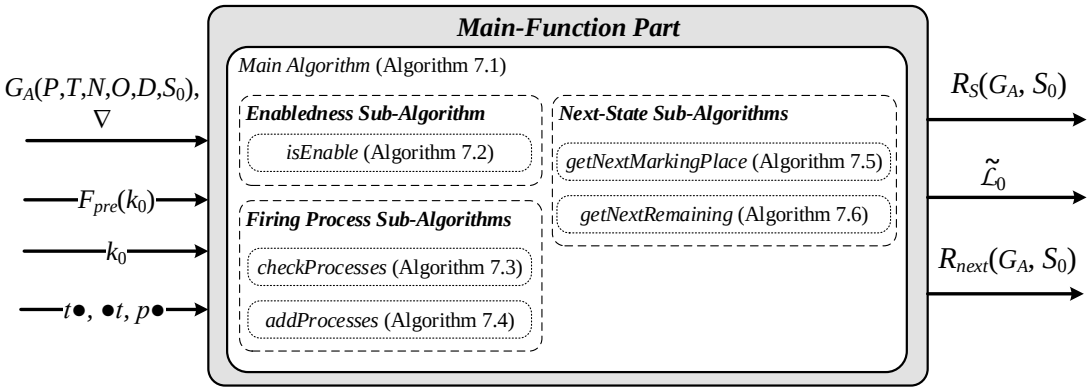


Figure 7.3. *Main algorithm and its sub-algorithms for TdAPN*

Here in *main algorithm* (Algorithm 7.1), the following sub-algorithms are used as:

- *Enabledness Sub-Algorithm* which is *isEnable* (given in Algorithm 7.2),
- *Firing Process Sub-Algorithms* which are *checkProcesses* (given in Algorithm 7.3) and *addProcesses* (given in Algorithm 7.4),
- *Next State Sub-Algorithms* which are *getNextMarkingPlace* (given in Algorithm 7.5) and *getNextRemaining* (given in Algorithm 7.6),
    - *getNextMarkingPlace* (Algorithm 7.5) is used to calculate $\boldsymbol{M}(k+1)$ according to the selection of $\phi \in \hat{E}(G_A, k)$ or $\phi = \emptyset$.
    - *getNextRemaining* (Algorithm 7.6) is used to calculate $\boldsymbol{\nabla}^R(k+1)$ by selecting $\phi \in \hat{E}(G_A, k)$ and $\emptyset$.

Algorithm 7.1. *Main algorithm of Main-Function Part*

Main-Algorithm	*ALGORITHM I - main*
Inputs	$G_A(P,T,N,O,D,S_0)$, ∇, k_0, $F_{pre}(k_0)$, $\bullet t$, $t\bullet$, $p\bullet$
Outputs	$R_S\,(G_A,S_0)$, $R_{next}\,(G_A,S_0)$, $\tilde{\mathcal{L}}_0$

1. $Set_{evaluated}(k_0) = \{S_0\ for\ S(k_0)\}$
2. $Set_{process}(k_0) = \{F_0\ for\ F_{pre}(k_0)\}$
3. $R_S\,(G_A,S_0) = Set_{evaluated}(k_0)$
4. $\tilde{\mathcal{L}}_0 = \emptyset$
5. $k = k_0$
6. while $Set_{evaluated}(k) \neq \emptyset$ do
7. : counter_init_state_label = last element in $R_S\,(G_A,S_0)$
8. : counter_state_label = counter_init_state_label
9. : $k = k + 1$
10. : for j = 1 to $|Set_{evaluated}(k-1)|$ do
11. : : $M_j(k-1) = \gamma_1\big([Set_{evaluated}(k-1)]_j\big)$
12. : : $\nabla^{R_j}(k-1) = \gamma_2\big([Set_{evaluated}(k-1)]_j\big)$
13. : : $F_j(k-1) = \big[Set_{process}(k-1)\big]_j$

Algorithm 7.1. (Continue) *Main algorithm of Main-Function Part*

Main-Algorithm	*ALGORITHM I - Main Algorithm*
Inputs	$G_A(P,T,N,O,D,S_0)$, ∇, k_0, $F_{pre}(k_0)$, $\bullet t$, $t\bullet$, $p\bullet$
Outputs	$R_S\ (G_A,S_0)$, $R_{next}\ (G_A,S_0)$, $\tilde{\mathcal{L}}_0$

: : $F_{pre}(k-1)$ =$F_j(k-1)$
: : $\hat{E}(G_A,k-1)$ = ***isEnable***$(P, T, N, M_j(k-1), F_{pre}(k-1),\ k-1, \bullet t, p\bullet)$
: : for i = 0 to $|\hat{E}(G_A,k-1)|$ do
: : : counter_state_label = counter_state_label + 1
: : : $S_{new} = S_{\text{counter\_state\_label}}$
: : : if i = 0 then
: : : : $\phi_i = \emptyset$
: : : else
: : : : $\phi_i = \left[\hat{E}(G_A,k-1)\right]_i$
: : : end if
: : : $F_{start}(k-1)$ = ***addProcesses***$(T, \phi_i, k-1)$
: : : $F(k-1) = F_{pre}(k-1) + F_{start}(k-1)$
: : : $M_{new}(k)$ = ***getNextMarkingPlace***$(P, T, N, O, D, k, M_j(k-1), F(k-1))$
: : : if $\nabla \neq \emptyset$ then
: : : : $\nabla^R_{new}(k)$ = ***getNextRemaining***$(P, T, \nabla, D,\ k, \nabla^{R_j}(k-1), F(k-1))$
: : : else
: : : : $\nabla^R_{new}(k)$ = [ ] ([ ] denotes an *empty vector with no dimension*)
: : : end if
: : : flag_duplication = false
: : : for x = 1 to $|R_S\ (G_A,S_0)|$ do
: : : : if $[R_S\ (G_A,S_0)]_x = S_{new}(k)$ then
: : : : : $Set_{next} \leftarrow \{([Set_{evaluated}(k-1)]_j, \phi_i), [R_S\ (G_A,S_0)]_x\}$
: : : : : delete $S_{new}(k)$
: : : : : flag_duplication = true
: : : : : counter_state_label = counter_state_label – 1
: : : : : break
: : : : end if
: : : end for
: : : if flag_duplication = true then
: : : : continue
: : : else
: : : : $R_S\ (G_A,S_0) \leftarrow S_{new}(k)$
: : : : $F_{new}(k)$ = ***checkProcesses***$(P, T, D, F(k-1), k)$
: : : : $F_{pre}(k) = F_{new}(k)$
: : : : $Set_{process}(k) \leftarrow F_{pre}(k)$
: : : : if $\nabla^R_{new}(k) = 0^{|\nabla| x 1}$ and $\hat{E}(G_A,k) = \emptyset$ then
: : : : : $\tilde{\mathcal{L}}_0 \leftarrow S_{new}$
: : : : end if
: : : : $R_{next}\ (G_A,S_0) \leftarrow \{([Set_{evaluated}(k-1)]_j, \phi_i), S_{new}(k)\}$
: : : end if
: : end for
: end for
: if counter_init_state_label +1 ≥ counter_state_label then
: : $Set_{evaluated}(k) = \{S_{\text{counter\_init\_state\_label}+1} : S_{\text{counter\_state\_label}}\}$
: else
: : $Set_{evaluated}(k) = \emptyset$
: end if
end while

***Enabledness Sub-Algorithm*:** This sub-algorithm is called and runned under *main algorithm* (Algorithm 7.1) of *Main-Function Part*. In order to determine the enabled transitions at $\boldsymbol{M}$(k) at time k, the sub-algorithm, namely *isEnable* (Algorithm 7.2), is developed using definitions in (3.1) and (3.2). Inputs of this sub-algorithm are P, T, N, $\boldsymbol{M}(k)$, $F(k)$, k, $\bullet t$ and $p\bullet$. Its output is the set of sets of simultaneously-enabled transition $\hat{E}(G_A, k)$ that also includes transitions in the set of transitions $E(G_A, k)$. The following steps are performed in order as:

- First of all, the sub-algorithm *isEnable* (Algorithm 7.2) finds the set of enabled transitions that is $E(G_A, k)$ based on the condition given in (3.1).
- Then, it checks all combinations (E_{combs} in the code, which refers to $\hat{E}(G_A, k) \subset 2^{E(G_A,k)} \backslash \emptyset$) of transitions $t \in E(G_A, k)$ in order to determine the set of simultaneously-enabled transitions, i.e., $\phi \subseteq E(G_A, k)$ based on the condition given in (3.2).
- Finally, it constructs the set of sets of simultaneously-enabled transitions, i.e., $\hat{E}(G_A, k)$.

Algorithm 7.2. *Enabledness sub-algorithm*

Sub-Algorithm	*ALGORITHM II - isEnable*
Inputs	$P, T, N, \boldsymbol{M}(k), F(k), k, \bullet t, p\bullet$
Outputs	$\hat{E}(G_A, k)$

1. $E(G_A, k) = \emptyset$
2. $\hat{E}(G_A, k) = \emptyset$
3. $t_{active} = \{t | t^{\lambda} \in F(k), \lambda \leq k\}$
4. for i = 1 to $|T|$ do
5. : counter_E = 0
6. : for j = 1 to $|P|$ do
7. : : if $[P]_j \in \bullet[T]_i$ then
8. : : : if $N([P]_j, [T]_i) \geq 1$ then
9. : : : : if $M([P]_j, k) \geq N([P]_j, [T]_i)$ then
10. : : : : : counter_E = counter_E+1
11. : : : : end if
12. : : : end if
13. : : end if
14. : end for
15. : if counter_E = $|\bullet[T]_i|$ then
16. : : $E(G_A, k) \leftarrow [T]_i$
17. : end if
18. end for
19. : $|E_{combs}| = \binom{|E(G_A,k)|}{z}$
20. for z = 1 to $|E(G_A, k)|$ do

Algorithm 7.2. (Continue) *Enabledness Sub-Algorithm*

Sub-Algorithm	*ALGORITHM II - isEnable*
Inputs	$P, T, N, \boldsymbol{M}(k), F(k), k, \bullet t, p\bullet$
Outputs	$\hat{E}(G_A, k)$

21. : for y = 1 to $|E_{combs}|$ do
22. : : $P_{union} = \emptyset$
23. : : flag_multi = 0
24. : : for j = 1 to $\left|[E_{combs}]_y\right|$ do
25. : : : $P_{union} \leftarrow P_{union} \cup \bullet\left[[E_{combs}]_y\right]_j$
26. : : end for
27. : : for i = 1 to $|P_{union}|$ do
28. : : : N_sum = 0
29. : : : $P_{union} \leftarrow P_{union} \cup \left[[E_{combs}]_y\right]_j$
30. : : : for x = 1 to $|[P_{union}]_i\bullet|$ do
31. : : : : if $[[P_{union}]_i\bullet]_x \in [E_{combs}]_y$ then
32. : : : : : if $N([P_{union}]_i, [[P_{union}]_i\bullet]_x) \geq 1$ then
33. : : : : : : N_sum += $N([P_{union}]_i, [[P_{union}]_i\bullet]_x)$
34. : : : : : : if $M([P_{union}]_i, k) <$ N_sum then
35. : : : : : : : flag_multi = 1
36. : : : : : : : break
37. : : : : : : end if
38. : : : : : else
39. : : : : : : flag_multi = 1
40. : : : : : : break
41. : : : : : end if
42. : : : : end if
43. : : : end for
44. : : : if flag_multi = 1 then
45. : : : : break
46. : : : end if
47. : : end for
48. : : if flag_multi = 0 then
49. : : : $\hat{E}(G_A, k) \leftarrow [E_{combs}]_y$
50. : : end if
51. : end for
52. end for
53. for i = 1 to $|\hat{E}(G_A, k)|$ do
54. : for j =1 to $\left|\left[\hat{E}(G_A, k)\right]_j\right|$ do
55. : : if $\left[\hat{E}(G_A, k)\right]_i \neq \emptyset$ then
56. : : : if $\left[\left[\hat{E}(G_A, k)\right]_i\right]_j \in t_{active}$ then
57. : : : : $\hat{E}(G_A, k) = \hat{E}(G_A, k) \setminus \left[\hat{E}(G_A, k)\right]_i$
58. : : : : $E(G_A, k) = E(G_A, k) \setminus \left[\left[\hat{E}(G_A, k)\right]_i\right]_j$
59. : : : : break
60. : : : end if
61. : : end if
62. : end for
63. end for

***Firing Process Sub-Algorithms*:** These sub-algorithms are called and runned under *main algorithm* (Algorithm 7.1) of *Main-Function Part*. A firing process $t^{\lambda} \in F_{start}(\lambda)$, $F_{start}(\lambda) \subseteq F(\lambda)$ starts at time $k = \lambda$ and it is completed at time $k = \lambda + d_t$ such that $t^{\lambda} \notin F_{pre}(\lambda + d_t)$, $F_{pre}(\lambda + d_t) \subseteq F(\lambda + d_t)$. In order to check the firing processes $t^{\lambda} \in F(k)$ whether it is completed at time $k + 1$ and in the set $F_{pre}(k + 1)$, the sub-algorithm, namely *checkProcesses* (Algorithm 7.3), is developed. Inputs of this sub-algorithm are $P, T, D,\ F(k-1)$ and k. Its output is the set of transitions whose firing processes continue, i.e., $F_{pre}(k) \subseteq F(k)$. In TdAPN, it is also checked whether a new firing process t^{λ} is to be added into the set $F_{start}(k)$ at time $k = \lambda$. For this purpose, the sub-algorithm, namely *addProcesses* (Algorithm 7.4), is also developed. Its inputs are T, ϕ and k, and its output is the set of transitions whose firing process is newly started at time k, i.e., $F_{start}(k) \subseteq F(k)$.

Algorithm 7.3. *Firing process sub-algorithm to check completed firing processes*

Sub-Algorithm	*ALGORITHM III - checkProcesses*
Inputs	$P, T\ D, F(k-1), k$
Outputs	$F_{pre}(k)$

1. $t_{active} = \{t | t^{\lambda} \in F(k-1), \lambda \leq k-1\}$
2. for i = 1 to $|T|$ do
3. : if $[T]_i \in t_{active}$ then
4. : if $[T]_i \in t_{active}$ then
5. : : if $k = \lambda^{[T]_i} + \max_{p \in [T]_i^{\bullet}} \{D(p, [T]_i)\}$ then
6. : : : $F_{pre}(k) = F_{pre}(k) \setminus ([T]_i)^{\lambda^{[T]_i}}$
7. : : end if
8. : end if
9. end for

Algorithm 7.4. *Firing process sub-algorithm to add new firing processes*

Sub-Algorithm	*ALGORITHM IV - addProcesses*
Inputs	T, ϕ, k
Outputs	$F_{start}(k)$

1. $F_{start}(k) = \emptyset$
2. $\lambda = k$
3. for j = 1 to $|T|$ do
4. : if $[T]_j \in \phi$ then
5. : : $F_{start}(k) \leftarrow ([T]_j)^{\lambda}$
6. : end if
7. end for

***Next State Sub-Algorithms*:** These sub-algorithms are called and runned under *main algorithm* (Algorithm 7.1) of *Main-Function Part*. First of all, in order to calculate the next marking vector $\boldsymbol{M}(k+1)$ as given in (3.3), the sub-algorithm, namely *getNextMarkingPlace* (Algorithm 7.5) is developed. Inputs of this sub-algorithm are P, T, N, O, D, $\boldsymbol{M}(k)$, $F(k)$ and k. Its output is the next marking vector $\boldsymbol{M}(k+1)$. The sub-algorithm *getNextMarkingPlace* (Algorithm 7.5) uses discrete-time unit impulse functions $\delta^{N}_{[P]_j}$ and $\delta^{O}_{[P]_j}$ to compute $\boldsymbol{M}(\mathrm{k}+1)$. Here, for the firing process $t^{\lambda} \in F(k)$, $\delta^{N}_{[P]_j}$ is used for a multiplier for denoting the (p_j, t) 'th element of N. Similarly, $\delta^{O}_{[P]_j}$ is used for a multiplier for denoting the (p_j, t) 'th element of O.

