The Modelling and Analysis of Command and Control Decision Processes using Extended Time Petri Nets

Fred David John Bowden
B.Sc. Hons. (Applied Mathematics)

Thesis submitted for the degree of

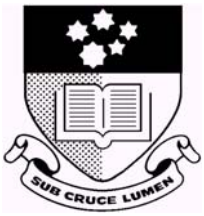
Doctor of Philosophy

at

The University of Adelaide

Department of Applied Mathematics

Faculty of Mathematical and Computer Sciences



June 2001

I dedicate this thesis to my wife Lesley.

Without whom this, along with many other things, would never have been possible. Thank you, for all your love, faith, patience, support and assistance.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

DECLARATION

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text of the thesis.

I give consent to this thesis being made available for photocopying and loan if accepted for the award of the degree.

Fred David John Bowden

ACKNOWLEDGEMENTS

I would like to thank my family for never doubting that I would complete this thesis. They always asked by “when” I would complete rather than “if”.

I would like to acknowledge the Defence Science and Technology Organisation for its support during this research. In particular I would like to thank John Coleby, Mike Davies, David Fogg, Paul Gaertner, Stephen Hood and Darryn Reid for their support at various stages.

I would like to thank my co-supervisors Jonathan Billington, Mike Davies and Peter Taylor. In particular I thank Peter and Mike for their comments on my final draft. I would also like to thank Lesley, Lars Kristensen and Darryn for reading and commenting on certain portions of my thesis at varying stages of its readiness.

Above all I would like to thank my principal supervisor Charles Pearce. He was initially my teacher, then my mentor and is now my friend. No one could ask for more than Charles has given me and it has been my privilege to have him as my supervisor. I hope that in the future our lives continue to be entangled.

ABSTRACT

Effective command and control is crucial to both military and non-military environments. Accurate representations of the processes associated with the inter and intra activities of nodes or agencies of such systems is essential in the analysis of command and control. One of the most important things is to be able to model the decision processes. These are the parts of the system that make decisions and then guide the direction of other elements in the system overall.

This thesis uses a new type of extended time Petri net to model and analyse command and control decision processes. A comprehensive review of existing time Petri net structures is given. This concludes with the introduction of a time Petri net structure that incorporates the most commonly used time structures. This extended time Petri net structure is then used in the definition of the basic modelling blocks required to model command and control decision processes. This basic modelling block forms the basis of the direct analysis techniques that are introduced in the thesis.

Due to the transient nature of the systems being modelled and the measures of interest a new type of measure is introduced, the mean conditional first hitting reward. This measure does not currently appear to be part of the stochastic process literature. Explicit procedures are given to determine the hitting probabilities and mean conditional first hitting reward for decision process models and discrete, continuous and semi-Markov chains. Finally the some extensions of the decision process sub-class are considered.

GLOSSARY OF SYMBOLS

κ	100	Sub-set of states of interest in reaching the first time
$\mathbf{\Lambda}$	106	Diagonal matrix containing elements a_i
$\Omega(\tau)$	127	Probability no transition fires before transition t
τ	25	Time
τ_{c_i}	135	Mean time spent in circuit C_i
τ_{s_k}	135	Mean holding time of absorbing transition R_k
ρ	26	Range of possible duration functions
$\boldsymbol{\mu}$’	123	Vector of mean times spent in i before going to j
$\mu_{i\,j}$	121	Mean time spent in i before going to j for a semi-Markov chain
A	89	Absorbing part of a decision process building block
a_i	100	Probability of reaching κ from state i
A_j	88	The j^{th} absorbing state of a decision process building block
^{A}M	26	Available tokens marking in a time Petri net
B	100	Finite set of states with access to κ
C	20	Incidence matrix
C	88	Circuit part of a decision process building block
c_i	88	First transition in the i^{th} circuit of a decision process building block

D	27	Duration function
${}^{E}D(t)(\tau)$	45	Enabling duration function of transition t
$E_{p\,t}(\tau)$	126	Enabling duration function for the arc from place p to transition t
${}^{E}M$	32	Enabling marking in a time Petri net
$\lvert {}^{E}M(t) \rvert$	36	Enabling degree of transition t
F	20	Firing sequence count vector
${}^{H}D(t)(\tau)$	46	Holding duration function of transition t
$H_{t\,p}(\tau)$	126	Holding duration function for the arc from transition t to place p
I	88	Initially marked place of a decision process building block
$I_i(\tau)$	127	Probability that transition C_i fires before time τ
$J_j(\tau)$	127	Probability that transition S_j fires before time τ
L	88	Number of absorbing states in a decision process building block
M	8	A marking
M0	7	Initial marking
$[M\rangle$	13	Reachability set for the marking M
$M[t\rangle M'$	13	Marking M' is the marking reached from M when transition t fires
$M[S\rangle M'$	13	M' is the marking reach from marking M with firing sequence σ
N	88	Number of circuits in the a decision process building

		block
P	7	Set of places
p	100	Transition probability matrix
p'	106	Transition probability matrix containing only states in B
$p \succ p'$	162	Place p is upstream to place p'
$p_{i\,j}$	99	Single step transition probability from state i to state j
${}^{P}M$	32	Token marking in a Petri net with enabling durations
q'	117	Vector with q_i as its elements
Q	112	Transition rate matrix
$q_{i\,j}$	112	Transition rate from state i to state j
q_i	112	$-\sum_{i \neq j} q_{i\,j}$
R_0^+	56	Nonnegative reals
R_i	103	Mean conditional first-passage reward from state i to κ
$R_{i\,j}$	103	Mean reward gained as a result of a transition form state i to state j
s_j	88	The j^{th} transition to an absorbing state of a decision process building block
T	7	Set of transitions
T	108	Mean conditional first-passage time matrix
$t^{\bullet}$	10	Set of output place for transition t
${}^{\bullet}t$	10	Set of input place for transition t
T_i	102	Mean conditional first-passage time from state i to κ
U	103	Dual matrix
$U_{i\,j}$	103	Dual transition probabilities

^{U}M	26	Unavailable tokens marking in a time Petri net
$\mathbf{v}$	105	Mean conditional next step passage reward vector
v_i	104	Mean conditional next step passage reward from state i
W	8	Weighted flow function
$\lfloor x \rfloor$	37	Greatest integer less than or equal to x
Z^+	7	Positive integers
Z_0^+	7	Nonnegative integers

CHAPTER 1 INTRODUCTION

Effective command and control is crucial to both military and non-military environments. In the analysis of command and control systems it is important to develop accurate representations of the processes associated with the inter and intra activities of nodes or agencies of that system. One of the most important things is to be able to model the decision processes. These are the parts of the system that make decisions and then guide the direction of other elements in the system overall.

In the modelling of any system a certain amount of abstraction is required, often necessitating stochastic ingredients. This is particularly true in the case of modelling human decision processes, which are usually too complex to represent completely. Also, realistic modelling of decision processes involves uncertainty in the way a model reacts to a given situation. Thus for an initial state there could be a number of possible final decisions. So the technique chosen to construct such a model must allow for stochastic behaviour in reaching the final state. Other desirable features of the chosen modelling technique include the ability to represent both concurrent and sequential activities, allow for modelling of resource sharing and accommodate both timed and non-timed events. Petri nets (PN) are able to represent all of the desirable features described above, and hence have been chosen as a tool to model decision processes in command and control systems. The use of PNs to model decision processes is not new and has been reported in many papers such as [28], [29], [31]-[34], [68], [96], [98]-[102], [108], [111], [133]-[135], [144], [157], [160], [174] and [179]-[181].

Another convenient feature of PNs is that models can be built in a modular fashion. This is particularly important in designing larger models and if parts or all of the model are to be

reused. The role-based design and implementation method ([28], [29], [33] and [34]) proposes a way of modelling the decision process of a command and control system such that part of complete models can be reused. Particular parts of a model can be presented to whatever level of detail required. PNs are seen as a way of representing role based models ([33]). This modelling method has been used in the design of decision processes as part of the distributed interactive C3I[1] effectiveness simulation environment, as described in [31], [32] and [68].

One of the key issues pursued in [28] and [29] is validating the decision process models. These papers describe the development of an explanation capability that allows a person unfamiliar with PNs to investigate how a PN model of a decision process works. Thus it allows a person who has expertise in a given decision process to comment on the way the model reacts to given situations. This capability is currently very fundamental in that it gives basic information such as possible reachable states, the path taken to reach a given state and initial states required for a given state to be reached.

The aim of the explanation capability described in [28] and [29] is to aid in the development of decision process models to be included as part of larger distributed simulations. However, this tool can be extended to carry out analysis in its own right. There are two measures that are very important not only in validating a model's suitability to be part of a larger system but also in analysing a decision process in general: the probability and the mean conditional time of reaching a given state for the first time. Further there is the issue of the mean conditional reward or cost of reaching a given state for the first time. This would be an invaluable measure in determining the cost of possible courses of action.

[1] Command, Control, Communications and Intelligence.

As described above, PNs are a technique for representing the decision processes of a command and control system. This thesis focuses on developing a way of representing decision processes in order to determine the probability of what will be defined as the mean conditional first-hitting reward. It begins by investigating the different ways of modelling time in PNs. Next it presents a modular approach, involving what is referred to as decision process building blocks, to build and analyse decision processes. Some of the developments arising in this thesis are:

- a new type of time PN representation, including the representation of rewards;
- the definition of a decision process building block;
- the definition of a new measure of a system, the mean conditional first-hitting reward;
- the theory behind calculating mean conditional first-hitting rewards in Markov chains;
- the determination of the probability of hitting a given state in a decision process; and
- the determination of the mean conditional first-hitting reward in a decision process.

PART I

PETRI NETS

PNs were originally developed in 1962 by Carl Adam Petri in his doctoral thesis titled "Communication with Automata" [140]. Since their development, PNs have been used to model many different types of discrete event systems, in fields including computer science (as in [137]), manufacturing systems ([12], [63], [84] and [165]), transport systems ([2]), project management ([93]), and more recently command and control ([11], [22], [28], [33], [34], [95], [98]-[102], [122], [133], [148] and [174]).

This Part of the thesis introduces the basic PN concepts that will be used. It concentrates on how time is represented in PNs and amalgamates the current time representations into a single representation to allow for the benefits of the alternative representations. The structure of this Part is as follows:

- Chapter 2 introduces PNs and gives a number of fundamental properties and analysis techniques.
- Chapter 3 defines and compares the different methods of representing time in PNs.
- Chapter 4 overviews the three ways of generating time delays.
- Chapter 5 discusses some of the subtle particulars of execution policies.
- In Chapter 6, the super-time Petri net class is introduced and used to represent all the previously defined time Petri nets.

- Chapter 7 extends the super-time Petri net class further by introducing the more general idea of rewards.
- Finally in Chapter 8 some concluding comments are presented.

CHAPTER 2 PETRI NETS

In this chapter the basic concepts of PNs will be introduced. It gives the PN definition that will be built on when time is introduced and defines many of the terms and notation used throughout this thesis. It also gives a brief outline of the main analysis techniques for PNs.

2.1 Basic Concepts and Definitions

A PN can be represented by a bipartite directed graph with two types of nodes: *places* and *transitions*. Pictorially, places are drawn as circles and represent features such as conditions, buffers, servers, resources and queues. Transitions are displayed as rectangles and represent activities in the system being modelled such as processors, algorithms and events. Directed arcs can only appear between different types of nodes. That is, a directed arc can only go from a place to a transition, or a transition to a place. As a representational abbreviation, when more than one arc is going in the same direction between two nodes, only one arc is drawn with a weight representing the actual number of arcs placed next to it. Figure 2.1 shows a PN with 5 places, 4 transitions and 12 arcs.

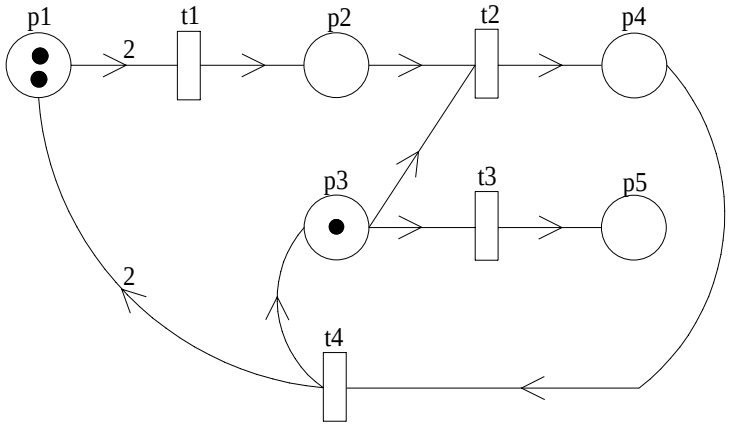


Figure 2.1: Example of a Petri net

Tokens make up the final element in a PN. Tokens are associated with places and are represented graphically by identical dots. The movement of tokens between places is controlled by the transitions of the PN. In a PN, the distribution of tokens defines the state of the system such as conditions satisfied, items in a buffer, number of free servers, availability of resources and entities in a queue. The distribution of tokens through the PN is called the PN's *marking*. The PN in Figure 2.1 has the marking of two tokens in place p1 and one in place p3.

The standard mathematical way of defining a PN is using a tuple. There are two main tuple representations. The one introduced by Reisig ([147]) is called a place transition system and is the 5-tuple

$PN = (P, T, F, W, M0)$[2]

where: $P = \{p1, p2, ..., pn\}$ is the set of places;

$T = \{t1, t2, ..., tm\}$ is the set of transitions;

$P \cap T = \varnothing$;

F is the flow relationship, which defines the set of arc relations between T and P, such that, $F \subseteq \{T \times P\} \cup \{P \times T\}$;

W is the weight function, which is a map: $F \rightarrow Z^+$, where Z^+ is the set of positive integers, giving the weights of the arcs; and

M0 is the initial marking, which is a mapping: $P \rightarrow Z_0^+$, where $Z_0^+ = Z^+ \cup \{0\}$, giving the number of tokens initial in each place.[3]

[2] Normally this definition includes a sixth element called the capacity. However, this is not relevant here and so has not been included. See [147] for more information on this.

An alternative representation is given by Peterson ([137] and [138]), who calls these PNs marked PNs and uses 5-tuple

PN = (P, T, I, O, M0),

where: P, T and M0 are as given in the earlier definition;

I is a mapping: $P \times T \rightarrow Z_0^+$ and defines the weights of the input arcs such that, if the input arc from p to t has a weight of *l*, then I(p, t) = *l*; and

O is a mapping: $T \times P \rightarrow Z_0^+$ and defines the weights of the output arc such that, if the output arc from t to p has a weight of *l*, then O(t, p) = *l*.

These two definitions have been introduced since it is a combination of these that will be used in this thesis. Here a PN will be defined by the 4-tuple

PN = (P, T, W, M0).

In this definition P, T and M0 are as defined earlier and W is the weighted flow function. The weighted flow function is a mapping: $\{T \times P\} \cup \{P \times T\} \rightarrow Z_0^+$. Thus, this is simply the same definition as Reisig's 5-tuple (P, T, F, W, M0) with W being defined directly from the places and transitions, allowing the explicit definition of F to be left out. It can also be considered to be a merging of the input and output arc functions, I and O, of Peterson's 5-tuple (P, T, I, O, M0) into the single function W.

Consider once more the PN in Figure 2.1. This PN has the tuple representation:

P = {p1, p2, p3, p4, p5};

T = {t1, t2, t3, t4};

[3] In this thesis general markings will be denoted by M.

W(p1, t1) = 2, W(p2, t2) = 1, W(p3, t2) = 1, W(p3, t3) = 1, W(p4, t4) = 1,

W(t1, p2) = 1, W(t2, p4) = 1, W(t3, p5) = 1, W(t4, p1) = 2, W(t4, p3) = 1; and

M0(p1) = 2, M0(p3) = 1.

For brevity, weighted flow definitions that are zero and unmarked places are not given here. This is often done in PNs.

There are a number of basic definitions that will be used during this thesis that will now be introduced.

Definition 2.1: Input arc

For $p \in P$ and $t \in T$ an *input arc*, (p, t), exists from place p to transition t if and only if $W(p, t) \neq 0$.

That is, the input arcs of a transition are all the directed arcs going from some place to the transition. In Figure 2.1 transition t2 has one input arc from both places p2 and p3. In a similar fashion the input arcs of a place can be defined, as in [171]. In this thesis an input arc will always refer to a transition input arc.

Definition 2.2: Output arc

For $p \in P$ and $t \in T$ an *output arc*, (t, p), exists from transition t to place p if and only if $W(p, t) \neq 0$.

That is, the output arcs of a transition are all the directed arcs going from the transition to any place. In Figure 2.1 transition t4 has two output arcs, one to place p1 and one to place p3. The

transition t4 is the only transition in this net that has more than one output arc. In a similar fashion the output arcs of a place can be defined as in [171]. In this thesis an output arc will always refer to a transition output arc.

Definition 2.3: Input place

The set of *input places* for $t \in T$, denoted by ${}^\bullet t \subseteq P$, is defined by ${}^\bullet t = \{p: p \in P \text{ and } W(p, t) \neq 0\}$.

That is, the input places of a transition are all those places with input arcs to that transition. In Figure 2.1, ${}^\bullet t1 = \{p1\}$, ${}^\bullet t2 = \{p2, p3\}$, ${}^\bullet t3 = \{p3\}$ and ${}^\bullet t4 = \{p4\}$. In a similar fashion the input transitions of a place, denoted by ${}^\bullet p$, are defined as all the transitions with place input arcs to the place, for example see [172].

Definition 2.4: Output place

The set of *output places* for $t \in T$, denoted by $t^\bullet \subseteq P$, is defined by $t^\bullet = \{p: p \in P \text{ and } W(t, p) > 0\}$.

That is, the output places of a transition are all those places connected by output arcs from that transition. In Figure 2.1 $t1^\bullet = \{p2\}$, $t2^\bullet = \{p4\}$, $t3^\bullet = \{p5\}$ and $t4^\bullet = \{p1, p3\}$. Once more, output transitions of a place, denoted by $p^\bullet$, can be defined in an analogous way. It should be noted that Definition 2.3 and Definition 2.4 hold for sets of transitions and places. For example in Figure 2.1 the set of transitions $T1 = \{t1, t2\}$ have ${}^\bullet T1 = \{p1, p2, p3\}$ and $T1^\bullet = \{p2, p4\}$.

The semantics of a PN are defined in two stages. First, if a transition is enabled, then it may fire.

Definition 2.5: Enabling of a transition

The transition $t \in T$ is enabled in marking M if and only if for all $p \in P$, $M(p) \geq W(p, t)$.

The PN in Figure 2.1 with the shown marking has the enabled transitions t1 and t3. Thus the multi-set of enabled transitions is {t1, t3}. The term multi-set has been used to illustrate the possibility of multiple enablings of the same transition. For example if there were two tokens in place p3 of Figure 2.1 then transition t3 would be enabled twice making the enabling multi-set {t1, t3, t3}.

Definition 2.6: Firing of a transition

When the enabled transition, t, *fires* (occurs) in the marking M, a new marking M' is obtained according to the firing rule

for all $p \in P$, $M'(p) = M(p) + W(t, p) - W(p, t)$.

Thus when a transition fires it removes tokens from the transition's input places and creates tokens in its output places. For example, if transition t1 in Figure 2.1 fires, then two tokens are removed from place p1 and one is created in place p2. Alternatively if transition t3 fires then one token is removed from place p3 and one token is created in place p5.

Definition 2.7: Firing sequence

Any possible sequence of transition firings is called a *firing sequence*.

A PN may have a number of different possible firing sequences for a given initial marking. Multiple firing sequences occur if more than one transition is enabled at the same time during the firing sequence. Consider the PN shown in Figure 2.1. Some possible firing sequences for this net with the shown initial marking are {t1, t2, t4, t1, t2, t4, t3, t1}, {t3, t1}, and {t1, t3}. In Murata [125] the firing sequence contains not only the list of transitions that fire but also the marking reached. This is also referred to as the execution sequence in [7].

Definition 2.8: Execution Sequence

An *execution sequence* is a firing sequence which includes not only the transitions that fire but also the initial, intermediate and final markings.

Consider the firing sequence {t3, t1} for the PN shown in Figure 2.1. This firing sequence has the three markings M0 = (2, 0, 1, 0, 0)[4], M1 = (2, 0, 0, 0, 1) and M2 = (0, 1, 0, 0, 1). Thus the execution sequence would be {M0, t3, M1, t1, M2}.

[4] In this representation, the markings are represented by ordered pentuples, where the first value specifies the number of tokens in place p1, the second the number in p2 and so on. Thus if there are two tokens in p1, one token in p3 and no tokens in the rest of the net, then (2, 0, 1, 0, 0) would be the specification of the marking.

Definition 2.9: Immediately Reachable

The marking M' is *immediately reachable* from M if there exists $t \in T$ enabled by M such that for all $p \in P$, $M'(p) = M(p) + W(t, p) - W(p, t)$. This is denoted by $M[t\rangle M'$.

In the PN of Figure 2.1, the marking with one token in place p2 and one token in place p3 is, through the firing of transition t1, immediately reachable from the marking shown.

Definition 2.10: Reachable

The marking M' is *reachable* from M if there exists a firing sequence S = {ti, tj, tk, …,tn} such that for all $p \in P$,

$$M'(p) = M(p) + W(ti, p) - W(p, ti) + W(tj, p) - W(p, tj) + W(tk, p) - W(p, tk) + W(tn, p) - W(p, tn).$$

This is denoted by $M[S\rangle M'$.

The marking of one token in place p4 is reachable from the illustrated marking when the firing sequence {t1, t2} occurs. It is possible that there may be a variety of ways to reach the same marking. For example, an alternative firing sequence which gives the marking of one token in place p4 from the marking shown in Figure 2.1 is {t1, t2, t4, t1, t2}.

Definition 2.11: Reachability Set

The set of markings reachable from a given initial marking are defined as the PN's *reachability set* for the given initial marking. The reachability set for the marking M is denoted $[M\rangle$. Thus M' is reachable from M if and only if $M' \in [M\rangle$.

The PN in Figure 2.1 with the shown initial marking has the reachability set

(2, 0, 1, 0 ,0), (0, 1, 1, 0, 0), (2, 0, 0, 0, 1), (0, 0, 0, 1, 0), (0, 1, 0, 0, 1).

Definition 2.12: Deadlock

A PN is said to have reached a *deadlock* when its execution halts.

This definition is not the usual one given but is more general and holds for PNs that include time. For the semantics defined above, deadlock occurs when the current marking does not enable any of the PN's transitions. This is the normal definition of deadlock. The more general definition was given here as in some PNs with time there is the possibility of having no enabled transitions and yet still not have reached a deadlock. This will become clear later in this Part.

The PN shown in Figure 2.1 reaches deadlock when the marking of one token in p2 and one token in p5 is reached. Markings that result in deadlock will be referred to as *absorbing markings* or *absorbing states*. A knowledge of the absorbing states is very important in system design as they define the states for which the system stops. Absorbing states may be desirable in some models as in the case of a decision maker, indicating a decision has been reached, or undesirable as in the case of a manufacturing plant, indicating production has stopped.

Definition 2.13: Loop

A loop exists between $p \in P$ and $t \in T$ if $p \in {}^\bullet t \cap t^\bullet$.

Thus, a loop exists in a PN if a place is both an input and output place to the same transition. The PN in Figure 2.1 has no loops.

Definition 2.14: Structural Conflict

The transitions t1, t2, …, tn are in *structural conflict* if $^\bullet t1 \cap {}^\bullet t2 \cap \ldots \cap {}^\bullet tn \neq \varnothing$, that is, if they share an input place.

Figure 2.1 has a structural conflict between t2 and t3 as these transitions share the input place p3. In [5] and other papers in the literature structural conflict is defined as existing between transitions tm and tl if

for all $p \in {}^\bullet tm \cap {}^\bullet tl$, $W(p, tm) \cap \{W(p, tl) - W(tl, p)\} \neq \varnothing$.

This varies from Definition 2.14 as it does not consider the case when loops exist. The definition given here, is used as loops are seen as an important part of the decision process building block that will be defined later.

Definition 2.15: Conflict

Consider the case when a given marking enables more than one transition. If the firing of an enabled transition disables some of the other enabled transitions, then the transitions which become disabled are said to be in *conflict* with the transition that fired. The set of transitions which are in conflict for a given marking are referred to as the marking's *conflict set* [47].

Conflict occurs in marking M if the removing of the input tokens of a transition enabled by M results in another enabled transition becoming disabled. For example, if the marking of one

token in place p2 and one in place p3 occurs in the PN shown in Figure 2.1, then transitions t2 and t3 are in conflict. Once more this varies from the usual definition of conflict which again does not consider conflict to exist in the case of loops. Due to the introduction of time later in this Part of the thesis it is important that this case is not ignored.

Definition 2.16: Concurrent enablings

If more than one transition is enabled by the same marking and the firing of any of these transitions does not disable any of the others, then the transitions are referred to as being *concurrently enabled*.

This definition is the same as that given in [8]. With the marking shown in Figure 2.1 the transitions t1 and t3 are concurrently enabled.

Definition 2.17: Confusion

Confusion exists when a marking has two (or more) concurrently enabled transitions but the firing of one before the other(s) means the still enabled transition(s) is (are) in conflict with a newly enabled transition. Thus confusion is the result of possible conflict.

Confusion is an aspect of PNs that is often overlooked although it is not a new concept, being introduced in papers such as [162]. The PN in Figure 2.1, with the marking shown, illustrates confusion as transitions t1 and t3 are concurrently enabled but the firing of t1 results in a conflict between t3 and the newly enabled transition t2. Although confusion does not directly influence the properties of a system it is important in the consideration of the PN semantics. If confusion exists in a PN then the order in which concurrently enabled transitions fire is

important. Confusion is considered in greater detail later when time is added to the PN structure.

Definition 2.18: Place invariant

The set of weights yi form the *place invariants* (S or P-invariant) for a PN if for all $M \in [M0\rangle$, $\sum_{pi \in P} yi\ M(pi) = c$.

Thus place invariants relate to the conservation of the number of tokens in the net for all reachable markings.

The PN in Figure 2.1 has the place invariant where place p1 has a weight of ½, places p2, p3 and p5 each a weight of 1, and place p4 a of weight 2. This gives for all $M \in [M0\rangle$ a constant value of 2.

Definition 2.19: Transition invariants

A *transition invariant* (T-invariant) is a firing sequence that returns the PN to its starting marking. That is, the firing sequence S is a T-invariant if $M[S\rangle M$.

Figure 2.1 has the T-invariant S = {t1, t2, t4}.

It should be noted that the definition given above is the same as that given in [2] but is not the definition usually used. This is discussed further in Section 2.2.2, where the more common definition of T-invariants will be given.

Definition 2.20: Circuit

A *circuit* is an execution sequence that returns the PN to its starting marking.

Thus in the definitions given here a circuit and a T-invariant are similar except a circuit includes the markings as well as the firing transitions.

Definition 2.21: Simple Circuit

A circuit where each transition and marking appears only once is called a *simple circuit*.

2.2 Analysis

There are two main forms of analysing a PN: reachability and matrix. This Section gives a brief introduction to these two.

2.2.1 Reachability Analysis

The primary way of analysing PNs is using their state space. This involves the generation of the complete reachability set and then using it to check for properties. The reachability set is represented by the *reachability graph*, which contains not only information about the reachable markings but also the firing sequences required to move between them.

The first entry in the reachability graph is the initial marking. Below this marking, each of the possible immediately reachable markings is listed. Directed arcs going from the initial marking to each of the immediately reachable markings are drawn and labelled with the

transition required to fire to reach the specified marking. This process is then repeated for each new marking. If a new immediately reachable marking is identical to one which appears earlier in the graph, then the generating marking is connected to the earlier marking by an arc labelled with the appropriate transition. If a PN is confusion free then the reachability graph can be reduced by firing all the concurrently enabled transitions at once, making the next node in the reachability graph that reached by firing of the set of concurrently enabled transitions. The reachability graph for the PN in Figure 2.1 is given in Figure 2.2.

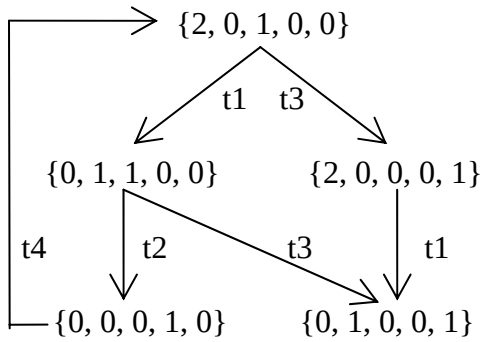


Figure 2.2: Reachability graph for the PN in Figure 2.1

A problem with this method is that for many practical systems the PN state space, and thus the reachability graph, is very large. The larger the reachability graph, the harder it is to identify properties. PNs may also have infinitely many states in which case the reachability graph would also have to be infinite. In [138] an algorithm is defined that will always generate a finite graph, but this does not get around the problem of dealing with very large state spaces. Much work has been done to help over come this problem, see references such as [46], [86], [132], [168] and [169].

2.2.2 Matrix Analysis

It is easy to see that the tuple representation can be placed in a matrix form. A PN with m transitions and n places is represented by the m×n matrix **C** called the *incidence matrix* ([58]). The elements of incidence matrix is defined by

$C_{j\,i} = W(tj, pi) - W(pi, tj)$.

The incidence matrix for the PN in Figure 2.1 is

$$\mathbf{C} = \begin{bmatrix} -2 & 0 & 0 & 2 \\ 1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The PN marking can be represented by a vector of size n called the *marking vector*, where $M_i = M(pi)$. Hence if there are *l* tokens in the i^{th} place then $M_i = l$. This means that the marking vector for the initial marking shown in Figure 2.1 is

M0 = [2 0 1 0 0].

The firing sequence of a PN can also be represented by a m-row vector, **F**, called the *firing vector*. In this vector, if the j^{th} transition fires *k* times in a given firing sequence, then $F_j = k$. Thus the firing sequence {t1, t2, t4, t1, t2} is represented by the vector

$\mathbf{F} = [2\ 2\ 0\ 1]^t$.

Here the “t” super-script stands for transpose.

If the firing sequence is known, the resulting marking can be calculated using

M = **M0** + **C** **F**.

In the PN shown in Figure 2.1, a firing sequence is {t1, t2, t4, t1, t2, t4, t3, t1} which gives the firing vector as $[3\ 2\ 1\ 2]^t$. The resulting marking would be given by

$$\mathbf{M}1 = \begin{bmatrix} 2 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -2 & 0 & 0 & 2 \\ 1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

The matrix representation can be used to detect properties such as reachability, P-invariants and T-invariants. In matrix analysis, P-invariants are defined to be given by the nonnegative and nonzero solutions to $\mathbf{W}\,\mathbf{C} = \mathbf{0}$ ([58]) and T-invariants by the nonnegative and nonzero solutions to $\mathbf{C}\,\mathbf{F} = \mathbf{0}$ ([58]). This is where the difference in the definitions of T-invariants arises. In the earlier definition, it was stated that a T-invariant related to a firing sequence that returned the PN to its initial marking. However, a nonnegative solution to $\mathbf{C}\,\mathbf{F} = \mathbf{0}$ is not a firing sequence but rather a vector containing a count of the number of firings of each transition in a firing sequence that returns the PN to its initial marking. Such a vector may not relate to a realisable firing sequence. This is a problem with the matrix method in general and comes from the fact that solutions to matrix equations are not guaranteed to be realisable in the PN.