Algorithm 7.5. *Next state sub-algorithm to calculate next marking vector*

Sub-Algorithm	*ALGORITHM V - getNextMarkingPlace*
Inputs	$P, T, N, O, D, k, \boldsymbol{M}(k), F(k)$
Outputs	$\boldsymbol{M}(k+1)$

1. $t_{active} = \{t | t^{\lambda} \in F(k), \lambda \leq k\}$
2. if $t_{active} \neq \emptyset$ then
3. : $M_{sum} = 0_{|P|x1}$
4. : for i = 1 to $|t_{active}|$ do
5. : : for j = 1 to $|P|$ do
6. : : : $\delta^{N}_{[P]_j} = \delta^{O}_{[P]_j} = 0$
7. : : : if $k = \lambda^{[t_{active}]_i}$ then
8. : : : : $\delta^{N}_{[P]_j} = 1$
9. : : : end if
10. : : : if $k = \lambda^{[t_{active}]_i} + D([P]_j, [t_{active}]_i)$ then
11. : : : : $\delta^{O}_{[P]_j} = 1$
12. : : : end if
13. : : : $[M_{sum}]_j = [M_{sum}]_j + \delta^{O}_{[P]_j} * O([P]_j, [t_{active}]_i) - \delta^{N}_{[P]_j} * N([P]_j, [t_{active}]_i)$
14. : : end for
15. : end for
16. : $M(k+1) = M(k) + M_{sum}$
17. else
18. : $M(k+1) = M(k)$
19. end if

Moreover, in order to calculate the next remaining time vector $\boldsymbol{\nabla}^{R}(k+1)$ as given in (3.4), the sub-algorithm, namely *getNextRemaining* (Algorithm 7.6) is developed. Inputs of this sub-algorithm are P, T, ∇, D, $\boldsymbol{\nabla}^{\boldsymbol{R}}(k)$, $F(k)$ and k. Its output is the next remaining time vector $\boldsymbol{\nabla}^{\boldsymbol{R}}(k+1)$. The sub-algorithm *getNextRemaining* (Algorithm 7.6) uses discrete-time unit impulse function $\delta_{[\nabla]_j}$ and the sum of discrete-time unit impulses

to compute $\boldsymbol{\nabla^R}(\mathrm{k}+1)$. Here, $\delta_{[\nabla]_j}$ is used for a multiplier for the (p, t) 'th element of D, where the j 'th time element h_p^t in the set ∇, and p is the place index of h_p^t while t stands for the firing process $t^\lambda \in F(k)$.

Algorithm 7.6. *Next state sub-algorithm to calculate next remaining time vector*

Sub-Algorithm	*ALGORITHM VI - getNextRemaining*
Inputs	$P, T, \nabla, D,\ k, \boldsymbol{\nabla}^R(k), F(k)$
Outputs	$\boldsymbol{\nabla}^R(k+1)$

1. $t_{active} = \{t | t^\lambda \in F(k), \lambda \leq k\}$
2. if $t_{active} \neq \emptyset$ then
3. : ${\nabla^R}_{sum} = 0_{|\nabla| x 1}$
4. : for i = 1 to $|t_{active}|$ do
5. : : for j = 1 to $|\nabla|$ do
6. : : : $\delta_{[\nabla]_j} = 0$
7. : : : if $k = \lambda^{[t_{active}]_i}$ then
8. : : : : $\delta_{[\nabla]_j} = 1$
9. : : : end if
10. : : : $[{\nabla^R}_{sum}]_j = [{\nabla^R}_{sum}]_j + \delta_{[\nabla]_j} * D(p\ part\ of\ [\nabla]_j, [t_{active}]_i)$
11. : : : for $l = \lambda^{[t_{active}]_i} + 1$ to $\lambda^{[t_{active}]_i} + D(p\ part\ of\ [\nabla]_j, [t_{active}]_i)$ then
12. : : : : if $k = l$ then
13. : : : : : $[{\nabla^R}_{sum}]_j = [{\nabla^R}_{sum}]_j - 1$
14. : : : : : break
15. : : : : end if
16. : : : end for
17. : : end for
18. : end for
19. : $\nabla^R(k+1) = \nabla^R(k) + {\nabla^R}_{sum}$
20. else
21. : $\nabla^R(k+1) = \nabla^R(k)$
22. end if

Using sub-algorithms *getNextMarkingPlace* (Algorithm 7.5) and *getNextRemaining* (Algorithm 7.6), the next state of TdAPN at time $k+1$ is computed as $S(k+1) = \{\boldsymbol{M}(k+1), \boldsymbol{\nabla}^R(k+1)\}$.

7.2. Algorithms to Construct Forbidden State Controller

In this section, algorithms to construct forbidden state controller, which avoid undesired states, are developed; as a result, a new part, namely *Forbidden State-Controller Part*, is created and integrated to *Main-Function Part* as shown in Figure 7.4.

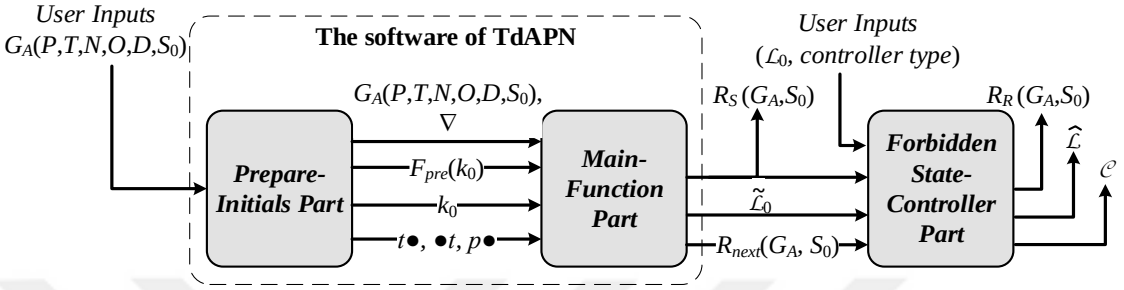


Figure 7.4. *Forbidden State-Controller Part for the software of TdAPN*

Forbidden State-Controller Part generates the expanded set of undesired states $\hat{\mathcal{L}}$ from the set of undesired states $\mathcal{L}_0$. Any set of $\mathcal{L}_0$ defined by the user or based on the desired behavior can be used in the controller. *Forbidden State-Controller Part* allows the user to select the controller type, where the controller can be a *deadlock avoidance controller*, or it can be a *reversibility enforcement controller* that enforces the system reversible and deadlock-free, or it avoids any undesired states defined by the user.

Forbidden State-Controller Part has a *main-controller algorithm* (given in Algorithm 7.7) and sub-algorithms as given in Figure 7.5.

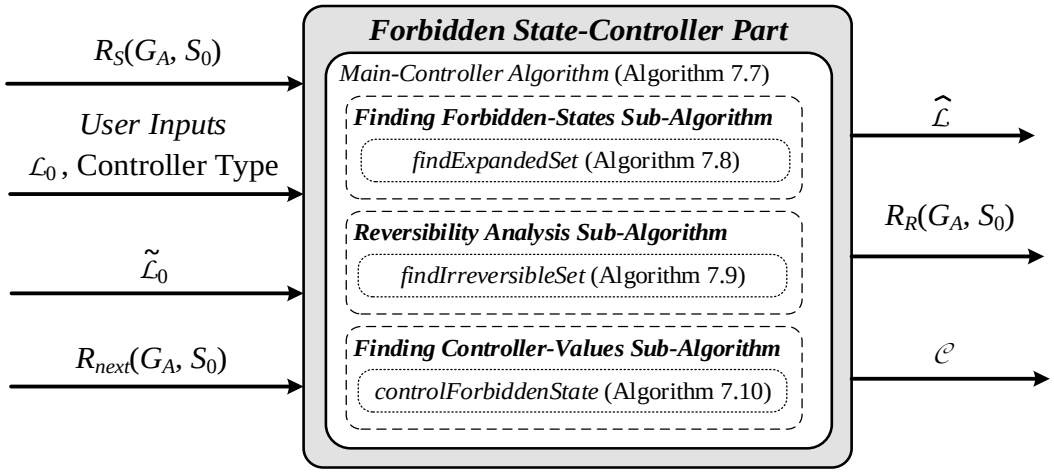


Figure 7.5. *Main-controller algorithm and its sub-algorithms for TdAPN*

Here, inputs of *main-controller algorithm* (Algorithm 7.7) are the reachability set $R_S(G_A, S_0)$, the set of deadlock states $\tilde{\mathcal{L}}_0$, the set of next states $R_{next}(G_A, S_0)$, and user inputs that are any initial set of undesired states $\mathcal{L}_0$ and the selection of the controller type (forbidding any desired state, deadlock avoidance or reversibility enforcement). Outputs of *main-controller algorithm* (Algorithm 7.7) are the expanded set of undesired states $\hat{\mathcal{L}}$, the set of irreversible states $R_R(G_A, S_0)$ and values of the controller function $\mathcal{C}$ according the controller type. Moreover, in this algorithm $\tilde{\mathcal{L}}$ represents the expanded set of deadlock states.

Algorithm 7.7. *Main-controller algorithm of Forbidden State-Controller Part*

Main-Algorithm	*ALGORITHM VII – main-controller*
Inputs	$R_S\ (G_A, S_0)$, $\tilde{\mathcal{L}}_0$, $R_{next}\ (G_A, S_0)$, $\mathcal{L}_0$, controller_type
Outputs	$\hat{\mathcal{L}}$, $R_R\ (G_A, S_0)$, $\mathcal{C}$

1. $\tilde{\mathcal{L}}$ = ***findExpandedSet***$(R_S\ (G_A, S_0), R_{next}\ (G_A, S_0), \tilde{\mathcal{L}}_0)$
2. $R_R\ (G_A, S_0)$ = ***findIrreversibleSet***$(R_S\ (G_A, S_0), R_{next}\ (G_A, S_0), \mathcal{L}_0)$
3. $\hat{\mathcal{L}} = \emptyset$
4. if controller_type is deadlock avoidance then
5. : $\hat{\mathcal{L}} = \tilde{\mathcal{L}}$
6. else if controller_type is enforcing the system reversible and deadlock-free then
7. : $\hat{\mathcal{L}} = R_R\ (G_A, S_0)$
8. else if controller_type is forbidding any desired states (such as user input) then
9. : $\hat{\mathcal{L}}$ = ***findExpandedSet***$(R_S\ (G_A, S_0), R_{next}\ (G_A, S_0), \mathcal{L}_0)$
10. else if controller_type is forbidding any desired states and deadlock avoidance then
11. : $\hat{\mathcal{L}}$ = ***findExpandedSet***$(R_S\ (G_A, S_0), R_{next}\ (G_A, S_0), \mathcal{L}_0 \cup \tilde{\mathcal{L}}_0)$
12. else if controller_type is forbidding any desired states and enforcing the system reversible then
13. : $\hat{\mathcal{L}}$ = ***findExpandedSet***$(R_S\ (G_A, S_0), R_{next}\ (G_A, S_0), \mathcal{L}_0 \cup R_R\ (G_A, S_0))$
14. end if
15. $\mathcal{C}$ = ***controlForbiddenState***$(R_S\ (G_A, S_0), R_{next}\ (G_A, S_0), \hat{\mathcal{L}})$

Main-controller algorithm (Algorithm 7.7), The following sub-algorithms are used as:

- *Finding Forbidden-States Sub-Algorithm* which is *findExpandedSet* (given in Algorithm 7.8),
- *Reversibility Analysis Sub-Algorithm* which is *findIrreversibleSet* (given in Algorithm 7.9),
- *Finding Controller-Values Sub-Algorithm* which is *controlForbiddenState* (given in Algorithm 7.10).

***Finding Forbidden-States Sub-Algorithm*:** This sub-algorithm is called and runned under *main-controller algorithm* (Algorithm 7.7) of *Forbidden State-Controller Part*. In order to find the expanded set of undesired states, i.e., $\hat{\mathcal{L}}$ from any given initial set of undesired states $\mathcal{L}_0$, the sub-algorithm, namely *findExpandedSet* (Algorithm 7.8) is developed using (5.3). Inputs of this sub-algorithm are $R_S\ (G_A, S_0)$, $R_{next}\ (G_A, S_0)$ and $\mathcal{L}_0$. Its output is the expanded set of undesired states $\hat{\mathcal{L}}$.

Algorithm 7.8. *Finding forbidden-states sub-algorithm*

Sub-Algorithm	*ALGORITHM VIII - findExpandedSet*
Inputs	$R_S\ (G_A, S_0)$, $R_{next}\ (G_A, S_0)$, $\mathcal{L}_0$
Outputs	$\hat{\mathcal{L}}$

1. $\hat{\mathcal{L}} = \mathcal{L}_0$
2. $R_{dummy} = R_S\ (G_A, S_0)$
3. $i = 0$
4. while $\mathcal{L}_i \neq \emptyset$ do then
5. : $R_{dummy} = R_{dummy} \setminus \mathcal{L}_i$
6. : $i = i + 1$
7. : $\mathcal{L}_i = \emptyset$
8. : for $j = 1$ to $\left|R_{dummy}\right|$ do
9. : : $To = \emptyset$
10. : : for $x = 1$ to $|R_{next}\ (G_A, S_0)|$ do
11. : : : compare = $\{(S_{present}, \phi), S_{resulting}\}$ of $[R_{next}\ (G_A, S_0)]_x$
12. : : : if $\left[R_{dummy}\right]_j = S_{present}$ in compare then
13. : : : : $To \leftarrow \{(\phi, S_{resulting})\}$ in compare
14. : : : end if
15. : : end for
16. : : counter = 0
17. : : for $y = 1$ to $|To|$ do
18. : : : if $S_{resulting}$ of $[To]_y \in \hat{\mathcal{L}}$ then
19. : : : : counter = counter + 1
20. : : : else if $S_{resulting}$ of $[To]_y \notin \hat{\mathcal{L}}$ and $S_{resulting}$ of $[To]_y = \left[R_{dummy}\right]_j$ then
21. : : : : counter = counter + 1
22. : : : end if
23. : : end for
24. : : if counter = $|To|$ then
25. : : : $\mathcal{L}_i \leftarrow \left[R_{dummy}\right]_j$
26. : : end if
27. : end for
28. : $\hat{\mathcal{L}} \leftarrow \mathcal{L}_i$
29. end while

***Reversibility Analysis Sub-Algorithm*:** This sub-algorithm is called and runned under *main-controller algorithm* (Algorithm 7.7) of *Forbidden State-Controller Part*. In order to find the set of irreversible states, i.e., $R_R\ (G_A, S_0)$ whose states violate the reversibility-property of TdAPN described in Definition 3.5, the sub-algorithm, namely *findIrreversibleSet* (Algorithm 7.9) is developed. Inputs of this sub-algorithm are $R_S\ (G, S_0)$, $R_{next}\ (G, S_0)$ and $\tilde{\mathcal{L}}_0$ defined in (3.8). Its output is the set of irreversible states $R_R\ (G_A, S_0)$.