There are two distinct problems with the matrix method. The first is the fact that the incidence matrix holds only net change in the number of tokens in a place, not the overall input and output requirements. So when a loop is included as part of the model, information is lost in the calculation of the elements of **C**. Another problem is that the firing vector does not represent the order in which transitions fire, just the number of firings of each. Thus all solutions given by this technique must been checked in terms of the semantics of the PN to

determine if they are an achievable firing sequence. For large models with long firing sequences this can be difficult.

CHAPTER 3 INTRODUCING TIME INTO PETRI NETS

There are three different ways of representing time durations in PNs: firing durations, holding durations and enabling durations. Since firing durations ([143]) are just a sub-class of holding durations (see [37] and Appendix I), only holding and enabling durations need to be considered[5]. The actual name used in the literature for each time representation varies with researchers. In general, timed Petri nets [155] are associated with holding durations and time Petri nets [118] and stochastic Petri nets ([7]) with enabling durations. A more comprehensive list of the different names used on PNs extended to include time is given in [176]. From this point on all PNs including time will be referred to as timed Petri nets (TPN).

An interesting aside is that there is some disagreement in the literature on the temporal interpretation of PNs without time. Papers [109], [110] and [175] consider transitions in non-timed PNs to fire the instant they are enabled. That is, each transition takes zero time to fire. Others, [64], [66], [117], [156], [163] and [164], consider PN transitions to take a time anywhere between zero and infinity to fire. Thus even on this simple matter there are conflicting views. This is further highlighted by the fact that there is not even a standard part of the PN to which time is assigned. In fact, time has been assigned to transitions, places and arcs. In all cases, the PN representation and semantics can be defined in a similar way. Throughout this chapter, the concepts of holding and enabling durations are explained using deterministic durations with global absolute time. In using global absolute time the clock is implicit and the ordering of events is such that time is non-decreasing. It is the shortest time that determines which of the execution step occurs next. Alternative duration functions are

[5] For completeness Appendix I discusses firing durations.

discussed in Chapter 4. Relative time, the alternative to absolute global time, will be considered in the last section of this chapter.

It should be noted that some of the definitions given in Chapter 2 now need to be reconsidered in light of the inclusion of time. Rather than dealing with this on an individual basis for each TPN representation, the changes to these definitions will be discussed further as they are relevant to the topics being covered in this chapter. The relevant definitions will also be redefined for TPNs in general in Chapter 6.

3.1 Holding Durations

Holding durations work by classifying tokens into two types, available and unavailable (also referred to as reserved and unreserved (non-reserved), see [61] and [88]). Available tokens can be used to enable a transition whereas unavailable tokens cannot. When an enabled transition fires, it removes and creates tokens in the same time instant. However, it can only remove available tokens and the tokens created are defined as unavailable. Unavailable tokens remain so for a specified time period, the holding time, before becoming available tokens. PNs with holding durations will be referred to as holding time Petri nets (HTPN).

The holding time of a token can be determined by the transition which created the token (as in [1], [2] and [131]), the output place in which the token was created (see [14], [110], [155] and [175]), or the output arc from the creating transition to the output place (as in [37] and [126]). The use of holding durations is increasing in popularity, particularly in high-level nets where the time an unavailable token becomes available can be stored as part of the token structure (as in [2], [74], [85], [131] and [177]). No definition for high-level nets is given here, because a definition would take considerable time and such nets are peripheral to this thesis.

Figure 3.1 illustrates the execution of a HTPN with holding durations associated with transitions. In this figure, filled tokens represent available tokens while unfilled ones are unavailable. Thus the tokens in places p1 and p2 of Figure 3.1(a) are both available so transition t1 is enabled at time $\tau = 0$[6]. On firing, this transition removes an available token from each of its input places, p1 and p2, and creates an unavailable token in p3 (see Figure 3.1(b)). Since the created token is unavailable it cannot be used to enable transition t2. This means that transition t2 is not enabled at this time and cannot be enabled until the unavailable token in p3 becomes available. In this example holding durations have been assigned to the transitions and so created tokens remain unavailable until the holding duration of their "creator" transition has passed. So the token in place p3 becomes available at $\tau = 1$, as shown in Figure 3.1(c), enabling transition t2. The HTPN goes through the same process with t2, as shown in Figure 3.1(d) and (e). The final state has now been reached as the HTPN is in deadlock.

Note that Figure 3.1(b) and (d) show the reason why deadlock was defined in the manner given in the previous chapter. In both these cases there are no enabled transitions, however, the execution of the PN has not ended. In particular, consider the case of the marking in Figure 3.1(b). This marking goes on to enable a further transition.

[6] Due to the use of t as a transition throughout this thesis time will be referred to by τ.

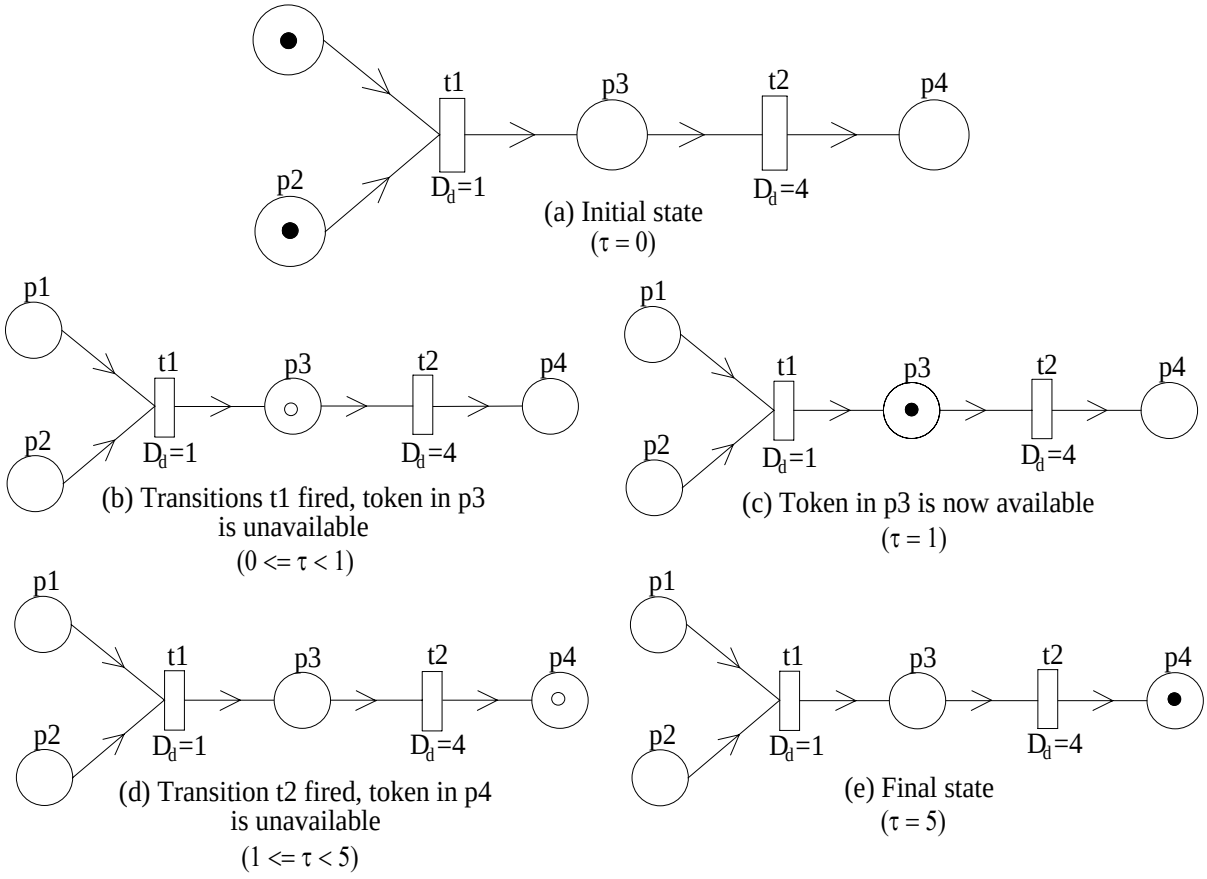


Figure 3.1: TPN with holding durations

Since there are two classes of tokens, there is a need to redefine the marking function. The marking is now made up of two parts, one defining the available tokens, the other the unavailable tokens. The available tokens function, ${}^{A}M$, is a mapping: $P \rightarrow Z_0^+$, that maps each place into the number of available tokens in that place. This takes the same form as the marking function of a PN. The unavailable tokens function ${}^{U}M$ is a mapping: $P \rightarrow \{(n, x): n \in Z_0^+ \text{ and } x \in \rho\}$, where ρ is the range of the possible timing functions. In the ordered pairs the first element is the number of unavailable tokens and the second the time that these tokens become available. For example in Figure 3.1(d) ${}^{U}M(p4) = \{(1, 5)\}$.

As with a PN, a HTPN can be represented by a multi-tuple. In this case the pentuple:

HTPN = (P, T, W, D, M0),

where P, T and W are as given in the PN definition of Chapter 2;

for each $t \in T$, $D(t)(\tau)$[7] is the distribution function of the durations associated with transition t. Further, deterministic duration functions which return the value of k with probability one and zero for other values will be written as $D_d(t) = k$ where

$$D_d(t)(\tau) = \begin{cases} 0 & \tau < k \\ 1 & \tau \geq k \end{cases}.$$

As with PNs, M0 is the initial marking of the HTPN. In this case M0 is the union of ${}^{A}M0$ and ${}^{U}M0$, where for place $p \in P$, ${}^{A}M0(p)$ is the initial number of available tokens in place p and ${}^{U}M0(p)$ that of unavailable tokens in p.

The TPN in Figure 3.1 has the pentuple definition:

P = {p1, p2, p3, p4},

T = {t1, t2},

W(p1, t1) = 1, W(p2, t1) = 1, W(t1, p3) = 1, W(p3, t2) = 1, W(t2, p4) = 1,

$D_d(t1) = 1$, $D_d(t2) = 4$

${}^{A}M0(p1) = 1$, ${}^{A}M0(p2) = 1$.

As before weighted flows of zero and the marking of places with no tokens are not included in the definition.

[7] For brevity, the distribution function for a given transition is referred to as the transition's duration function, that is, $D(t)(\tau)$ will be referred to as D(t). Chapter 4 outlines the various forms that duration functions take.

The semantics of the HTPN can be defined using its pentuple definition. The enabling of a transition in a HTPN is very similar to that of the PN. The transition t is enabled if

for all $p \in {}^{\bullet}t$, ${}^{A}M(p) \geq W(p, t)$.

Thus a transition is enabled if sufficient available tokens are in its input places.

If transition t fires at time x then

for all $p \in {}^{\bullet}t$, ${}^{A}M'(p) = {}^{A}M(p) - W(p, t)$ and

for all $p \in t^{\bullet}$, ${}^{U}M'(t) = {}^{U}M(t) \cup \{(W(t, p), x+y)\}$, where y is the holding time of transition t and is generated by the duration function D(t).

In the above definition it is assumed that the holding durations have been assigned to transitions. If this were not the case it would be simply a matter of using the appropriate holding time function D. Thus in the case of holding duration assigned to places, the duration function would be given by D(p), and for output arcs by D(t, p). These alternative possibilities will be discussed in more detail shortly.

There is another step to the semantics of a HTPN, when unavailable tokens become available tokens. At time x+y for $(n, x+y) \in {}^{U}M'(p)$

${}^{A}M'(p) = {}^{A}M(p) + n$ and

${}^{U}M'(p) = {}^{U}M(p) \backslash \{(n, x+y)\}$.

This definition allows for the case of concurrent firings of the same transition. A concurrent firing would simply mean adding more than one unavailable token to the output places of the firing transition. In terms of the semantics this would mean adding multiple entries to ${}^{U}M$, one for each concurrent firing.

Until now the focus has been on transition holding durations. Some modellers prefer to associate holding durations with places or output arcs. This does not affect the semantics of a HTPN, rather it just redefines where the holding durations come from. That is, it changes the domain of D from T (the transitions) to P (the places) or T×P (the arcs).

When place holding durations are used, the time delays refer to the length of time that unavailable tokens created in a given place, by any transition, remain unavailable. Holding durations were first assigned to places in [154] and also appear in [14], [59], [110], [155], [171] and [175]. In [16] holding durations are assigned to both places and transitions.

One technique is not intrinsically better than the other. Rather they are two different ways of modelling and the one chosen largely depends on what is being modelled. The reason that time is more often associated with transitions is that transitions are generally seen to represent events or activities in a model while places are considered to be conditions. Thus it is natural to consider events or activities to take time rather than time to be related to conditions. However, there are some situations where this is not the case. It is stated in [15], [16] and [155] that associating time with places or with transitions has equivalent modelling power. It is up to the modeller to decide which technique will represent the system being modelled in a more natural fashion.

One way to add further flexibility to HTPNs is by assigning time to output arcs ([37] and [126]). In this case the holding durations taken by unavailable tokens are determined by the output arcs of the firing transition. This will be further discussed in Chapter 6.

A slight variation to the definition given here appears in [88]. In this paper, enabled transitions reserve their input tokens in the input places. The tokens are then defined as being unavailable for the required time. Once the time delay has passed the transition fires, removing the input tokens and creating available tokens in the output places. This approach runs into problems when a place is the input place to more than one transition, as it may not be clear which unavailable tokens in a given place were used to enable which transition. Thus, extra "bookkeeping" is required to keep track of the tokens.

One way of representing is to flag tokens with their holding times. This is sometime referred to as using time stamps. This approach allows for the merging of ${}^{A}M$ and ${}^{U}M$ into one function. The new marking function, ${}^{T}M$, is a mapping: $P \rightarrow \{x: x \in \rho\}$, where as stated earlier ρ is the range of the duration functions. That is, each token is now identified by the time it becomes available. Clearly if this approach is taken then the semantics must be changed. For completeness these semantics will now be given.

Define the set B(p, x) as the set of tokens in place p available at time x or less, that is,

$$B(p, x) = \{b: b \in {}^{T}M(p) \wedge (b \leq x)\}.$$

Define the set $C_n(p, x)$ as the subset of B(p, x) containing the n smallest time values in B(p, x). Since time is not explicitly represented, the notational abbreviations B(p) and $C_n(p)$ are used to represent B(p, x) and $C_n(p, x)$ at $\tau = x$.

The transition $t \in T$ is enabled at the present time if

$$\text{for all } p \in {}^{\bullet}t, \quad |B(p)| \geq W(p, t),$$

where $|A|$ is the cardinality of the set A.

If the transition $t \in T$ fires at time x with a holding time of y, given by the function D(t), then

for all $p \in {}^\bullet t$ and $b \in B(p)$, $M'(p) = M(p) \backslash C_{W(p, t)}(p)$ and

for all $p \in t^\bullet$, W(t, p) of the x+y elements are added to M(p) to form M'(p).

This notation will not be used in the remainder of this thesis. It has been presented for completeness as the author was unable to find a reference that gave a formal definition for holding times in high level PNs.

3.2 Enabling Durations

The alternative way of representing time in PNs is using enabling durations. This technique was first introduced by Merlin to model recoverable communication protocols ([118]). With enabling durations the firing of the transition is done immediately, that is, tokens are removed and created in the same instant and all tokens are the same. However, before a transition can fire it must be enabled for the associated enabling duration. That is, time delays are represented by forcing transitions to be enabled for a specified period of time before they can fire. TPNs with enabling durations will be referred to as enabling time PNs (ETPN) from now on.

Consider the PN shown in Figure 3.2(a), which is structurally the same as the example used in the previous section (Figure 3.1). As in the former case transition t1 is enabled at $\tau = 0$. However, now before the transition can fire it must remain enabled for t1's enabling duration, that is, 1 time unit. So transition t1 does not get to fire until $\tau = 1$, at which instant it removes the tokens from the input places p1 and p2 and creates an available token in its output place p3 (see Figure 3.2(b)). Now t2 goes through a similar process as t1 to give the final marking in Figure 3.2(c) at $\tau = 5$.

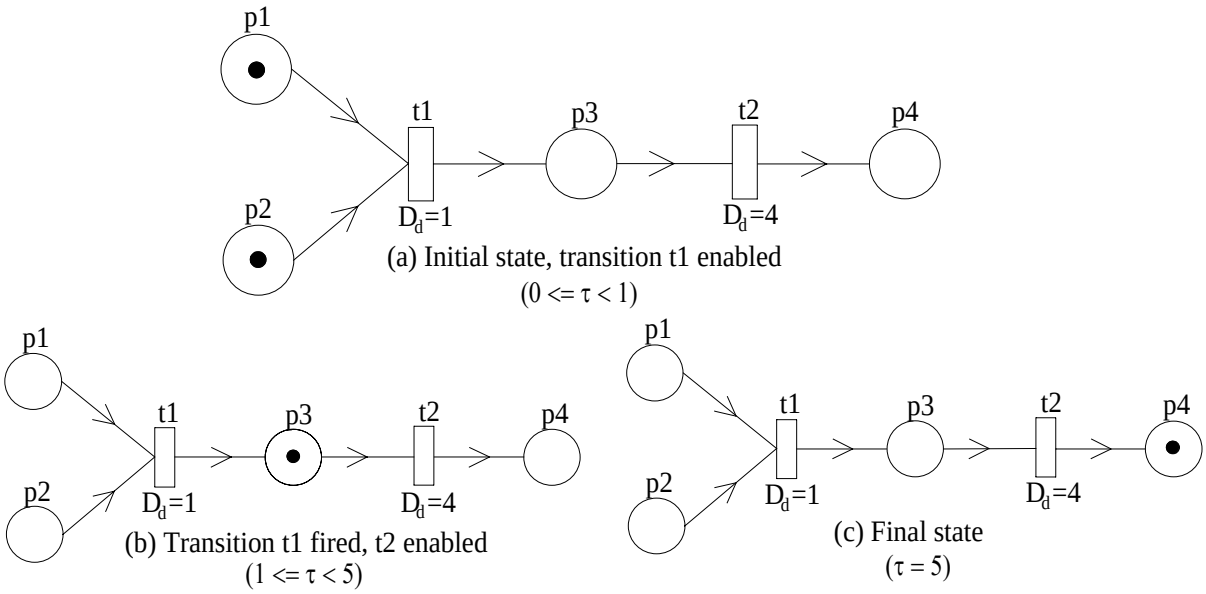


Figure 3.2: TPN with enabling durations

As in the case of HTPNs the formal definition for ETPNs can be expressed by the same 6-tuple as given in Section 3.1. In this case the marking function is made up of two parts, one relating to the places and one to the transitions. The first part, ${}^{P}M$, is a mapping: $P \rightarrow Z_0^+$ and relates to the number of tokens in each place, as in the marking function of PNs. The second part, ${}^{E}M$, is a mapping: $T \rightarrow \rho$ and maps the transitions into the remaining enabling time of the transition. If a transition t is not enabled then ${}^{E}M(t)$ undefined.

The ETPN semantics can be formally defined using its tuple notation. The transition $t \in T$ is enabled at time x if

for all $p \in {}^{\bullet}t$, ${}^{P}M(p) \geq W(p, t)$ and

${}^{E}M(t)$ is undefined.

Thus the transition is enabled if its input token requirements are met and it does not currently have an enabling time associated with it[8]. There is another step to the enabling of transitions in ETPNs the scheduling of the firing of enabled transitions. If transition t is enabled at time x, then

$^{E}M'(t) = x+y,$

where y is the enabling duration of transition t given by the function D(t).

The firing of an ETPN transition is done in one step. If $^{E}M(t) = x$, where $x \leq {}^{E}M(t')$ for all $t' \in T\backslash\{t\}$, then at time x

for all $p \in P$, $^{P}M'(p) = {}^{P}M(p) + W(t, p) - W(p, t)$ and

$^{E}M'(t)$ is undefined.

There is a major practical problem with the above definition. If a net includes loops and conflict, then the above definition can fail to work correctly. Consider the very simple ETPN shown in Figure 3.3. If there is a token in place p1, then both transitions t1 and t2 are enabled. That is, $^{E}M(t1) = 4$ and $^{E}M(t2) = 3$, so t2 fires first at $\tau = 3$. Now because t2 is part of a loop, this means that the token p1 is removed and created at this time. But the above semantics do not consider how this effects t1. Under the above defined semantics t1 would fire at $\tau = 4$. Clearly this should not be the case as in the firing of t2 the token in place p1 was removed disabling t1. Thus an additional step in the semantics needs to be included, the disabling of conflicting transitions.

[8] This second condition will be returned to shortly.

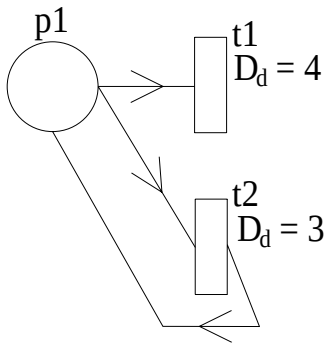


Figure 3.3: Simple ETPN illustrating problems caused by loops

If the transition t fires at time x, then

for all $t' \in T$, where ${}^{E}M'(t')$ is defined, if there exists $p \in {}^{\bullet}t$ such that $W(p, t) \geq {}^{P}M(p) - W(p, t')$, then ${}^{E}M'(t')$ is undefined.

This is the disabling of all transitions disabled by the firing of t. The application of the first step in the semantics, that is, the enabling of transitions, will re-enable any transitions that become enabled by the creation of tokens in the output places. Thus the above step would disable t1 in Figure 3.3 for it to be re-enabled in the next step. For this example these two steps would be repeated continuously.

The above definition, which is similar to that presented in [53], does not allow for concurrent enablings of the same transition. Allowing for concurrent enablings of the same transition has been overlooked in the ETPN literature. Papers [24], [60], [116], [146] and [151], do mention the concept but say it will not be considered for the sake of simplicity. This assumption certainly does simplify a number of issues, as will be seen in this Section and Section 5.2. In [27], [72] and [106], it is stated that concurrent enablings of the same transition are not considered without explanations as to why. In [39] the authors state that concurrent enablings of the same transition requires multiple timing distributions to be assigned to transitions. As

will be seen in this thesis, this is not the case unless the system being modelled requires it. The authors in [70] go further in excluding concurrent firing of any type, in either timed or non-timed PNs. In [4], [12], [13], [41], [50], [73], [113], [114], [145], [156] and [171], this issue is avoided by only considering TPNs that do not allow for this to occur in their structure. It should be noted that when exponential enabling times are used, such as in [7], [82], [120], [128] and [159], the memoryless property of the exponential distribution negates the need to consider concurrent enablings. Papers [19], [75], [104] and [117], define ETPNs for which concurrent enablings of the same transition could occur but simply ignore the possibility. In [8] and [47] marking dependent generation functions are introduced and used to represent multiple enablings of the same transition. However, as will be discussed later, this approach requires the construction of marking dependent time generation functions that take into account enablings and disablings at different epochs of multiply-enabled transitions as well as requiring detailed information about the reachable states.

The only papers that the author found which dealt with concurrent enablings of the same transition in ETPNs were [23], [25], [47] and [48]. Of these only [23] and [25] discussed (although very briefly in the case of [23]) the possible ramifications of allowing for concurrent enablings of the same transition. Section 5.2 discusses in more detail the problems that can arise when concurrent enablings of transitions are allowed.

The definition given earlier can be extended to allow for concurrent firings by letting the transition part of the marking, ${}^{E}M$, map T to a multi-set of times, in this case enabling times. Thus ${}^{E}M$ is a mapping: $T \rightarrow \{x: x \in \rho\}$. Before considering the semantics of an ETPN with concurrent enablings of the same transition, a number of definitions are required.

Definition 3.1: Enabling degree

The number of concurrent enablings of a transition t in the current marking is the transition's *enabling degree*.

Thus the enabling degree of transition t, denoted by $|{}^{E}M(t)|$, is the number of elements currently in the multi-set ${}^{E}M(t)$. This definition is also used in [8] and [47]. Three types of transitions are considered in these papers, *single server transitions* which have a maximum enabling degree of one, *multiple server transitions* with a specified maximum enabling degree and *infinite server transitions* which have no bounds on the enabling degree. The semantics defined earlier assumed that each transition was a single server transition. The semantics defined below consider all transitions to have infinite enabling degree, leaving it to the modeller to use input places to restrict the enabling degree where required.

Definition 3.2: Fresh enabling

A *fresh enabling* of a transition is an enabling that was not possible in the previous marking.

A fresh enabling may be the first enabling of a transition or it may be a later enabling. However, a fresh enabling is an enabling for which there does not currently exist an element in ${}^{E}M(t)$. The ETPN semantics can now be redefined to allow for concurrent enablings.

The transition $t \in T$ has fresh enabling at time x if

$$\text{for all } p \in {}^{\bullet}t, \quad {}^{P}M(p) \geq (|{}^{E}M(t)| + 1)\ W(p, t).$$

If there is a fresh enabling of the transition t at time x which has enabling duration y, then

$${}^{E}M'(t) = {}^{E}M(t) \cup \{x+y\}.$$

This new element is added at the end of the current multi-set.

This definition can be made more general by extending it to generate the number of fresh enablings allowed by the current marking. In the marking M the number, n, of fresh enablings of t is given by

$$\min_{\text{for all } p \in {}^\bullet t} \left(\left\lfloor \frac{{}^P M(p)}{W(p,t)} \right\rfloor - \left| {}^E M(t) \right| \right),$$

where $\lfloor x \rfloor$ is the greatest integer value less than or equal to x. For each fresh enabling at time x an enabling duration y_i (i = 1, 2, …, n) is determined by the function D(t). The elements $x+y_i$, i = 1, 2, …, n, are added to ${}^E M(t)$. Thus in total n elements are added to ${}^E M(t)$. Also if transition t is scheduled to fire at $\tau = x$, then

for all $p \in P$ ${}^P M'(p) = {}^P M(p) + W(t, p) - W(p, t)$ and

${}^E M'(t) = {}^E M(t) \setminus \{x\}$.

As with non-concurrent enabling there is a need to define a disabling step. In this case there is the chance that multiple disablings could occur in the same step. Problems related to this are dealt with in Section 5.2. Suppose the transition t fires at time x,

if for all $t' \in T$, where ${}^E M'(t') \neq \{\varnothing\}$, there exists $p \in {}^\bullet t$ such that

$(\left| {}^E M(t') \right| + 1)\ W(p, t') \geq {}^P M(p) - W(p, t)$,

then an enabling must be removed from ${}^E M(t')$.

There is the possibility that more than one enabling must be removed. In this case the number of enablings of transition t' that must be removed when transition t fires is given by

$$\min_{\text{for all } p \in {}^\bullet t} \left(\left| {}^E M(t') \right| - \left\lfloor \frac{{}^P M(p) - W(p,t)}{W(p,t')} \right\rfloor \right).$$

The semantics defined above assume that transitions always fire once their enabling time expires. Some definitions of ETPNs do not force this to happen. In [164] and [156] two types of firing rules are considered, strong and weak firing rules. Transitions with s*trong firing rules* fire immediately their enabling time expires. Transitions with *weak firing rules* are not forced to fire once their enabling time expires, however, if they do not fire at this time they cannot fire later. Weak firing rules are an implicit way of representing time outs. If a transition does not fire when its enabling time expires it is considered to have timed out. A problem here is that the designer does not have any direct control on which transitions time out and which do not. To do this transitions must be further classified into two types, those that time out and those that do not. There is also the need to specify a way of determining when the transition does and does not time out, something not considered in [164], since this only considers reachability analysis. One could also argue that this "hiding" of time outs within transitions makes it harder to determine where time outs occur in the system, as there do not exist any places that are marked when a time out occurs. The advantage of this approach is that it does reduce the model and state space size.

An interesting development in ETPNs was given in [171] and used in [37], [64] and [80]. In this paper the enabling timing durations were assigned to the transition input arcs. Figure 3.4 shows an example of time associated with transition input arcs. When time is assigned in this manner under the definition given in [171], it is the same as placing the maximum of these values on the related transition. Under this definition the preconditions of the transitions must first hold before the timing is considered. That is, all the input places must contain the required number of tokens before its enabling time is considered. The enabling time of the transition is then the largest value associated with the input arcs. In Figure 3.4 transitions t1 and t2 are enabled to fire. Now t1 is scheduled to fire at $\tau = 5$, due to the arc from p2 to t1

having this duration, and t2 at $\tau = 2$. Thus t2 fires disabling t1. The domain of the duration functions is now the input arcs instead of the transitions. As with relating holding durations to places or arcs rather than transitions, arc enabling durations are simply a different way of modelling the same type of delays as transition enabling durations. Their advantage is that they allow greater modelling flexibility.

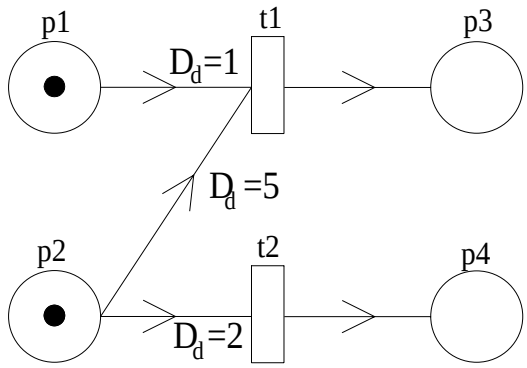


Figure 3.4: ETPN with duration functions associated with input arcs

In [64] Diaz and Sénac give an alternative interpretation to the way enabling times assigned to input arcs work. Here the input arc (p, t) is considered to become enabled as soon as W(p, t) tokens are present in the place p. Following enablement, the enabling time of the input arc begins to count down. It is this reduced time which is used in calculating the enabling time of a transition once the transition is enabled. This means that each arc has an individual starting time and the resulting time that the transition fires depends not so much on the time the transition became enabled, as on the time at which the input arc conditions were satisfied. This approach appears to contradict the concept of a transition not being activated until all its input conditions are satisfied, particularly when some of the different semantics introduced in [64] are considered. However, it has proved useful for modelling hypermedia systems, as

shown in [152] and [153]. These papers also give good examples of systems where assigning time to arcs is more natural.

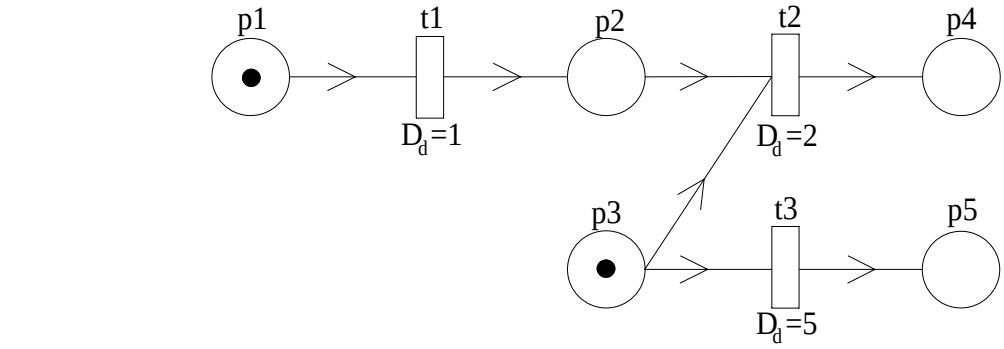
In the above semantics the question of conflict has not been considered in any detail. In ETPNs this is dealt with using the race policy, that is, the transition with the lowest enabling time fires first. This is a very simple way of resolving conflict. However, now consideration must be given to how a newly disabled transition will react to such a firing. Section 5.2 deals with this question as well as considering how the choice of a given policy effects the PN behavior and the analysis techniques that can be applied to the model.

3.3 Comparison of Holding and Enabling Durations

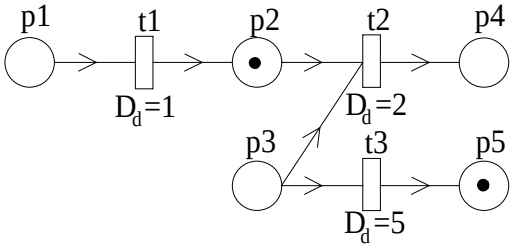
The difference in semantics between holding durations and enabling durations is best illustrated by a TPN with confusion. Consider the TPN shown in Figure 3.5(a). If holding durations are used, transitions t1 and t3 fire concurrently at $\tau = 0$, placing one unavailable token in place p2 and one in p5. The token in p2 becomes available at $\tau = 1$ and the token in p5 at $\tau = 5$. No further transitions are enabled by this marking. Thus this is the final state of the HTPN, shown in Figure 3.5(b). By contrast if enabling durations are used, initially transitions t1 and t3 are enabled and t1 is scheduled to fire after one time unit and t3 after five. Thus at $\tau = 1$, t1 fires removing a token from p1 and creating a token in p2. This enables transition t2 to fire after two time units have passed, that is, when $\tau = 3$. Transition t3 is still enabled and still scheduled to fire at $\tau = 5$. Thus t2 fires (before t3) at $\tau = 3$, removing the tokens from p2 and p3, and placing a token in p4. No further transitions are enabled so this is the final state of the ETPN (see Figure 3.5(c)). This example shows how changing the timing policy of the TPN can cause a dramatic change to the way the net executes. Both final markings are possible in the PN, but when deterministic time is included only one can be

reached in TPNs. It is the interruption to the enabling of t3 in the case of enabling durations that changes the outcome of the model. It was the requirement to model such interruptions in systems with time outs that led to the development of enabling durations.

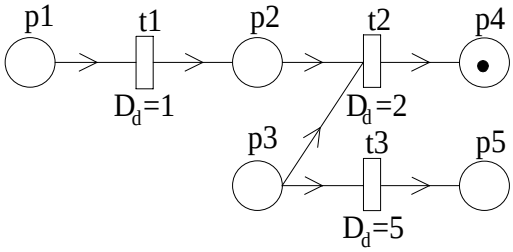
Holding durations can be modelled using enabling durations if *immediate transitions* ([6]) are used. Immediate transitions have zero enabling durations and always fire before transitions with nonzero enabling durations. As outlined in [30] and [117], a simple rule for transforming a HTPN to an equivalent ETPN representation is to precede each transition by an immediate transition output place pair. An equivalent ETPN to the HTPN shown in Figure 3.5(b) is given in Figure 3.5(d). In this example there is no need to include the first immediate transition, which is included only for completeness. In fact immediate transitions need only to be placed before those transitions that can be in conflict with another transition. To be safe this should be done for each transition that is in structural conflict with another one.



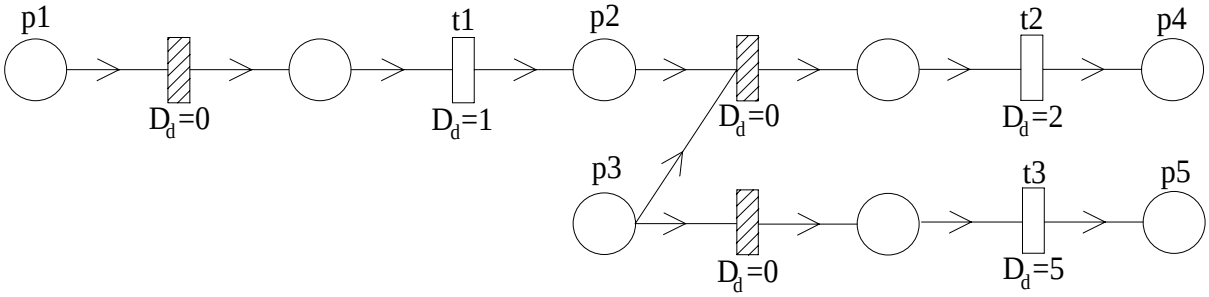
(a) Initial marking of a TPN with confusion



(b) Final marking using holding times



(c) Final marking using enabling times



(d) Representing holding times with enabling times

Figure 3.5: TPN showing the difference between holding and enabling durations

The above transformation from HTPNs to ETPNs can be defined on the tuple definitions given earlier. Let P be the set of places and T the transitions in the HTPN.

1. Each place in the HTPN is mapped to a place with the same name in the ETPN. These places form a subset of places, ${}_{O}P$, in the ETPN.
2. If the transition $t \in T$ is not in structural conflict with any other transition in T, then it is mapped directly to ${}_{H}t$ in the ETPN, with $W(p, {}_{H}t) = W(p, t)$, $W({}_{H}t, p) = W(t, p)$ and

$D(_Ht) = D(t)$. Thus the transition in the ETPN has the same input and output places as the transition in the HTPN as well as having the same timing function.

3. Each transition $t \in T$ which is in structural conflict with another transition in T is mapped to two transitions, $_It$ and $_Et$, and a place, $_tp$.

   3.1. For all places $p \in P(=_OP)$ the transition $_It$, formed from $t(\in T)$, has $W(p, _It) = W(p, t)$, $W(_It, _tp) = 1$ and $D_d(_It) = 0$. That is, the transition $_It$ has the same input places as the original transition $t \in T$ and one output arc to the newly created place $_tp$. This transition is an immediate transition.

   3.2. The second created transition $_Et$ has $W(_tp, _Et) = 1$, $W(_Et, p) = W(t, p)$ for all places $p \in P(=_OP)$ and $D(_Et) = D(t)$. That is, the second transition has the newly constructed place $_tp$ as its input, the original transitions output places as its outputs and the same time function as the HTPN transition. Thus tokens held in place p of the HTPN are held in place $_tp$ of the equivalent ETPN.

4. The initial marking is transformed in two parts.

   4.1. The available tokens are directly transformed, that is, $^PM_0(p) = {}^AM_0(p)$ for all $p \in P(=_OP)$.

   4.2. If place $p \in P$ initially has unavailable tokens, then for each $(n, x) \in {}^UM(p)$ a new place, $_Up$, and a new transition, $_Ut$, are constructed. The transition $_Ut$ has one input arc from $_Up$ and one output arc, with weight n to place p, that is, $W(_Up, _Ut)) = 1$ and $W(_Ut, p) = n$. The enabling time to this transition is x and place $_Up$ initially has one token in it, that is, $D(_Ut) = x$ and $M_0(_Up) = 1$. It should be noted that $_Ut$ will only ever fire once.

Note that the flow function for all possible arcs not outlined above is set to zero. Let $_IT$ be the set of transitions $_It$, $_ET$ the set of transitions $_Et$, $_HT$ the set of transitions $_Ht$, $_UT$ the set of

transitions $_{U}t$, $_{t}P$ the set of places $_{t}p$ and $_{U}P$ the set of places $_{U}p$. The set of transitions of the ETPN is the union of the sets $_{I}T$, $_{E}T$, $_{H}T$ and $_{U}T$, and the set of places in the ETPN is made up of the sets $_{0}P$, $_{t}P$ and $_{U}P$. This transformation holds only for transition holding times. However, similar transformations can be constructed when time is associated with places or input arcs.

The reverse transformation cannot be defined as there is no way of representing interruptions of ETPNs in HTPNs. In [182] interruptions were introduced into FTPNs in the form of special arcs called escape arcs. Escape arcs go from a place, referred to as the escape place, to a transition. If a transition is in the process of firing and a token arrives in one of its escape places, then the firing of the transition stops and the tokens are returned to their input places. This new type of arc goes a long way towards duplicating the interruptions of enabling times. However, to do this the conflict resolution function used must be defined in such a way as to ensure that the actions of the HTPN mimic those of an ETPN. This means defining a function that is dependent on how long a given transition was in the process of firing before it was interrupted.

3.4 Combination Model

In [145] (and [146]), Razouk and Phelps define a PN structure where both firing and enabling durations are used. Although this technique creates a very powerful and versatile modelling method, most researchers have not used it. A comprehensive search showed a number of references to this paper but only a few recognised it as a combined model ([37], [58], [66], [78], [124], [130], [136], [158], [175], [176] and [183]). Of these only [37], [58], [66] and [136] use this combined approach. Due to the fact that a definition of TPNs with firing

durations is not given in the main text, this section will discuss the combination model in terms of HTPNs and ETPNs.

Consider the TPN shown in Figure 3.6(a). This is the same PN with the same initial state as in Figure 3.1 and Figure 3.2, however, now enabling and holding durations have been assigned to each transition. As before initially transition t1 is enabled. This time it must be enabled for 0.6 of a time unit before it can fire. Thus at $\tau = 0.6$ transition t1 fires (see Figure 3.6(b)). Due to the holding duration of 0.4 the token created in place p3 does not become available until $\tau = 1$ (see Figure 3.6(c)). At this time transition t2 is enabled. This transition has an enabling time of 1 so it fires at $\tau = 2$, as shown in Figure 3.6(d). The holding duration of transition t2 is 3 and so the token created in p4 is not available until $\tau = 5$ (see Figure 3.6(e)). Thus as before the same marking is reached at the same time as in the previous examples. As discussed in Section 3.3 the major difference between HTPNs and ETPNs is that in ETPNs enabled transitions can be interrupted during their enabling. When both enabling and firing times are used the transition can be interrupted only during the enabling part of the process. Clearly a HTPN can be represented by this time representation by setting the enabling durations of all the transition to zero. Similarly an ETPN can be represented in this structure by setting the firing durations of all the transition to zero. Thus this time representation allows for a compact way of representing the features of both HTPNs and ETPNs.

The same tuple definition can be used for this combination model as was used in the earlier time representations. In this definition P, T and W are as given earlier. Now D maps to two duration functions, the first relating to the enabling time function and the second to the holding time function. Let ${}^{E}D(t)$ be the function determining enabling durations for transition

t, that is D(t) = (^{E}D(t), •). Similarly define ^{H}D(t) to be the function which specifies the holding time for transition t, where D(t) = (•, ^{H}D(t)).

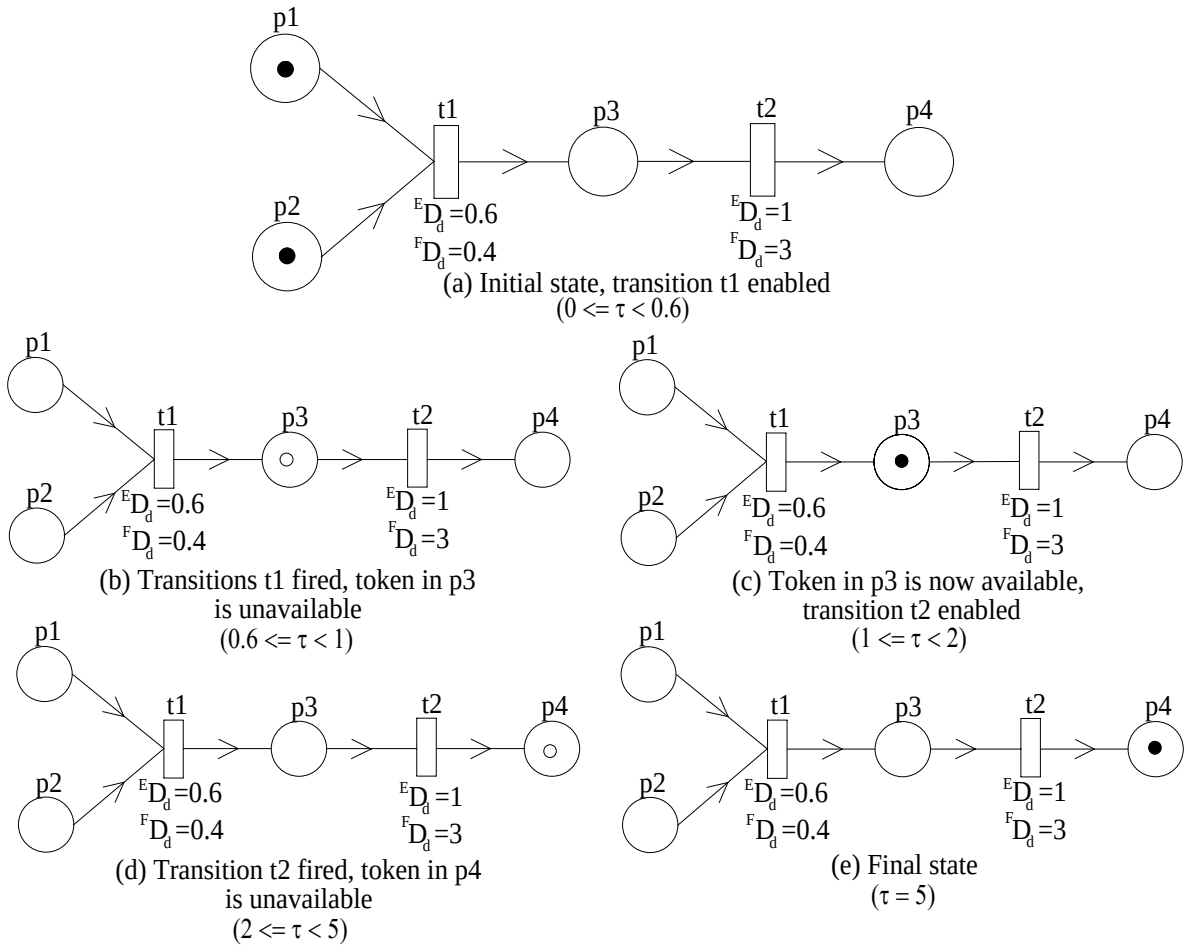


Figure 3.6: TPN with enabling and holding times.

As before M_0 relates to the initial marking of the TPN. The marking in this case is made up of 3 elements, the available tokens, ^{A}M, remaining enabling times, ^{E}M, and the unavailable tokens, ^{U}M. The first two are as defined in Section 3.1 and ^{E}M is as defined in Section 3.2. This definition is an adaptation of that given by Razouk and Phelps in [145] and [146] where holding times have been used instead of firing times. This change allows for concurrent

firings of the same transition as well removing some other problems with the original definition. However, it retains the principle of the original concept.

The semantics of this model are a combination of those for holding and enabling durations. The number of fresh enablings of transition $t \in T$ is given by

$$\min_{\text{for all } p \in {}^\bullet t} \left(\left\lfloor \frac{{}^{A}M(p)}{W(p,t)} \right\rfloor - \left| {}^{E}M(t) \right| \right).$$

For each fresh enabling of the transition t at time x, the element x+y is added to the remaining enabling time, where y is generated separately by ${}^{E}D(t)$ for each fresh enabling. That is, if t is freshly enabled at time x with the enabling duration y, which is determined by the function ${}^{E}D(t)$, then

$${}^{E}M'(t) = {}^{E}M(t) \cup \{x+y\}.$$

Consider the situation when transition t is to fire at the current time, x. Then at $\tau = x$

for all $p \in P$, $\quad {}^{A}M'(p) = {}^{A}M(p) - W(p, t)$,

$${}^{E}M''(t) = {}^{E}M'(t) \setminus \{x\} \text{ and}$$

for all $p \in P$ and $W(t, p) > 0$, $\quad {}^{U}M'(p) = {}^{U}M(p) \cup \{(W(t, p), x+y)\}$, where y is the holding duration of transition t given by ${}^{H}D(t)$.

The other process in the semantics relates to unavailable tokens becoming available. At $\tau = x+y+z$ for $(n, x+y+z) \in {}^{U}M(p)$,

$${}^{A}M'(p) = {}^{A}M(p) + W(t, p) \text{ and}$$

$${}^{U}M'(p) = {}^{U}M(p) \setminus \{(n, x+y+z)\}.$$

As stated at the beginning of this chapter the clock is implicit. The ordering of events is such that time is non-decreasing. Thus the ordering of the firing of a transition and the availability

of tokens depends on the times in ${}^{E}M$ and ${}^{U}M$. The shortest of these times determines which of these steps occurs first.

3.5 Relative Time

So far in this section all the discussion has focused on the concept of absolute global time. In [23]-[25] relative time is used. To understand the concept of relative time, consider the case of HTPN with the unavailable marking of ${}^{U}M(p1) = \{(2, 3), (3, 7)\}$ and ${}^{U}M(p2) = \{(1, 4), (3, 9)\}$. This implies that the next token(s) to become available will be two tokens in place p1. In absolute global time this means that the global clock progresses to time + 3 and then the tokens become available. In relative time this means that the tokens also become available in 3 time units only now there is no global clock, but rather each of the other times in the marking are reduced by the 3 time units making the new unavailable marking ${}^{U}M(p1) = \{(3, 4)\}$ and ${}^{U}M(p2) = \{(1, 1), (3, 6)\}$. Thus when relative times are used there is no need to retain a global clock as only the time remaining before an event occurs is actually used. In this thesis the use of relative times will not be considered further as the analysis that is of interest relies on a global absolute clock.

CHAPTER 4 GENERATING DURATIONS

The TPN literature distinguishes four different, although not distinct, forms that the duration functions take: deterministic, stochastic, interval and fuzzy. Of these deterministic, stochastic and interval will be discussed in depth. The fuzzy duration function will be briefly introduced, as it is are currently only a fringe form of duration function. Each of these has been applied to the different types of time representations defined in Chapter 3. The form that the functions take depends on the system being modelled, analysis being conducted and the information available about the time delays. Throughout this chapter time durations will be discussed in relation to transitions, however, the discussions are equally valid when time is assigned to places or arcs.

In early HTPN models, as in [143], deterministic time delays were used in a similar way to the examples given in Chapter 3. In [166] and [167] the positive integers are used, in [14], [17], [81] and [109] the nonnegative reals and in [149] the nonnegative rationals. This is the simplest representation used, in this case the holding or enabling durations taking a fixed value for all firings of t. Thus

$$D_d(t)(\tau) = \begin{cases} 0 & \tau < k \\ 1 & \tau \geq k \end{cases}.$$

With stochastic time delays the duration of an event is determined by a probability distribution function. Originally the exponential distribution was used, as in [3], [120], [128], [159] and [182]. In these early works the duration functions were the set of exponential distributions and were expressed in terms of the transition firing rates. This was extended to allow the durations to be generated by an arbitrary distribution function (see [4], [18], [45],

[75], [103] and [123]). Thus the holding (enabling) duration of transition t at a particular firing (enabling) is given by the random variable y, with the probability distribution function $D(t)(\tau)$. Stochastic time delays can mean that not only are the times of the durations non-deterministic but the final marking may also become non-deterministic. This is because for HTPNs the time tokens become available and for ETPNs the time transitions fire can impact the firing sequence and thus the final marking.

Interval time delays were first introduced in [118] and [119]. In this case the durations defined in the TPN are the minimum and maximum time that a transition takes to fire. The actual duration for a particular firing (enabling) of a transition is considered to take some value in this interval. Studies in this area generally do not specify how the time is sampled, simply saying that it was calculated from this interval, as in [24], [116], [117], [119] and [171]. In this case the actual duration is not important, it is only the interval of possible durations that is considered. This type of representation is useful if the delays in the real system are variable but not enough is known about delays to determine their distribution. In this case the duration functions of take the form of $\rho \times \rho$. In [25], [27], [39], [60], [97], [149] and [151], ρ consists of the positive rational numbers while in [2], [117] and [119] it is the positive reals. Generally interval durations of this form are used for the calculation of performance bounds ([1], [25] and [27]). Thus the holding (enabling) duration of transition t is $y \in [\min, \max]$.

The relationship between the three time generation functions is very simple. Clearly deterministic duration functions are a subset of both stochastic and interval duration functions. Also, as mentioned above interval duration functions are similar to stochastic duration functions in that they specify a set of possible durations. The difference is that in the

case of stochastic duration functions a probability of selecting a particular time is also specified. In papers [89] and [151] the idea of interval durations is extended so that a probability density function is associated with each interval making stochastic and interval duration functions equivalent.

Further variations are introduced by researchers who allow for duration functions which are marking dependent ([5], [8], and [120]) and history dependent ([7]). Marking dependent duration functions are often used to represent different types of queues. Such features can also be represented using concurrent enablings of the same transition. Concurrent enablings of the same transition represents an infinite-server queue. This can be restricted by having restrictive input places that reduce the number of concurrent firings. Figure 4.1 shows how an input place can be used to control the number of concurrent firing of a transition and thus be used to represent different types of servers.

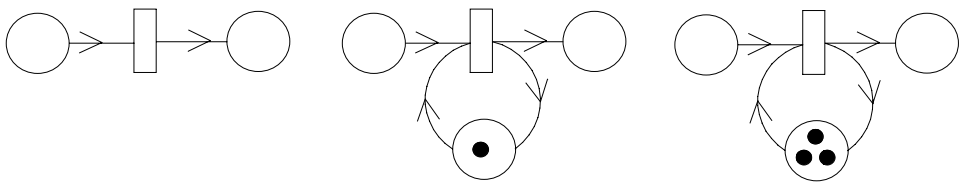


Figure 4.1: Examples of multi-server queues

This final method of specifying duration functions is by fuzzy duration functions. In [126] these are assigned to output arcs of HTPNs. This is similar to the work in [89] and [151] where probability functions are added to interval generating functions.

The remainder of this thesis focuses of stochastic generation functions, because interval timings are not relevant to the analysis techniques discussed in the latter Part.

CHAPTER 5 EXECUTION STRATEGIES

The execution strategies outlined in Chapter 3 give the general enabling and firing rules of TPNs. These strategies varied depending on the time representation used and need to be extended when conflict is considered. In some papers, for example [1], [2], [64] and [167], conflict is not discussed as the generation of the reachability graph is used for analysis. In these cases, as with non-timed PNs, all paths are considered and a mechanism for deciding which transitions should fire is not required. This approach is generally taken when interval duration functions are used, while in [42], [175] and [182] only nets in which conflict never occurs are considered. The effect of conflict is now considered separately for holding and enabling durations.

5.1 Holding Durations

The most common ways of resolving conflict in HTPNs is with the use of firing frequencies, as in [81] and [184], or firing weights, as in [55] and [142]. In these specifications each transition is assigned a value which determines how often this transition fires in relation to other transitions enabled at the same time. For example, consider the net in Figure 3.5 with one token in each of the places p2 and p3. This causes a conflict between transitions t2 and t3. Let the firing frequencies 0.75 and 0.25 be assigned to transitions t2 and t3 respectively. This means that when conflict occurs between these transitions, transition t2 will fire with a probability of three quarters and t3 one quarter of the time. With both the above extensions the PN tuple is extended to involve another element which defines the firing frequency of each transition, as in [81]. The difference between firing frequencies and firing weights is that in the case of firing frequencies the sum of firing frequencies of the enabled transitions for all the reachable markings must equal one, while for the case of firing weights this is not true.

Firing weights work in a similar way to firing frequencies as they determine the probability of a transition firing over other transitions it is in conflict with. Here the actual probability that a given transition fires in a marking is calculated from the ratio of the transition's firing weight to the sum of the firing weights of the other transitions enabled at the same time. This is a trivial difference in itself but does make a difference to model construction and execution. It has been mentioned as in more complex conflict resolution policies this subtle difference can have a large impact.

Conflict resolution or pre-selection policies can be grouped into two types: global and local ([7]). Global policies resolve conflicts throughout the whole PN, as in the case of firing frequencies, while local ones choose only between a subset of transitions, as with firing weights. Pre-selection policies can be as simple as the assigning of firing frequencies or may be a function of a TPN's history and marking.

Global preselection policies can lead to model design problems. In the TPN in Figure 3.5 it was easy to see where conflict existed. However, for larger nets it is not always as easy. Some papers, such as [5], even suggest that the only way to be sure that a policy is correct is by generating the reachability graph of the corresponding PN before the policy is specified. Although this may be true in some cases it is more likely that what is required is a study of those transitions in structural conflict. Such a study may not help in cases of confusion where order in which concurrent transitions fire is also important. A counter argument is that in creating the model the designer should be or at least become aware of all instances of conflict

and confusion[9], and construct a preselection policy so that the model reflects the behaviour of the real system.

Unlike global policies, local ones operate only on a certain set of transitions. Since the transitions included in a given specification are well defined the state space does not need to be calculated before the policy is specified. However, this technique can lead to problems in the case of undetected conflict or confusion. In such cases the local policy may not be extensive enough to correctly resolve some conflicts.

One way around the problems of global preselection policies is using priorities. Each transition is given a priority. Only transitions with the same priority levels can be in conflict as the enabling of higher priority transitions disables all the transitions with lower priorities. Thus the modeller can specify which events are in conflict in the real system, give the corresponding transitions in the model the same priority level and then determine the relevant policy for each group of transitions with the same priority level. The use of priorities is a simple but effective way of converting a global preselection policy into a local one. To see how priorities work consider the case where the transitions t1, t2, t3, t4 and t5 are enabled with the pre-selection policy defined as $f(t1) = (1, 0.5)$, $f(t2) = (3, 0.75)$, $f(t3) = (2, 0.5)$, $f(t4) = (3, 0.75)$ and $f(t5) = (3, 0.5)$, where the first value is the priority and the second the firing weight. Thus the transitions t2, t4 and t5 are in conflict and t2 and t4 fire with

[9] Confusion is only an issue in the case where there are some transitions with zero holding durations. The problem of conflict resolution becomes more complicated as the chosen policy must not only resolve conflict but also resolve the problem of confusion and so must select which transition fires first even if no conflict exists.

probability 3/8 and t5 with probability 1/4. It should be noted that the problem caused by conflicting transitions in HTPNs is the same one faced by conflicting immediate transitions in ETPNs ([47]).

In [123] a HTPN representation is defined in which the modeller has the freedom to define the selection policy of their choice. It may be using frequencies, priorities, through the use of modeller-defined functions or any combination of these. Generally, the more complicated the policy the more the modeller needs to know about the behaviour of the PN before the policy can be defined.

The pre-selection policy used in a HTPN becomes the sixth element of the tuple definition as in [183]. The form of this function depends on the pre-selection policy chosen. Consider the case of firing frequencies. In this case the pre-selection policy is a mapping: $T \rightarrow [0, 1]$. In the case of firing weights the function is a mapping: $T \rightarrow Z_0^+$. Priorities with firing weights give rise to a mapping: $T \rightarrow Z_0^+ \times R_0^+$, where R_0^+ denotes the nonnegative reals. In this case the first value is the priority and the second the transition weight.

Some consideration must also be given to the changes pre-selection policies make to the HTPN semantics. The enabling semantics are unchanged but the firing semantics must be altered. The firing process is now preceded by the application of the selection process which determines which of the currently enabled transitions actually fires. Once a transition has been chosen to fire the firing proceeds as defined in Section 3.1. In the generation of the reachability graph this means generating only those paths with non-null probabilities. The

definition of a pre-selection policy may also include the order in which concurrently enabled transitions fire is defined, whereas previously this may not have been the case.

As touched on earlier, this is not required if the probability of reaching a given state is unimportant and the entire state space is being explored. That is, in cases such as this all firings need to be explored and so the ordering of transition firings in not relevant.

5.2 Enabling Durations

In ETPN conflict between two transitions is resolved using the race policy in which the transition with the shorter enabling duration fires first. Although this would appear to be a straightforward way of solving conflict there are still a number of issues that must be considered.

The first issue is how to determine which enablings of a transition, t say, are removed from ${}^{E}M(t)$ when the number of possible enablings of t is reduced by the firing of a conflicting transition, that is, a *disabling policy* must be determined. This is an issue overlooked in most the literature. Two exceptions are [23] and [25]. Both these papers use the complications of multiple enablings as a justification for not allowing for them[10]. Because these papers use interval duration functions some of the complications mentioned here do not arise. However,

[10] As highlighted by these papers a secondary issue that will not be dealt with here arises when interval enabling durations are used. In this case there is a need for a firing choice function. So if the remaining enabling intervals of a given transition overlap the firing choice function can be used to determine which of the current enablings of the transition fires first.

they serve to highlight the fact that the use of multiple enablings has added complications as will be seen in the following discussion on disabling policies.

There are numerous disabling policies that can be used. The simplest is to use a first-in first-out policy. This policy is easy to execute by virtue of the fact that new elements are added to the end of ${}^{E}M$. However, it may not represent the system being modelled.

Consider the PN shown in Figure 5.1. To illustrate the problems that can arise let $D_d(t1) = 3$, $D_d(t2) = 5$ and let D(t3) be stochastic. Also let the place portion of the initial marking be ${}^{P}M_0(p1) = 1$, ${}^{P}M_0(p2) = 1$ and ${}^{P}M_0(p3) = 1$ as shown. For this marking, transitions t1, t2 and t3 are enabled. Transitions t2 and t3 are in conflict. Assume that D(t3) generates an enabling time of 6, that is, the transition enabling part of the marking is ${}^{E}M(t1) = \{3\}$, ${}^{E}M(t2) = \{5\}$ and ${}^{E}M(t3) = \{6\}$. The defined semantics means that transition t1 fires at $\tau = 3$ changing the place portion of the marking to ${}^{P}M(p2) = 2$ and ${}^{P}M(p3) = 2$. Assume that the second enabling of transition t3 is an enabling of 1 so ${}^{E}M(t2) = \{5, 10\}$ and ${}^{E}M(t3) = \{6, 4\}$. The next scheduled firing is of transition t3 at $\tau = 4$. This firing will cause the disabling of one of the enablings of transition t2 and thus require the removal of an element from ${}^{E}M(t2)$. However, if the policy of first-in first-out is used then 5 will be removed from ${}^{E}M(t2)$, although this enabling does not actually relate to the firing of t3 at $\tau = 4$. In fact this policy means that transition t2 does not fire at all. In this example it is easy to see that the second enabling of the transition t2 actually relates to the second enabling of transition t3 and thus when t3 fires at $\tau = 4$ it is the second enabling of t2 that should be removed, that is, the one set for 10 time units. In this case a more representative rule might be to remove the element corresponding to the transition that just fired. So for the above situation the second enabling of transition t2 would be removed meaning transition t2 fires at $\tau = 5$.