Algorithm 7.9. *Reversibility analysis sub-algorithm*

Sub-Algorithm	*ALGORITHM IX - findIrreversibleSet*
Inputs	$R_S\ (G_A, S_0)$, $R_{next}\ (G_A, S_0)$, $\tilde{\mathcal{L}}_0$
Outputs	$R_R\ (G_A, S_0)$

1. $\tilde{\mathcal{L}}$ = ***findExpandedSet***$(R_S\ (G_A, S_0), R_{next}\ (G_A, S_0), \tilde{\mathcal{L}}_0)$
2. $R_R\ (G_A, S_0) = \tilde{\mathcal{L}}$
3. $R_{dummy} = R_S\ (G_A, S_0) \setminus \tilde{\mathcal{L}}$
4. $R_0 = \tilde{\mathcal{L}}$
5. $l = 0$
6. while $R_l \neq \emptyset$ do then
7. : $R_{dummy} = R_{dummy} \setminus R_l$
8. : $l = l + 1$
9. : $R_l = \emptyset$
10. : for $j = 1$ to $\left|R_{dummy}\right|$ do
11. : : $To = \emptyset$
12. : : for $x = 1$ to $|R_{next}\ (G_A, S_0)|$ do
13. : : : compare = $\{(S_{present}, \phi), S_{resulting}\}$ of $[R_{next}\ (G_A, S_0)]_x$
14. : : : if $\left[R_{dummy}\right]_j = S_{present}$ in compare then
15. : : : : $To \leftarrow \{(\phi, S_{resulting})\}$ in compare
16. : : : end if
17. : : end for
18. : : for $y = 1$ to $|To|$ do
19. : : : if $S_{resulting}$ of $[To]_y \in R_R\ (G_A, S_0)$ then
20. : : : : $R_l \leftarrow \left[R_{dummy}\right]_j$
21. : : : : break
22. : : : end if
23. : : end for
24. : end for
25. : $R_R\ (G_A, S_0) \leftarrow R_l$
26. end while

***Finding Controller-Values Sub-Algorithm*:** This sub-algorithm is called and runned under *main-controller algorithm* (Algorithm 7.7) of *Forbidden State-Controller Part*. In order to determine values of the controller function $\mathcal{C}(S,\phi)$ that is able to disable the transition(s) in the set ϕ at the state $S \in R_S\,(G,S_0)$, the sub-algorithm, namely *controlForbiddenState* (Algorithm 7.10) is developed using (5.4). Inputs of this sub-algorithm are $R_S\,(G,S_0)$, $R_{next}\,(G,S_0)$ and $\hat{\mathcal{L}}$. Its output is values of the controller function as $\mathcal{C}(S,\phi) \in \{0,1\}$.

Algorithm 7.10. *Finding controller-values sub-algorithm*

Sub-Algorithm	*ALGORITHM X - controlForbiddenState*
Inputs	$R_S\,(G_A,S_0)$, $R_{next}\,(G_A,S_0)$, $\hat{\mathcal{L}}$
Outputs	$\mathcal{C}(S,\phi)$

1. $\mathcal{C} = 1$
2. for j = 1 to $|R_S\,(G,S_0)|$ do
3. : $To = \emptyset$
4. : for $x = 1$ to S_{next} do
5. : : compare = $\{(S_{present},\phi),S_{resulting}\}$ of $[S_{next}]_x$
6. : : if $[R_S\,(G,S_0)]_j = S_{present}$ in compare then
7. : : : $To \leftarrow \{(\phi,S_{resulting})\}$ in compare
8. : : end if
9. : : for $y = 1$ to $|To|$ do
10. : : : if $S_{resulting}$ of $[To]_y \in \hat{\mathcal{L}}$ then
11. : : : : if ϕ of $[To]_y \neq \emptyset$ then
12. : : : : : $\mathcal{C}([R_S\,(G,S_0)]_j,\phi) = 0$
13. : : : : end if
14. : : : end if
15. : : end for
16. : end for
17. end for

8. CONCLUSION, DISCUSSION AND PROPOSALS

8.1. Conclusion

A novel mathematical modeling method has been presented for Timed PNs, where time delays are assigned to arcs. In Timed PNs, system states are defined by the change of tokens in PNs and their movements (flow). In the proposed model, a triangular representation, called time element, that allows monitoring of tokens in transition, namely flowing tokens, has been defined. The proposed Timed-Arc PN overcomes the main drawback of Timed PNs by including time elements in the PN. It is important to monitor flowing tokens in Timed PNs because a complete picture of states of time-delayed systems is required in many practical systems. The proposed triangular graphical representation transforms the representation of Timed PN into a tripartite graph including places, transitions, and time elements. The tripartite structure of Timed-Arc PNs allows the net to start at any state and any initial-time instant. The state of the system and the remaining time of the work/operations are shown as a vector. Furthermore, discrete-time unit impulse functions are used to compute the marking and the remaining time vectors. The former is used for indicating the status of places, while the latter represents the status of time elements. Such impulse functions are used to trigger appearances and disappearances of tokens at places and at time elements such that the state evolution in places and time elements is described in terms of these impulse functions. The use of discrete-time unit impulse functions allows for obtaining new states; as a result, the next state of the proposed Timed-Arc PN is formally computed using the marking and the remaining time vectors with discrete-time unit impulse functions. All situations of tokens, i.e., all situations of states can be computed. This makes possible to obtain the reachability set and the timed-reachability graph (tree) enhanced by the time information as long as the state of the system is expressed by the marking and remaining time vectors of TdAPN. In addition, the controller design is also presented for time-delayed systems that is modeled by the proposed mathematical model of TdAPNs. Basic behavioral properties of the proposed TdAPNs have been defined by using the reachability set in order to permit analysis of the proposed approach. Using the reachability set and behavioral properties of the proposed Timed-Arc PNs, a forbidden state controller for time-delayed systems has been designed in order to make the system avoid undesired states, such as deadlock states, and to enforce the reversibility. Algorithms for constructing the reachability set and

designing a forbidden state controller for Timed-Arc PNs have been developed and simulated using MATLAB. In addition; case studies of modeling and designing for real systems, such as manufacturing, railway, and automotive systems, have been carried out using the proposed approach. These case studies are not limited to the scope of these topics. Timed-Arc PNs can also be applied to autonomous operations or decision mechanisms of self-driving cars/autonomous robots, etc.

The proposed Timed-Arc PN is compared with Timed PNs and Stretched PNs. In terms of firing processes, Timed-Arc PN provides the user to observe states mathematically and graphically. Timed-Arc PN has a number of time elements according to the number of non-zero-time delayed outgoing arcs; in addition, non-zero-time delayed time elements are considered in the state of Timed-Arc PNs. The graphical representation of Timed-Arc PNs is useful to show temporary disappearance in the representation of Timed PNs, and one time element is sufficient to represent corresponding flowing tokens while flowing tokens are represented in newly created places in the representation of Stretched PNs. These newly created places are generated by a stretching procedure, and their numbers are proportional to non-unity time delays defined in the Timed PN before stretching. The stretching procedure results in an increase in matrices and the marking vector. This drawback of Stretched PNs causes an increase in the computation time for constructing the reachability set for Timed PNs. On the contrary, Timed-Arc PN is able to represent time delays defined in Timed PNs using only one time element; as a result; this feature of the proposed method allows fast computational time to construct the reachability set. Computational times to construct the reachability set and to construct necessary matrices (e.g. input and output matrices) and sets (e.g. a set of input/output places connected to a transition, a set of time elements) are considered as criteria for the performance metric.

The concept of the proposed Timed-Arc PNs is clear and concise. This approach is used with unit time, in terms of appropriate time slots, which are readily handled by computers in practical applications. Thus, any model constructed using the proposed approach could be easily implemented for specific time-delayed systems and related algorithms. In the proposed Timed-Arc PNs, exact time durations are used such that deterministic time labels attached to outgoing arcs have no time intervals; as a result, the use of deterministic time values instead of time intervals can, therefore, provide less complexity and serve to decrease the computational time. It also presents fewer

difficulties when implementing algorithms. Moreover, in the model, a firing process can start at any time.

The proposed Timed-Arc PN provides an overall model for large-scale and complex systems. This novel approach considers the complete dynamic evolution of time-delayed systems; as a result, it allows the user to see all situations of states for time-delayed systems such that it gives a complete model for time-delayed systems. Therefore, it is possible to see the complete picture of the system with deterministic time delays. Furthermore, behavioral properties and algorithms for the proposed Timed-Arc PNs have been presented. It is a useful feature for Timed PNs because the reachability set for time-delayed and dynamic systems is completely constructed. Any Timed PN with firing or holding durations can be converted into the tripartite structure of Timed-Arc PNs; thus, the reachability set of such Timed PNs is obtained using the proposed approach. Moreover, the timed-reachability graph (tree) enhanced by the time information is generated using the relation between states; as a result, the deterministically time-delayed system is depicted in the full dynamic schema. This graph allows pointing out problematic states, such as deadlocks, in time-delayed and dynamic systems; thus, it is possible to take measures. In this study, in order to avoid such undesired states, a forbidden state controller approach is developed for the proposed model.

In conclusion, the proposed Timed-Arc PN is a useful modeling tool for Timed PNs as long as any complicated systems, which include deterministic time delays, can be solved by using the proposed model. Obtaining the reachability set of the net is a crucial point for solving the system as well as in every modeling approach. The proposed Timed-Arc PN allows the complete reachability set for a time-delay and dynamic system. Thus, this property allows seeing the complete picture of such systems and analyzing the behaviors of such systems; as a result, a forbidden state controller for Timed PNs, which provides a reversibility enforcement and deadlock avoidance controller, can be developed by using this strong feature. Algorithms have shown that the proposed Timed-Arc PN is applicable and has given admissible results.

8.2. Discussion

Each model described in this paper offers a deterministic-timed approach for Timed PNs. In real applications, time delays are commonly expressed as time intervals. Using time intervals can be more realistic than using fixed durations. However, using time

intervals may result in higher complexity to construct the reachability set. Thus, deterministic time delays are considered. These time delays can be can be determined by considering the most probable time delay of the system, such as the mean value.

It is the fact that time delays are expressed in terms of seconds in real applications; as a result, these delays are discretized into time slots using an appropriate sampling period in order to adapt into the model of Timed PNs. This approximation might cause a loss of information. The sampling period should be carefully selected.

8.3. Proposals

Future directions and proposals to develop and improve the proposed Timed-Arc PNs are as follows:

- The controller for the proposed Timed-Arc PNs is based on behavioral properties. The structure of this Timed-Arc PNs is very similar to the structure of the basic (untimed) PNs; thus, structural properties could be similarly investigated and a structural controller can be developed for the proposed Timed-Arc PNs.
- Time delays of Timed-Arc PNs are deterministic; thus, they are not sensitive to any changes in time labels. A method can be developed, where these time labels are defined as time intervals or stochastic over time elements. On the other hand, Timed-Arc PNs can be extended to an adaptive model that is capable of changing its deterministic time-labels according to current conditions. This makes Timed-Arc PNs more realistic and online model.
- The proposed Timed-Arc PN does not allow any active transition to fire during its firing process. The model may be developed to allow any transition whose firing process still continues. This may cause the remaining time vector to become a matrix or the set of vectors. On the other hand, an arc-stretching procedure, namely Arc-Stretched Petri Nets, can be studied like Transition or Place-Stretched PNs.
- There are different types of arcs, such as enabling, reading, test arcs, etc. These can be adapted into the proposed model of Timed-Arc PNs.
- When constructing the model of Timed-Arc PN, it needs matrices whose elements are mostly zero. These matrices are used in computations. In order to improve computational time, algorithms for the proposed Timed-Arc PN can be developed considering sparse matrices.

- Time elements can be structurally used to trigger some events using special arcs that should be defined for the proposed Timed-Arc PN.
- The proposed Timed-Arc PN uses a discrete-time unit impulse function to compute the next state. This approach can readily be applied on both deterministic Timed PNs with holding or enabling durations and deterministic Timed-Place PNs.
- Today's trend topics are autonomous and intelligent systems, autonomous and unmanned vehicles, self-driving cars, smart automation systems, smart home systems, intelligent transportation systems, decision mechanisms for such systems etc. Applications/Case studies can be carried out in these fields using the proposed Timed-Arc PNs. Timed-Arc PN can be used as a verification tool for such systems so as to guarantee functional safety. Timed-Arc PN can be used to model the behaviors of such systems.

REFERENCES

[1] Yufka, A., Özkan, H.A. and Aybar, A. (2018). Modeling Basic Components of Railway Systems Using Timed Arc Petri Nets. *Proceeding 5th International Conference on Control, Decision and Information Technologies*, Thessaloniki, Greece, Apr. 10-13, 2018, CODIT'18. pp. 427-432. IEEE.

[2] Giua A. and Seatzu C. (2008). Modeling and Supervisory Control of Railway Networks Using Petri Nets. *IEEE Transactions on Automation Science and Engineering*, 5(3), pp. 431-445.

[3] Proth, J.M. and Xie, X. (1996). *Petri Nets: a Tool for Design and Management of Manufacturing Systems*. Wiley.

[4] Zuberek, W.M. (2000). Timed Petri nets in modeling and analysis of cluster tools, *IEEE Transactions on Robotics and Automation*, 17(5), 562- 575.

[5] Cassandras, C. G. and Lafortune, S. (2008). *Introduction to Discrete Event Systems, Second Edition*. New York: Springer US.

[6] Wang, J. (1998). *Time Petri Nets, Theory and Application*. Kluwer Academic.

[7] Murata, T. (1989). Petri Nets - Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4), 541 – 580.

[8] Zhou, M. and DiCesare, F. (1993). *Petri net Synthesis for Discrete Event Control of Manufacturing Systems*. Massachusetts: Kluwer Academic.

[9] Aybar, A. and İftar, A. (2010). Decentralized supervisory controller design to avoid deadlock in Petri nets. *International Journal of Control*, 76(13), 1285-1295.

[10] Zuberek, WM. (1991). Timed Petri nets - definitions, properties, and applications. *Microelectronics and Reliability*. 31(4), 627-644.

[11] Aybar, A. and İftar, A. (2012). Supervisory controller design to enforce some basic properties in timed-transition Petri nets using stretching. *Nonlinear Analysis: Hybrid Systems*. 6, 712-729.

[12] Aybar, A. and İftar, A. (2008). Deadlock Avoidance Controller Design for Timed Petri Nets Using Stretching. *IEEE Systems Journal*, 2(2), 178-188.

[13] Aybar, A. and İftar, A. (2006). Supervisory Controller Design for Timed Petri Nets. *Proceeding 1st International Conference on System of Systems Engineering*, Los Angeles, CA, USA, Apr. 24-26, 2006, SoSE'06. 9641, pp. 59-64. IEEE.

[14] Aybar, A. and ˙İftar, A. (2009). Supervisory Controller Design to Enforce Some Basic Properties in Timed Petri Nets. *Proceeding 13th IFAC International Symposium on Information Control Problems in Manufacturing*, Moscow, Russia, June 3-5, 2009, INCOM'09. pp. 940-945.

[15] Bernd Walter, W. (1983). Timed Petri-Nets for Modelling and Analyzing Protocols with Real-Time Characteristics. *Proceeding 3rd IFIP International Workshop on Protocol Specification, Testing, and Verification*, Ruschlikon, Switzerland, May 31 - June 2, 1983. pp. 149-159. North-Holland.

[16] Bolognesi, T., Lucidi, F. and Trigila, S. (1990). From Timed Petri Nets to Timed LOTOS. *Proceeding 10th International Symposium on Protocol Specification, Testing and Verification*, Ottawa, Canada, June 12-15, 1990, IFIP X. pp. 395-408.