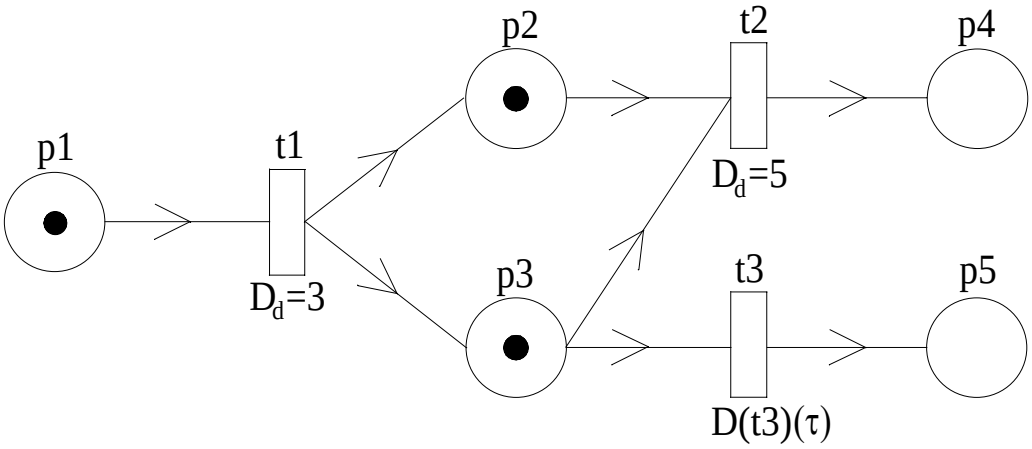


Figure 5.1: ETPN With Concurrent Enabling of the Same Transition

Sometimes it is not as easy to determine which enablings of a given transition in a conflict set relates the enablings of the other transitions in the conflict set. Consider the example of the ETPN shown in Figure 5.2. In this case the enabling durations of transitions t1 and t2 are deterministic while the enabling of transition t3 is stochastic. Assume that the first two enabling durations generated by D(t3) are 10 and 5. Thus ${}^{E}M_0(t1) = \{4\}$, ${}^{E}M_0(t2) = \{7\}$ and ${}^{E}M_0(t3) = \{10, 5\}$, so transition t1 fires at $\tau = 4$. This firing results in the enabling of transition t2 for a second time making ${}^{E}M(t2) = \{7, 11\}$. The next firing is at $\tau = 5$ when transition t3 fires. This firing actually disables one of the enablings of t2, but which one? Since tokens are indistinguishable there in no real way of saying which of the two tokens in p3 relates to which enabling. Thus a further rule must be defined. The main problem is not defining a rule but realising its consequences and making sure that these do not invalidate the model.

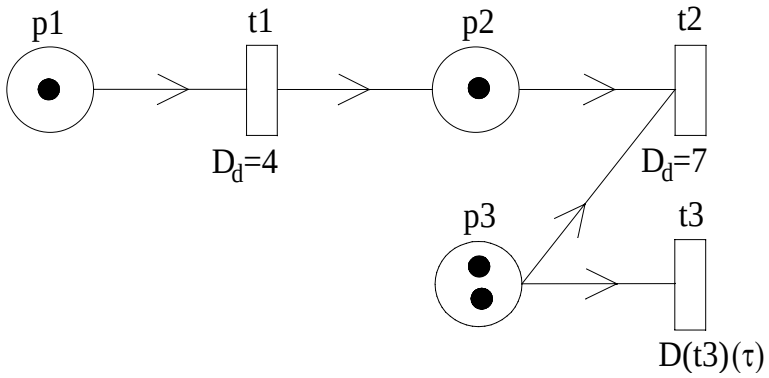


Figure 5.2: ETPN With Concurrent Enabling of the Same Transition

In summary there are different disabling policies that can be defined. In their simplest form these policies can be for example first-in first-out, last-in first-out, smallest time removed or largest time removed. In all these cases the disabling policy is independent of the actual situation that lead to the enablings of the transition. Another policy that is independent of the situation is that of a random selection, where a probability function is defined for each transition and this function is used to determine which enabling will be removed. For example, if there were three enablings of a transition and a recent firing resulted in the transition now being only enabled two times, then the probability function could be used to generate three random numbers and the enabling corresponding to the smallest value would be removed. A more complex disabling policy is to try and keep track of which tokens lead to each enabling. However, as shown above this is not always possible and so an alternative disabling policy must also be specified for those cases when the disabling cannot be related to the token removed by a conflicting transition. Policies that trace the tokens removed and use this to determine the disabling policy no longer view tokens as indistinguishable.

As mentioned in Section 3.2, some articles, for example [8] and [47], use marking dependent generation functions to represent concurrent enablings of the same transition. These papers

express that concurrent enablings of the same transition can be dealt with using marking dependent generation functions. Although this may be true in some cases, to truly represent concurrent enablings of the same transition the time generations functions must be marking and history dependent. As an example of this consider the PN in Figure 5.2 with the enabling durations given by $D_d(t1) = 6$, $D_d(t2) = 2$ and $D_d(t3) = 7$. Now in the case of the initial marking M(p1) = 2 and M(p3) = 2 with concurrent enablings of the same transition, transition t1 fires twice at $\tau = 6$ and t3 fires twice at $\tau = 7$ reaching a final state of M(p2) = 2. For the case of the initial marking M(p2) = 2 and M(p3) = 2 with concurrent enablings of the same transition, transition t2 fires twice at $\tau = 2$ reaching a final state of no tokens in the net. There is no way of representing both these behaviours with the same net definition using marking dependent generation functions that are not history dependent. Further complications are introduced when multiple enablings and disablings of transitions occur at different epochs. Also for marking dependent generation functions to work there is a need to know the complete state space before specifying the generation functions.

Having decided on a disabling policy there is still a need to answer the question of what enabling time a disabled transition should take the next time it is enabled. This question is solved by defining a memory policy. In [7] and [8] three *memory policies* are defined. These are essentially the same policies as in [19] only defined in a more straightforward way.

The first method is referred to as *resampling*. In resampling, each time a transition fires, a new value of the enabling time is assigned for each enabled transition. That is, the time durations distribution functions of all enabled transitions are resampled irrespective of whether or not their enabling conditions were affected by the firing. This approach is of little practical use (except in the case when exponentially distributed durations are used) as it

removes one of the most important advantages of modelling with PNs, a true representation of concurrency. A PN cannot be considered to have concurrent events if every time an event occurs all the other events that are concurrently in the process of occurring stop and have to restart. Having the enabling times recalculated each time another transition fires has this effect and so removes concurrency from the model. However, this approach does allow the PN reachability graph to be considered as a semi-Markovian process, as outlined in [53]. This may also be an appropriate representation of some systems. If resampling is used then the disabling selection policy can be ignored as all enablings are removed and enabling times are recalculated after each firing.

Age memory refers to the case where transitions remember how long they were enabled regardless of which transitions fire. That is, even if a transition becomes disabled, it still remembers the remaining time it is required to be enabled before it will fire and next time it becomes enabled it uses this time as its enabling duration. This method is a good way of representing shared processor systems.

As an example of how the above two semantics work, consider the PN in Figure 2.1 with the shown marking. Let the enabling durations for each transition be $D_d(t1) = 2$, $D_d(t2) = 3$, $D_d(t3) = 6$ and $D_d(t4) = 1$. Initially transitions t1 and t3 are enabled to fire at times 2 and 6. Regardless of which of the two semantics is used, transition t1 will fire at $\tau = 2$ removing and creating tokens. It is at this point that the two methods differ.

If resampling is used then when t1 fires, all of the enabled transitions must now resample their enabling duration functions to determine when they will be scheduled to fire. Transition t2 is enabled to fire at $\tau = 5$, after being enabled for 3 time units, while transition t3 is enabled to

fire at $\tau = 8$, after being enabled for 6 time units. Thus t2 fires first disabling t3. The next transition to fire is t4 which fires at $\tau = 6$ restarting the same process with a time shift of 6 to the right. So for this memory policy transition t3 never fires.

In the case of age memory things do not turn out quite the same way. When transition t1 fires the first time, the enabling time of transition t3 is not recalculated so it is still scheduled to fire at $\tau = 6$. However, transition t2 still fires first at $\tau = 5$, as its enabling duration is 3. This action disables transition t3 when it has only 1 time unit remaining on its enabling time. The remaining enabling time to transition t3 is remembered for the next time it becomes enabled, which occurs after the firing of transition t4 at $\tau = 6$. As before transition t1 is scheduled to fire at $\tau = 8$, however, the new enabling of transition t3 has only 1 time unit remaining in its age memory and so is enabled to fire at $\tau = 7$. Thus it is transition t3 which fires next at $\tau = 7$ and then t1 at $\tau = 8$, reaching a deadlock that could not be reached using resampling.

An extension of age memory defined by me in [37] is to have *age memory with loss*. In this case the age memory slowly degrades between enablings of the transition. For example, if the memory degradation function was time dependent, then if there was a sufficiently long time between enablings the transition will lose any memory of previously being enabled. Age memory with loss is a good way of representing human actions. Consider the case when someone is working on a task and gets interrupted by a higher priority task. The person carries out the second task leaving the first where it is. The longer the time until the first task is returned to, the more time the person must spend refamiliarising themself with the original task. However, if the task is returned too soon after it was left then there will be minimal loss.

The final memory policy is called *enabling memory*. In this method transitions retain the time they have been enabled as long as they remain enabled. This is the rule used in the enabling duration description given in Section 3.2 and Section 3.3. This method provides a simple way of representing resource-sharing systems where the resource can be used by only one process at any time and the process must complete its work to make any gains. It is this technique that is used in most ETPNs.

It is possible to make the rules defined above part of the time generation function, meaning that all transitions do not have to have the same rule. This results in a model that combines all three memory policies.

In [26] and [161] the age and enabling memory policies are referred to as pre-emptive resume and pre-emptive repeat different. Yet another rule is also introduced, that of pre-emptive repeat identical. This is a policy that is a compromise between age and enabling memories. In this case a disabled transition retains its enabling duration but restarts at the beginning of this time when it is re-enabled. This is just a special case of age memory with loss, where the loss is always all the work done so far. In the case of deterministic enabling durations the pre-emptive repeat identical policy is the same as the resampling policy. Also introduced in [26] are some special types of arcs that reset the enabling durations of transitions further enhancing the time representation.

As mentioned earlier the inclusion of immediate transitions means ETPNs must also use preselection policies in a similar way to that discussed in Section 5.1. A preselection policy may also need to be used if there is the possibility that two or more conflicting transitions have the same enabling duration. In this case the race policy cannot decide which of the

transitions will fire first. Thus despite the use of the race policy ETPNs can have the same difficulties in resolving conflicts as HTPNs. A similar situation arises if conflicting transitions have the same enabling times. In this case some kind of preselection policy must be used to determine which will fire.

CHAPTER 6 SUPER-CLASS

In this section a new type of TPN is introduced. All the above defined TPNs form sub-classes of this TPN. For this reason it will be referred to as the super-class TPN (STPN). The STPN was first introduced by me in [37]. Here a more comprehensive definition will be given.

Although any of the TPNs introduced can be represented with ETPNs, using ETPNs may mean that the relationship between the system being modelled and the model is blurred, making it harder to construct and interpret such models. However, it is still advantageous to have a way of representing all types of TPNs in one form. This new representation makes it easier to translate from one form to another as well as allowing greater modelling flexibility. The STPN definition has enabling durations associated with the input arcs and holding durations with the output arcs. This representation allows modelling of resource holding (as modelled in HTPNs) and interruptions (as modelled in ETPNs) in the most flexible way possible (using arc durations).

6.1 Super-Class Definition

Consider the TPN in Figure 6.1, which has enabling times assigned to input arcs and holding times assigned to output arcs. Figure 6.1(a) shows the initial state in which transition t1 is enabled. The semantics used mean that the enabling time of a transition is given by the maximum value of its input arcs. Alternatives to this have been mentioned in Section 3.2. In the case of transition t1 there are two input arcs, one from place p1 and one from place p2. One has the enabling duration 0.4 and the other 0.2. This means that the overall enabling duration is 0.4 given by the arc from place p1 to transition t1. So transition t1 fires at $\tau = 0.4$, removing the available tokens and creating unavailable tokens. The holding time of each

created tokens depends on the output arc from the transition to the output place the token is created in. In Figure 6.1 transition t1 has two output places p3 and p4. The output arc from transition t1 to place p3 has the holding duration 0.4 while the output arc from transition t1 to place p4 has the holding duration 0.6. Thus the token created in place p3 by the firing of transition t1 at $\tau = 0.4$ will be available at $\tau = 0.8$ (see Figure 6.1(b) and Figure 6.1(c)). The token created in place p4 by the firing to transition t1 at $\tau = 0.4$ will be available at $\tau = 1$ (see Figure 6.1(d)).

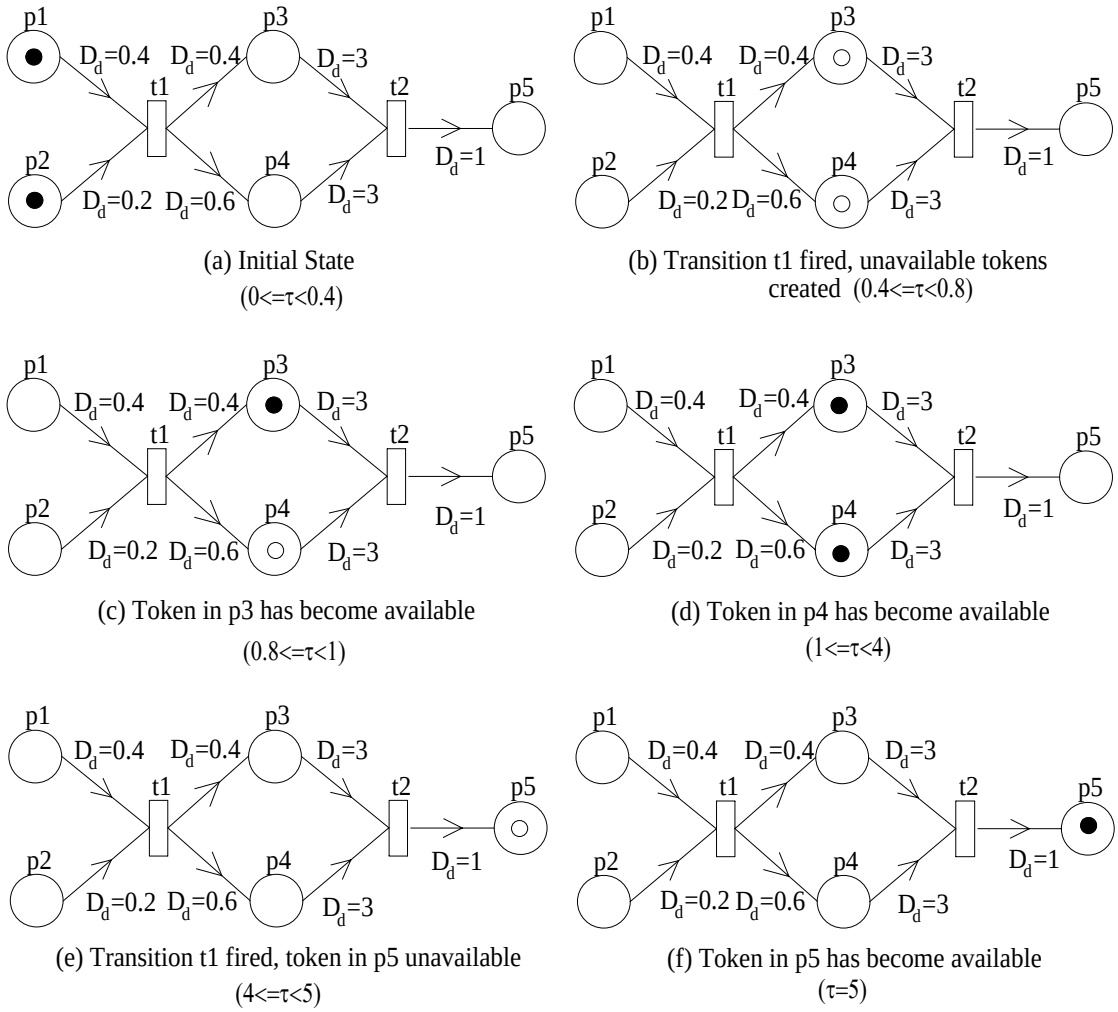


Figure 6.1: Example STPN

The tuple representation for the STPN, as defined for other TPNs in Chapter 3, is given by

STPN = {P, T, W, D, M_0}.

In this definition P, T and W are as defined earlier. The duration functional D is defined on both the input and output arcs. That is, D(•) defines the duration function of a specified arc. In the case of the input arc, D(p, t) defines the enabling duration function for the input arc (p, t). For the output arc (t, p), D(t, p) defines the holding duration function of the W(t, p) tokens created in place p by the firing of transition t. Also for all $p \in P$ and $t \in T$ where W(t, p) = 0, define D(t, p)(τ) as the improper distribution function given by D(t, p)(τ) := 0 for all finite τ. Similarly for all $p \in P$ and $t \in T$ with W(p, t) = 0, define D(p, t)(τ) as the improper distribution function given by D(p, t)(τ) := 0 for all finite τ. Assigning durations to arcs gives greatest modelling flexibility. This way transition, place and arc durations can be defined by the same TPN definition. As before M_0 is the initial marking, only now the marking takes a different form.

The marking is made up of three parts: available tokens (^{A}M), unavailable tokens (^{U}M) and remaining enabling times (^{E}M). Each of these takes the same form as the equivalent marking functions defined earlier. That is, ^{U}M is a mapping: $P \rightarrow \{(n, x): n \in Z_0^+ \text{ and } x \in \rho\}$, ^{A}M is a mapping: $P \rightarrow Z_0^+$ and ^{E}M is a mapping: $T \rightarrow \{x: x \in \rho\}$.

The semantics of STPNs can be defined, as with the other TPNs, using the pentuple definition. The semantic definition given here actually relates only to strong firings. The reason for this will be given later. Alternative firing rules, such as those mentioned in Section 3.2 are not considered here. The number of fresh enablings of transition $t \in T$ is given by

$$\min_{\text{for all } p \in {}^\bullet t} \left(\left\lfloor \frac{{}^{A}M(p)}{W(p,t)} \right\rfloor - \left| {}^{E}M(t) \right| \right).$$

For each fresh enabling an enabling time for the transition must be generated. Thus if there are n fresh enablings then n enabling times must be generated[11]. Define $y_{p,t}$, generated from the duration function D(p, t), as the enabling time associated with the arc (p, t). If transition t is freshly enabled at $\tau = x$, then

${}^{E}M'(t) = {}^{E}M(t) \cup \{x+y\}$, where $y = \max\{y_{p,t}: \text{for all } p \in {}^\bullet t\}$.

If transition t is scheduled to fire at time x then

for all $p \in {}^\bullet t$, ${}^{A}M'(p) = {}^{A}M(p) - W(p, t)$,

${}^{E}M(t) = {}^{E}M(t) \backslash \{x\}$ and

for all $p \in t^\bullet$, ${}^{U}M(p) = {}^{U}M'(p) \cup \{(W(t, p), x+y)\}$, where y is the holding duration of the arc (t, p) and is generated using D(t, p).

As discussed earlier there is a need to recheck the enabled transitions when a transition fires in case a conflicting transition has become disabled. In the case of the STPNs there is no need for an extra step to be defined as created tokens take the form of unavailable tokens in the new marking. Making a simple application of the enabling rule sufficient to determine which transitions may now have fewer enablings than before.

The other step in the semantics of a STPN occurs when unavailable tokens become available. If $\{(n, x)\} \in {}^{U}M(p)$ then at $\tau = x$

${}^{A}M'(p) = {}^{A}M(p) + n$ and

[11] In the case of interval delays this means generating n intervals.

${}^{U}M'(p) = {}^{U}M(p)\backslash\{(n, x)\}$.

This definition allows for the case of concurrent enabling and firings of the same transition. Both the process of firing a transition and that of a token becoming available will be referred to as being part of the firing process of a STPN.

As with the previously defined semantics there are still a number of issues. Due to the concurrent enablings of the same transition a disabling selection policy needs to be considered. There is a need for a memory policy and possibly a preselection policy. The final consideration is which firing operation takes precedence. This has already been discussed for transitions with zero enabling times, in which case a preselection policy for the transitions is specified. However, now a preselection policy must be specified to determine if tokens become available before transitions fire or vice versa. Thus, if a transition is to fire at the same time as a token becomes available, which event goes first?

As an example of how a preselection policy affects a STPN, consider the PN in Figure 2.1 where time durations are assigned to the arcs. If the marking was ${}^{A}M(p3) = 1$, ${}^{U}M(p2) = \{(1, 2)\}$, ${}^{E}M(t3) = \{2\}$ and t2 has a zero enabling time, then at $\tau = 2$ transition t3 is scheduled to fire and the token in place p2 is scheduled to become available. If transitions fire first, then t3 fires and the STPN reaches an absorbing state once the tokens become available. However, if tokens become available first then transition t2 could fire before t3 and so the same state is not reached.

Once these three selection policies (preselection, disabling and memory) have been specified the semantics of the STPN are fully defined.

Now consider the representation of existing TPN structures using the STPN. To form one of the previously defined TPNs using the STPN, restrictions must be placed on the form of the duration function D and in some cases minor changes to the STPN semantics must be made.

6.2 Representing Holding Durations

With HTPNs the enabling durations selected are always zero, that is, for all $t \in T$ and $p \in P$ $D(p, t)(\tau) = 1$ for all $\tau \geq 0$. If holding durations are assigned to output arcs, then this is the only restriction that is required. When holding durations are assigned to transitions a further two restrictions apply, as specified below.

Restriction A

All the output arcs from a given transition have the same duration function. Formally, for all $t \in T$ and for all $p, p' \in t^{\bullet}$, $D(t, p) = D(t, p')$.

Restriction B

Each time a transition fires, the holding durations of the tokens created by the output arcs from it are the same for all the created tokens and calculated from a single sampling of the relevant duration function. That is, for all $t \in T$ and for all $p \in t^{\bullet}$, $D(t, p)$ is only sampled once each time t fires.

In the case of places with holding durations we get one restriction that is similar to restriction A. However, now each output arc to the same place has the same duration function. This can be expressed as , for all $p \in P$ and for all $t, t' \in p^{\bullet}$, $D(t, p) = D(t', p)$.

6.3 Representing Enabling Durations

For ETPNs the holding durations are always zero, that is, for all $t \in T$ and $p \in t^{\bullet}$, $D(p, t)(\tau) = 1$ for all $\tau \geq 0$. This is the only restriction required when enabling durations are associated with input arcs. For transition enabling durations, restrictions C and D below must hold. These restrictions are similar to restrictions A and B but relate to the input arcs instead of the output arcs.

Restriction C

All the input arcs from a given transition have the same duration function. Formally, for all $t \in T$ and for all $p, p' \in {}^{\bullet}t$, $D(p, t) = D(p', t)$.

Restriction D

Each time an enabling duration is determined for a transition, the enabling durations of the input arcs to the transition are all the same and are calculated by a single sampling of this duration function. That is, for all $t \in T$ and for all $p \in {}^{\bullet}t$, $D(p, t)$ is only sampled once for each enabling of t.

6.4 Representing Transition Enabling and Holding Durations

When enabling and holding durations are assigned to transitions, restrictions A, B, C, and D hold.

6.5 Revisiting Some PN Definitions

This section reconsiders some of the definitions given for PNs in Chapter 2. Those that have not changed have not be included in this discussion.

Definition 2.7': Firing sequence

Any possible sequence of transition firings is called a firing sequence.

This definition is the same as for PNs. However, now a firing sequence can include a combination of both transitions firing and tokens becoming available. Each element in the firing sequence is expressed by the duration function that caused it. Thus the firing sequence shown in Figure 6.1 would be expressed as {D(p1, t1), D(t1, p3), D(t1, p4), D(p3, t2), D(t2, p5)}. Note that, it is the largest time that determines the entry in the firing sequence. So in the firing of transition t1 only the duration relating to the arc (p1, t1) was expressed as this timing was greater than that of the arc (p2, t1).

Definition 2.8': Execution Sequence

An *execution sequence* is a firing sequence which includes not only the firing processes but the markings created as well.

As with the firing sequence definition this is the same as that given originally. The point to note here is that the firing process now involves both the firing of transitions and the process of tokens becoming available.

Definition 2.9': Immediately Reachability

The marking M' is immediately reachable from M if either

t is enabled by M and for all $p \in P$, ${}^{A}M'(p) = {}^{A}M(p) - W(p, t)$,

${}^{U}M'(p) = {}^{U}M(p) \cup (W(t, p), y)$ and ${}^{E}M'(t) = {}^{E}M(t) \backslash \{y\}$,

or

$(n, x) \in {}^{U}M(p)$ and ${}^{U}M'(p) = {}^{U}M(p) \backslash \{(n, x)\}$, ${}^{A}M'(p) = {}^{A}M(p) + n$ and ${}^{E}M'$ is given by adding an enabling to ${}^{E}M$ for each fresh enabling resulting from the placing of n tokens in place p.

The properties of deadlock, structural conflict and conflict were defined in Section 2.1 in such as way as to allow them to be applied to both TPNs and PNs. Deadlock was defined as the state where none of the steps in the semantics can occur. In the case of PNs this meant no more transitions were enabled, however, for STPNs this means no transitions are enabled and there are no unavailable tokens. Structural conflict and conflict were defined to include transitions sharing input places even if some of those transitions did not reduce the number of tokens in the place. Clearly if time is included, the net change in the number of available tokens is what is important. Confusion now can exist not only between firings but also between firing processes. This along with conflict is dealt with by the selection policies.

Definition 2.18': Place invariants

The set of values yi form the *place invariants* for a TPN if for all $M \in [M0\rangle$

$$\sum_{pi \in P} yi \left({}^{U}M(pi) + {}^{A}M(pi) \right) = c .$$

6.6 Analysis

A major question still to be determined for STPNs is how performance analysis can be carried out. Clearly it is possible to apply those existing analysis technique applicable to a given subclass of TPNs to their STPN representation. Also using the semantics defined above it is possible to generate the model's state space, that is, its reachability graph.

In generating such a state space it should be noted that after the firing of each transition, ${}^{E}M$ should be checked to see if any current enabling has been disabled by the recent firing. For a given marking the total number of enablings needing to be removed is given by $\left|{}^{E}M(t)\right| - \left|{}^{E}M'(t)\right|$. The checking of ${}^{E}M$ before this final stage is done to ensure that loops result in the disabling of other transitions that share them as input places. If this were not the case, then loops with zero durations on their input and output arcs would not result in the above mentioned disablings. This then becomes part of the firing process of the transition.

A new node in the reachability graph should be defined each time a transition fires or a token becomes available. Even those markings that exist for zero time should be represented by a node in the reachability graph. Such states can be removed later. The arcs between nodes are labelled with the duration function that brought about the change. By using the procedure outlined above the state space of the STPN can be defined in the form of its reachability graph.

Figure 6.2 shows the reachability graph of the STPN in Figure 6.1. This reachability graph is more general as it does not use the specified deterministic durations on the arcs of the STPN in Figure 6.1. Strictly speaking the firing sequence {Max(D(p1, t1), D(p2, t1)), D(t1, p4),

D(t1, p3) - D(t1, p4)} is not possible in Figure 6.1 since D(t1, p3) < D(t1, p4). However, the more general form was used to show how complex the reachability graph has become by the introduction of durations on arcs. Also if D(t1, p3) = D(t1, p4), which is not possible for the duration functions defined in Figure 6.1, then these tokens become available concurrently.

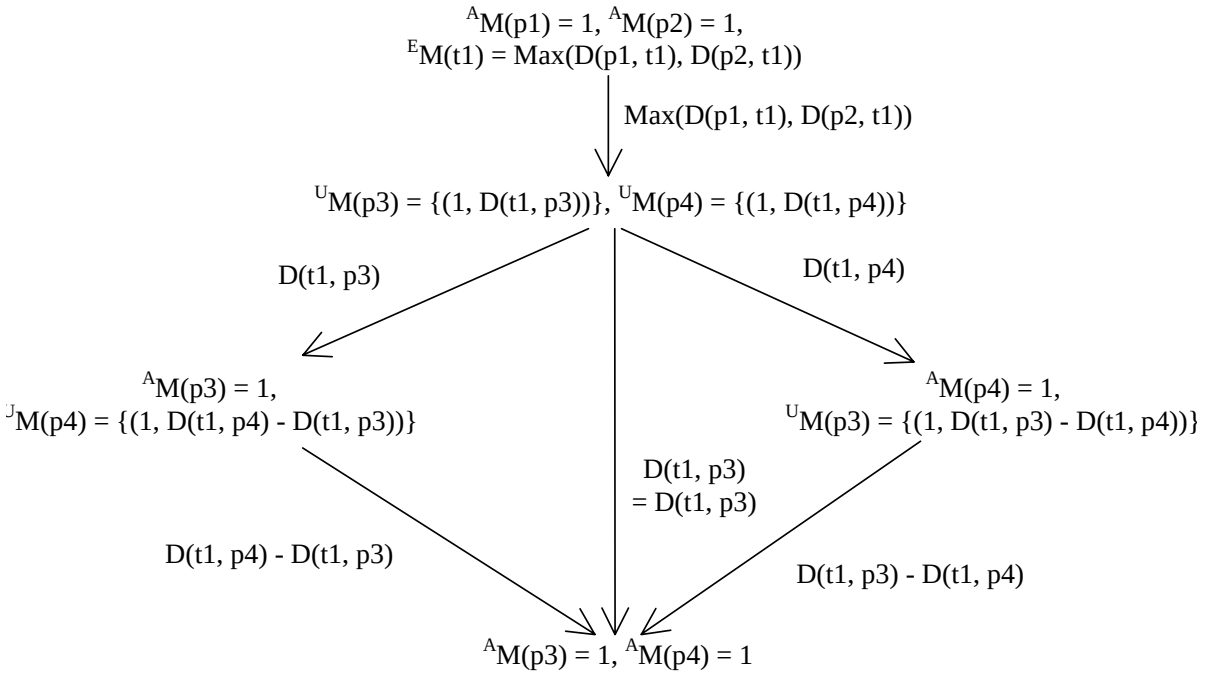


Figure 6.2: Reachability graph of the STPN in Figure 6.1

Even for this simple example with no conflict the state space has been more than doubled when compared to the state space of the non-timed PN. The state space explosion is even worse in the case of the PN shown in Figure 2.1 with durations assigned to its arcs. In this case when enabling memory is used the state space expands from the 5 states shown in Figure 2.2 to 36 states. As with the example given above this may be reduced depending on the choice of durations functions.

An alternative analysis technique is that of simulation. To make any such simulation more efficient, a number of changes should be made to how the above semantics should be applied. The main issue is the checking of the enabling of transitions. Instead of checking all the transitions in the net, it is more efficient to check for fresh enablings only for those transitions whose input places have had newly available tokens placed in them. That is, when unavailable tokens become available in place p, for all $t \in T$ such that $p \in {}^{\bullet}t$, the transition t is freshly enabled n times, where n is given by

$$\min_{\text{for all } p \in {}^{\bullet}t} \left(\left\lfloor \frac{{}^{P}M(p)}{W(p,t)} \right\rfloor - \left| {}^{E}M(t) \right| \right).$$

The second step in the enabling process is unchanged. There is still the need to initially check all the transitions, but after that one needs only to check transitions in whose input places have experienced a change in the number of available tokens. The other change that can be made to make things more efficient is to only check those transitions in structural conflict with the fired transition for disenabling. Technically it is actually better to check only the transitions in conflict with the fired transition. However, as stated earlier determining such conflict sets can be difficult without the complete state space.