[17] Hanisch, H.M. (1993). Analysis of Place/Transition Nets with Timed-Arcs and its Application to Batch Process Control. *Proceeding 14th International Conference on Application and Theory of Petri Nets*, Chicago, Illinois, USA, June 21-25, 1993, ICATPN'93. 691, pp. 282-299.

[18] Abdulla, P. A. and Nyl´en, A. (2001). Timed Petri nets and BQOs. *Proceeding 22nd International Conference on Application and Theory of Petri Nets*, Newcastle upon Tyne, UK, June 25-29, 2001, ICATPN'01. 2075, pp. 53-70. Springer.

[19] Jacobsen, L., Jacobsen, M., Møller, M.H. and Srba, J. (2011). Verification of Timed-Arc Petri Nets. *Proceeding 37th International Conference on Current Trends in Theory and Practice of Computer Science*, Nový Smokovec, Slovakia, Jan. 22-28, 2011, SOFSEM'11. 6543, pp. 46-72. Springer.

[20] Bowden, F.D.J. (2000). A brief survey and synthesis of the roles of time in Petri nets. *Mathematical and Computer Modelling*. 31, 55-68.

[21] Freedman, P. (1991). Time, Petri nets, and robotics. *IEEE Transactions on Robotics and Automation*. 7(4), 417-433.

[22] Zhu, J. and Denton, R. (1988). Timed Petri Nets and their Application to Communication Protocol Specification. *Proceeding 21st International Conference on Century Military Communications - What's Possible?' Military Communications*, San Diego, CA, USA, Oct. 23-26, 1988, MILCOM'88. 1, pp. 195-199. IEEE.

[23] Nielsen, M., Sassone, V. and Srba, J. (2001). Towards a Notion of Distributed Time for Petri Nets (Extended Abstract). *Proceeding 22nd International Conference on Applications and Theory of Petri Nets*, Newcastle upon Tyne, U.K, June 25–29, 2001, ICATPN'01, 2075, pp. 23-31.

[24] Bowden, F.D.J. (2001) *The modelling and analysis of command and control decision processes using extended time petri nets*. Doctor of Philosophy Dissertation. Adelaide, Australia: The University of Adelaide, Faculty of Mathematical and Computer Sciences.

[25] Sener, I., Kaymakci, O.T., Ustoglu I. and Cansever, G. (2016). Specification and formal verification of safety properties in a point automation system. *Turkish Journal of Electrical Engineering and Computer Sciences*. 24, 1384-1396.

[26] Sieverding, S., Ellen, C. and Battram, P. (2013). Sequence Diagram Test Case Specification and Virtual Integration Analysis using Timed-Arc Petri Nets. *Proceeding 10 th International Workshop on Formal Engineering approaches to Software Components and Architectures*, Rome, Italy, March 23, 2013, FESCA'13. 108, pp. 17-31.

[27] Jensen, P.G., Larsen, K.G. and Srba, J. (2016). Real-Time Strategy Synthesis for Timed-Arc Petri Net Games via Discretization, *Proceeding 23 rd International Symposium on Model Checking Software*, Eindhoven, The Netherlands, April 7-8, 2016, SPIN'16. 9641, pp. 129-146.

[28] Yufka, A., Özkan, H.A. and Aybar, A. (2016). A Formal Method and Novel Graphical Representation for Deterministic Timed-Arc Petri Nets. *Proceeding National Conference on "Otomatik Kontrol Ulusal Toplantisi"*, Eskişehir, Turkey, Sep. 29 – Dec. 01, 2016, TOK'16. pp. 209-213.

[29] Yufka, A., Özkan, H.A. and Aybar, A. (2017). Timed Arc Petri Nets: The Time-Element Approach. *Proceeding 10th International Conference on Electrical and Electronics Engineering*, Bursa, Turkey, Nov. 30 – Dec. 2, 2017, ELECO'17. pp. 794-798. IEEE.

[30] Yufka, A., Özkan, H.A. and Aybar, A. (2018). Timed Arc Petri Nets: The Impulsive Approach. *Proceeding 5th International Conference on Control, Decision and*

Information Technologies, Thessaloniki, Greece, Apr. 10-13, 2018, CODIT'18. pp. 409-414. IEEE.

[31] Yufka, A., Özkan, H.A. and Aybar, A. (2019). Reachability Set Algorithms for Timed-Arc Petri Nets. *Proceeding 395th International Conference on Electrical and Electronics Engineering*, Barcelona, Spain, Feb. 11-12, 2019, ICEEE'19. (Accepted)

[32] Aybar, A. and ˙İftar, A. (2012). Supervisory controller design for timed-place Petri nets. *Kybernetika*, 48, 1114-1135.

[33] Chu, F. and Xie, X.L. (1997). Deadlock analysis of Petri nets using siphons and mathematical programming. *IEEE Transactions on Robotics and Automation*. 13(6), pp. 793-804.

[34] Hou, YF., Li, ZW., Al-Ahmari, A.M., El-Tamimi, AA.M. and Nasr, E.A. (2014). Extended Elementary Siphons and Their Application to Liveness-Enforcement of Generalized Petri Nets. *Asian Journal of Control*. 16(6), pp. 1789–1810.

[35] Cabasino, M.P., Affiliated, A.G. and Seatzu, C. (2013). Structural Analysis of Petri Nets. M.P.Cabasino, A.G. Affiliated and C. Seatzu (Ed.), in *Control of Discrete-Event Systems - Automata and Petri Net Perspectives* (433, pp. 213-233). London: Springer-Verlag.

[36] Ramadge, P. J. G. and Wonham, W. M. (1989). The control of discrete event systems. *Proceedings of the IEEE*. 77(1), pp. 81-98.

[37] Aybar, A. and ˙İftar, A. (2013). Supervisory Controller Design to Enforce Basic Properties in Timed-Place Petri Nets. *Proceeding 6th IFAC International Conference on Management and Control of Production and Logistics*, Fortaleza, Brazil, Sep. 11-13, 2013, MCPL'13. pp. 486-492.

[38] Aybar, A. and İftar, A. (2003). Controller design to enforce boundedness, liveness, and reversibility in Petri nets. *Proceeding 7th IFAC International Workshop on Intelligent Manufacturing Systems*, Budapest, Hungary, Apr. 6-8, 2003, IMS'03. pp. 199 - 204.

[39] Aybar, A. and İftar, A. (2003). Decentralized controller design to enforce boundedness, liveness, and reversibility in Petri nets. *Proceeding European Control Conference*, Cambridge, UK, Sep. 1-4, 2003, ECC'03. pp. 1681-1686. IEEE.

[40] Aybar, A. and İftar, A. (2005). Centralized and decentralized supervisory controller design to enforce boundedness, liveness, and reversibility in Petri nets. *International Journal of Control*. 78(8), pp. 537–553.

[41] Li, ZW and Zhou, MC. (2004). Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*. 34(1), pp. 38 - 51.

[42] Durmus, M.S., Takai, S. and Soylemez, M.T. (2014). Fault Diagnosis in Fixed-Block Railway Signaling Systems: A Discrete Event Systems Approach. *IEEJ Transactions on Electrical and Electronic Engineering*. 9, pp. 523-531.

[43] Comite Europeen de Normalisation Electrotechnique. (2011). EN50128:2011 - Railway applications - Communication, Signaling and Processing Systems - Software for Railway Control and Protection Systems.

[44] Staines, A.S. (2009). Modeling and Analysis of Real Time Control Systems: A Cruise Control System Case Study. *Recent Advances in Technologies*, pp. 561-576.

[45] Kramer, J. and Magee, J. (1997). Exposing the Skeleton in the Coordination Closet. *Proceeding 2nd International Conference on Coordination Languages and Models*, Berlin, Germany, Sep. 1–3, 1997, COORDINATION'97. 1282, pp. 18-31.

[46] Aybar, A. and İftar, A. (2009). Representation of the State of Timed-Place Petri Nets Using Stretching. *Proceeding 4th International IFAC Workshop on Discrete-Event System Design*, Gandia, Spain, Oct. 6–8, 2009. 42(21), pp. 72-77. Elsevier.

[47] Aybar, A. and İftar, A. (2013). Supervisory Controller Design to Enforce Basic Properties in Timed-Place Petri Nets. *Proceeding 6th International IFAC Conference on Management and Control of Production and Logistics*, Fortaleza, Brazil, Sep. 11–13, 2013. 46(24), pp. 486-492. Elsevier.

[48] Aybar, A. and İftar, A. (2012). Supervisory controller design to enforce some basic properties in timed-transition Petri nets using stretching. *Nonlinear Analysis: Hybrid Systems*. 6, pp. 712-729.

[49] Jones, N.D., Landweber, L.H. and Lien, Y.E. (1977). Complexity of Some Problems in Petri Nets. *Theoretical Computer Science*. 4, pp. 277-299.

APPENDIX-1 – THE REACHABILITY SET OF TdAPN (NOT REVERSIBLE)

In this appendix, the reachability set of TdAPN in Figure 5.2 is given in Table A1.1. Descriptions of columns in Table A1.1 are similar to explanations for Table 3.3.

Table A1.1. *Reachability set for TdAPN in Figure 5.2*

k	The State of TdAPN		ϕ	*Next State*	*k*	The State of TdAPN		ϕ	*Next State*
0	$^{\dagger}S_0$	{[2 0 0 0]', [0 0 0]'}	-	$^{\dagger}S_0$	4	S_{13}	{[0 1 0 0]', [2 0 0]'}	-	S_{20}
			$\{t_1\}$	S_1				$\{t_4\}$	S_{21}
			$\{t_2\}$	S_2		$^{\dagger}S_{14}$	{[1 0 0 1]', [0 0 0]'}	-	$^{\dagger}S_{14}$
			$\{t_1, t_2\}$	S_3				$\{t_1\}$	S_{15}
1	S_1	[1 0 0 0]', [2 0 0]'}	-	S_4				$\{t_2\}$	S_{16}
			$\{t_2\}$	S_5				$\{t_5\}$	$^{\dagger}S_9$
	S_2	{[1 0 0 0]', [0 1 0]'}	-	$^{\dagger}S_6$				$\{t_1, t_5$	S_{13}
			$\{t_1\}$	S_7				$\{t_2, t_5$	S_{10}
	S_3	{[0 0 0 0]', [2 1 0]'}	-	S_8		S_{15}	{[0 0 0 1]', [2 0 0]'}	-	S_{21}
2	S_4	{[1 0 0 0]', [1 0 0]'}	-	$^{\dagger}S_9$				$\{t_5\}$	S_{20}
			$\{t_2\}$	S_{10}		S_{16}	{[0 0 0 1]', [0 1 0]'}	-	$^{\dagger}S_{17}$
	S_5	{[0 0 0 0]', [1 1 0]'}	-	$^{\dagger}S_{11}$				$\{t_5\}$	S_{11}
	$^{\dagger}S_6$	{[1 0 1 0]', [0 0 0]'}	-	$^{\dagger}S_6$		$^{\dagger}S_{17}$	{[0 0 1 1]', [0 0 0]'}	-	$^{\dagger}S_{17}$
			$\{t_1\}$	S_7				$\{t_5\}$	S_{11}
			$\{t_2\}$	S_{12}		S_{18}	{[0 0 0 0]', [0 0 3]'}	-	S_{22}
	S_7	{[0 0 1 0]', [2 0 0]'}	-	S_8		*S_{19}	{[0 0 2 0]', [0 0 0]'}	-	*S_{19}
3	S_8	{[0 0 1 0]', [1 0 0]'}	-	$^{\dagger}S_{11}$	5	S_{20}	{[0 1 0 0]', [1 0 0]'}	-	$^{\dagger}S_{23}$
	$^{\dagger}S_9$	{[1 1 0 0]', [0 0 0]'}	-	$^{\dagger}S_9$				$\{t_4\}$	$^{\dagger}S_{24}$
			$\{t_1\}$	S_{13}		S_{21}	{[0 0 0 1]', [1 0 0]'}	-	$^{\dagger}S_{24}$
			$\{t_2\}$	S_{10}				$\{t_5\}$	$^{\dagger}S_{23}$
			$\{t_4\}$	$^{\dagger}S_{14}$		S_{22}	{[0 0 0 0]', [0 0 2]'}	-	S_{25}
			$\{t_1, t_4\}$	S_{15}	6	$^{\dagger}S_{23}$	{[0 2 0 0]', [0 0 0]'}	-	$^{\dagger}S_{23}$
			$\{t_2, t_4\}$	S_{16}				$\{t_4\}$	$^{\dagger}S_{24}$
	S_{10}	{[0 1 0 0]' [0 1 0]'}	-	$^{\dagger}S_{11}$		$^{\dagger}S_{24}$	{[0 1 0 1]', [0 0 0]'}	-	$^{\dagger}S_{24}$
			$\{t_4\}$	$^{\dagger}S_{17}$				$\{t_4\}$	$^{\dagger}S_{26}$
	$^{\dagger}S_{11}$	{[0 1 1 0]', [0 0 0]'}	-	$^{\dagger}S_{11}$				$\{t_5\}$	$^{\dagger}S_{23}$
			$\{t_3\}$	S_{18}				$\{t_4, t_5$	$^{\dagger}S_{24}$
			$\{t_4\}$	$^{\dagger}S_{17}$		S_{25}	{[0 0 0 0]', [0 0 1]'}	-	$^{\dagger}S_0$
	S_{12}	{[0 0 1 0]', [0 1 0]'}	-	*S_{19}		$^{\dagger}S_{26}$	{[0 0 0 2]', [0 0 0]'}	-	$^{\dagger}S_{26}$
								$\{t_5\}$	$^{\dagger}S_{24}$

† *denotes relaxed states and * denotes relaxed and deadlock states..*

APPENDIX-2 – THE REACHABILITY SET OF MANUFACTURING EXAMPLE

In this appendix, the reachability set of the manufacturing-systems example in Figure 6.2 is given in Table A2.1. Descriptions of columns in Table A2.1 are similar to explanations for Table 3.3.

Table A2.1. *Reachability Set for TdAPN in Figure 6.2*

k	**The State of TdAPN**		ϕ	***Next State***
0	†S_0	{[2 0 1 0 0 1 0 0 1 2], [0 0 0 0 0 0]'}	-	S_0
			$\{t_1\}$	S_1
1	S_1	{[1 0 0 0 0 1 0 0 1 2]', [2 0 0 0 0 0]'}	-	S_2
2	S_2	{[1 0 0 0 0 1 0 0 1 2]', [1 0 0 0 0 0]'}	-	S_3
3	†S_3	{[1 1 0 0 0 1 0 0 1 2]', [0 0 0 0 0 0]'}	-	S_3
			$\{t_2\}$	S_4
4	†S_4	{[1 0 1 1 0 1 0 0 0 2]', [0 0 0 0 0 0]'}	-	S_4
			$\{t_1\}$	S_5
			$\{t_3\}$	S_6
			$\{t_1, t_3\}$	S_7
5	S_5	{[0 0 0 1 0 1 0 0 0 2]', [2 0 0 0 0 0]'}	-	S_8
			$\{t_3\}$	S_9
5	†S_6	{[1 0 1 0 1 0 0 0 1 2]', [0 0 0 0 0 0]'}	-	S_6
			$\{t_1\}$	S_7
			$\{t_4\}$	S_{10}
			$\{t_1, t_4\}$	S_{11}
5	S_7	{[0 0 0 0 1 0 0 0 1 2]', [2 0 0 0 0 0]'}	-	S_9
			$\{t_4\}$	S_{12}
6	S_8	{[0 0 0 1 0 1 0 0 0 2]', [1 0 0 0 0 0]'}	-	S_{13}
			$\{t_3\}$	S_{14}
6	S_9	{[0 0 0 0 1 0 0 0 1 2]', [1 0 0 0 0 0]'}	-	S_{14}
			$\{t_4\}$	S_{15}
6	†S_{10}	{[1 0 1 0 0 1 1 0 0 2]', [0 0 0 0 0 0]'}	-	S_{10}
			$\{t_1\}$	S_{11}
			$\{t_5\}$	S_{16}
			$\{t_1, t_5\}$	S_{17}
6	S_{11}	{[0 0 0 0 0 1 1 0 0 2]', [2 0 0 0 0 0]'}	-	S_{12}
			$\{t_5\}$	S_{18}
6	S_{12}	{[0 0 0 0 0 1 1 0 0 2]', [1 0 0 0 0 0]'}	-	S_{15}
			$\{t_5\}$	S_{19}
7	†S_{13}	{[0 1 0 1 0 1 0 0 0 2]', [0 0 0 0 0 0]'}	-	S_{13}
			$\{t_3\}$	S_{14}

† *denotes relaxed states and * denotes relaxed and deadlock states.*

Table A2.1. (Continue) *Reachability Set for TdAPN in Figure 6.2*

k	**The State of TdAPN**		ϕ	***Next State***
7	†S_{14}	{[0 1 0 0 1 0 0 0 1 2]', [0 0 0 0 0 0]'}	-	S_{14}
			$\{t_2\}$	S_{20}
			$\{t_4\}$	S_{15}
7	†S_{15}	{[0 1 0 0 0 1 1 0 0 2]', [0 0 0 0 0 0]'}	-	S_{15}
			$\{t_5\}$	S_{19}
7	S_{16}	{[1 0 1 0 0 1 0 0 0 1]', [0 1 1 1 0 0]'}	-	S_{21}
			$\{t_1\}$	S_{22}
7	S_{17}	{[0 0 0 0 0 1 0 0 0 1]', [2 1 1 1 0 0]'}	-	S_{23}
7	S_{18}	{[0 0 0 0 0 1 0 0 0 1]', [1 1 1 1 0 0]'}	-	S_{24}
7	S_{19}	{[0 1 0 0 0 1 0 0 0 1]', [0 1 1 1 0 0]'}	-	S_{24}
8	*S_{20}	{[0 0 1 1 1 0 0 0 0 2]', [0 0 0 0 0 0]'}	-	S_{20}
8	†S_{21}	{[2 0 1 0 0 1 0 1 1 1]', [0 0 0 0 0 0]'}	-	S_{21}
			$\{t_1\}$	S_{22}
			$\{t_6\}$	S_{25}
			$\{t_1, t_6\}$	S_{26}
8	S_{22}	{[1 0 0 0 0 1 0 1 1 1]', [2 0 0 0 0 0]'}	-	S_{23}
			$\{t_6\}$	S_{27}
8	S_{23}	{[1 0 0 0 0 1 0 1 1 1]', [1 0 0 0 0 0]'}	-	S_{24}
			$\{t_6\}$	S_{28}
8	†S_{24}	{[1 1 0 0 0 1 0 1 1 1]', [0 0 0 0 0 0]'}	-	S_{24}
			$\{t_2\}$	S_{29}
			$\{t_6\}$	S_{28}
9	S_{25}	{[2 0 1 0 0 1 0 0 0 1]', [0 0 0 0 1 1]'}	-	S_0
			$\{t_1\}$	S_1
9	S_{26}	{[1 0 0 0 0 1 0 0 0 1]', [2 0 0 0 1 1]'}	-	S_2
9	S_{27}	{[1 0 0 0 0 1 0 0 0 1]', [1 0 0 0 1 1]'}	-	S_3
9	S_{28}	{[1 1 0 0 0 1 0 0 0 1]', [0 0 0 0 1 1]'}	-	S_3
9	†S_{29}	{[1 0 1 1 0 1 0 1 0 1]', [0 0 0 0 0 0]'}	-	S_{29}
			$\{t_1\}$	S_{30}
			$\{t_3\}$	S_{31}
			$\{t_1, t_3\}$	S_{32}
10	S_{30}	{[0 0 0 1 0 1 0 1 0 1]', [2 0 0 0 0 0]'}	-	S_{33}
			$\{t_3\}$	S_{34}

† *denotes relaxed states and * denotes relaxed and deadlock states.*

Table A2.1. (Continue) *Reachability Set for TdAPN in Figure 6.2*

k	**The State of TdAPN**		ϕ	***Next State***
10	†S_{31}	{[1 0 1 0 1 0 0 1 1 1]', [0 0 0 0 0 0]'}	-	S_{31}
			$\{t_1\}$	S_{32}
			$\{t_4\}$	S_{35}
			$\{t_6\}$	S_{36}
			$\{t_1, t_4\}$	S_{37}
			$\{t_1, t_6\}$	S_{38}
10	S_{32}	{[0 0 0 0 1 0 0 1 1 1]', [2 0 0 0 0 0]'}	-	S_{34}
			$\{t_4\}$	S_{39}
			$\{t_6\}$	S_{40}
11	S_{33}	{[0 0 0 1 0 1 0 1 0 1]', [1 0 0 0 0 0]'}	-	S_{41}
			$\{t_3\}$	S_{42}
11	S_{34}	{[0 0 0 0 1 0 0 1 1 1]', [1 0 0 0 0 0]'}	-	S_{42}
			$\{t_4\}$	S_{43}
			$\{t_6\}$	S_{44}
11	†S_{35}	{[1 0 1 0 0 1 1 1 0 1]', [0 0 0 0 0 0]'}	-	S_{35}
			$\{t_1\}$	S_{37}
			$\{t_5\}$	S_{45}
			$\{t_1, t_5\}$	S_{46}
11	S_{36}	{[1 0 1 0 1 0 0 0 0 1]', [0 0 0 0 1 1]'}	-	S_6
			$\{t_1\}$	S_7
11	S_{37}	{[0 0 0 0 0 1 1 1 0 1]', [2 0 0 0 0 0]'}	-	S_{39}
			$\{t_5\}$	S_{47}
11	S_{38}	{[0 0 0 0 1 0 0 0 0 1]', [2 0 0 0 1 1]'}	-	S_9
11	S_{39}	{[0 0 0 0 0 1 1 1 0 1]', [1 0 0 0 0 0]'}	-	S_{43}
			$\{t_5\}$	S_{48}
11	S_{40}	{[0 0 0 0 1 0 0 0 0 1]', [1 0 0 0 1 1]'}	-	S_{14}
12	†S_{41}	{[0 1 0 1 0 1 0 1 0 1]', [0 0 0 0 0 0]'}	-	S_{41}
			$\{t_3\}$	S_{42}
12	†S_{42}	{[0 1 0 0 1 0 0 1 1 1]', [0 0 0 0 0 0]'}	-	S_{42}
			$\{t_2\}$	S_{49}
			$\{t_4\}$	S_{43}
			$\{t_6\}$	S_{44}
12	†S_{43}	{[0 1 0 0 0 1 1 1 0 1]', [0 0 0 0 0 0]'}	-	S_{43}
			$\{t_5\}$	S_{48}
12	S_{44}	{[0 1 0 0 1 0 0 0 0 1]', [0 0 0 0 1 1]'}	-	S_{14}
12	S_{45}	{[1 0 1 0 0 1 0 1 0 0]', [0 1 1 1 0 0]'}	-	S_{50}
			$\{t_1\}$	S_{51}
12	S_{46}	{[0 0 0 0 0 1 0 1 0 0]', [2 1 1 1 0 0]'}	-	S_{52}

† *denotes relaxed states and * denotes relaxed and deadlock states.*

Table A2.1. (Continue) *Reachability Set for TdAPN in Figure 6.2*

k	**The State of TdAPN**		ϕ	***Next State***
12	S_{47}	{[0 0 0 0 0 1 0 1 0 0]', [1 1 1 1 0 0]'}	-	S_{53}
12	S_{48}	{[0 1 0 0 0 1 0 1 0 0]', [0 1 1 1 0 0]'}	-	S_{53}
13	*S_{49}	{[0 0 1 1 1 0 0 1 0 1]', [0 0 0 0 0 0]'}	-	S_{49}
13	†S_{50}	{[2 0 1 0 0 1 0 2 1 0]', [0 0 0 0 0 0]'}	-	S_{50}
			$\{t_1\}$	S_{51}
			$\{t_6\}$	S_{54}
			$\{t_1, t_6\}$	S_{55}
13	S_{51}	{[1 0 0 0 0 1 0 2 1 0]', [2 0 0 0 0 0]'}	-	S_{52}
			$\{t_6\}$	S_{56}
13	S_{52}	{[1 0 0 0 0 1 0 2 1 0]', [1 0 0 0 0 0]'}	-	S_{53}
			$\{t_6\}$	S_{57}
13	†S_{53}	{[1 1 0 0 0 1 0 2 1 0]', [0 0 0 0 0 0]'}	-	S_{53}
			$\{t_2\}$	S_{58}
			$\{t_6\}$	S_{57}
14	S_{54}	{[2 0 1 0 0 1 0 1 0 0]', [0 0 0 0 1 1]'}	-	S_{21}
			$\{t_1\}$	S_{22}
14	S_{55}	{[1 0 0 0 0 1 0 1 0 0]', [2 0 0 0 1 1]'}	-	S_{23}
14	S_{56}	{[1 0 0 0 0 1 0 1 0 0]', [1 0 0 0 1 1]'}	-	S_{24}
14	S_{57}	{[1 1 0 0 0 1 0 1 0 0]', [0 0 0 0 1 1]'}	-	S_{24}
14	†S_{58}	{[1 0 1 1 0 1 0 2 0 0]', [0 0 0 0 0 0]'}	-	S_{58}
			$\{t_1\}$	S_{59}
			$\{t_3\}$	S_{60}
			$\{t_1, t_3\}$	S_{61}
15	S_{59}	{[0 0 0 1 0 1 0 2 0 0]', [2 0 0 0 0 0]'}	-	S_{62}
			$\{t_3\}$	S_{63}
15	†S_{60}	{[1 0 1 0 1 0 0 2 1 0]', [0 0 0 0 0 0]'}	-	S_{60}
			$\{t_1\}$	S_{61}
			$\{t_4\}$	S_{64}
			$\{t_6\}$	S_{65}
			$\{t_1, t_4\}$	S_{66}
			$\{t_1, t_6\}$	S_{67}
15	S_{61}	{[0 0 0 0 1 0 0 2 1 0]', [2 0 0 0 0 0]'}	-	S_{63}
			$\{t_4\}$	S_{68}
			$\{t_6\}$	S_{69}
16	S_{62}	{[0 0 0 1 0 1 0 2 0 0]', [1 0 0 0 0 0]'}	-	S_{70}
			$\{t_3\}$	S_{71}

† *denotes relaxed states and * denotes relaxed and deadlock states.*

Table A2.1. (Continue) *Reachability Set for TdAPN in Figure 6.2*

k	The State of TdAPN		ϕ	*Next State*
16	S_{63}	{[0 0 0 0 1 0 0 2 1 0]', [1 0 0 0 0 0]'}	-	S_{71}
			$\{t_4\}$	S_{72}
			$\{t_6\}$	S_{73}
16	†S_{64}	{[1 0 1 0 0 1 1 2 0 0]', [0 0 0 0 0 0]'}	-	S_{64}
			$\{t_1\}$	S_{66}
16	S_{65}	{[1 0 1 0 1 0 0 1 0 0]', [0 0 0 0 1 1]'}	-	S_{31}
			$\{t_1\}$	S_{32}
16	S_{66}	{[0 0 0 0 0 1 1 2 0 0]', [2 0 0 0 0 0]'}	-	S_{68}
16	S_{67}	{[0 0 0 0 1 0 0 1 0 0]', [2 0 0 0 1 1]'}	-	S_{34}
16	S_{68}	{[0 0 0 0 0 1 1 2 0 0]', [1 0 0 0 0 0]'}	-	S_{72}
16	S_{69}	{[0 0 0 0 1 0 0 1 0 0]', [1 0 0 0 1 1]'}	-	S_{42}
17	†S_{70}	{[0 1 0 1 0 1 0 2 0 0]', [0 0 0 0 0 0]'}	-	S_{70}
			$\{t_3\}$	S_{71}
17	†S_{71}	{[0 1 0 0 1 0 0 2 1 0]', [0 0 0 0 0 0]'}	-	S_{71}
			$\{t_2\}$	S_{74}
			$\{t_4\}$	S_{72}
			$\{t_6\}$	S_{73}
17	*S_{72}	{[0 1 0 0 0 1 1 2 0 0]', [0 0 0 0 0 0]'}	-	S_{72}
17	S_{73}	{[0 1 0 0 1 0 0 1 0 0]', [0 0 0 0 1 1]'}	-	S_{42}
18	*S_{74}	{[0 0 1 1 1 0 0 2 0 0]', [0 0 0 0 0 0]'}	-	S_{74}

† *denotes relaxed states and * denotes relaxed and deadlock states.*

APPENDIX-3 – THE REACHABILITY SET OF RAILWAY EXAMPLE

In this appendix, the reachability set of the railway-systems example in Figure 6.5 is given in Table A3.1. Descriptions of columns in Table A3.1 are similar to explanations for Table 3.3.