CHAPTER 7 REWARDS IN TIMED PETRI NETS

In the early 1990s the idea of rewards was introduced into Petri nets in [52]. This was a concept that does not seem to have been picked up in the literature. This may be because, in general, the measure of a system's performance is time and so there is not a great demand for the more general use of rewards. However, this is not the case here. There is interest not only in time but also in the reward (cost) of the system.

There are three reward functions defined in [52]. The first allocates a reward dependent on the time spent in a given marking. The second allocates a reward once a transition fires. Finally a reward can be gained from the stochastic process as a whole. Of these three rewards only the first two can be easily represented by the definition given here.

A reward STPN (RSTPN) can be represented by the 8-tuple

$$RSTPN = \{P, T, W, D, C, M_0\}.$$

In this definition P, T, W, D and M_0 are as defined earlier. The functional C is defined for each arc in the same way as the duration functional and gives the function that determines the reward gained by the firing of the transition associated with the arc in question. The reward function can be a function of time. Thus in the case of the input arc, C(p, t) defines the reward duration function for the input arc (p, t). Let $C(p, t)(\tau) = r_{p\,t}(\tau)$, which defines the reward gained from the input place p when transition t fires after being enabled for time τ. For the output arc (t, p), $C(t, p)(\tau)$ $(=r_{t\,p}(\tau))$ defines the reward gained by the holding of W(t, p) tokens in place p for time τ by the firing of transition t. Also for all $p \in P$ and $t \in T$ where $W(\bullet) = 0$, define $r(\bullet)(\tau) := 0$ for all τ.

Above it was stated that the reward could be time dependent. However, it may also be time independent, giving a fixed reward for each removal or creation of a token. In addition, it may be relevant at times to relate the rewards to transitions instead of actual input arcs. In fact under the semantics defined in Chapter 6 this would be more logical than assigning it to arcs, as transition enabling time is determined by taking the maximum of input arc enabling times.

The basic idea of including rewards to the STPN structure is not complex. It is just another element of the net. It is during the analysis that it has its greatest impact, where the reward gained must be calculated. An example of the types of measures that can be calculated using rewards will be shown in Part II.

CHAPTER 8 PART I CONCLUSIONS

In [64] it is stated that the advantage of using ETPNs with time on the input arcs is that models with this time representation achieve a closer resemblance to the real system than using transition times, making them easier to construct and understand. Although in many models this may not be true, the principle behind this statement is important. Every good modeller knows that when representing a system the choice of modelling technique should depend on the system being modelled and the purpose of the model. It is these factors that should determine the time representation chosen by a modeller. As shown earlier it is possible to represent a HTPN as an ETPN, but even in the simple example given in Figure 3.5 the model size increases considerably. Another example of this is the ETPN in Figure 1 of [44]. As a HTPN this model would have its number of nodes reduced by 40%, its number of arcs reduced by 30% and the number of states in its state space reduced by 30%. Such a transformation may also obscure the link between the real system and the model. Despite this there is a need to allow the different time representations to be understood by all researchers. Ideally, what is required is a transformation that takes ETPNs and converts them to HTPNs similar to the one defined for converting HTPNs to ETPNs. However, this is not possible. The best possible outcome is to introduce a standard form that can be understood by all researchers but allows either representation to be used. In the case of time related to transitions this has, in a way, been done, as outlined in Section 3.4. This does not have the flexibility of allowing for the cases where time is not associated with transitions. This is one of the reasons for my definition of STPN representation. STPNs give modellers a common definition to base their models on. It also gives a common definition that can be used in computer tools and analysis techniques being developed. It should be noted that the STPN definition gives an overarching syntactical representation. The semantic definition given does

not have the flexibility of some other definitions found in the literature. Issues such as the enabling of arcs (as introduced by Diaz and Sénac in [64]) and strong and weak firings have not been included here. However, these concepts can be introduced if required, although as outlined in Section 3.2 the use of weak firings rules is not recommended.

The confusion created by these multiple representations can be seen in [164]. In this paper it is stated that representing time in places or transitions does not give equivalent modelling power. This, as the authors state, is in contradiction to many other papers which state the equivalence between these two representations. To demonstrate the difference in modelling power the authors use an example. Unfortunately the problem here is that the transition timings used in the example are enabling times while the place timings are holding times. Clearly, as illustrated in this Part of the thesis these two representations are not equivalent. What previous papers have expressed as being equivalent is the representation of holding times in places or transitions.

The problem of a lack of a standard has been recognised by the PN research community and will hopefully be overcome with the development of the PN standard ([139]). Work on this standard is still in its infancy but hopefully the STPN can be incorporated as the mechanism to represent time. Putting this issue aside, PNs are a useful way of representing discrete event systems. They are particularly useful in representing conflict, concurrence, synchronisation and resource sharing. The introduction of time has meant that PN models can now also be used for performance analysis.

The STPN is not only a standard representation in which all other TPN representations can be placed, it is also the most flexible TPN representation available. It allows for the holding of

resources during an activity, in the same way as HTPNs, as well as representing the interruption of activities, as in ETPNs. It also allows time to be assigned to any part of the PN or a variety of different parts, that is, time can be assigned to places in one part, transitions in another and arcs in yet another.

Another advantage of the STPN is that all analysis techniques developed under the super-class can be applied to the sub-classes and those developed for the sub-classes may be more easily extended to other sub-classes due to the common representation. A major issue that still needs to be considered in much greater detail is the issue of analysis, not only the consideration of new analysis techniques, but also how existing methods can be applied to the new structure. The next chapter outlines a direct analysis technique for a sub-class of STPNs.

PART II

EVALUATION OF

DECISION PROCESSES

In the modelling of any system a certain amount of abstraction is required, often necessitating probabilistic ingredients. This is particularly true in the case of modelling human (or non-human) decision processes as such systems are too complex to represent completely. Also, realistic modelling requires uncertainty in the way a model reacts to a given situation. Thus for some initial state there could be a number of possible final or absorbing states that may be reached. This Part considers the calculation of the probability of reaching a given absorbing state and the mean reward (cost) until absorption in the given state. Unlike much other PN research ([4], [53] and [76] to note but a few) the aim here is to look at transient behaviour. It should be noted that traditional transient analysis, as in [50], [73] and [114], is more concerned with the distribution of states before an equilibrium is reached. Here the interest is in absorption, or more generally the first-hitting properties for states that may have a probability less than one of being reached. The modelling and analysis approach outlined is planned to be applied in two areas: course of action analysis and decision process model analysis and validation.

In considering possible courses of action it is important that the overall cost of each action is considered, that is, the *mean conditional reward*. Thus the theories defined in this paper can be used to determine the “best” course of action. The use of rewards means that any measure

can be used to determine what “best” means for given situations. For example, the rewards may relate to the time taken to conduct a course of action, the resources used in a course of action, the possible risk to personnel to carry out a course of action or any combination of these.

When developing models of decision processes either for inclusion in a larger model or to be directly analysed, two important measures are the probability of reaching a given decision and the mean time to reach that decision. These values are particularly useful to any subject matter adviser who might be validating a model as they can be related back to the real system without any knowledge of the mathematical detail of the model. They are also valuable measures in analysing any decision process directly. The use of these types of measures to validate decision processes is a further extension of the validation methods outlined in [28].

This Part specifies what is meant by a decision process and shows how it can be modelled and analysed using PNs. The structure of this part is as follows:

- Chapter 9 defines a command and control decision process and how it can be modelled using a STPN.
- Chapter 10 gives a general review of current PN analysis techniques is given. This makes clear that the area is still the subject of considerable ongoing research
- Chapter 11 introduces some basic concepts of Markov chains and in some cases improvements are made to current theory. Also the concept of conditional first-hitting rewards is introduced, which is not currently found the Markov chain literature.
- In Chapter 12 the STPN decision process building block is directly analysed to determine the absorption probabilities and mean conditional absorption rewards for this representation. An algorithm is defined to show how the analysis techniques developed

for the STPN decision process building block can be used to analyse a complete PN made up of these building blocks.

- Chapter 13 considers some extensions of the basic building block are considered.

CHAPTER 9 DECISION PROCESS MODEL

As an example of a decision process consider the case of a friendly military unit which is given the mission of finding and destroying a particular enemy unit. Initially the friendly unit seeks to find the designated target. Suppose there are four possible alternative outcomes of the searching:

- the friendly unit is found and neutralised by the enemy,
- the target is located,
- the desired target is not found or
- the desired target is not observed so the unit decides to rest before continuing its search.

Once the enemy target is located a battle begins. There are three possible outcomes of such a battle:

- the friendly unit is neutralised,
- the enemy unit is neutralised or
- no resolution is reached and the battle continues.

Figure 9.1 shows a PN model of this decision process. Each of the eight component processes is represented by a transition in the PN. Table 9.1 indicates which transition corresponds to which process. With the initial marking shown in Figure 9.1 there are three possible absorbing states (a token in p2, p5 or p6). These relate respectively to the three possible outcomes of the system: the friendly unit is neutralised before it locates its target; the friendly unit is neutralised by the target during the battle; or the mission is successfully completed. That is, the enemy unit is neutralised.

The PN in Figure 9.1 is an example of the type of model that will be considered in this thesis. This example will be used to illustrate the analysis techniques that are developed to calculate absorption probabilities and mean conditional absorption rewards.

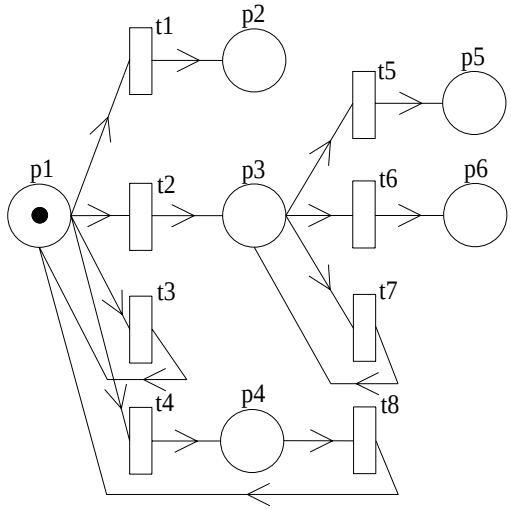


Figure 9.1: Sample decision process

The basic building block that underlies the decision processes considered here is made up of two parts. One comprises a number of decisions and actions that take the system from its initial state back to the initial state. In the example above such a sequence of events occurs when the friendly unit is unable to find the target so decides to rest before continuing to search. The second part of the decision process involves a set of possible decisions (events) regarding actions that move the system one step closer to a final decision (an absorbing marking in PN terms). The location of the target by the friendly unit is an example of an action that shifts the systems towards an absorbing marking.

Table 9.1: Descriptions of transitions in Figure 9.1

Transition	Description
t1	The friendly unit is located and neutralised by the enemy.
t2	The target is located and the friendly unit prepares to attack.
t3	The desired target is not located and the friendly unit continues to search.
t4	The desired target is not located so the unit decides to rest before continuing its search.
t5	The friendly unit is neutralised during the battle.
t6	The enemy unit is neutralised during the battle.
t7	No resolution is reach and the battle continues.
t8	The unit rests before returning to the search.

The decision process building block is made up of a series of simple circuits and a number of transitions that lead to possible absorbing states. Figure 9.2 shows the decision process building block. The initial state of this decision process has a token is in place I and is referred to as state I. The system comprises N circuits each starting with a transition c_i, where i = 1 to N. The simple circuits are referred to as the C part of the system. The circuit which begins with c_i is called the i^{th} circuit. In addition there are L transitions referred to as absorbing transitions. Each absorbing transition leads to an absorbing state of the decision process building block and consequently there are L possible absorbing states. The j^{th} absorbing state is reached when a token is in place A_j, which occurs when the transition s_j fires, where j = 1 to L. The marking with one token in place A_j is referred to as state j. Each of the absorbing states relates to an outcome that can be reached by the decision process. The

absorbing part of the decision process will be referred to as the A part of the system. The place A_j at times will be referred to as an absorbing place j. State I is considered part of the C system. The restrictive building block allows for a wide variety of systems to be modelled and will be generalised later in this Part.

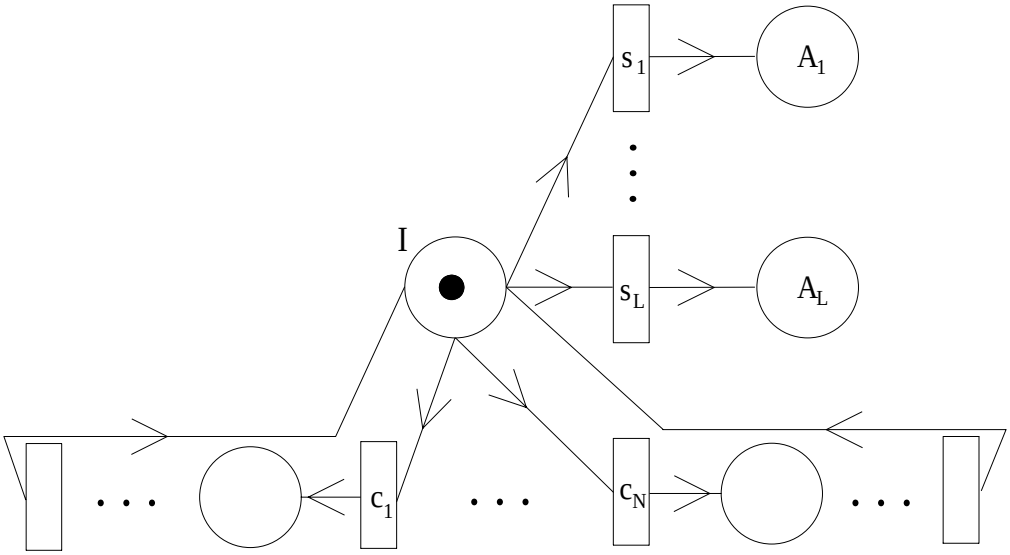
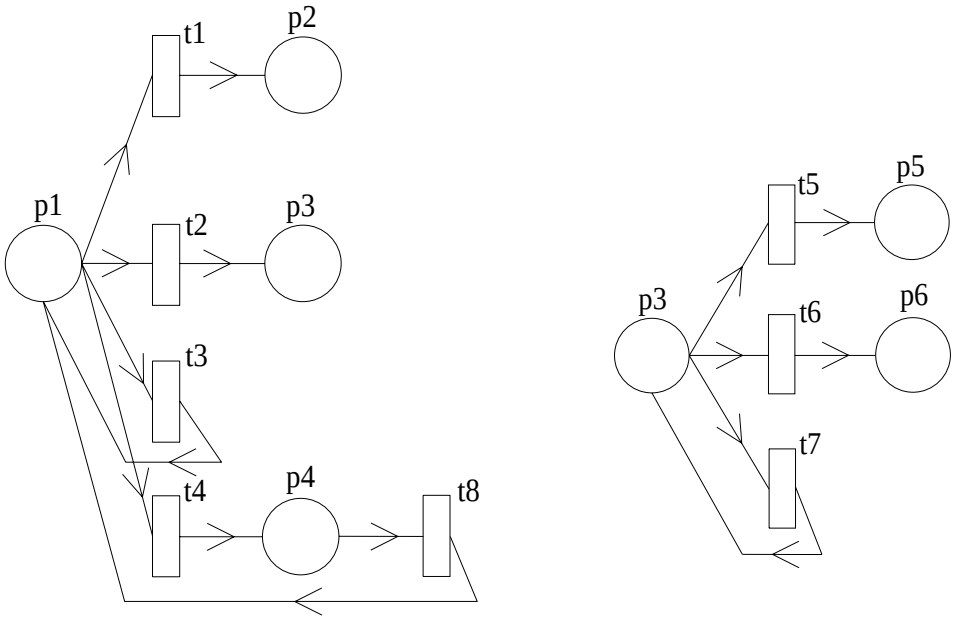


Figure 9.2: The decision process building block

The example given in Figure 9.1 has two building blocks. These are shown separately in Figure 9.3. The initial building block, shown in Figure 9.3(a), comprises two simple circuits, one starting with t3 and the other with t4 so N = 2. There are two absorbing markings (L = 2), one when a token is in place p2 and the other when a token is in place p3. That is, two outcomes can be reached; the friendly unit is neutralised or the target is located. For the second building block shown in Figure 9.3(b) N = 1 and L = 2, since there is one circuit and two absorbing states. One absorbing state corresponds to a token is in place p5, representing the case when the friendly unit is neutralised during the battle. The other absorbing state

occurs when a token is in place p6, representing the case when the target is neutralised during the battle.



(a) Initial building block (b) Final building block

Figure 9.3: Decision process building blocks for Figure 9.1

Developing an analysis technique for the decision process basic building block makes it possible to produce an recursive method of calculating the absorption probability and mean conditional absorption reward for each of the absorbing states of a total system. The techniques developed allow the calculation of the probability that the final marking has one token in place A_j as well as the mean reward gained in reaching this state, given the state is reached. These results can then be used in a recursive manner to analyse a full system.

CHAPTER 10 CURRENT TPN ANALYSIS TECHNIQUES

This chapter gives a brief overview of the current types of analysis carried out on TPNs. It considers structural, behavioural and performance analysis. Although structural and behavioural analysis techniques are important, they are of limited relevance to the analysis in this thesis. This Section will give an overview of these techniques and concentrate on performance analysis.

As with PNs one of the key ways of analysing TPNs is using their corresponding reachability graphs. These can be constructed using the semantics defined in Part I. The key requirement is in the definition of a state and the transitions between states. In papers [1], [24], [25], [27], [117], [151] and [153] ways of generating TPN reachability graphs are defined.

10.1 Structural and Behavioural Analysis

When PNs were first introduced they were used to carry out analysis on the structure and behaviour of a system, that is, search for structural and behavioural properties such as deadlocks and T-invariants. The introduction of time into PNs has meant that performance analysis can be carried out using PN models. Therefore, most of the research on TPNs has been in this area and most structural and behavioural analysis is carried out using the techniques developed for PNs without time. Although these techniques are still valid the interpretation of such analysis is important. For example, a deadlock predicted by PN techniques may not exist in the corresponding TPN.

Some researchers have worked on problems of structural analysis of TPNs. In [166] a method of converting a TPN with deterministic firing durations into a PN is described. Thus the results of PN structural analysis can be applied to the converted PN. In [167] the underlying timed process of TPNs with deterministic firing durations is considered. The underlying timed process representation is then used to study structural and behavioural properties of the TPN. In [25], [116] and [119] the reachability graph is used to carry out this type of analysis, while [59] and [171] develop theorems for TPNs which define when given PN properties hold. In [117] the issue is discussed of which PN properties also hold for the corresponding TPN with deterministic firing times. These papers give the general feel for the work done on structural and behavioural analysis. That is, work in this area is either looking at links between structural and behavioural analysis of PNs or analysis of the reachability graph.

10.2 Performance Analysis

The main advantages of including time in PNs is that it allows for performance analysis. The work in this area is vast and this section will only outline the main areas of this work, giving references to papers where more details can be found.

The first type of performance analysis that will be considered is that of performance bounds. Generally performance bounds are used in conjunction with interval durations, as in [1], [25] and [89]. This is done by generating a reachability graph of the TPN. This reachability graph contains not only the obtainable markings, but also the minimum and maximum times it takes to reach the marking, allowing the analyst to read off these values and identify problems such as underutilised resources and critical paths. In [2] performance bounds are used to the analysis of railway stations to determine station operating schedules, minimise train waiting durations and determine train platform assignments. A problem with this type of analysis is

that it depends on the time intervals assigned to the transitions. If the intervals are large then the bounds quickly become large and make any overall performance bounds meaningless.

A popular way of analysing PNs with stochastic timings is by constructing reachability graphs which are Markovian or semi-Markovian processes. This was originally done with ETPNs, where the time an event takes is considered to have an exponential distribution (see [78], [79], [120], [128] and [159]). Under certain restrictions this work has been extended to allow this type of analysis on more general distributions (see [5], [6] and [8]). The main problem with this is that it reduces the class of TPNs that can be analysed. In [4], [50], [73], [77], [113] and [114] deterministic and exponential enabling durations, are used, where the number of concurrently enabled deterministic transitions is restricted to one so that the reachability graph is still Markovian. This has been extended, in papers [56], [72] and [105], where the restriction on the number of concurrently enabled non-exponential transitions is loosened. In [18] generally distributed enabling durations are considered, however, the reachability graph of these nets is only semi-Markovian under given conditions on the enabling of transitions with general duration functions. In [141] concurrent enabling of transitions with non-exponential enabling generation functions is allowed for but only if the tranistions become enabled concurrently. In [53] the resampling execution policy, which is not always a realistic representation of a system (see Section 5.2), is used to ensure that the reachability graph of a TPN with deterministic and exponential enabling durations is a Markov process. Another way used to introduce non-exponential distributions into ETPNs is to model these distributions within the PN using exponential distributions as in [8] and [173]. The above references show the type of work being done in the area of Markovian reachability analysis. A feature of this approach is that (for better or worse) a stochastic process can be modelled by someone who is not fully aware of its complexities. The information obtainable from such

analysis is not restricted to equilibrium values (references [8], [9], [42], [48], [58], [78], [82], [87], [106], [112], [113], [161] and [172] are a sample of TPN equilibrium analysis), although these are by far the most popular measures considered. Recently there has been some research into the area of transient analysis, for example [50], [56], [73] and [114], which look at the system behaviour before equilibrium is reached.

If the state space is chosen in such a way that the process on it is non-Markovian it is still possible to carry out some Markovian analysis. There are three ways in the literature of dealing with state spaces that are not Markovian. The first is the use of state space aggregation, see [4], [18], [40] and [53]. This is where the state space is partitioned in such a way that the process on the partitions are memoryless. The complication here is in defining the transition rates between the partitions. Alternatively supplementary variables can be used (see [104]). In this case the states of the reachability graph are expanded to include all the required information so that each state is memoryless. However, this leads to a very complex state space which is very difficult to analyse. In [104], [105] and [115] this problem is overcome using numerical techniques. The third method is to use Markov regenerative or renewal processes as in [67], [75], [87] and [141]. This is case the measures of interest are such that they can be calculated from a state space that regenerates at given points. An issue all these techniques is that they are applicable only to a sub-class of TPN models. In [54] a review of some different ETPNs is considered. This paper discusses how the increase in modelling flexibility decreases ease of analysis.

The main problem with both the above mentioned analysis techniques is that as the models become more complicated their state spaces, that is, the reachability graphs, grow rapidly making their generation time consuming and any analysis more difficult. The problem of state

space explosions is one that affects all techniques that use the reachability graph for analysis. In [40], [76] and [92] the problem of the size of the state space is dealt with using numerical techniques. This is taken one step further in [115] where numerical techniques are defined over a distributed network of computers. Papers [51] and [178], approach this problem by carrying out modular analysis, while in [62] a combination of modular and numerical analysis to overcome the state space explosion problem. In [151] the generation of the reachability graph is restricted to those states with a sufficiently high probability of occurring. The probability of reaching a given state is generated simultaneously with the reachability graph and those states with a probability lower than the specified threshold probability do not have their successive states generated. This increases the effectiveness of the analysis but at the cost of accuracy and potentially leaves the analysis open to missing important states that the process may reach from initial states of low probability. In [97], [146] and [156] reduction rules are defined to reduce the TPN and thus reduce the state space generated. Although these techniques do appear to go some way toward helping with state space explosions, what they are really doing is increasing the number of states that can be analysed without overcoming the basic problem.

Some direct analysis techniques have been derived for a restricted class of PNs. In [16] analysis is carried out directly on a sub-class of HTPNs. In this case the class of PNs has been reduced considerably to allow for analysis that does not require the generation of the reachability graph. To overcome the problem of state space explosion, [43] decomposes a sub-class of ETPNs so that an iterative technique can be applied directly to the smaller ETPNs in approximating some performance measures. Another area of work involves the calculation of performance bounds, as in [13], [42], [48], [107], [172] and [106], where the net structure is used to set up a system of equations that must hold for equilibrium to be

reached. These equations are then solved using linear programming techniques to determine performance bounds.

As with most modelling techniques, analytic performance analysis of TPNs can be carried out only under restrictive conditions. This is not to say these techniques are not useful, only that they do not provide the solution to every problem. If the system being modelled can be represented by a given class of TPNs, then the appropriate technique can be applied. If this is not the case the only alternative is to use simulation. Due to the discrete nature and tuple structure of TPNs they can be easily simulated using computers. Many papers, for example [109], [110], [122], [130] and [131], use simulation as an analysis tool for TPNs. There exist a number of packages that allow a user to draw a TPN and then simulate the resulting net. One such package is described in [123]. In [10] a technique for applying parallel simulation methods to TPNs is described, thus allowing more rapid simulation of large nets. Also [65] looks at the use of structural properties to improve the speed of simulations by improving the management of the scheduling of events. A problem with simulation is that any set of simulation runs may not search the complete state space so the results may not be conclusive.

Unfortunately none of the above approaches has considered the problem of finding the mean reward (cost) to reach a given state for the first time, given the state is reached. That is, the *mean conditional first-passage reward*. This is one of the key questions that will be considered in this thesis. There was some promising work done at Duke University in the late 80's but this was not developed beyond the basic definition stage. This work is discussed further in Section 11.4.

One lesson that is clear from the TPN performance literature is that any new analysis is generally applicable only to a sub-class of TPNs. This is also the case in the analysis presented here. Unfortunately an extensive search of the literature located no papers discussing methods of calculating the mean conditional first-passage reward either directly on a TPN or on a Markov process defined from the state space of a TPN.

CHAPTER 11 MARKOVIAN ANALYSIS

The majority of TPN analysis is achieved using the state space. This is due to the fact that direct analysis techniques are generally restricted to special classes of TPNs. In the case of TPNs with stochastic timings, state space analysis generally means analysing the underlying Markov process. The question of how Markovian analysis can be used to determine the absorption probability and mean conditional first-passage (first-hitting) reward (cost) is the primary focus of this chapter.

The literature pertaining to these ideas utilises a variety of forms. Many of the key ideas already appear in [90], though predicated to determining necessary and sufficient conditions for transience or recurrence of communicating classes in discrete-time Markov chains. In [83] the question of determining the probability that, for an initial transient state, the process stays forever amongst the transient states is addressed. They proceed to the consideration of absorption probabilities in persistent states and the associated absorption times. Papers [57] and [91] make use of notions from potential theory for a general treatment. In [69] and [127] simular results to the above references are given as is a brief consideration of rewards in the case of discrete-time Markov processes.

This chapter begins with some of the basic concepts of Markov chains. A more detailed and thorough treatment of the Markov structures on which we build is given in the above mentioned references as well as [94], [129] and [150].

11.1 Discrete-Time Markov Chains

Definition 11.1: Discrete-time (homogeneous) Markov chain

A *discrete-time (homogeneous) Markov chain* is a process with the following four characteristics:

1. A countable number of states.
2. Changes of state occur only at a countable number of time points, which without loss of generality we can label 0, 1, 2, …. For definiteness we define X_n to be the state of the process just before time n.
3. $P(X_{n+1} = j \mid X_n = i, X_{n-1} = h, \ldots, X_0 = k) = P(X_{n+1} = j \mid X_n = i)$ (memoryless or Markovian property).
4. $P(X_{n+1} = j \mid X_n = i) = P(X_{m+1} = j \mid X_m = i)$, that is, probability of changing state is independent of the absolute time.

Thus a *discrete-time (homogeneous) Markov chain* is a process with a countable set, S, of states for which, given the history of the process to date, the probability of moving to a given state is dependent only on the current state. The fourth condition is referred to as the time-homogeneity property. In this thesis all the discrete-time Markov chains are time-homogeneous and so for brevity will be referred to as discrete-time Markov chains instead of discrete-time homogeneous Markov chains.

Definition 11.2: Transition Probabilities

The probability $P(X_{n+1} = j \mid X_n = i)$ is called the *transition probability* from i to j and is written $p_{i\,j}$.

The statistical behaviour of a discrete-time Markov chain is defined by its initial state, X_0, and the transition probabilities, p_{ij}.

Definition 11.3: Transition matrix

The matrix **p** with elements p_{ij} which are the transition probabilities of a Markov process is defined as the *transition matrix* for the Markov process.

Definition 11.4: Access

The state i is said to have access to state j if there is a non-null probability of reaching state j from state i in a positive number of steps.

In this and the following chapter let κ be a nonempty set of states in a discrete-time Markov chain and B the set of states which have access to κ but do not belong to κ. Suppose B is also nonempty. For $i \in B$, denote by a_i the probability that for initial state i the process ever reaches κ.

Theorem 11.1

The probabilities a_i ($i \in B$) provide the minimal nonnegative solution to the equations

$$a_i = \sum_{j \in \kappa} p_{ij} + \sum_{j \in B} p_{ij} a_j \quad (i \in B).$$

Proof:

That a_i satisfies the equations follows from the theorem of total probability applied to the event of reaching κ from initial state i, conditioning on the outcome of the first step.

For minimality, let (b_i) be any nonnegative solution to the equations in Theorem 11.1. Then

$$b_r \geq \sum_{j\in\kappa} p_{rj} \quad (r\in B). \qquad \text{(eq1)}$$

Let $(\grave{p}_{ij})$ denote the restriction of (p_{ij}) to B and denote by $\left(p_{ij}^{(n)}\right)$ the matrix of corresponding n-step transition probabilities. Then for each $n\geq 1$

$$b_i \geq \sum_{j\in\kappa} p_{ij} + \sum_{m=1}^{n}\sum_{r\in B} \grave{p}_{ir}^{(m)} \sum_{j\in\kappa} p_{rj} \quad (i\in B). \qquad \text{(eq2)}$$

This follows by mathematical induction. For the basis, note that since

$$b_i = \sum_{j\in\kappa} p_{ij} + \sum_{r\in B} p_{ir}\, b_r\,, \qquad \text{(eq3)}$$

then from (eq1)

$$\begin{aligned} b_i &\geq \sum_{j\in\kappa} p_{ij} + \sum_{r\in B} p_{ir} \sum_{j\in\kappa} p_{rj} \\ &= \sum_{j\in\kappa} p_{ij} + \sum_{m=1}^{1}\sum_{r\in B} \grave{p}_{ir}^{(1)}\, p_{rj} \qquad . \end{aligned}$$

For the inductive step, assume that (eq2) holds for some $n \geq 1$. Then by (eq3) and (eq1)

$$\begin{aligned} b_i &\geq \sum_{j\in\kappa} p_{ij} + \sum_{r\in B} p_{ir} \left(\sum_{j\in\kappa} p_{rj} + \sum_{m=1}^{n}\sum_{s\in B} \grave{p}_{rs}^{(m)} \sum_{j\in\kappa} p_{sj} \right) \\ &= \sum_{j\in\kappa} p_{ij} + \sum_{r\in B} \grave{p}_{ir} \sum_{j\in\kappa} p_{rj} + \sum_{m=1}^{n}\sum_{s\in B} \grave{p}_{is}^{(m+1)} \sum_{j\in\kappa} p_{sj} \\ &= \sum_{j\in\kappa} p_{ij} + \sum_{m=1}^{n+1}\sum_{r\in B} \grave{p}_{ir}^{(m)} \sum_{j\in\kappa} p_{rj} \qquad , \end{aligned}$$

giving the inductive step.

Letting $n \to \infty$ in (eq2) yields

$$b_i \geq \sum_{j\in\kappa} p_{ij} + \sum_{m=1}^{\infty}\sum_{r\in B} \grave{p}_{ir}^{(m)} \sum_{j\in\kappa} p_{rj} \quad (i\in B). \qquad \text{(eq4)}$$

The right-hand side gives the probability of ever reaching κ from i conditioned by the time of first entry, so that (eq4) reads

$$b_i \geq a_i. \qquad \nu$$

It is implicit in this formulation that there may be taboo states entry to which precludes the subsequent entry of the process into κ. If m is such a taboo state then of course $m \notin B \cup \kappa$.

The above version of the basic theorem for hitting probabilities has the advantages of minimising the size of the state space involved in calculations and of removing the need for any boundary conditions.

A corresponding version of the standard first-passage time theorem is the following.

Theorem 11.2

Suppose that $a_i = 1$ for all $i\in B$ and let T_i denote the first-passage time to κ from initial state $i\in B$. Then $(T_i)_{i\in B}$ is the minimal nonnegative solution to the equations

$$T_i = 1 + \sum_{j\in B} p_{ij} T_j \qquad (i\in B).$$

This theorem is inappropriate when $a_i < 1$ for any state $i\in B$. In this situation the first-passage time from i to κ is infinite with positive probability and so $T_i = \infty$. Thus the theorem does not provide a useful first-passage time result if κ is reached from i with probability less than

unity. An appropriate version can be formulated based on only those paths which reach κ. However, this result is subserved by a yet more general result involving rewards, which is detailed below. A reward R_{ij} is associated with the single-step transition from state i to state j. This reward can be an arbitrary function of i and j. This induces for an in general multi-step transition from i to j an aggregated reward equal to the sum of the rewards associated with the individual single-step transitions involved. Define for $i \in B$ R_i as the mean reward associated with first-passage to κ from initial state i, conditional on such passage occurring.

It is useful for the evaluation of the rewards R_i to define the dual as the embedded chain with the one-step transition probabilities

$$U_{ij} := \frac{p_{ij} a_j}{a_i}$$

for $i, j \in B$ and put $\mathbf{U} := (U_{ij})_{i,\, j \in B}$. The entry U_{ij} may be interpreted as the probability of a single-step transition from i to j, conditional on the process eventually entering κ.

Note that for $i \in B$

$$a_i = \sum_{j \in \kappa} p_{ij} + \sum_{j \in B} p_{ij} a_j ,$$

so that

$$\sum_{j \in B} U_{ij} + \sum_{j \in \kappa} \frac{p_{ij}}{a_i} = 1 .$$

This means the matrix **U** is substochastic.

Denote by $U_{ij}^{(n)} = (U^n)_{ij}$ $(n \geq 1)$ the matrix of corresponding n-step conditional transition probabilities. Define

$$U_{ij}^{(0)} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}.$$

Also, define

$$v_i := \sum_{j \in B \cup \kappa} \frac{p_{ij} a_j R_{ij}}{a_i}, \quad (i \in B) \qquad (\textbf{eq5})$$

where

$a_j := 1$ for $j \in \kappa$.

The interpretation of v_i is as the expected reward from the first step conditional on κ being reached in that step or later.

Theorem 11.3

The mean conditional rewards R_i satisfy

$$R_i = v_i + \sum_{j \in B} U_{ij} R_j \quad (i \in B). \qquad (\textbf{eq6})$$

Proof:

Represent by A_i the event that $X_n \in \kappa$ for some $n \geq 1$ and $X_0 = i \in B$. Then by conditioning on the first step

$$R_i = \sum_{j \in \kappa} P(X_1 = j \mid A_i) R_{ij} + \sum_{j \in B} P(X_1 = j \mid A_i)(R_{ij} + R_j). \qquad (\textbf{eq7})$$

For $j \in \kappa$,

$$P(X_1 = j \mid A_i) = \frac{P(\{X_1 = j\} \cap A_i)}{P(A_i)} = \frac{P(X_1 = j \mid X_0 = i)}{P(A_i)} = \frac{p_{ij}}{a_i}. \qquad (\textbf{eq8})$$

For $j \in B$,

$$P(X_1 = j \mid A_i) = \frac{P(\{X_1 = j\} \cap A_i)}{P(A_i)} = \frac{P(\{X_1 = j\} \cap A_j \mid X_0 = i)}{P(A_i)} = \frac{P(X_1 = j \mid X_0 = i)\, P(A_j)}{P(A_i)}.$$

Thus for $j \in B$

$$P(X_1 = j \mid A_i) = \frac{p_{ij} a_j}{a_i} \qquad \textbf{(eq9)}$$

Substituting from equations (eq8) and (eq9) into equation (eq7) gives

$$R_i = \sum_{j \in \kappa} \frac{p_{ij}}{a_i} R_{ij} + \sum_{j \in B} \frac{p_{ij} a_j}{a_i} (R_{ij} + R_j) = \sum_{j \in \kappa} \frac{p_{ij}}{a_i} R_{ij} + \sum_{j \in B} \frac{p_{ij} a_j}{a_i} R_{ij} + \sum_{j \in B} \frac{p_{ij} a_j}{a_i} R_j.$$

Thus

$$R_i = \frac{1}{a_i} \sum_{j \in B \cup \kappa} p_{ij} a_j R_{ij} + \sum_{j \in B} \frac{p_{ij} a_j}{a_i} R_j \,. \qquad \textbf{(eq10)}$$

The desired result follows by substituting from equation (eq5) for the first summation on the right-hand side. ν

In many applications, B contains a finite number of states. Since each state $i \in B$ has access to κ and **U** is substochastic, **I - U** is invertible, where **I** is the $|B| \times |B|$ identity matrix. Also define $\mathbf{R} := (R_i)_{i \in B}$ and $\mathbf{v} := (v_i)_{i \in B}$. Then (eq6) can be expressed as

$$\mathbf{R} = (\mathbf{I} - \mathbf{U})^{-1} \mathbf{v}.$$

In fact, if

$\mathbf{\Lambda} := (\mathrm{diag}(a_i))_{i\in B}$,

and **p'** be the matrix $(p_{ij})_{i,\, j\in B}$ then

$$\mathbf{R} = (\mathbf{I} - \mathbf{\Lambda}^{-1}\,\mathbf{p}'\,\mathbf{\Lambda})^{-1}\,\mathbf{v}. \qquad (\mathbf{eq11})$$

An alternative form that equation (eq6) can take, namely

$$R_i = \sum_{j\in B\cup\kappa} U_{ij}\left(R_{ij} + R_j\right) \ (i\in B),$$

where $R_j := 0$ for $j\in\kappa$. The original form of Theorem 11.3 was presented to allow for the derivation of the matrix form of the result. These forms will be used interchangeably without distinction.

An advantage of looking at rewards is that they are fairly general, allowing factors other than time. For example, a reward may relate to the combination of the time and the resources required. Different single-step transitions may be associated with different rewards. Note also that the reward associated with a transition from i to j can be function of i alone or a function of j alone.

Corollary 11.1

Suppose that $a_i = 1$ for all $i\in B$. Then $(R_i)_{i\in B}$ satisfies the equations

$$R_i = v_i + \sum_{j\in B} p_{ij} R_j \qquad (i\in B).$$

Proof:

This result simply follows by substituting $a_i = 1$, for all $i \in B$, into equation (eq10). ν

This is similar to the general mean first-passage reward as found in references [69] and [127].

A set of matrix equations can be expressed for this case also by adapting equation (eq11) when |B| is finite. When $a_i = 1$ then $\mathbf{\Lambda} = \mathbf{I}$ so equation (eq11) becomes

$$\mathbf{R} = (\mathbf{I} - \mathbf{p}')^{-1}\,\mathbf{v}. \qquad \textbf{(eq12)}$$

Consider now the special case when the first-passage reward is just the total time taken to reach κ. This is given by $R_{ij} = 1$.

Definition 11.5: Mean conditional first-passage time

The *mean conditional first-passage time,* T_i, from state $i \in B$ to the set of states κ is the mean time it takes to go from i to any of the states in κ given such a passage occurs.

Corollary 11.2

The mean conditional first-passage time to κ from initial state $i \in B$ satisfies the equations

$$T_i = 1 + \sum_{j \in B} U_{ij} T_j \quad (i \in B). \qquad \textbf{(eq13)}$$

Proof:

To get this result it must be shown that $v_i = 1$ when $R_{ij} = 1$. Substituting $R_{ij} = 1$ in equation (eq5) gives

$$v_i = \sum_{j \in B \cup \kappa} \frac{p_{ij} a_j}{a_i}.$$

Since $a_i = \sum_{j \in B \cup \kappa} p_{ij} a_j$, the desired result follows. ν

Suppose $|B| < \infty$. Putting $\mathbf{T} = (T_i)_{i \in B}$ and writing **1** for a column vector of length $|B|$ consisting of ones allows (eq13) to be expressed in the form

$$\mathbf{T} = (\mathbf{I} - \Lambda^{-1} \mathbf{p}' \Lambda)^{-1} \mathbf{1}. \qquad (\mathbf{eq14})$$

This result is presented in [35] and [38].

Corollary 11.3 (Theorem 11.2)

Suppose that $a_i = 1$ for all $i \in B$. Then $(T_i)_{i \in B}$ satisfies the equations

$$T_i = 1 + \sum_{j \in B} p_{ij} T_j \quad (i \in B).$$

Proof:

This result follows directly by substituting $a_i = 1$ for all $i, j \in B$ into equation (eq13). ν

Using equation (eq14) for finite B and the case when $a_i = 1$ for each $i \in B$ gives $\Lambda = \mathbf{I}$ and so

$$\mathbf{T} = (\mathbf{I} - \mathbf{p}')^{-1} \mathbf{1}.$$

The existence of an explicit formula for R when B is finite means that Theorem 11.3 defines each R_i uniquely in that case. The following strengthening of Theorem 11.3 provides a unique solution in the general case under light conditions.

Theorem 11.4

If either

a) $R_{ij} \geq 0$ for all $i \in B$ and $j \in B \cup \kappa$, or

b) R_{ij} is bounded below and each conditional mean first-passage time $T_i < \infty$,

then $(R_i)_{i \in B}$ is the minimal nonnegative solution to

$$R_i = v_i + \sum_{j \in B} U_{ij} R_j \quad (i \in B).$$

Proof: Consider first case (a). By considering the contribution to R_i for the first or (n+1) step ($n \geq 1$). A characterisation of R_i is given by

$$R_i = v_i + \sum_{n=1}^{\infty} \sum_{j \in B} U_{ij}^{(n)} v_j \ ,$$

which can be expressed as

$$R_i = \sum_{n=0}^{\infty} \sum_{j \in B} U_{ij}^{(n)} v_j \ .$$

It will be shown by mathematical induction that if $(S_i)_{i \in B}$ is any nonnegative solution to (eq6) then

$$S_i \geq \sum_{m=0}^{n} \sum_{j \in B} U_{ij}^{(m)} v_j \qquad \textbf{(eq15)}$$

for each $i \in B$ and any nonnegative n.

From (eq6)

$$S_i \geq v_i = \sum_{j \in B} U_{ij}^{(0)} v_i ,$$

which gives the basis (n = 0) for the induction.

For the inductive step, suppose equation (eq15) holds for all $i \in B$ for some nonnegative integer n. Then substituting this into (eq6) gives

$$\begin{aligned} S_i &\geq v_i + \sum_{j \in B} U_{ij} \sum_{m=0}^{n} \sum_{k \in B} U_{jk}^{(m)} v_k \\ &= \sum_{j \in B} U_{ij}^{(0)} v_j + \sum_{m=0}^{n} \sum_{k \in B} U_{jk}^{(m+1)} v_k \\ &= \sum_{m=0}^{n+1} \sum_{j \in B} U_{ij}^{(m)} v_j \qquad , \end{aligned}$$

giving the inductive step so (eq15) is proved.

Now letting $n \to \infty$ yields

$$S_i \geq \sum_{m=0}^{\infty} \sum_{j \in B} U_{ij}^{(m)} v_j = R_i$$

as required.

Now consider case (b). We may associate with the process another process with the same transition probabilities but rewards

$$Z_{ij} := R_{ij} + K \text{ where } K := -\inf_{\substack{i \in B \\ j \in B \cup \kappa}} \left(R_{ij}\right)$$

Note that $K < \infty$, since R_{ij} is bounded below.

The conditional first-passage reward for the new process for initial state i is $R_i + K\,T_i \geq 0$, where under the assumptions of the theorem, each $T_i < \infty$. Thus the new process is covered by case (a). The desired result follows. ν

The fact that Theorem 11.3 gives the smallest nonnegative solution means that the results in Corollary 11.1, Corollary 11.2 and Corollary 11.3 also give smallest nonnegative solutions. Furthermore in the cases of Corollary 11.2 and Corollary 11.3 in which the reward gained simply relates to time the reward is always positive, so condition (b) of Theorem 11.4 need not be considered.

11.2 Continuous-Time Markov Chains

A continuous-time Markov chain differs from a discrete-time Markov chain in that the process is now defined for a time parameter in the nonnegative reals instead of nonnegative integers. Denote the state at time τ by X(τ).

Definition 11.6: Continuous-time (homogeneous) Markov chain

A *continuous-time (homogeneous) Markov chain* is a process $(X(\tau), \tau \in \mathbf{R}^+)$ which has the properties

1. A countable number of states.

2. For all finite sequence of times $\tau_1 < \tau_2 < \ldots < \tau_n \in \mathbf{R}^+$ and corresponding states $x_1, x_2, \ldots, x_n \in S$ whenever the conditioning path has positive probability $P(X(\tau_n) = x_n \,|\, X(\tau_{n-1}) = x_{n-1}, \ldots, X(\tau_1) = x_1) = P(X(\tau_n) = x_n \,|\, X(\tau_{n-1}) = x_{n-1})$.

3. $P(X(\tau + s) = j \,|\, X(s) = i) = P(X(\tau + s') = j \,|\, X(s') = i) = P_{ij}(t) \geq 0$ for all s, s' and $\tau \in \mathbf{R}^+$ and $i, j \in S$.

The above definition means that the occupancy time of a state before it moves has an exponential distribution. The mean firing rates of the exponential distributions are given by $\lim_{\tau \to 0^+} \frac{P_{ij}(\tau)}{\tau} = q_{ij}$ for $i \neq j$. Since all the continuous-time Markov chains considered here are time-homogeneous, for brevity they will simply be referred to as continuous-time Markov chains.

Definition 11.7: Q-matrix

The firing rate of each transition is given by the *Q-matrix*, **Q**, which is defined over the state space S and has the properties:

1. $\lim_{\tau \to 0^+} \frac{P_{ij}(\tau)}{t} = q_{ij} \geq 0$ exists for all $i \neq j$;

2. $0 \leq -q_{ii} := \sum_{j \neq i} q_{ij} \leq \infty$ for all i.

Traditionally $q_i := -q_{ii}$.

Definition 11.8: Embedded discrete-time Markov chain

The *embedded discrete-time Markov chain* of a continuous-time Markov chain is given by considering the system only at those instances when the process changes state. It has transition probabilities given by

$$p_{ij} = \begin{cases} q_{ij}/q_i & i \neq j \\ 0 & i = j \end{cases}.$$

The hitting probabilities are the same in the continuous-time process and on its embedded chain, so that $(a_i)_{i\in B}$ is given by Theorem 11.1.

Suppose now that a reward is gained when a transition from state i to state j occurs. In the general case the reward may be dependent on the time spent in state i before the transition occurs as well as on i and j. For this case the reward from a transition is given by the function $r_{ij}(t)$, where t is the time spent in state i before the transition to state j.

Here the same definition of v_i and U_{ij} as were given in the discrete-time Markov chain are used. Also define

$$R_{ij} = q_i r_{ij}^*(q_i)$$

where $r_{ij}^*(q_i)$ is the Laplace transform of reward function $r_{ij}(\tau)$.

Theorem 11.5

If R_i is the mean conditional first-passage reward to κ from initial state $i \in B$, then $(R_i)_{i\in A}$ satisfies

$$R_i = v_i + \sum_{j \in B} U_{ij} R_j \quad (i \in B). \qquad \textbf{(eq16)}$$

Proof:

Because the process is only being viewed at the times that it changes state the proof of Theorem 11.3 gives the general form of the solution to mean conditional first-passage reward. The only difference is in the form that R_{ij} takes in the continuous-time case.

The expected reward from a single transition from i to j will be given by

$$R_{ij} = \int_0^\infty r_{ij}(\tau)\, dG_{ij}(\tau) \qquad \textbf{(eq17)}$$

where $G_{ij}(t)$ is the time distribution function which determines the time spent in i before the transition to j occurs, given such a transition occurs. Thus $G_{ij}(\tau) = 1 - e^{-q_i \tau}$. It should be noted that G_{ij} is parameterised by q_i and not q_{ij}, because in the case of a continuous-time Markov chain the rate at which the transition from state i to state j occurs depends on all the transitions out of i.

Substituting G_{ij} into (eq17) gives

$$R_{ij} = \int_0^\infty r_{ij}(\tau)\, q_i\, e^{-q_i \tau}\, d\tau = q_i r_{ij}^*(q_i), \qquad \textbf{(eq18)}$$

where $r_{ij}^*(q_i)$ is the Laplace transform of $r_{ij}(\tau)$.

The desired result follows by substituting from (eq18) into (eq6). ν

Since the form of the result is the same as that of the discrete-time case, for $|B| < \infty$ the matrix representation of continuous-time case is also given by (eq11). As with the theorem the only difference is that the representation of $R_{i\,j}$ in the calculation of **v**.

As in the discrete case, the problems of mean first-passage reward, mean conditional first-passage time and mean first-passage time are all special cases of the above theorem. Also when the reward is independent of the time spent in the state, the embedded discrete-time Markov chain can be used to determine the reward. This result also flows from Theorem 11.5, as in this case

$$R_{ij} = \int_0^\infty r_{ij} q_i e^{-q_i \tau}\, d\tau = r_{ij} \int_0^\infty q_i e^{-q_i \tau}\, d\tau = r_{ij},$$

since

$$\int_0^\infty q_i e^{-q_i \tau} d\tau = 1.$$

The discrete-time case is thus a special case of the continuous-time case, as expected.

Corollary 11.4

Suppose that $a_i = 1$ for all $i \in B$. Then $(R_i)_{i \in B}$ satisfies the equations

$$R_i = v_i + \sum_{j \in B} p_{ij} R_j \,. \; (i \in B).$$

Proof:

This result follows directly form Theorem 11.5 by substituting $a_i = 1$ for all $i \in B \cup \kappa$ in the same way that Corollary 11.1 followed from Theorem 11.3. ν

As with the conditional case, the matrix representation of this solution is equivalent to that in the discrete-time case. So for $|B| < \infty$ the matrix solution of Corollary 11.4 is given by (eq12), where the R_{ij} defined in Theorem 11.5 is used to calculate **v**.

Corollary 11.5

The mean conditional first-passage time to κ from initial state $i \in B$ satisfies the equations

$$T_i = \frac{1}{q_i} + \sum_{j \in B} U_{ij} T_j \quad (i \in B).$$

Proof:

When the reward gained is the expired time, then $r_{ij}(\tau) = \tau$ so

$$R_{ij} = \int_0^\infty r_{ij}(\tau)\, q_i e^{-q_i \tau} d\tau = q_i \int_0^\infty \tau\, e^{-q_i \tau} d\tau = \frac{1}{q_i}.$$

Thus

$$v_i = \frac{1}{a_i} \sum_{j \in B \cup \kappa} p_{ij} a_j \frac{1}{q_i} = \frac{1}{q_i}, \qquad \textbf{(eq19)}$$

since

$$a_i = \sum_{j \in B \cup \kappa} p_{ij} a_j .$$

The desired result follows by substituting from (eq19) into (eq16). ν

Once more using the fact that **U** is substochastic, the matrix form of this result in the case of $|B| < \infty$ can be given as

$$\mathbf{T} = (\mathbf{I} - \Lambda^{-1} \mathbf{p}' \Lambda)^{-1} \mathbf{q}',$$

where **T**, **I**, Λ and **p**' are as defined earlier and **q**' := $(q_i)_{i \in B}$.

This result is the same as that presented in [35], [38], [49] and [170].

Corollary 11.6

Suppose that $a_i = 1$ for all $i \in B$. Then $(Ti)_{i \in B}$ satisfies the equations

$$T_i = \frac{1}{q_i} + \sum_{j \in B} p_{ij} T_j \quad (i \in B).$$

Proof:

In the case when $a_i = 1$, as with the discrete case, $U_{ij} = p_{ij}$ so the result in Corollary 11.5 simply reduces to this variation of the standard mean time result. ν

Once more using the fact that **p**' is substochastic, the matrix form of this result when $|B| < \infty$ can be given as

$$\mathbf{T} = (\mathbf{I} - \mathbf{p}')^{-1} \mathbf{q}',$$

where **T**, **I**, **p**' and **q**' are as defined earlier.

The minimality result for Theorem 11.5 is the same as that for the discrete case as the solution takes the same form. Thus provided the conditions of Theorem 11.4 hold this result is also true for the continuous case. It should be noted that the conditions of the theorem restrict only

the expected rewards, that is, the values of R_{ij}'s. So $r_{ij}(t)$ need not be well-behaved so long as the expected reward behaves as specified in Theorem 11.4. Similarly Corollary 11.4, Corollary 11.5 and Corollary 11.6 give minimal solutions as with the equivalent discrete cases.

11.3 Semi-Markov Chain

Definition 11.9: Semi-Markov chain

Suppose $(X_n)_{n=0}^{\infty}$ is a Markov chain with states X_0, X_1, X_2, … entered at times S_0, S_1, S_2, …, further suppose that

$$F_{ij}(\tau) = P(S_{n+1} - S_n \le \tau \mid X_n = i, X_{n+1} = j).$$

If $z(\tau)$ is the state of the above defined process at time τ, then $(z(\tau))_{\tau \ge 0}$ is called a Semi-Markov chain.

Semi-Markov chains are a further extension of continuous-time Markov chains. The difference is that now a general distribution can be used to determine the time before a transition occurs. A continuous-time Markov chain is a semi-Markov chain with

$$F_{ij}(\tau) = 1 - e^{-q_i \tau}.$$

A semi-Markov chain is defined by the initial state and

$$\begin{aligned} p_{ij}(\tau) &= P(X_{n+1} = j, S_{n+1} - S_n \le \tau \mid X_n = i) \\ &= P(X_{n+1} = j \mid X_n = i)\, P(S_{n+1} - S_n \le \tau \mid X_n = i, X_{n+1} = j). \\ &= P(X_{n+1} = j \mid X_n = i)\, F_{ij}(\tau) \end{aligned}$$

In this definition $P(X_{n+1} = j \mid X_n = i)$ is the transition probabilities of going from state i to state j of the embedded discrete-time Markov chain. So as with continuous-time Markov

chain in the semi-Markov chain, the probability of reaching the set of states κ given the process starts in state i is determined by the embedded Markov chain. So it is given by Theorem 11.1.

Consider the case where a reward is gained when a transition is made from state i to state j. In the general case the reward is dependent on the time spent in state i before the transition occurs. Again let $r_{ij}(\tau)$ be the reward gained if the time in state i before the transition to state j is t.

Here the same definition of v_i and U_{ij} as were given in the discrete and continuous-time Markov chains are used. Also define

$$R_{ij} := \int_0^\infty r_{ij}(\tau)\, dF_{ij}(\tau) \qquad \textbf{(eq20)}$$

as the reward gained from the transition from state i to state j, given the transition occurs.

Theorem 11.6

The mean conditional first-passage reward R_i satisfies

$$R_i = v_i + \sum_{j \in B} U_{ij} R_j \quad (i \in B).$$

Proof:

The proofs of Theorem 11.3 and Theorem 11.5 give the general form of the solution to mean conditional first-passage reward. The only difference is in the form taken by R_{ij}. In this case the form of R_{ij} is given by equation (eq17), where $G_{ij}(\tau) = F_{ij}(\tau)$. ν

Since the form of the result is the same as that of the discrete-time and continuous-time cases, for $|B| < \infty$ the matrix representation of semi-Markov case is also given by (eq11). As in the theorem the only difference here is in the representation of R_{ij}. Thus **v** is determined using (eq20) instead of the original definition of R_{ij}.

As in the discrete-time and continuous-time cases the problems of mean first-passage reward, mean conditional first-passage time and mean first-passage time are all special cases of the above theorem.

Corollary 11.7

Suppose that $a_i = 1$ for all $i \in B$. Then $(R_i)_{i \in B}$ satisfies the equations

$$R_i = v_i + \sum_{j \in B} p_{ij} R_j \quad (i \in B).$$

Proof:

This result follows directly form Theorem 11.6 by substituting $a_i = 1$ for all $i \in B$ in the same way as done in the proof of Corollary 11.1. ν

As with the conditional case the matrix representation of this solution is equivalent to that in the discrete and continuous-time cases. So for $|B| < \infty$ the matrix solution of Corollary 11.7 is given by (eq12), where v_i is defined using R_{ij} as defined in earlier in this section.

Before proceeding to give the time results, denote the mean time spent in i before going to j, conditional on this transition occurring, by μ_{ij}, which is given by

$$\mu_{ij} = \int_0^\infty \tau \, dF_{ij}(\tau) .$$

Corollary 11.8

The mean conditional first-passage time to κ from initial state $i \in B$ satisfies the equations

$$T_i = v_i + \sum_{j \in B} U_{ij} T_j \quad (i \in B).$$

Proof:

When the reward gained is elapsed time, then $r_{ij}(\tau) = \tau$ so

$$R_{ij} = \int_0^\infty r_{ij}(\tau) \, dF_{ij}(\tau) = \int_0^\infty \tau \, dF_{ij}(\tau) = \mu_{ij} .$$

Thus

$$v_i = \sum_{j \in A \cup \kappa} \frac{p_{ij} a_j \mu_{ij}}{a_i} . \qquad \nu$$

Once more since **U** is substochastic, for $|B| < \infty$ the result in Corollary 11.8 can be expressed as equation (eq11), where $R_{ij} = \mu_{ij}$.

Denote the mean time spent in i by μ_i which is given by

$$\mu_i = \int_0^\infty \tau \, dH_i(\tau) , \qquad \textbf{(eq21)}$$

where

$$H_i(\tau) = \sum_j p_{ij}\, F_{ij}(\tau).$$

Corollary 11.9

Suppose that $a_i = 1$ for all $i \in B$. Then $(T_i)_{i\in B}$ satisfies the equations

$$T_i = \mu_i + \sum_{j\in B} p_{ij} T_j \quad (i \in B).$$

Proof:

Expanding (eq21) gives

$$\begin{aligned}
\mu_i &= \int_0^\infty \tau\, dH_i(\tau) \\
&= \int_0^\infty \tau\, d\left[\sum_j p_{ij} F_{ij}(\tau)\right] \\
&= \int_0^\infty \tau \sum_j p_{ij}\, dF_{ij}(\tau) \\
&= \sum_j p_{ij} \int_0^\infty \tau\, dF_{ij}(\tau) \\
&= \sum_j p_{ij}\, \mu_{ij}.
\end{aligned}$$

Now in the case when $a_i = 1$ for all i, then $B \cup \kappa$ is the complete state space so

$$v_i = \sum_{j\in B\cup\kappa} p_{ij}\mu_{ij} = \sum_j p_{ij}\mu_{ij} = \mu_i .$$

Also as with the discrete case, $U_{ij} = p_{ij}$, so the result in Corollary 11.8 simply reduces to the desired result. ν

Once more using the fact that **p**' is substochastic, the matrix form of this result where $|B| < \infty$ can be given as

$$\mathbf{T} = \left(\mathbf{I} - \mathbf{p}'\right)^{-1}\boldsymbol{\mu}',$$

where **T**, **I** and **p**' are as defined earlier and $\boldsymbol{\mu}' := (\mu_i)_{i \in B}$.

The minimality result is the same as that for the discrete and continuous cases as the solution takes the same form. Thus provided the two conditions of Theorem 11.4 hold then $(R_i)_{i \in B}$ is the smallest nonnegative solution to (eq16) due to Theorem 11.4. Once more this also holds for the results in the corollaries to Theorem 11.4.

11.4 Final Remarks

This chapter has primarily explored ways of calculating the mean first-passage rewards for different types of Markov processes. Such measures have been largely ignored in the literature and are very useful measures for some systems. It should be noted that in the mid to late 80's, some work was done on conditional first-passage times. In this work the state spaces of ETPNs with exponential and deterministic generation functions assigned to transitions were constructed in such a way as to be continuous or semi-Markov processes. This work seems to begin with [19], where the concept of mean first-passage time to a set of markings was considered but only in the case when reaching these states was certain. This was then extended in [18] which discussed briefly the concept of conditional distributions for the first-passage time to a given κ after starting in state i at $\tau = 0$, defined as $F_\kappa(\tau)$. These results also appear to much the same level in [19]-[21], [71] and [107]. These papers focus on reliability modelling of fault-tolerant computer systems and give a general formula for the mean conditional first-passage time to a given state. This formula is

$$F_{\kappa}(\tau) = \frac{P_{i\kappa}(\tau)}{P_{i\kappa}(\infty)},$$

where $P_{i\kappa}(\tau)$ is the probability of reaching κ in time $\leq \tau$. No indication is given of how to calculate any of these values.

This chapter has presented methods for calculating absorbing probabilities and mean conditional first-hitting rewards for discrete, continuous and semi Markov chains. It should be noted that this analysis can be extended to the analysis of non-Markov processes if state aggregation or supplementary variable techniques are used. These techniques have been briefly noted in Section 10.2.

The focus of this thesis is the assessment of decision processes and courses of action. The remainder of this Part focuses on the problem at hand, that of determining absorbing probabilities and mean conditional first-hitting rewards for decision processes. In particular it considers the case where a decision process is modelled using a subclass of STPNs defined in the introduction to this Part.

CHAPTER 12 STPN DECISION PROCESS BUILDING BLOCK

Consider the case when time in the decision process building block is represented by enabling durations associated with input arcs and holding durations are associated with output arcs. Thus STPNs are used.

To simplify the analysis, initially a simpler decision process building will be used. Here the circuits are loops, as shown in Figure 12.1. The reason for this simplification will become evident later.