Table A3.1. *Reachability Set for TdAPN in Figure 6.5*

k	**The State of TdAPN**		ϕ	***Next State***
0	†S_0	{[1 0 1 0 0 1 0 0 1]', [0 0]'}	-	S_0
			$\{t_4\}$	S_1
			$\{t_7\}$	S_2
			$\{t_4, t_7\}$	S_3
1	†S_1	{[1 0 0 1 0 1 0 0 1]', [0 0]'}	-	S_1
			$\{t_3\}$	S_0
			$\{t_3, t_7\}$	S_3
1	†S_2	{[1 0 1 0 0 0 1 0 1]', [0 0]'}	-	S_2
			$\{t_2\}$	S_4
			$\{t_4\}$	S_3
			$\{t_6\}$	S_0
			$\{t_4, t_6\}$	S_1
1	†S_3	{[1 0 0 1 0 0 1 0 1]', [0 0]'}	-	S_3
			$\{t_3\}$	S_2
			$\{t_6\}$	S_1
			$\{t_3, t_6\}$	S_0
2	S_4	{[0 0 1 0 0 0 1 1 0]', [0 2]'}	-	S_5
			$\{t_4\}$	S_6
			$\{t_6\}$	S_7
			$\{t_4, t_6\}$	S_8
3	S_5	{[0 0 1 0 0 0 1 1 0]', [0 1]'}	-	S_9
			$\{t_4\}$	S_{10}
			$\{t_6\}$	S_{11}
			$\{t_4, t_6\}$	S_{12}
3	S_6	{[0 0 0 1 0 0 1 1 0]', [0 1]'}	-	S_{10}
			$\{t_5\}$	S_{13}
			$\{t_6\}$	S_{12}
			$\{t_5, t_6\}$	S_{14}
3	S_7	{[0 0 1 0 0 1 0 1 0]', [0 1]'}	-	S_{11}
			$\{t_4\}$	S_{12}
4	S_8	{[0 0 0 1 0 1 0 1 0]', [0 1]'}	-	S_{12}
			$\{t_5\}$	S_{14}
4	†S_9	{[0 1 1 0 0 0 1 1 0]', [0 0]'}	-	S_9
			$\{t_4\}$	S_{10}
			$\{t_6\}$	S_{11}
			$\{t_4, t_6\}$	S_{12}

† *denotes relaxed states.*

Table A3.1. (Continue) *Reachability Set for TdAPN in Figure 6.5*

k	**The State of TdAPN**		ϕ	***Next State***
4	†S_{10}	{[0 1 0 1 0 0 1 1 0]', [0 0]'}	-	S_{10}
			$\{t_5\}$	S_{13}
			$\{t_6\}$	S_{12}
			$\{t_5, t_6\}$	S_{14}
4	†S_{11}	{[0 1 1 0 0 1 0 1 0]', [0 0]'}	-	S_{11}
			$\{t_4\}$	S_{12}
4	†S_{12}	{[0 1 0 1 0 1 0 1 0]', [0 0]'}	-	S_{12}
			$\{t_5\}$	S_{14}
4	†S_{13}	{[0 1 0 1 1 0 1 0 0]', [0 0]'}	-	S_{13}
			$\{t_6\}$	S_{14}
4	†S_{14}	{[0 1 0 1 1 1 0 0 0]', [0 0]'}	-	S_{14}
			$\{t_1\}$	S_{15}
5	S_{15}	{[0 0 0 1 0 1 0 0 1]', [3 0]'}	-	S_{16}
			$\{t_3\}$	S_{17}
			$\{t_7\}$	S_{18}
6	S_{16}	{[0 0 0 1 0 1 0 0 1]', [2 0]'}	-	S_{19}
			$\{t_3\}$	S_{20}
			$\{t_7\}$	S_{21}
6	S_{17}	{[0 0 1 0 0 1 0 0 1]', [2 0]'}	-	S_{20}
			$\{t_4\}$	S_{19}
			$\{t_7\}$	S_{22}
			$\{t_4, t_7\}$	S_{21}
6	S_{18}	{[0 0 0 1 0 0 1 0 1]', [2 0]'}	-	S_{21}
			$\{t_3\}$	S_{22}
			$\{t_6\}$	S_{19}
			$\{t_3, t_6\}$	S_{20}
7	S_{19}	{[0 0 0 1 0 1 0 0 1]', [1 0]'}	-	S_1
			$\{t_3\}$	S_0
			$\{t_7\}$	S_3
7	S_{20}	{[0 0 1 0 0 1 0 0 1]', [1 0]'}	-	S_0
			$\{t_4\}$	S_1
			$\{t_7\}$	S_2
			$\{t_4, t_7\}$	S_3
7	S_{21}	{[0 0 0 1 0 0 1 0 1]', [1 0]'}	-	S_3
			$\{t_3\}$	S_2
			$\{t_6\}$	S_1
			$\{t_3, t_6\}$	S_0
7	S_{22}	{[0 0 1 0 0 0 1 0 1]', [1 0]'}	-	S_2
			$\{t_4\}$	S_3
			$\{t_6\}$	S_0
			$\{t_4, t_6\}$	S_1

† *denotes relaxed states.*

APPENDIX-4 – THE REACHABILITY SET OF AUTOMOTIVE EXAMPLE

In this appendix, the reachability set of the automotive-systems example in Figure 6.9 is given in Table A4.1. Descriptions of columns in Table A4.1 are similar to explanations for Table 3.3.

Table A4.1. *Reachability Set for TdAPN in Figure 6.9*

k	**The State of TdAPN**		ϕ	***Next State***
0	†S_0	{[0 0 0 0 0 0 0 0 0 0 0 0 1]', [0 0 0 0 0 0 0 0 0 0 0 0 0]'}	-	S_0
			$\{t_{11}\}$	S_1
1	†S_1	{[1 1 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0 0 0 0]'}	-	S_1
			$\{t_1\}$	S_2
			$\{t_2\}$	S_3
			$\{t_1, t_2\}$	S_4
2	S_2	{[0 1 0 0 0 0 0 0 0 0 0 0 0]', [3 0 0 0 0 0 0 0 0 0 0 0 0]'}	-	S_5
			$\{t_2\}$	S_6
2	S_3	{[1 0 0 0 0 0 0 0 0 0 0 0 0]', [0 1 0 0 0 0 0 0 0 0 0 0 0]'}	-	S_7
			$\{t_1\}$	S_8
2	S_4	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [3 1 0 0 0 0 0 0 0 0 0 0 0]'}	-	S_9
3	S_5	{[0 1 0 0 0 0 0 0 0 0 0 0 0]', [2 0 0 0 0 0 0 0 0 0 0 0 0]'}	-	S_{10}
			$\{t_2\}$	S_{11}
3	S_6	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [2 1 0 0 0 0 0 0 0 0 0 0 0]'}	-	S_{12}
3	†S_7	{[1 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]'}	-	S_7
			$\{t_1\}$	S_8
			$\{t_3\}$	S_{13}
			$\{t_1, t_3\}$	S_{14}
3	S_8	{[0 0 0 1 0 0 0 0 0 0 0 0 0]', [3 0 0 0 0 0 0 0 0 0 0 0 0]'}	-	S_9
			$\{t_3\}$	S_{15}
3	S_9	{[0 0 0 1 0 0 0 0 0 0 0 0 0]', [2 0 0 0 0 0 0 0 0 0 0 0 0]'}	-	S_{12}
			$\{t_3\}$	S_{16}
3	S_{10}	{[0 1 0 0 0 0 0 0 0 0 0 0 0]', [1 0 0 0 0 0 0 0 0 0 0 0 0]'}	-	S_{17}
			$\{t_2\}$	S_{18}
4	S_{11}	{[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]'}	-	S_{19}
4	S_{12}	{[0 0 0 1 0 0 0 0 0 0 0 0 0]', [1 0 0 0 0 0 0 0 0 0 0 0 0]'}	-	S_{19}
			$\{t_3\}$	S_{20}
4	S_{13}	{[1 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 2 0 0 0 0 0 0 0 0 0 0]'}	-	S_{21}
			$\{t_1\}$	S_{22}
4	S_{14}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [3 0 2 0 0 0 0 0 0 0 0 0 0]'}	-	S_{23}

† *denotes relaxed states.*

Table A4.1. (Continue) *Reachability Set for TdAPN in Figure 6.9*

k	**The State of TdAPN**		ϕ	***Next State***
4	S_{15}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [2 0 2 0 0 0 0 0 0 0]'}	-	S_{24}
4	S_{16}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [1 0 2 0 0 0 0 0 0 0]'}	-	S_{25}
5	†S_{17}	{[0 1 1 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0]'}	-	S_{17}
			$\{t_2\}$	S_{18}
			$\{t_4\}$	S_{26}
			$\{t_2, t_4\}$	S_{27}
5	S_{18}	{[0 0 1 0 0 0 0 0 0 0 0 0 0]', [0 1 0 0 0 0 0 0 0 0]'}	-	S_{19}
			$\{t_4\}$	S_{28}
5	†S_{19}	{[0 0 1 1 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0]'}	-	S_{19}
			$\{t_3\}$	S_{20}
			$\{t_4\}$	S_{28}
			$\{t_3, t_4\}$	S_{29}
5	S_{20}	{[0 0 1 0 0 0 0 0 0 0 0 0 0]', [0 0 2 0 0 0 0 0 0 0]'}	-	S_{25}
			$\{t_3, t_4\}$	S_{30}
5	S_{21}	{[1 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 1 0 0 0 0 0 0 0]'}	-	S_{31}
			$\{t_1\}$	S_{32}
5	S_{22}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [3 0 1 0 0 0 0 0 0 0]'}	-	S_{33}
5	S_{23}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [2 0 1 0 0 0 0 0 0 0]'}	-	S_{34}
5	S_{24}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [1 0 1 0 0 0 0 0 0 0]'}	-	S_{35}
5	S_{25}	{[0 0 1 0 0 0 0 0 0 0 0 0 0]', [0 0 1 0 0 0 0 0 0 0]'}	-	S_{35}
			$\{t_3, t_4\}$	S_{36}
6	S_{26}	{[0 1 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 2 0 0 0 0 0 0]'}	-	S_{37}
			$\{t_2\}$	S_{38}
6	S_{27}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 1 0 2 0 0 0 0 0 0]'}	-	S_{39}
6	S_{28}	{[0 0 0 1 0 0 0 0 0 0 0 0 0]', [0 0 0 2 0 0 0 0 0 0]'}	-	S_{39}
			$\{t_3\}$	S_{40}
6	S_{29}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 2 2 0 0 0 0 0 0]'}	-	S_{41}
6	S_{30}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 1 2 0 0 0 0 0 0]'}	-	S_{42}
6	†S_{31}	{[1 0 0 0 0 1 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0]'}	-	S_{31}
			$\{t_1\}$	S_{32}
			$\{t_5\}$	S_{43}
			$\{t_1, t_5\}$	S_{44}
6	S_{32}	{[0 0 0 0 0 1 0 0 0 0 0 0 0]', [3 0 0 0 0 0 0 0 0 0]'}	-	S_{33}
			$\{t_5\}$	S_{45}
6	S_{33}	{[0 0 0 0 0 1 0 0 0 0 0 0 0]', [2 0 0 0 0 0 0 0 0 0]'}	-	S_{34}
			$\{t_5\}$	S_{46}

† *denotes relaxed states.*

Table A4.1. (Continue) *Reachability Set for TdAPN in Figure 6.9*

k	**The State of TdAPN**		ϕ	***Next State***
6	S_{34}	{[0 0 0 0 0 1 0 0 0 0 0 0 0]', [1 0 0 0 0 0 0 0 0 0 0]'}	-	S_{35}
			$\{t_5\}$	S_{47}
6	†S_{35}	{[0 0 1 0 0 1 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0 0]'}	-	S_{35}
			$\{t_4\}$	S_{36}
			$\{t_5\}$	S_{47}
			$\{t_4, t_5\}$	S_{48}
6	S_{36}	{[0 0 0 0 0 1 0 0 0 0 0 0 0]', [0 0 0 2 0 0 0 0 0 0 0]'}	-	S_{42}
			$\{t_5\}$	S_{49}
7	S_{37}	{[0 1 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 1 0 0 0 0 0 0 0]'}	-	S_{50}
			$\{t_2\}$	S_{51}
7	S_{38}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 1 0 1 0 0 0 0 0 0 0]'}	-	S_{52}
7	S_{39}	{[0 0 0 1 0 0 0 0 0 0 0 0 0]', [0 0 0 1 0 0 0 0 0 0 0]'}	-	S_{52}
			$\{t_3\}$	S_{53}
7	S_{40}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 2 1 0 0 0 0 0 0 0]'}	-	S_{54}
7	S_{41}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 1 1 0 0 0 0 0 0 0]'}	-	S_{55}
7	S_{42}	{[0 0 0 0 0 1 0 0 0 0 0 0 0]', [0 0 0 1 0 0 0 0 0 0 0]'}	-	S_{55}
			$\{t_5\}$	S_{56}
7	S_{43}	{[1 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 1 0 0 0 0 0 0]'}	-	S_{57}
			$\{t_1\}$	S_{58}
7	S_{44}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [3 0 0 0 1 0 0 0 0 0 0]'}	-	S_{59}
7	S_{45}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [2 0 0 0 1 0 0 0 0 0 0]'}	-	S_{60}
7	S_{46}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [1 0 0 0 1 0 0 0 0 0 0]'}	-	S_{61}
7	S_{47}	{[0 0 1 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 1 0 0 0 0 0 0]'}	-	S_{61}
			$\{t_4\}$	S_{62}
7	S_{48}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 2 1 0 0 0 0 0 0]'}	-	S_{63}
7	S_{49}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 1 1 0 0 0 0 0 0]'}	-	S_{64}
8	†S_{50}	{[0 1 0 0 1 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0 0]'}	-	S_{50}
			$\{t_2\}$	S_{51}
			$\{t_6\}$	S_{65}
			$\{t_2, t_6\}$	S_{66}
8	S_{51}	{[0 0 0 0 1 0 0 0 0 0 0 0 0]', [0 1 0 0 0 0 0 0 0 0 0]'}	-	S_{52}
			$\{t_6\}$	S_{67}
8	†S_{52}	{[0 0 0 1 1 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0 0]'}	-	S_{52}
			$\{t_3\}$	S_{53}
			$\{t_6\}$	S_{67}
			$\{t_3, t_6\}$	S_{68}

† *denotes relaxed states.*

Table A4.1. (Continue) *Reachability Set for TdAPN in Figure 6.9*

k	**The State of TdAPN**		ϕ	***Next State***
8	S_{53}	{[0 0 0 0 1 0 0 0 0 0 0 0 0]', [0 0 2 0 0 0 0 0 0 0]'}	-	S_{54}
			$\{t_6\}$	S_{69}
8	S_{54}	{[0 0 0 0 1 0 0 0 0 0 0 0 0]', [0 0 1 0 0 0 0 0 0 0]'}	-	S_{55}
			$\{t_6\}$	S_{70}
8	†S_{55}	{[0 0 0 0 1 1 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0]'}	-	S_{55}
			$\{t_5\}$	S_{56}
			$\{t_6\}$	S_{70}
			$\{t_5, t_6\}$	S_{71}
8	S_{56}	{[0 0 0 0 1 0 0 0 0 0 0 0 0]', [0 0 0 0 1 0 0 0 0 0]'}	-	S_{64}
			$\{t_6\}$	S_{72}
8	†S_{57}	{[1 0 0 0 0 0 0 1 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0]'}	-	S_{57}
			$\{t_1\}$	S_{58}
8	S_{58}	{[0 0 0 0 0 0 0 1 0 0 0 0 0]', [3 0 0 0 0 0 0 0 0 0]'}	-	S_{59}
8	S_{59}	{[0 0 0 0 0 0 0 1 0 0 0 0 0]', [2 0 0 0 0 0 0 0 0 0]'}	-	S_{60}
8	S_{60}	{[0 0 0 0 0 0 0 1 0 0 0 0 0]', [1 0 0 0 0 0 0 0 0 0]'}	-	S_{61}
8	†S_{61}	{[0 0 1 0 0 0 0 1 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0]'}	-	S_{61}
			$\{t_4\}$	S_{62}
8	S_{62}	{[0 0 0 0 0 0 0 1 0 0 0 0 0]', [0 0 0 2 0 0 0 0 0 0]'}	-	S_{63}
8	S_{63}	{[0 0 0 0 0 0 0 1 0 0 0 0 0]', [0 0 0 1 0 0 0 0 0 0]'}	-	S_{64}
8	†S_{64}	{[0 0 0 0 1 0 0 1 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0]'}	-	S_{64}
			$\{t_6\}$	S_{72}
9	S_{65}	{[0 1 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 3 0 0 0 0]'}	-	S_{73}
			$\{t_2\}$	S_{74}
9	S_{66}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 1 0 0 0 3 0 0 0 0]'}	-	S_{75}
9	S_{67}	{[0 0 0 1 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 3 0 0 0 0]'}	-	S_{75}
			$\{t_3\}$	S_{76}
9	S_{68}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 2 0 0 3 0 0 0 0]'}	-	S_{77}
9	S_{69}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 1 0 0 3 0 0 0 0]'}	-	S_{78}
9	S_{70}	{[0 0 0 0 0 1 0 0 0 0 0 0 0]', [0 0 0 0 0 3 0 0 0 0]'}	-	S_{78}
			$\{t_5\}$	S_{79}
10	S_{74}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 1 0 0 0 2 0 0 0 0]'}	-	S_{83}
10	S_{75}	{[0 0 0 1 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 2 0 0 0 0]'}	-	S_{83}
			$\{t_3\}$	S_{84}
10	S_{76}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 2 0 0 2 0 0 0 0]'}	-	S_{85}

† *denotes relaxed states.*

Table A4.1. (Continue) *Reachability Set for TdAPN in Figure 6.9*

k	**The State of TdAPN**		ϕ	***Next State***
10	S_{77}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 1 0 0 2 0 0 0 0]'}	-	S_{86}
10	S_{78}	{[0 0 0 0 0 1 0 0 0 0 0 0 0]', [0 0 0 0 0 2 0 0 0 0]'}	-	S_{86}
			$\{t_5\}$	S_{87}
10	S_{79}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 1 2 0 0 0 0]'}	-	S_{88}
10	S_{80}	{[0 0 0 0 0 0 0 1 0 0 0 0 0]', [0 0 0 0 0 2 0 0 0 0]'}	-	S_{88}
11	S_{81}	{[0 1 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 1 0 0 0 0]'}	-	S_{89}
			$\{t_2\}$	S_{90}
11	S_{82}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 1 0 0 0 1 0 0 0 0]'}	-	S_{91}
11	S_{83}	{[0 0 0 1 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 1 0 0 0 0]'}	-	S_{91}
			$\{t_3\}$	S_{92}
11	S_{84}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 2 0 0 1 0 0 0 0]'}	-	S_{93}
11	S_{85}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 1 0 0 1 0 0 0 0]'}	-	S_{94}
11	S_{86}	{[0 0 0 0 0 1 0 0 0 0 0 0 0]', [0 0 0 0 0 1 0 0 0 0]'}	-	S_{94}
			$\{t_5\}$	S_{95}
11	S_{87}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 1 1 0 0 0 0]'}	-	S_{96}
11	S_{88}	{[0 0 0 0 0 0 0 1 0 0 0 0 0]', [0 0 0 0 0 1 0 0 0 0]'}	-	S_{96}
12	$^{\dagger}S_{89}$	{[0 1 0 0 0 0 1 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0]'}	-	S_{89}
			$\{t_2\}$	S_{90}
12	S_{90}	{[0 0 0 0 0 0 1 0 0 0 0 0 0]', [0 1 0 0 0 0 0 0 0 0]'}	-	S_{91}
12	$^{\dagger}S_{91}$	{[0 0 0 1 0 0 1 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0]'}	-	S_{91}
			$\{t_3\}$	S_{92}
12	S_{92}	{[0 0 0 0 0 0 1 0 0 0 0 0 0]', [0 0 2 0 0 0 0 0 0 0]'}	-	S_{93}
12	S_{93}	{[0 0 0 0 0 0 1 0 0 0 0 0 0]', [0 0 1 0 0 0 0 0 0 0]'}	-	S_{94}
12	$^{\dagger}S_{94}$	{[0 0 0 0 0 1 1 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0]'}	-	S_{94}
			$\{t_5\}$	S_{95}
12	S_{95}	{[0 0 0 0 0 0 1 0 0 0 0 0 0]', [0 0 0 0 1 0 0 0 0 0]'}	-	S_{96}
12	$^{\dagger}S_{96}$	{[0 0 0 0 0 0 1 1 0 0 0 0 0]', [0 0 0 0 0 0 0 0 0 0]'}	-	S_{96}
			$\{t_7\}$	S_{97}
13	S_{97}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 7 7 0 0]'}	-	S_{98}
14	S_{98}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 6 6 0 0]'}	-	S_{99}
15	S_{99}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 5 5 0 0]'}	-	S_{100}

† *denotes relaxed states.*

Table A4.1. (Continue) *Reachability Set for TdAPN in Figure 6.9*

k	The State of TdAPN		ϕ	*Next State*
16	S_{100}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 4 4 0 0]'}	-	S_{101}
17	S_{101}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 3 3 0 0]'}	-	S_{102}
18	S_{102}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 2 2 0 0]'}	-	S_{103}
19	S_{103}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 1 1 0 0]'}	-	S_{102}
20	†S_{104}	{[0 0 0 0 0 0 0 0 1 1 0 0 0]', [0 0 0 0 0 0 0 0 0 0]'}	-	S_{104}
			$\{t_8\}$	S_{105}
			$\{t_9\}$	S_{106}
			$\{t_8, t_9\}$	S_{107}
21	S_{105}	{[0 0 0 0 0 0 0 0 0 1 0 0 0]', [0 0 0 0 0 0 0 0 4 0]'}	-	S_{108}
			$\{t_9\}$	S_{109}
21	S_{106}	{[0 0 0 0 0 0 0 0 1 0 0 0 0]', [0 0 0 0 0 0 0 0 0 3]'}	-	S_{110}
			$\{t_8\}$	S_{111}
21	S_{107}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 4 3]'}	-	S_{112}
22	S_{108}	{[0 0 0 0 0 0 0 0 0 1 0 0 0]', [0 0 0 0 0 0 0 0 3 0]'}	-	S_{113}
			$\{t_9\}$	S_{114}
22	S_{109}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 3 3]'}	-	S_{115}
22	S_{110}	{[0 0 0 0 0 0 0 0 1 0 0 0 0]', [0 0 0 0 0 0 0 0 0 2]'}	-	S_{116}
			$\{t_8\}$	S_{117}
22	S_{111}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 4 2]'}	-	S_{118}
22	S_{112}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 3 2]'}	-	S_{119}
23	S_{113}	{[0 0 0 0 0 0 0 0 0 1 0 0 0]', [0 0 0 0 0 0 0 0 2 0]'}	-	S_{120}
			$\{t_9\}$	S_{121}
23	S_{114}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 2 3]'}	-	S_{122}
23	S_{115}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 2 2]'}	-	S_{123}
23	S_{116}	{[0 0 0 0 0 0 0 0 1 0 0 0 0]', [0 0 0 0 0 0 0 0 0 1]'}	-	S_{124}
			$\{t_8\}$	S_{125}
23	S_{117}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 4 1]'}	-	S_{126}
23	S_{118}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 3 1]'}	-	S_{127}
23	S_{119}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 2 1]'}	-	S_{128}
24	S_{120}	{[0 0 0 0 0 0 0 0 0 1 0 0 0]', [0 0 0 0 0 0 0 0 0 1 0]'}	-	S_{129}
			$\{t_9\}$	S_{130}
24	S_{121}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 1 3]'}	-	S_{131}

† *denotes relaxed states.*

Table A4.1. (Continue) *Reachability Set for TdAPN in Figure 6.9*

k	**The State of TdAPN**		ϕ	***Next State***
24	S_{122}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 1 2]'}	-	S_{132}
24	S_{123}	{[0 0 0 0 0 0 0 0 0 0 0 0 0]', [0 0 0 0 0 0 0 0 1 1]'}	-	S_{133}
24	†S_{124}	{[0 0 0 0 0 0 0 0 1 0 0 1 0]', [0 0 0 0 0 0 0 0 0 0 0]'}	-	S_{124}
			$\{t_8\}$	S_{125}
24	S_{125}	{[0 0 0 0 0 0 0 0 0 0 0 0 1 0]', [0 0 0 0 0 0 0 0 0 4 0]'}	-	S_{126}
24	S_{126}	{[0 0 0 0 0 0 0 0 0 0 0 0 1 0]', [0 0 0 0 0 0 0 0 0 3 0]'}	-	S_{127}
24	S_{127}	{[0 0 0 0 0 0 0 0 0 0 0 0 1 0]', [0 0 0 0 0 0 0 0 0 2 0]'}	-	S_{128}
24	S_{128}	{[0 0 0 0 0 0 0 0 0 0 0 0 1 0]', [0 0 0 0 0 0 0 0 0 1 0]'}	-	S_{133}
25	†S_{129}	{[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]'}	-	S_{129}
			$\{t_9\}$	S_{130}
25	S_{130}	{[0 0 0 0 0 0 0 0 0 0 0 1 0 0]', [0 0 0 0 0 0 0 0 0 0 3]'}	-	S_{131}
25	S_{131}	{[0 0 0 0 0 0 0 0 0 0 0 1 0 0]', [0 0 0 0 0 0 0 0 0 0 2]'}	-	S_{132}
25	S_{132}	{0 0 0 0 0 0 0 0 0 0 0 1 0 0]', [0 0 0 0 0 0 0 0 0 0 1]'}	-	S_{132}
25	†S_{133}	{[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]'}	-	S_{133}
			$\{t_{10}\}$	S_0

† *denotes relaxed states.*

APPENDIX-5 – THE TIMED-REACHABILITY TREE FOR THE MANUFACTURING EAMPLE

In this appendix, the timed-reachability tree for the manufacturing-systems example in Figure 6.2 is shown in Figure A5.1. Descriptions of this graph are similar to explanations for Figure 3.8.

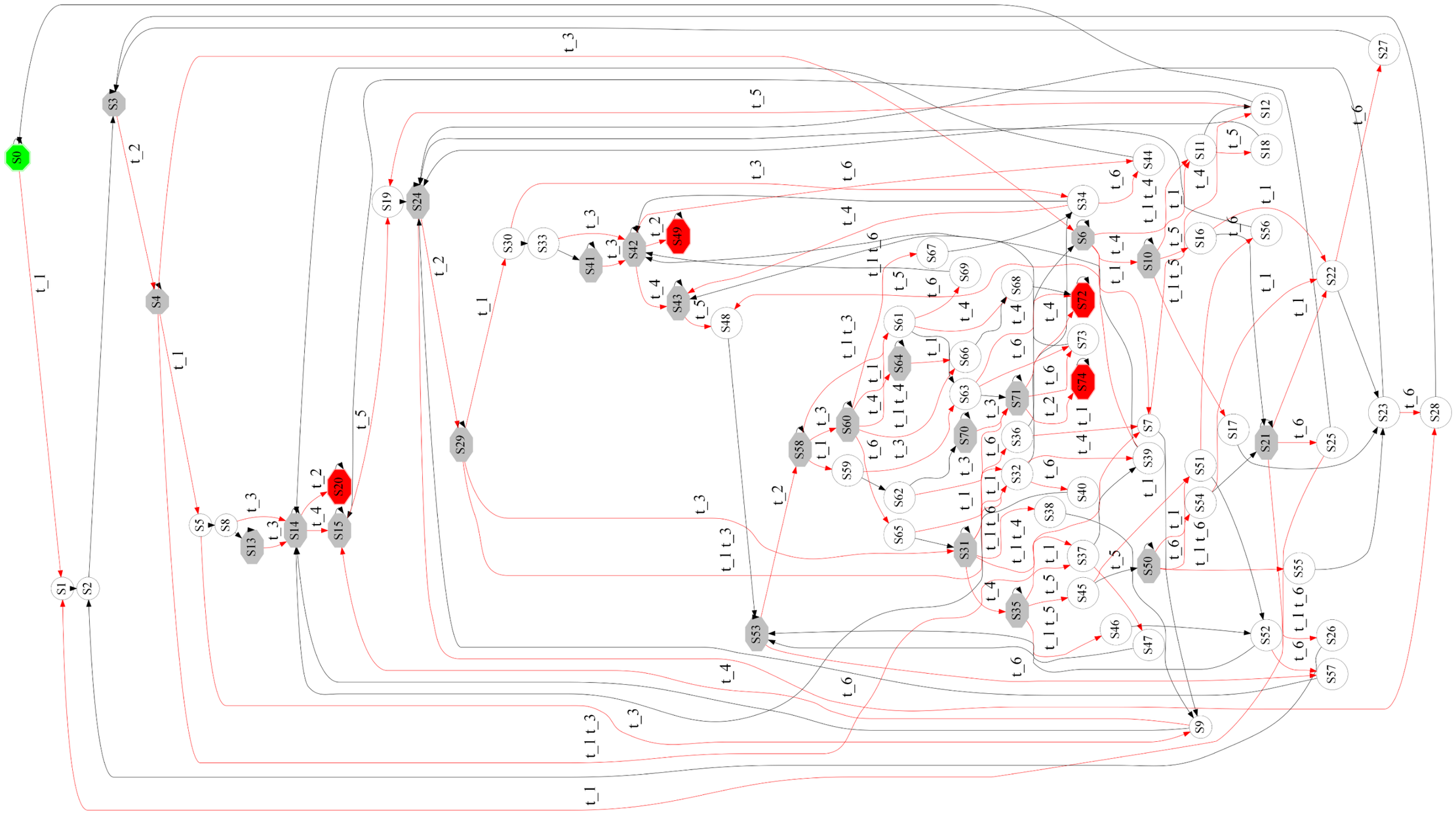


Figure A5.1. *Timed-reeachability tree for TdAPN in Figure 6.2*

CURRICULUM VITAE

Name Surname : Alpaslan YUFKA

Foreign Language : English (Advanced, YÖKDİL 03/2018: 81.25),
Romanian (Beginner), German (Beginner)

Birth Place and Year : İzmir, Turkey / 1984

E-Mail : ayufka@gmail.com (alternatively, alpaslan.yufka@gmail.com)

Education

- 2019, Eskişehir Technical University, Natural Science, Department of Electrical-Electronics Engineering, Electrical-Electronics Engineering (Ph.D., English), Eskişehir, Turkey, GPA: 3.86/4.
- 2018, UDACITY, Nanodegree of "Intro to Self-Driving Cars", (Online) USA. https://graduation.udacity.com/confirm/Z3CPSFC4
- 2015, Anadolu University, The Faculty of Business Administration, Business Administration (B.B.A.), Eskişehir, Turkey, GPA: 2.70/4.
- 2010, Eskişehir Osmangazi University, Natural Science, Department of Electrical-Electronics Engineering, Electrical-Electronics Engineering (M.Sc.), Eskişehir, Turkey, GPA: 3.50/4.
- 2009, Eskişehir Osmangazi University, The Faculty of Engineering and Architecture Administration, Department of Electrical-Electronics Engineering, Electrical-Electronics Engineering (B.Sc., English), Eskişehir, Turkey, GPA: 3.62/4.
- 2008, Technical University of Cluj Napoca, Computer Science and Automation (ERASMUS, English & Romanian), Cluj Napoca, Romania, GPA: 3.33/4.
- 2002, İzmir Nevvar Salih İşgören High School, Foreign Language Aided Science (English), İzmir, Turkey, GPA: 4.19/5.