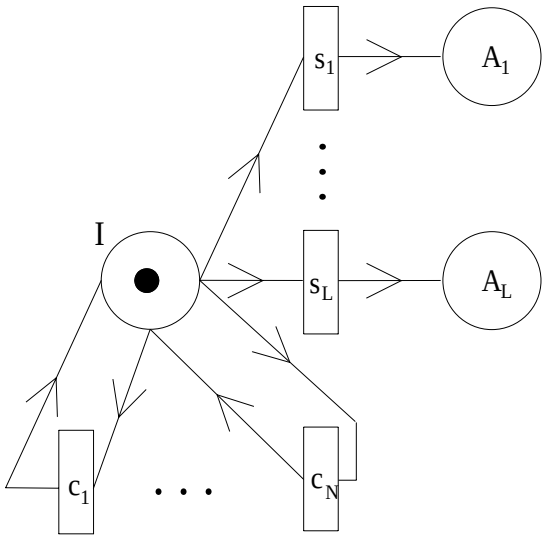


Figure 12.1: Simple decision process building block

The state space of the simple decision process building block is very similar to the PN, see Figure 12.2. This similarity is due to the almost Markovian nature of the building block. Thus the Markov chain results given in Chapter 11 can be used to develop theories on the

absorption probabilities and the mean conditional first-hitting rewards for the state space and thus the STPN building block.

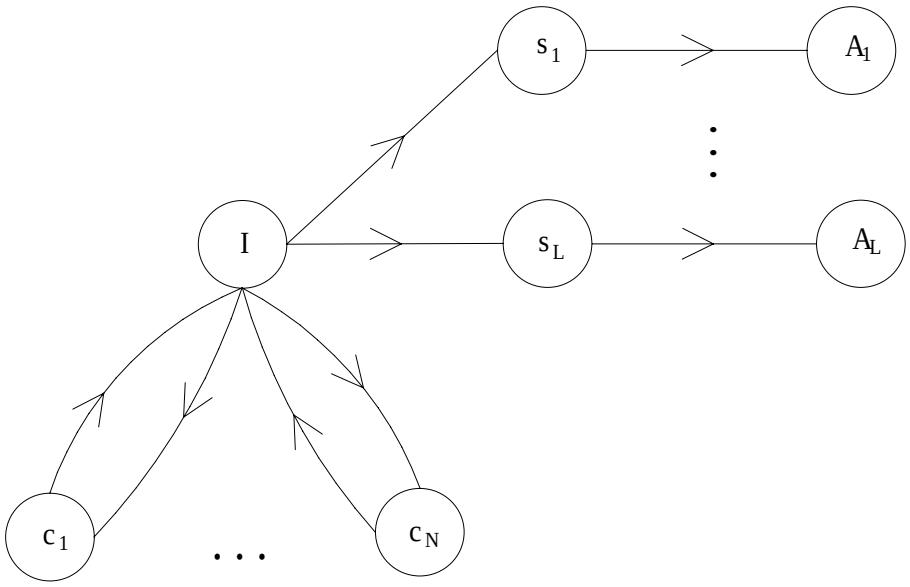


Figure 12.2: Simple decision process building block state space

12.1 Analysis

The delay (time) functions associated with the arcs will be referred to in terms of their transitions. This can be done without confusion as each transition has at most one input and output arc. Define

$E_{p\,t}(\tau) := D(p, t)(\tau)$ and

$H_{t\,p}(\tau) := D(t, p)(\tau)$.

Also for the brevity in the case of the decision process building block define

$E_{c\,i}(\tau) := D(I, c_i)(\tau)$ $(i = 1$ to $N)$ and $E_{s\,j}(\tau) := D(I, s_j)(\tau)$ $(j = 1$ to $L)$ as the enabling duration functions for the C and A transitions; and

$h_{c\,i}(\tau) := D(c_i, I)(\tau)$ $(i = 1$ to $N)$ and $h_{s\,j}(\tau) := D(s_j, A_k)(\tau)$ $(j = 1$ to $L)$ as the holding duration function for the C and A transitions.

For this analysis the race policy will be used to resolve conflict and enabling memories will be used as the memory policy.

In considering the states of the simple decision process building block, state i is defined as a state in c_1 to c_N. Similarly j is used to describe states in s_1 to s_L. What is of interest are the probability and mean conditional first-passage reward associate with reaching state A_k, for $k \in [1, \ldots, L]$.

Define $\Omega(\tau)$:= P(no transition fires before time τ). Then

$$\Omega(\tau) := \left\{\prod_i \left(1 - E_{ci}(\tau)\right)\right\} \times \left\{\prod_j \left(1 - E_{sj}(\tau)\right)\right\}.$$

Define $I_i(\tau)$ to be the probability that transition c_i fires before time τ. Note that for c_i to fire no other transition can fire first. Similarly define $J_j(\tau)$ as the probability that transition s_j fires before time τ. So,

$$\begin{aligned} I_i(\tau) &= \int_0^\tau P\left(\begin{array}{c}\text{transition } c_i \text{ fires in the time period } (u, u + du) \\ \text{and no transition fires before time } u\end{array}\right) \\ &= \int_0^\tau P(\text{no transitions fire before time } u) \times \\ &\quad P(\text{transition } c_i \text{ fires in the time period } (u, u + du) \mid \\ &\quad \text{no transition before time } u). \end{aligned}$$

If there is a positive probability that two or more transitions fire at the same time, then the above result does not hold. In such cases the race policy cannot be used to resolve conflict so an extra conflict resolution policy must be specified. If an alternative conflict resolution policy were specified for those transitions that could have identical enabling durations, then

the following results could be adjusted to allow for this. This situation will not be considered further.

Expanding the conditional probability in $I_i(\tau)$ gives

$$I_i(\tau) = \int_0^\tau P(\text{no transitions fire before time u}) \times \frac{P(\text{transition } c_i \text{ fires in the time period (u, u + du)})}{P(\text{transition } c_i \text{ does not fire before time u})}.$$

Therefore

$$I_i(\tau) = \int_0^\tau \left\{\prod_{m=1}^{N}\left(1 - E_{cm}(u)\right)\right\}\left\{\prod_{j=1}^{K}\left(1 - E_{sj}(u)\right)\right\}\frac{dE_{ci}(u)}{\left(1 - E_{ci}(u)\right)}$$
$$= \int_0^\tau \frac{\Omega(u)\, dE_{ci}(u)}{\left(1 - E_{ci}(u)\right)}.$$

A similar argument gives

$$J_j(\tau) = \int_0^\tau \frac{\Omega(u)\, dE_{sj}(u)}{1 - E_{sj}(u)}.$$

The probability that a transition in the c system fires by time t is given by

$$I(\tau) = \sum_{i=1}^{N} I_i(\tau).$$

The probability that a transition in the A system fires by time t is given by

$$J(\tau) = \sum_{j=1}^{K} J_j(\tau).$$

This means that the probability that transition c_i (s_j) fires is given by $I_i(\infty)$ ($J_j(\infty)$). So in terms of the Markov process in Figure 12.2

$$p_{Is_i} = I_i(\infty) \text{ and } p_{Ic_j} = J_j(\infty).$$

Also the probability that a C (A) transition fires is $I(\infty)$ ($J(\infty)$). That is,

$$\sum_{i=1}^{N} p_{Ic_i} = I(\infty), \quad \sum_{j=1}^{M} p_{Is_j} = J(\infty) \text{ and } I(\infty) + J(\infty) = 1.$$

Now

$$P(\text{i fires at time } \tau \text{ or earlier} \mid \text{i fires}) = \frac{P(\text{i fires at time } \tau \text{ or earlier} \cap \text{i fires})}{P(\text{i fires})}$$
$$= \frac{P(\text{i fires at time } \tau \text{ or earlier})}{P(\text{i fires})}$$
$$= \frac{I_i(\tau)}{I_i(\infty)}.$$

Similarly

$$P(\text{j fires at time } \tau \text{ or earlier} \mid \text{j fires}) = \frac{J_j(\tau)}{J_j(\infty)}.$$

Theorem 12.1

In a STPN simple decision process building block the absorption probability of state k, given that the probability that two transition will have the same enabling time is zero, is given by

$$P(k) = \frac{J_k(\infty)}{J(\infty)}. \qquad (\textbf{eq22})$$

Proof:

For the state space shown in Figure 12.2, $\kappa = A_k$ and $B = \{I, c_1, \ldots, c_N, s_k\}$, which means Theorem 11.1 gives

$$a_w = p_{wA_k} + \sum_{x \in B} p_{wx} a_x .$$

For all $x \in B$, $p_{s_k x} = 0$ and $p_{s_k A_k} = 1$, so $a_{s_k} = 1$.

Also $p_{c_i w} = 0$ for all $w \neq I$ and $p_{c_i I} = 1$, so $a_{c_i} = p_{c_i I} a_I = a_I$.

Further $p_{IA_k} = 0$ and so

$$a_I = \sum_{x \in B} p_{Ix} a_x = \sum_{i=1}^{N} p_{Ic_i} a_{c_i} + p_{Is_k} a_{s_k} = \sum_{i=1}^{N} p_{Ic_i} a_I + p_{Is_k}.$$

Thus

$$a_I = \frac{p_{Is_k}}{1 - \sum_{i=1}^{N} p_{Ic_i}} = \frac{J_k(\infty)}{1 - I(\infty)} = \frac{J_k(\infty)}{J(\infty)},$$

as required. ν

The above result means that the probability of reaching a given absorbing state is independent of the number of elements in the circuits. Therefore the above result holds not only for the simple decision process building block but also for the more complex ones with non-loop circuits. Accordingly we have the following.

Corollary 12.1

The probability of reaching the absorbing state j of a STPN decision process building block is given by Theorem 12.1.

Theorem 12.2

The mean conditional first-passage reward for state A_k of a STPN simple decision process building block is given by

$$R_I = \frac{1}{J(\infty)} \sum_{i=1}^{N} \left(\int_0^\infty r_{Ic_i}(\tau)\, dI_i(\tau) + I_i(\infty) \int_0^\infty r_{c_i I}(\tau)\, dH_{ci}(\tau) \right) + \frac{1}{J_k(\infty)} \int_0^\infty r_{Is_k}(\tau)\, dJ_k(\tau) + \int_0^\infty r_{s_k A_k}(\tau)\, dH_{sk}(\tau), \quad \textbf{(eq23)}$$

where $r_{I\,w}(\tau)$ is the reward gained by spending time τ in I before a transition w ($\in \{c_1, \ldots, c_N\} \cup \{s_k\}$) fires, $r_{c_i I}(\tau)$ the reward gained by the token in I being held for time τ after transition c_i fires and $r_{s_k A_k}(\tau)$ the reward gained by the token spending a holding time of τ in place A_k after the firing of transition s_k.

Proof:

As with Theorem 12.1 the results from Chapter 11 can be applied to the state space shown in Figure 12.2 to get the desired result. In this case Theorem 11.6 is used. Once more $\kappa = A_k$ and $B = \{I, c_1, \ldots, c_N, s_k\}$ so applying Theorem 11.6 gives

$$R_x = v_x + \sum_{w \in B} U_{xw} R_w \quad \text{where } v_x = \sum_{w \in B \cup \kappa} \frac{p_{xw} a_w R_{xw}}{a_x} \text{ for all } x \in B.$$

Since $p_{s_k w} = 0$ for all $w \neq A_k$,

$$R_{s_k} = v_{s_k} + U_{s_k A_k} R_{A_k},$$

where $v_{s_k} = \frac{p_{s_k A_k} a_{A_k} R_{s_k A_k}}{a_{s_k}}$. Also $p_{s_k A_k} = a_{s_k} = a_{A_k} = 1$ and $R_{A_k} = 0$, so

$$R_{s_k} = R_{s_k A_k}.$$

For $x \in \{c_1, \ldots, c_N\}$, $p_{xw} = 0$ for all $w \neq I$ giving

$$R_x = v_x + U_{xI} R_I$$

where $v_x = \dfrac{p_{xI} a_I R_{xI}}{a_x}$. Also $p_{xI} = 1$ and $a_x = a_I$, so

$$R_x = R_{xI} + R_I .$$

For x=I,

$$R_I = v_I + \sum_{w \in B} U_{Iw} R_w$$

where $v_I = \sum_{w \in B} \dfrac{p_{Iw} a_w R_{Iw}}{a_I}$. Also since $p_{IA_k} = 0$,

$$\begin{aligned} R_I &= \sum_{w \in B} U_{Iw} \left(R_{Iw} + R_w \right) \\ &= U_{Is_k} \left(R_{Is_k} + R_{s_k} \right) + \sum_{i=1}^{N} U_{Ic_i} \left(R_{Ic_i} + R_{c_i} \right) \\ &= \frac{p_{Is_k}}{a_I} \left(R_{Is_k} + R_{s_k A_k} \right) + \sum_{i=1}^{N} p_{Ic_i} \left(R_{Ic_i} + R_{c_i I} + R_I \right) \end{aligned}$$

Thus

$$\begin{aligned} R_I &= \frac{1}{1 - \sum_{i=1}^{N} p_{Ic_i}} \left[\frac{p_{Is_k}}{a_I} \left(R_{Is_k} + R_{s_k A_k} \right) + \sum_{i=1}^{N} p_{Ic_i} \left(R_{Ic_i} + R_{c_i I} \right) \right] \\ &= R_{Is_k} + R_{s_k A_k} + \frac{1}{J(\infty)} \sum_{i=1}^{N} I_i(\infty) \left(R_{Ic_i} + R_{c_i I} \right) \quad , \end{aligned} \qquad \textbf{(eq24)}$$

since

$$\frac{p_{Is_k}}{a_I} = \frac{J_j(\infty)}{J_j(\infty) \big/ J(\infty)} = J(\infty) \text{ and } 1 - \sum_{i=1}^{N} p_{Ic_i} = 1 - I(\infty) = J(\infty) .$$

The expected reward gained by the firing of transition c_i given this transition occurs is given by

$$R_{Ic_i} = \int_0^\infty r_{Ic_i}(\tau)\, d\left[\frac{I_i(\tau)}{I_i(\infty)}\right] = \frac{1}{I_i(\infty)}\int_0^\infty r_{Ic_i}(\tau)\, dI_i(\tau)\left(= \frac{1}{I_i(\infty)}\int_0^\infty \frac{r_{Ic_i}(\tau)\,\Omega(\tau)}{1-E_{ci}(\tau)}\, dE_{ci}(\tau)\right). \quad (\textbf{eq25})$$

Thus R_{Ic_i} is the expected reward gained in state I before a transition is made to state c_i. Similarly

$$R_{Is_j} = \int_0^\infty r_{Is_j}(\tau)\, d\left[\frac{J_j(\tau)}{J_j(\infty)}\right] = \frac{1}{J_j(\infty)}\int_0^\infty r_{Is_j}(\tau)\, dJ_j(\tau)\left(= \frac{1}{J_j(\infty)}\int_0^\infty \frac{r_{Is_j}(\tau)\,\Omega(\tau)}{1-E_{sj}(\tau)}\, dE_{sj}(\tau)\right)$$

is the expected reward for going from state I to state s_j.

Since there is only one possible transition out of states c_i and s_k, the expected rewards from these states are simply given by

$$R_{c_i I} = \int_0^\infty r_{c_i I}(\tau)\, dH_{ci}(\tau) \text{ and} \quad (\textbf{eq26})$$

$$R_{s_k A_k} = \int_0^\infty r_{s_k A_k}(\tau)\, dH_{sk}(\tau). \quad (\textbf{eq27})$$

Substituting from equations (eq25) to (eq27) into (eq24) provides (eq23), the required result.

ν

To get the more general result that applies to the decision process building block (as shown in Figure 9.2) the meaning of equation (eq26) must be examined in greater detail. In a broader sense this term gives the expected reward gained in the circuit. Thus to expand this result from the simple decision process building block is simply a matter of determining the expected reward of the more complex circuit.

Consider the PN shown in Figure 12.3. This is representation of the circuits shown in Figure 9.2. The corresponding state space is given in Figure 12.4. Clearly the expected reward for such a state space is the sum of the expected rewards of the individual state transitions, which yields the following corollary.

Corollary 12.2

The mean conditional first-passage reward for state A_k of a STPN decision process building block is given by

$$R_I = \frac{1}{J(\infty)} \sum_{i=1}^{N} \left(\int_0^{\infty} r_{Ic_i}(\tau)\, dI_i(\tau) + I_i(\infty)\, R_{c_i} \right) + \frac{1}{J_k(\infty)} \int_0^{\infty} r_{Is_k}(\tau)\, dJ_k(\tau) + \int_0^{\infty} r_{s_k A_k}(\tau)\, dH_{sk}(\tau), \qquad \textbf{(eq28)}$$

where $r_{I\,w}(\tau)$ and $r_{s_k A_k}(\tau)$ are as defined in Theorem 12.2 and R_{c_i} is the expected reward gained in circuit c_i.

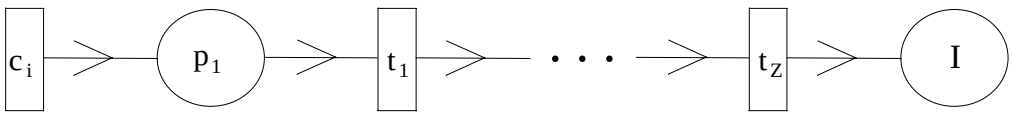


Figure 12.3: Circuit as stand alone PN

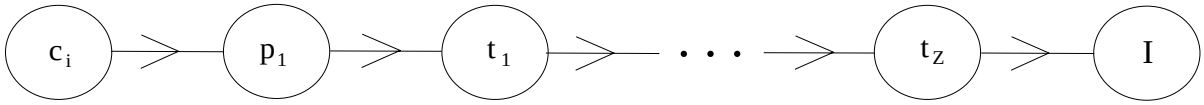


Figure 12.4: State space of PN in Figure 12.3

Corollary 12.3

The mean first-passage time for state A_k of a STPN decision process building block is given by

$$T_I = \frac{1}{J(\infty)} \sum_{i=1}^{N} \left(\int_0^\infty \tau \, dI_i(\tau) + I_i(\infty) \, \tau_{c_i} \right) + \frac{1}{J_k(\infty)} \int_0^\infty \tau \, dJ_k(\tau) + \tau_{s_k},$$

where τ_{c_i} is the mean time spent in the circuit c_i and τ_{s_k} the expected holding time of transition R_k.

Proof:

If the reward of interest is the time taken, then $r_{ij}(\tau) := \tau$ for all i and j. Substituting this into equation (eq28) provides

$$\begin{aligned} R_I &= \frac{1}{J(\infty)} \sum_{i=1}^{N} \left(\int_0^\infty \tau \, dI_i(\tau) + I_i(\infty) \, R_{c_i} \right) + \frac{1}{J_k(\infty)} \int_0^\infty \tau \, dJ_k(\tau) + \int_0^\infty \tau \, dH_{sk}(\tau) \\ &= \frac{1}{J(\infty)} \sum_{i=1}^{N} \left(\int_0^\infty \tau \, dI_i(\tau) + I_i(\infty) \, \tau_{c_i} \right) + \frac{1}{J_k(\infty)} \int_0^\infty \tau \, dJ_k(\tau) + \tau_{s_k}. \end{aligned}$$

This yields the desired result, since when the reward gained is given by the time taken $R_{c_i} = \tau_{c_i}$. ν

Consider the case when the transitions have exponentially-distributed enabling durations. Let the firing rate of the transition c_i, i = 1 to N, be λ_i and that of s_j, j = 1 to L be μ_j. This means that

$$E_{ci}(\tau) = 1 - e^{-\lambda_i \tau} \qquad \textbf{(eq29)}$$

and

$$E_{sj}(\tau) = 1 - e^{-\mu_j \tau}. \qquad \textbf{(eq30)}$$

Also define $\Lambda := \sum_{i=1}^{N} \lambda_i$ and $M := \sum_{j=1}^{L} \mu_j$.

Equations (eq29) and (eq30) give

$$\Omega(\tau) = e^{-(\Lambda+M)\tau}, \quad I_i(\tau) = \frac{\lambda_i}{\Lambda + M}\left(1 - e^{-(\Lambda+M)\tau}\right), \quad I_i(\infty) = \frac{\lambda_i}{\Lambda + M},$$

$$J_j(\tau) = \frac{\mu_j}{\Lambda + M}\left(1 - e^{-(\Lambda+M)\tau}\right), \quad J_j(\infty) = \frac{\mu_j}{\Lambda + M} \text{ and } J(\infty) = \frac{M}{\Lambda + M}.$$

Corollary 12.4

The absorption probability of state k of a decision process building block with exponentially distributed enabling durations is given by

$$P(k) = \frac{\mu_k}{M}.$$

Proof: This follows by substituting the above stated values of $J_k(\infty)$ and $J(\infty)$ into (eq22). ν

Corollary 12.5

The mean conditional absorption time of a decision process building block with exponentially distributed enabling durations is given by

$$T_j = \frac{1}{M}\left[1+\sum_{i=1}^{N}\tau_i\lambda_i\right]+\tau_j.$$

Proof: Since

$$\int_0^\infty \tau\, dI_i(\tau) = \frac{\lambda_i}{(\Lambda+M)^2} \quad \text{and} \quad \int_0^\infty \tau\, dJ_j(\tau) = \frac{\mu_j}{(\Lambda+M)^2},$$

then

$$\begin{aligned} T_j &= \frac{\Lambda+M}{M}\left(\sum_{i=1}^{N}\frac{\lambda_i}{(\Lambda+M)^2}+\sum_{i=1}^{N}\frac{\lambda_i}{\Lambda+M}\kappa_i\right)+\frac{\dfrac{\mu_i}{(\Lambda+M)^2}}{\dfrac{\mu_i}{\Lambda+M}}+\tau_j \\ &= \frac{1}{M}\left[\frac{\Lambda}{\Lambda+M}+\sum_{i=1}^{N}\lambda_i\kappa_i\right]+\frac{1}{\Lambda+M}+\tau_j, \end{aligned}$$

which on simplification gives the desired result. ν

Note that, as before, this result is independent of the distribution function of the second transition in the circuit. So the enabling duration distributions of the transitions in the circuits do not have to be exponential.

12.2 HTPN Building Blocks

Consider the case when a building block has only holding durations. In this case $I_i(\infty)$ and $J_j(\infty)$ are the relative firing probabilities that transitions c_i and s_j. Clearly, from the derivation of the result in Theorem 12.1, this result holds for the special case of HTPN building blocks.

As with the definition of HTPNs as a sub-class of STPNs in Section 6.2, there is a need to determine the meaning of $I_i(\tau)$ and $J_j(\tau)$. This redefinition is very similar to these earlier cases. These two functions take the form of step functions each taking a value of zero for $\tau < 0$ and the values of $I_i(\infty)$ and $J_j(\infty)$ for $\tau \geq 0$. Thus, the derivative of these functions is the Dirac delta centred at zero. Placing this into the result in Theorem 12.2, gives the equivalent result for HTPNs.

Corollary 12.6

The mean conditional first-passage reward for state k of a HTPN decision process building block is given by

$$R_I = \frac{r_{Is_k}(0)}{J(\infty)} + \frac{1}{J(\infty)} \sum_{i=1}^{N} I_i(\infty) R_{c_i} + \frac{r_{Is_k}(0)}{J_k(\infty)} + \int_0^{\infty} r_{s_k A_k}(\tau)\, dH_{sk}(\tau) .$$

Further, if the reward is time then Corollary 12.3 simplifies to give the following result.

Corollary 12.7

The mean conditional absorption time in a state k of a HTPN decision process building block is given by

$$T_k = \frac{1}{J(\infty)} \sum_{i=1}^{N} I_i(\infty) \tau_i + \tau_k .$$

12.3 ETPN Building Blocks

Finally consider the case when the decision process building block has only enabling times. In this case the result in Theorem 12.1 holds directly. Also, as outlined in Part I,

$$H_{ci}(\tau) = \begin{cases} 0 & \tau < 0 \\ 1 & \tau \geq 0 \end{cases} \text{ and}$$

$$H_{sj}(\tau) = \begin{cases} 0 & \tau < 0 \\ 1 & \tau \geq 0 \end{cases}.$$

Thus for this special case Theorem 12.2 can be rewritten to allow for zero holding times.

Corollary 12.8

The mean conditional absorption reward for state k of an ETPN decision process building block is given by

$$R_I = \frac{1}{J(\infty)} \sum_{i=1}^{N} \left(\int_0^{\infty} r_{Ic_i}(\tau)\, dI_i(\tau) + I_i(\infty) R_{c_i} \right) + \frac{1}{J_k(\infty)} \int_0^{\infty} r_{Is_k}(\tau)\, dJ_k(\tau) + r_{s_k A_k}(0).$$

Similarly the result were rewards relate to time can be given.

Corollary 12.9

The mean conditional absorption time for state j of an ETPN decision process building block is given by

$$T_j = \frac{1}{J(\infty)} \left(\sum_{i=1}^{N} \int_0^{\infty} \tau\, dI_i(\tau) + \sum_{i=1}^{N} I_i(\infty)\, \varepsilon_i \right) + \frac{1}{J_j(\infty)} \int_0^{\infty} \tau\, dJ_j(\tau),$$

where ε_i is the sum of the mean enabling times of the transitions in the i^{th} circuit excluding the first transition.

12.4 Example of Decision Process Analysis

The decision process shown in Figure 9.1 will now be used to illustrate the application of the results in this section. Some sample time functions for this STPN are given in Table 12.1. The enabling durations of transitions t1 to t4 relate to the time spent by the friendly unit searching for its target. In the case of t4 more time is spent searching as the unit knows that once the decision to rest is made the search will not resume for a relatively long period of time. Due to the exponential distribution being selected for these transitions, the mean time before one of the transitions t1 to t4 fires is the reciprocal of the sum of the rates of the enabled transitions. Thus the individual rates of the transitions really reflect only the relative probability of a certain transition firing. The holding duration of transition t1 reflects the time it takes for the friendly unit to be neutralised by the enemy. The holding duration of transition t2 relates to the time the unit spends preparing to attack the target. The holding duration of t3 is the time taken to determine where next to search for the enemy. The holding duration of transition t4 is the time spent making camp, resting and breaking camp. The enabling durations of transitions t5 to t7 are the times spent in battle between the friendly unit and the target. If no outcome is reached then the holding duration of t7 is the time between direct exchanges. If the friendly unit is neutralised then the holding duration of transition t5 is the time taken to confirm the neutralisation of the friendly unit and vice versa in the case of transition t6. Due to the fact that the result is independent of enabling time distribution functions of transitions internal to a circuit no distribution is given for the enabling distribution of transition t8. The enabling time of this transition relates to the time spent resting and the holding time to the time taken to pack up camp and returning to the search.

Table 12.1: Sample mean durations for the STPN in Figure 9.1

Transition	Enabling duration, $(\lambda_i)^{-1}$ or $(\mu_j)^{-1}$		Mean holding duration (s), τ_i or ϕ_j
	Distribution	Parameters	
t1	Exponential	$\mu_1 = 1/5$	$\phi_1 = 10$
t2	Exponential	$\mu_2 = 1/5$	$\phi_2 = 10$
t3	Exponential	$\lambda_3 = 1/5$	$\tau_3 = 5$
t4	Exponential	$\lambda_4 = 1/50$	$\tau_4 = 50$
t5	Exponential	$\mu_5 = 1/10$	$\phi_5 = 10$
t6	Exponential	$\mu_6 = 1/10$	$\phi_6 = 10$
t7	Erlang	$\alpha_7 = 2$, $\lambda_7 = 1/10$	$\tau_7 = 20$
t8	-	$\lambda_8 = 1/50$	$\tau_8 = 100$

The results for the decision process building block shown in Figure 9.3(a) are given in Table 12.2. The results for the second decision process building block shown in Figure 9.3(b) are given in Table 12.3. For these tables the values for P(j) are calculated using Corollary 12.4 while those T_I are calculated using Corollary 12.5.

Table 12.2: Results for STPN in Figure 9.3(a) with the values in Table 12.1

Decision (state j is reached)	Probability of reaching decision (P(j))	Mean conditional time to reach decision (T_j)
The friendly unit was neutralised before it located the target (state 2).	½	7
The target was found (state 3).	½	7

The results for the second decision process building block shown in Figure 9.3(b) are given in Table 12.3. The results in Table 12.3 are the values for the probability and mean conditional absorption time for the PN in Figure 9.3(b) where the initial marking is one token in place p3. Thus, these results relate only to this building block, not the overall model.

Table 12.3: Results for STPN in Figure 9.3(b) for the values in Table 12.1

Decision (state j is reached)	Probability of reaching decision (P(j))	Mean conditional time to reach decision (T_j)
The friendly unit was neutralised during the battle (state 5').	½	10.2
The mission was completed (state 6').	½	10.2

As earlier the results in Table 12.3 are the values for the probability and mean time given for the PN in Figure 9.3(b) where the initial marking is one token in place p3. Thus, these results relate only to the movement of the token from place p3 to either place p5 or place p6. By

combining the results in Table 12.2 and Table 12.3 we can derive the overall results of the decision process shown in Figure 9.1. These are given in Table 12.4.

Table 12.4: Results for STPN in Figure 9.1 for the values in Table 12.1

Decision (state j is reached)	Probability of reaching decision (P(j))	Mean conditional time to reach decision (T_{p1})
The friendly unit was neutralised before it found the target (state 2).	$P(2) = 1/2$	$T_{p1} = 7$
The friendly unit was neutralised during the battle (state 5).	$P(5) = P(3) \times P(5') = 1/4$	$T_{p1}(3) + T_{p3}(5) = 17.2$
The mission was completed (state 6).	$P(6) = P(3) \times P(6') = 1/4$	$T_{p1}(3) + T_{p3}(6) = 17.2$

12.5 Algorithm for Analysing a Decision Process

An algorithm for analysing a STPN decision process can be established to calculate the absorption probability and mean conditional absorption reward for each absorbing state.

In this algorithm two lists are used, the marking list and the absorption list. The marking list contains a list of markings (this list is not the complete reachability set) along with the absorption probabilities and mean conditional absorption rewards for each of these markings. These values are referred to as the *local absorption probabilities* and *local mean conditional absorption rewards* as the markings they relate to may not be absorption markings, but rather the initial place of the next building block. The absorption list is the algorithm output and takes a similar form to the marking list but contains only the absorption probabilities and

mean conditional absorption rewards of the absorbing markings. If the local absorption probabilities and local mean conditional absorption rewards of other markings in the marking list are of interest then these can also be retained.

The algorithm is as follows.

1 Start with the initial marking in the marking list.

2 Set the local absorption probability of the initial marking to be 1 and its mean conditional absorbing reward to 0.

3 For each marking, M, in the marking list:

3.1 Determine all the circuits involving the current marking. The first transition of each circuit is a transition in C.

3.2 Calculate the mean reward gained in each of the circuits. These are the R_{c_i} values in Corollary 12.2.

3.3 Determine all the transitions enabled by M but not in C. These transitions make up A.

3.4 For each immediately reachable marking, M', reached by the firing of an s transition:

3.4.1 Add this marking to the marking list.

3.4.2 Apply Theorem 12.1 to calculate the absorption probability of this immediately reachable marking.

3.4.3 Apply Corollary 12.2 to calculate the mean conditional absorption reward for this marking.

3.4.4 Multiply the absorption probability, given by step 3.4.2, by the local probability of M, to get the local absorption probability of M'.