Job Experience

- 2016 - Present, Otokar Automotive and Defense Industry Inc., Turkey (KOC Holding), Sakarya, Turkey
    - (Military Vehicles R&D) Specialist Engineer – Electronic Mission-Equipment Systems
    - (Military Vehicles R&D) Specialist Engineer – Electrical-Electronics System
        - System Engineering and Integration of Military Electronic Mission Equipment Systems for Military Armored Vehicles (Cobra, Cobra II, Arma, Tulpar)
        - Concept Studies on Military Self-Driving Cars, Military Autonomous and Unmanned Ground Vehicles
        - Electrical and Electronic Subsystem Architecture and Interfaces
        - Requirement Management
        - Benchmarking and selection of electronic mission equipments from different suppliers
        - System/Subsystem Validation and Verification, Test Planning
        - System/Subsystem Design Documentation
        - Technical Consultancy to Project Side
- 2011 - 2016, Otokar Automotive and Defense Industry Inc., Turkey (KOC Holding), Sakarya, Turkey
    - (S.S.B. ALTAY Main Battle Tank R&D) Electrical-Electronics System Engineer
        - System Engineering and Integration of Military Electronic Mission Equipment Systems for ALTAY
        - System Engineering Steps of SRR, SSR, PDR, CDR and TRR
        - System/Subsystem Qualification Steps of SSQT and SQT
        - Military Standards: MIL-STD-810, 461, 464, 498, 1275D, 1472, 1521, STANAG 4579, 4347
        - System/Subsystem Validation and Verification, Test Planning
        - Electrical and Electronic Subsystem Architecture and Interfaces
        - Requirement Management
        - System/Subsystem Design Documentation

- 2008 - 2010, Eskişehir Osmangazi University Artificial Intelligence and Robotics Laboratory, Eskişehir, Turkey
    - Laboratory Assistant and Researcher
        - Research Engineer at TUBITAK: 107E064's project that is namely Mobile Robot Route Planning for Complete Coverage of Dynamic Indoor Environments.
        - Research Engineer at Thesis, which is Motion Planning and Control Scheme for Cooperative Transportation by Multiple Mobile Robots.
            - Path Planning, Indoor Localization,
            - Robot Motion Planning and Control Systems on Autonomous Mobile Robots,
            - Software development in C/C++ Object Oriented Programming and MATLAB,
            - Background in Artificial Intelligence and Computer Vision
- 2007, Eskişehir TUSAŞ Engine Industries Inc. (TEI), Eskişehir, Turkey
    - Engineer-Intern at the Department of Electronics Maintenance

<u>Publications Related to This Thesis</u>

Published papers are as follows:

- Yufka, A., Özkan, H.A. and Aybar, A. (2016). A Formal Method and Novel Graphical Representation for Deterministic Timed-Arc Petri Nets. *Proceeding National Conference on "Otomatik Kontrol Ulusal Toplantisi"*, Eskişehir, Turkey, Sep. 29 – Dec. 01, 2016, TOK'16. pp. 209-213.
- Yufka, A., Özkan, H.A. and Aybar, A. (2017). Timed Arc Petri Nets: The Time-Element Approach. *Proceeding 10th International Conference on Electrical and Electronics Engineering*, Bursa, Turkey, Nov. 30 – Dec. 2, 2017, ELECO'17. pp. 794-798. IEEE.
- Yufka, A., Özkan, H.A. and Aybar, A. (2018). Timed Arc Petri Nets: The Impulsive Approach. *Proceeding 5th International Conference on Control, Decision and Information Technologies*, Thessaloniki, Greece, Apr. 10-13, 2018, CODIT'18. pp. 409-414. IEEE.

- Yufka, A., Özkan, H.A. and Aybar, A. (2018). Modeling Basic Components of Railway Systems Using Timed Arc Petri Nets. *Proceeding 5th International Conference on Control, Decision and Information Technologies*, Thessaloniki, Greece, Apr. 10-13, 2018, CODIT'18. pp. 427-432. IEEE.
- Yufka, A., Özkan, H.A. and Aybar, A. (2019). Reachability Set Algorithms for Timed-Arc Petri Nets. *Proceeding 395th International Conference on Electrical and Electronics Engineering*, Barcelona, Spain, Feb. 11-12, 2019, ICEEE'19. (Presented)

Publications in the preparation/submission stage are as follows:

- Yufka, A., Özkan, H.A. and Aybar, A. (2019). "A Mathematical Model of Timed-Arc Petri Nets Using Time Elements".
- Yufka, A., Özkan, H.A. and Aybar, A. (2019). "Forbidden State Controller for Timed-Arc Petri Nets".

Other Publications

Technical Reports

1. (2011-2017)(**Classified**) Some technical data packages (TDP) of T.R. M.S.B. Undersecretariat for Defence Industries (S.S.M.) ALTAY (Turkish National Main Battle Tank) Project for some specific electrical/electronic subsystems
    - Benchmarks
    - Request for Information (RFI)
    - System/Subsystem Specification Documents (SSS, SGÖ)
    - System/Subsystem Software Specification Documents (YGÖ)
    - System/Subsystem Design Description Documents (SSDD, STT)
    - System/Subsystem Interface Control Documents (ICD, AKD)
    - System/Subsystem Validation Procedures (KMTP)
    - System/Subsystem Validation Records and Reports
    - System/Subsystem Failure Analysis Documents (HAR)
    - System/Subsystem Engineering Change Proposals (ECP, MDT)
    - TTRD/System/Subsystem Requirement Change Proposals
    - Minutes of Meeting (MoM)

2. (2009) Yazıcı A., Parlaktuna O., Kapanoğlu M., Özkan M., Sipahioğlu A., Kirlik G. ve **Yufka A.** "*Mobile Robot Route Planning for Complete Coverage of Dynamic Indoor Environments*", The Final Report of The Scientific and Technical Research Council of Turkey (TUBITAK), Nov., Eskişehir, Project 107E064.

Journal Papers

1. (2018) **A. Yufka**, H.A. Özkan and A. Aybar, "BUG-0, 1, 2 Algoritmalari ve Petri Ağı Modelleri", *Anadolu University Journal of Science and Technology B- Theoritical Sciences*, vol. 6(2), pp. 129-139, October 2018, DOI: 10.20290/aubtdb.421865.
2. (2015) **A. Yufka** and M. Ozkan, "Formation-based control scheme for cooperative transportation by multiple mobile robots", *International Journal of Advanced Robotic Systems*, vol. 12:120(9), pp. 1-15, September 2015, ISSN: 1729-8806.
3. (2010) Ozkan M., Kirlik G., Parlaktuna O., **Yufka A.**, and Yazici A. "Fault-Tolerant Control Architecture for Multi-Robot Sensor-Based Coverage", *International Journal of Advanced Robotic Systems*, Special issue on Robotics for Risky Interventions and Environmental Surveillance, vol. 7 (1), pp. 67-74, ISSN 1729-8806.

Conference Papers

1. (2019) **A. Yufka**, H.A. Özkan and A. Aybar, "*Reachability Set Algorithms for Timed-Arc Petri Nets*". Proceeding 395th International Conference on Electrical and Electronics Engineering, Feb. 11-12, 2019, Barcelona, Spain. (Presented and in Publish)
2. (2018) **A. Yufka**, H.A. Özkan and A. Aybar, "*Modeling Basic Components of Railway Systems Using Timed Arc Petri Nets*", Proceeding of the 5th Int. Conf. on Control, Decision and Information Technologies, April 10-13, Thessaloniki, Greece, pp. 427-432. IEEE.
3. (2018) **A. Yufka**, H.A. Özkan and A. Aybar, "*Timed Arc Petri Nets: A Novel Representation*", Proceeding of the 5th Int. Conf. on Control, Decision and Information Technologies, April 10-13, Thessaloniki, Greece, pp. 409-414. IEEE.

4. (2017) **A. Yufka**, H.A. Özkan and A. Aybar, "*Timed Arc Petri Nets: The Time-Element Approach*", Proceeding of the 10th International Conf. on Electrical and Electronics Engineering, pp. 794-798, 30 Nov.- 2 Dec., Bursa, Turkey, pp. 794-798. IEEE.
5. (2016) **A. Yufka**, H.A. Özkan and A. Aybar, "*A Formal Method and Novel Graphical Representation for Deterministic Timed-Arc Petri Nets*", Proceeding of the National Conf. on Otomatik Kontrol Ulusal Toplantısı, pp. 209- 213, 29 Sep.- 1 Dec., Eskisehir, Turkey. ISBN: 978-605-9975-13-1.
6. (2015) **A. Yufka** and A. Aybar, "*Line estimation for a line-following mobile robot*", Proceeding of the 9th International Conf. on Electrical and Electronics Engineering, pp. 890-893, 26 – 28 Nov., Bursa, Turkey. ISBN: 978-1-4673-7912-0. IEEE.
7. (2014) M.S. Özdemir and **A. Yufka**, "*Application of MCDM Methods for a Group of Nonholonomic Mobile Robots to Determine the Best Route and the Most Suitable Robot*", Proceeding of the 13th International Symposium on the Analytic Hierarchy Process, pp. 183 - 185, 28 June, Washington, USA. ISBN: 978-1-888603-30-9.
8. (2013) **A. Yufka** and A. Aybar, "*BUG algorithm analysis using Petri net*", Proceeding of the 8th International Conf. on Electrical and Electronics Engineering, pp. 507 - 511, 28 - 30 Nov, Bursa, Turkey. ISBN: 978-1-4799-2888-0. IEEE.
9. (2010) **Yufka A**., Ozkan M., and Parlaktuna O. "*Formation-Based Cooperative Transportation by a Group of Non-holonomic Mobile Robots*", Proceeding of the 2010 IEEE International Conf. on Systems Man and Cybernetics, pp. 3300-3307, 10 – 13 Oct., Istanbul, Turkey. ISBN: 978-1-4244-6586-6. IEEE. (that it was a finalist for the: "IEEE International Conference on Systems, Man, and Cybernetics Best Student Paper Award" at SMC'10)
10. (2010) Gurler N., Cevher F.Y., **Yufka A.**, and Parlaktuna O. "*Direct-Motion Parallel Parking for a Vehicle with and without a Trailer*", Proceeding of the International Symposium on INnovations in Intelligent SysTems and Applications, pp. 108-112, 21-24 June, Kayseri, Turkey. ISBN 978-975-6478-58-5.

11. (2010) **Yufka A.** and Yazici A. (2010). "*An Intelligent PID Tuning Method for an Autonomous Mobile Robot*", Proceeding of the International Workshop on Unmanned Vehicles, pp. 130-133, 10-12 June, Istanbul, Turkey.
12. (2010) **Yufka A.** and Ozkan M. "*Cooperative Transportation by Multiple Autonomous Non-holonomic Mobile Robots*", Proceeding of the International Workshop on Unmanned Vehicles, pp. 134-139, 10-12 June, Istanbul, Turkey.
13. (2010) Altintasi C., Biberoglu M., **Yufka A.**, and Parlaktuna O. "*Direct-Motion Diagonal and Perpendicular Parking for a Vehicle with and without a Trailer*", Proceeding of the 1st International Symposium on Computing in Science & Engineering, pp. 559-563, 3-5 June, Aydın, Turkey. ISBN 978-605-61394-1-3.
14. (2010) **Yufka A.** and Ozkan M. "*Formation-Based Cooperative Transportation by a Group of Non-Holonomic Forklift-Type Mobile Robots*", Proceeding of the 1st International Symposium on Computing in Science & Engineering, pp. 131, 3-5 June, Aydın, Turkey. ISBN 978-605-61394-0-6.
15. (2010) **Yufka A.**, Özkan M., ve Parlaktuna O. "*Formation-Based Cooperative Transportation by Multiple Mobile Robots"*, Proceeding of the National Conf. on Otomatik Kontrol Ulusal Toplantısı, pp. 520-525, vol. PeB2-3, 21-23 Sep., Gebze, Kocaeli.
16. (2009) **Yufka A.**, Yazıcı A., and Parlaktuna O. "*A Smooth Path Generation Approach for Sensor-based Coverage Path Planning*", Proceeding of the 6th International Conf. on Electrical and Electronics Engineering, pp. 375-379, vol. 2, 5-8 Nov., Bursa, Turkey. ISBN 978-9944-89-820-1. IEEE.
17. (2009) **Yufka A.** and Parlaktuna O. "*Performance Comparison of the BUG's Algorithms for Mobile Robots*", Proceeding of the International Symposium on INnovations in Intelligent Systems and Applications, pp. 416-421, 29 June – 1 July, Trabzon, Turkey. ISBN 978-975-6983-58-4.
18. (2009) **Yufka A.** and Parlaktuna O. "*Performance Comparison of the BUG Algorithms for Mobile Robots*", Proceeding of the 5th International Symposium on Advanced Technologies, pp. 61-65, 13-15 May, Karabuk, Turkey. ISBN 978-605-60681-0-2.

19. (2008) Dobrucalı O., **Yufka A.**, Kaleci B., Özkan M., Parlaktuna O. "*Multi-Camera Localization System for Mobile Robots*", Proceeding of the 5th National Symposium on Electrical-Electronics and Computer Engineering, pp. 402-406, vol. Elektrik-Kontrol, 26-30 Nov., Bursa, Turkey. ISBN 978-9944-89-637-5.
20. (2008) **Yufka A.**, Dobrucalı O., Kaleci B., Özkan M., Parlaktuna O. "*Image-Based Localization System for Mobile Robots*", Proceeding of the National Conf. on Otomatik Kontrol Ulusal Toplantısı, pp. 47-51, vol. 1, 13-15 Nov., İstanbul, Turkey.

Dissertations

1. (2019) **Yufka A.** "*Timed-Arc Petri Nets Modeling and Forbidden State Control Approach*", Dissertation of the Doctor of Philosophy Program on Electrical-Electronics Engineering, Institute of Natural Sciences, Eskişehir Technical University, Eskisehir, Turkey.
2. (2010) **Yufka A.** "*Motion Planning and Control Scheme for Cooperative Transportation by Multiple Mobile Robots*", Dissertation of the Master of Science Program on Electrical-Electronics Engineering, No: 0078929 (TEZ 3901 2010 k.1), YÖK Ref No: 372574, Institute of Natural Sciences, Eskişehir Osmangazi University, Eskisehir, Turkey.
3. (2008) **Yufka A.** "*Indoor Localization System for Mobile Robots*", Synthesis and Design Project of the Bachelor of Science Program on Electrical-Electronics Engineering, Faculty of Engineering and Architecture, Eskişehir Osmangazi University, Eskisehir, Turkey.

Honors and Awards

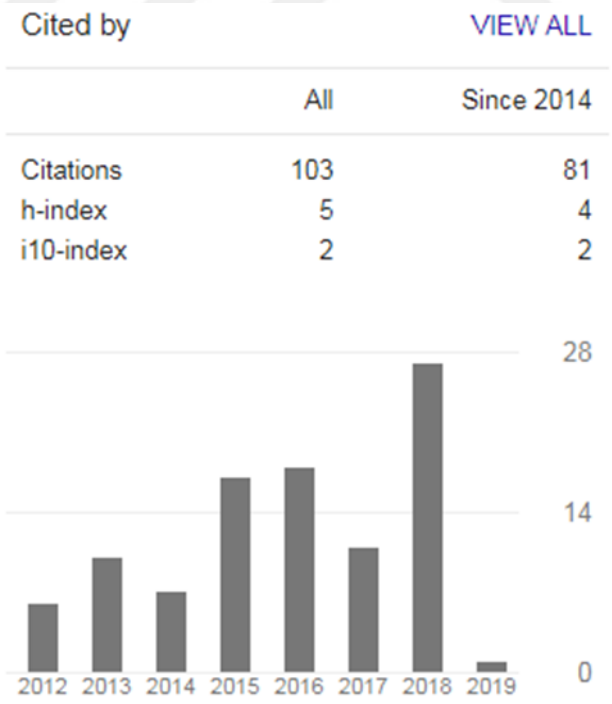
- 2017, 5-years service-award at Otokar Automotive and Defense Ind. Inc.
- 2010, Finalist for the: "IEEE International Conference on Systems, Man, and Cybernetics Best Student Paper Award"
- 2009, The Scientific and Technical Research Council of Turkey (TUBITAK) Research Scholarship for The Scientific Robotics Project
- 2008, High Honor Student at University

- 2007, European Commission – Education & Training Scholarship for The ERASMUS Programme
- 2007, High Honor Student at University
- 1999, Honor Student at High School
- 1997, Honor Student at Secondary School

Organizations

- 2017–2018, IEEE Power Electronics Society (ID:92439534)
- 2017–2018, IEEE Control Systems Society (ID:92439534)
- 2015–2018, IEEE Robotics and Automation Society (ID:92439534)
- 2015–2018, The Institute of Electrical and Electronics Engineers (IEEE) (ID:92439534)
- 2013–continue, OTOKAR Robotics Society (Founder and President)
- 2011–2012, International Council on Systems Engineering (INCOSE) (ID:32068)

Google Scholar Statistics



Last access: on the 04th April of 2019