3.4.5 Add the mean conditional absorption reward, given in step 3.4.3, to the local mean conditional absorption reward of M, to get the local mean conditional absorption reward of M'.

3.5 If there are no transitions enabled by M, the local absorption probability and local mean conditional absorption reward along with M are stored in the absorption list as the absorption probability and mean conditional absorption reward of the absorbing state M.

3.6 Remove M from the marking list.

4 Read off from the absorption probabilities and mean conditional absorption rewards for the absorbing states given in the absorption list.

This algorithm can be simplified if all circuits are reduced to loops before the process begins. This will simplify step 3.1 by requiring only the search for loops.

One of the problems with this algorithm is that it does not allow for more complex circuits. The next chapter looks at this problem.

CHAPTER 13 EXTENDING THE BASIC BUILDING BLOCK

This chapter considers how to analyse some extended building blocks. Initially Corollary 12.1 and Corollary 12.2 are used to reduce complex C elements to loops. Next the problem of the C system containing absorbing states is considered. Finally the case when there is more than one path to an absorbing place is dealt with.

13.1 Reducing More Complex Circuits

In the previous sections the C system was made up of simple circuits, as shown in Figure 9.2. As outlined in Section 12.5 simple circuits could be reduced to loops using the holding duration component of the STPN. This was a very simple case of net reduction which was made possible by the time representation used. This section considers reduction of more complex C systems to single loops using Corollary 12.2. Consider the case where a circuit has within it a number of circuits. Figure 13.1 shows an example of such a STPN.

To help describe how such STPNs can be reduced and thus analysed in the same way as those defined earlier, some terminology will now be introduced.

Definition 13.1: Branching place

A *branching place* is a place which is the input to more than one transition. That is, if the number of elements in the set of output transitions, $p^{\bullet}$, for place p is greater than 1, then p is a branching place.

In Figure 13.1 places p1, p4 and p5 are branching places.

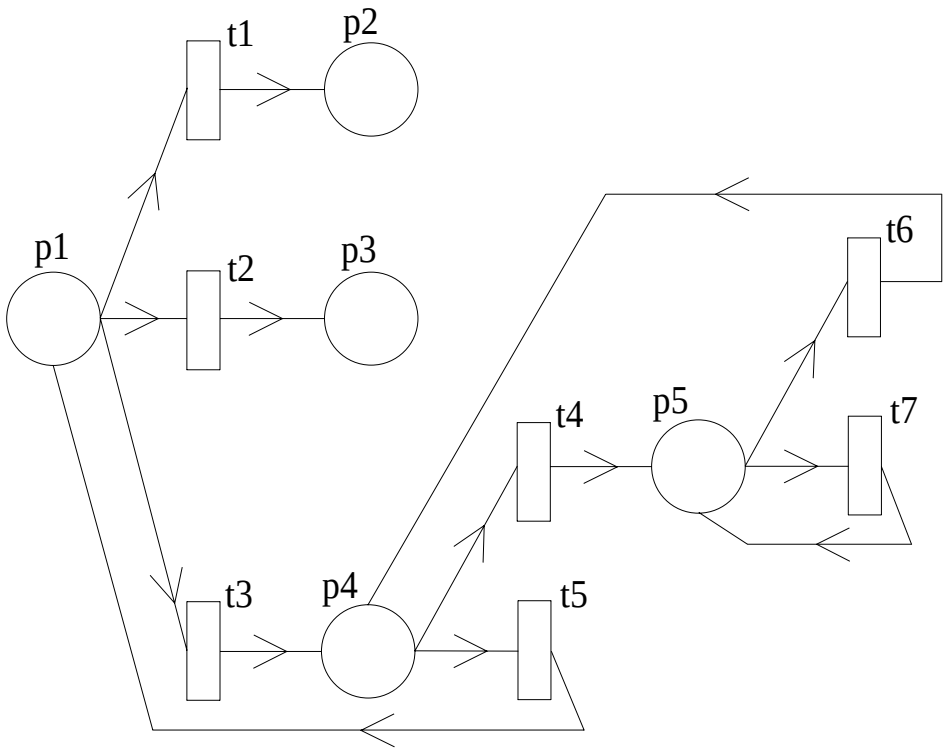


Figure 13.1: Example of sub-circuits

Definition 13.2: Sub-branching place

The branching place p' is a *sub-branching place* with respect to place p, if there exists a marking M in a circuit with the initial marking M_0 ($\neq$M) such that $M_0(p)\neq 0$ and $M(p')\neq 0$. So a sub-branching place is a place inside a circuit which starts at a marking for which the sub-branching place has no tokens.

In Figure 13.1 places p4 and p5 are sub-branching places with respect to p1 since both are branching places and part of the c_3 part of the STPN[12]. There is some symmetry in this definition in that a sub-branching place of a branching place has the original branching place as a sub-branching place.

Definition 13.3: Sub-circuit

The circuit C', with initial marking M', is a *sub-circuit* of the circuit C, with initial marking M, if $M' \in C \backslash \{M\}$ and $C' \neq C$. So a sub-circuit is a circuit within another circuit.

It should be noted for the above definition to hold at least one place marked by the marking M' must be a branching place. Figure 13.1 has a number of sub-circuits. Consider the circuit c_3 = {p1, t3, p4 ,t5}, where the place name, p, is given to represent the marking where there is a single token in place p[13]. This circuit has sub-circuit c_4 = {p4, t4, p5, t6} which itself has the sub-circuit c_7 = {p5, t7}.

The process of removing sub-circuits may need to be applied several times to a circuit before all its sub-circuits are removed. Consider the simple circuit c', with initial marking p, which is

[12] Thoughout this Chapter for brevity circuits will be referred to by their initial transition. Thus the circuit involving t3 is referred to as c_3.

[13] The notation of using the place name to represent the marking can (and will) be used for all the STPN decision processes that will be considered in the remainder of this thesis. Although this is an abuse of notation it is convenient and makes many aspects of the descriptions easier to follow. When the meaning is unclear the marking will be explicitly stated.

a sub-circuit of the simple circuit c, so $p \in c$. Let $t(\in c)$ and $p'(\in c)$ be the transition and marking such that $p[t>p'$. Finally for place p let ${}^{\bullet}p=\{t'\}$. The following steps will remove the sub-circuit c'.

1. Use Corollary 12.2 to calculate the mean conditional absorption reward of p' from p.
2. Add the result from step 1 to the mean reward of circuit c. That is, this value is added to R_c.
3. Set $t'^{\bullet} = \{p'\}$, removing c' and t from the STPN.

Although the above procedure has been applied to a sub-branching place with only one circuit to remove (c' above), it is valid if there are multiple circuits that need to be removed. Also if there are multiple sub-circuits with different branching places these can be removed one at a time using the above procedure.

Consider the circuit {p4, t4, p5, t6} and its sub-circuit {p5, t7} in Figure 13.1. To remove the sub-circuit the mean conditional absorption reward of p4 from p5 must be calculated. This is then added to the mean reward R_{c_4} gained in circuit c_4. Finally set $t4^{\bullet} = \{p4\}$, removing t7, t6 and p4 from the STPN as shown in Figure 13.2.

The above procedure can then be applied a second time to remove the inner circuit {p4, t4'} of the reduced net shown in Figure 13.2. After this next application of reduction the circuit will be a loop allowing Corollary 12.2 to be applied to the overall STPN to calculate the mean conditional absorption reward. The absorption probabilities are not changed by this reduction as it does not change any of the c or s transition enabling durations.

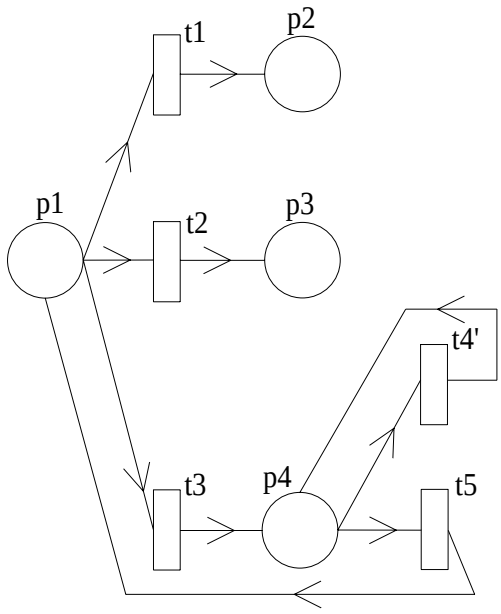


Figure 13.2: First reduction of the CTPN in Figure 13.1

13.2 Absorbing States as Part of Circuits

In the previous section the problem of circuits within circuits was considered. In this section this problem is taken one step further by allowing absorbing places and circuits as part of circuits. So as in the previous section the circuits can have sub-branching places in them, only now these branches may lead to absorbing states. To calculate the mean conditional absorption rewards, each branching process which can lead to an absorbing marking is considered separately.

Consider the STPN shown in Figure 13.3 with the displayed marking. This STPN has three circuits: {p1, t3, p4, t8}, {p1, t4} and {p2, t5, p5, t9}. It has four absorbing states, represented by a token in place p3, p6, p7 or p8.

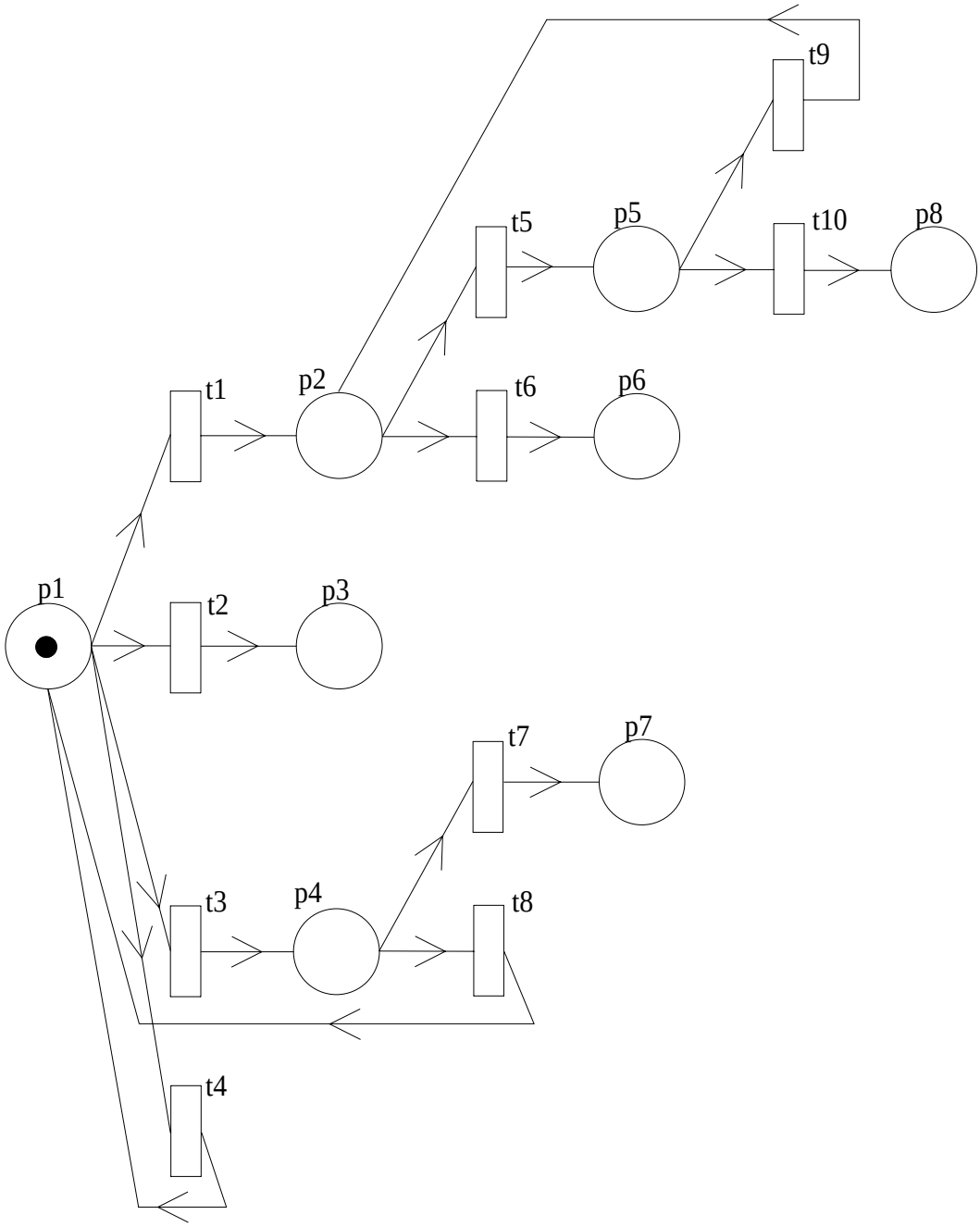


Figure 13.3: Example of an absorbing state within a circuit

To help describe how to analyse STPNs with absorbing states as part of their circuits, some more definitions are helpful.

Definition 13.4: Non-re-enabling transition

A *non-re-enabling transition* for the marking M is a transition that can fire only once for the given marking M. That is, transition t is non-re-enabling if it is enabled for some marking M and is never enabled for all markings in the set $[M'\rangle$, where $M[t\rangle M'$.

By definition a non-re-enabling transition cannot be part of a circuit.

In Figure 13.3 transitions t1, t2, t6, t7 and t10 are non-re-enabling transitions. Transitions t2, t6, t7 and t10 result in absorbing markings while transition t1 does not. If a transition is non-re-enabling for the initial marking then it is simply called non-re-enabling.

Definition 13.5: Absorbing branching place

An *absorbing branching place* is a branching place for which at least one of its output transitions is a non-re-enabling transition. Thus if for some non-re-enabling transition $t \in p^\bullet$ and $p^\bullet \backslash t \neq \{\varnothing\}$ then p is an absorbing branching place.

The set of absorbing branching places is a subset of the net's branching places. The branching places, p1, p2, p4 and p5, in Figure 13.3 are all absorbing branching places, p1 because of transitions t1 and t2, p2 because of t6, p4 because of t7 and p5 because of t10. On the other hand, places p4 and p5 of Figure 13.1 are branching places but not absorbing branching places.

Definition 13.6: Initially marked place

The place in a decision process which is initially marked is called the *initially marked place*.

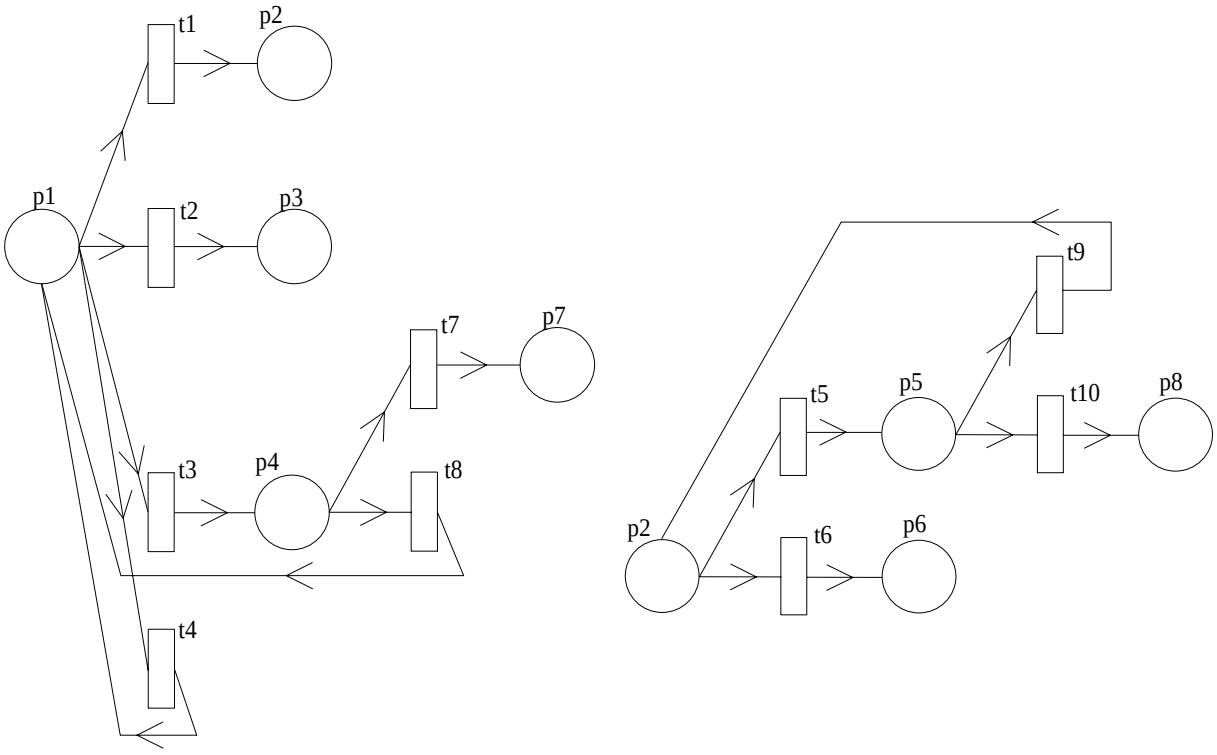
This definition can hold for either a building block or an entire net. For example, for the building blocks of the STPN in Figure 9.1, shown in Figure 9.3, the initially marked places are p1 and p3 while for the entire net the initially marked place is p1. The decision process building block shown in Figure 9.2 has the initially marked place I.

Definition 13.7: Absorbing sub-branching place

An *absorbing sub-branching place* with respect to place p is a sub-branching place with respect to p for which at least one of its output transitions is a non-re-enabling transition.

The set of absorbing sub-branching places for a given STPN is a sub-set of the net's sub-branching places.

In Figure 13.3 place p4 is an absorbing sub-branching place with respect to place p1 and vice versa. Also place p5 is an absorbing sub-branching place with respect to place p2 and vice versa. Note that place p2 is not an absorbing sub-branching place with respect to place p1 as it is not part of a circuit involving p1. When the absorbing sub-branching place is with respect to the initially marked place, then it is referred to as an absorbing sub-branching place. So place p4 of both Figure 13.3 and Figure 13.4(a) are absorbing sub-branching places. Also place p5 of Figure 13.4(b) is an absorbing sub-branching place.



(a) First building block of Figure 13.3 (b) Second building block of Figure 13.3

Figure 13.4: Building blocks of the STPN in Figure 13.3

Definition 13.8: Initial component

The set containing the initially marked place and its absorbing sub-branching places is called the *initial component.*

In Figure 13.3 and Figure 13.4(a) the initial component is {p1, p4}. The STPN in Figure 13.4(b) the initial component {p2, p5}.

The algorithm in Section 12.5 did not include any way of defining the decision process building blocks. This is because any STPN built from the decision process building blocks of

the form shown in Figure 9.2 is such that the algorithm automatically goes through each of its building blocks without explicitly defining them. This is also true in the case when circuits contain circuits as defined in Section 13.1. However, if there are absorbing sub-branching places then it is very important that the decomposition is carried out correctly. It is the decomposition that allows this extended structure to be analysed.

The first step in analysing a STPN with absorbing places as part of its circuits is to divide the STPN into similar building blocks as described in Section Chapter 9. As before the central element of any building block is its initially marked place, shown as I in Figure 9.2. For the STPN in Figure 13.3, places p1 and p2 are the initially marked places of the two building blocks. Once these places have been determined the building blocks can be defined around them.

For simplicity consider the case when the STPN's initially marked place is an absorbing branching place. If this is not the case then the following process can be started at the first absorbing branching place reached. The building blocks can be constructed by using an algorithm based around the initially marked places.

1 The STPN's initially marked place is the initial entry in the initially marked place list.

2 For each place, p, in the initially marked place list:

  2.1 The place p is the initially marked place of a newly formed building block.

  2.2 For each non-re-enabling transition enabled by a token in place p:

    2.2.1 This transition is added to the list of transitions in A for the building block.

    2.2.2 The output place of this transition is an absorbing place of the building block.

    2.2.3 The output place of this transition is added to the initially marked place list.

2.3 Any transition part of a possible firing sequence that does not contain a transition in R forms part of the C system of this building block.

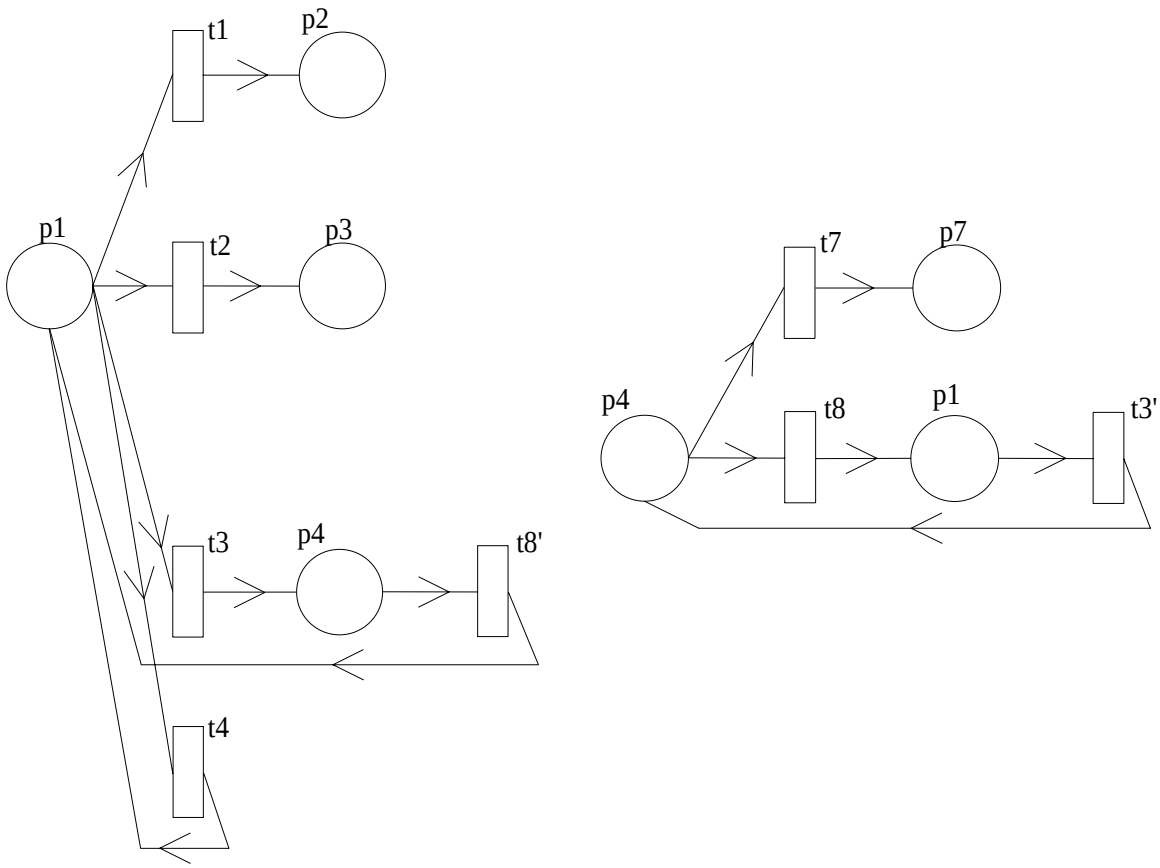
Thus each building block contains the initially marked place, all non-re-enabling transitions enabled by a token in this place and their output places, and all parts of the net that are in some way connected to the circuits of the initially marked place. Figure 13.4 shows the building block breakdown of the STPN shown in Figure 13.3.

It should be noted that the building blocks are initially marking dependent. For example, if place p4 was the initially marked place of the STPN in Figure 13.3, then there would only be one building block involving the entire net. This is because the only one non-re-enabling transition is enabled when there is a token in place p4 and its output place is an absorbing place.

Having determined the decision process building blocks, next each of the different absorbing processes within the building block is separated into different STPNs, referred to as components. Each element, p, of the initial component set forms a different component with p as its initially marked place. For each place, p', which is an absorbing sub-branching-place with respect to p, all but one of p' conflict transitions are removed. The transition that remains is the one in the circuit containing place p.

Consider the building block shown in Figure 13.4(a). Since its initial component set has 2 elements, p1 and p4, two components must be constructed. The first component, which has p1 as its initially marked place, is shown in Figure 13.5(a). In this case the set of output transitions of the absorbing sub-branching place p4 is reduced to the single transition t8'.

Figure 13.5(b) shows the second component, which has p4 as its initially marked place and the absorbing sub-branching place p1, has all its output transitions except for t3 removed.



(a) First component of Figure 13.4(a) (b) Second component of Figure 13.4(a)

Figure 13.5: Two components of STPN in Figure 13.4(a)

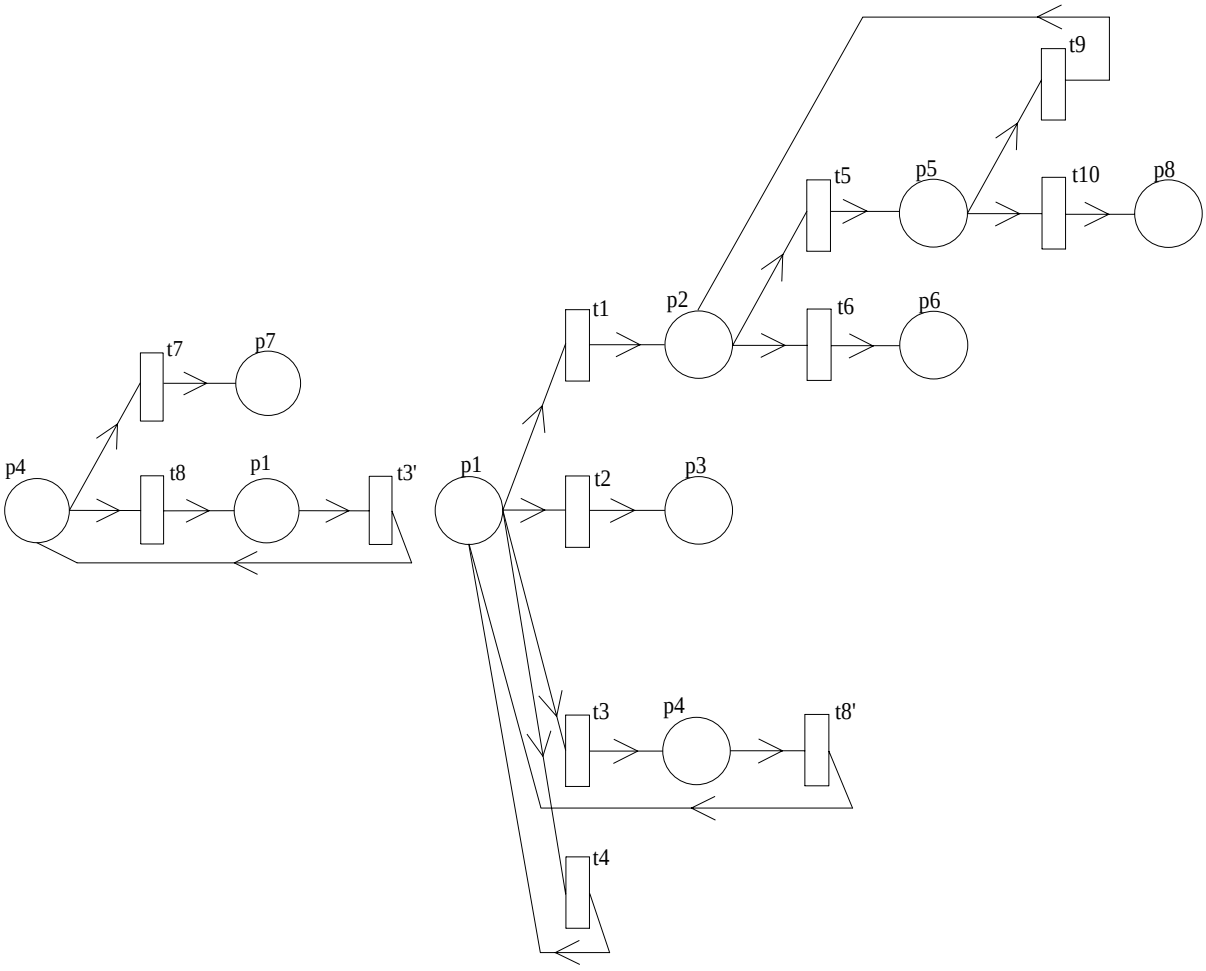
The removed transitions cannot be ignored as they will influence the reward gained from the circuit, as well as change the probability of reaching a given absorbing marking. Thus the probability of staying in the circuit must be calculated along with the mean reward gained in the circuit conditional on staying in the circuit.

The best time to calculate the influence of the removed part of the net is during its removal. This is done by considering the initial component of each building block. Let t be the transition that is not removed in the construction of the component STPN and p_t be the output place of this transition. What needs to be calculated is the probability of firing transition t1 (given by Corollary 12.1) and mean conditional absorption reward of place p_t (given by Corollary 12.2). The absorption probability calculated here is the probability of remaining within the circuit. This process may be repeated multiple times. Each time it is repeated the different absorbing probabilities are multiplied by the value calculated so far to form an overall probability of remaining in the component. The mean conditional absorption reward in place p_t is used in the component as the mean reward gained in the firing of transition t. The application of Corollary 12.2 to the component (with the new mean reward gained by firing t) gives the mean conditional absorption reward of the absorbing places remaining in the component. When Corollary 12.1 is applied to the component the result is multiplied by the probability of remaining in the component to give the probability of absorption for each absorption place in the component.

Consider the case when the absorbing branching place of a component is not the initially marked place of the building block. For example, the absorbing branching place of the component in Figure 13.5(b) is p4 while the initially marked place of this components building block is p1. In this case the mean conditional reward to get from the initially marked place of the building block to the absorbing branching place of the component must be added to the mean conditional absorption reward of the component. In Figure 13.5(b) this means the mean conditional reward gained by firing transition t3’ is added to the mean conditional absorption reward of p7 given by the component in Figure 13.5(b). As before a recursive

algorithm can be defined to determine the absorption probabilities and mean conditional absorption rewards of all absorbing branching places.

There is one further matter to consider. In some cases the components must be broken into building blocks and then components. This breakdown must continue until the system is in a form that can be analysed. Consider once more the STPN in Figure 13.3 where the initially marked place is p4. In this case initially there is only one building block which becomes the two components shown in Figure 13.6. Clearly the component shown in Figure 13.6(b) is made up of two building blocks with the initially marked places p1 and p2. The second building block of this component is made up of two components. This illustrates how such a breakdown may occur.



(a) First component (b) Second component

Figure 13.6: Components of the STPN in Figure 13.3 with initially marked place p4

13.3 Crossovers

Consider the case when there is more than one non-re-enabling transition that has the same output place. This is a situation that has not been considered yet, but, it is not difficult to allow for. Indeed this has already been done implicitly in a number of places. To illustrate this consider the STPN in Figure 13.7. In this example the absorbing branching place p3 can be

reached by the two non-re-enabling transitions t1 and t4. Once more some definitions will make it easier to describe how to analyse this type of structure.

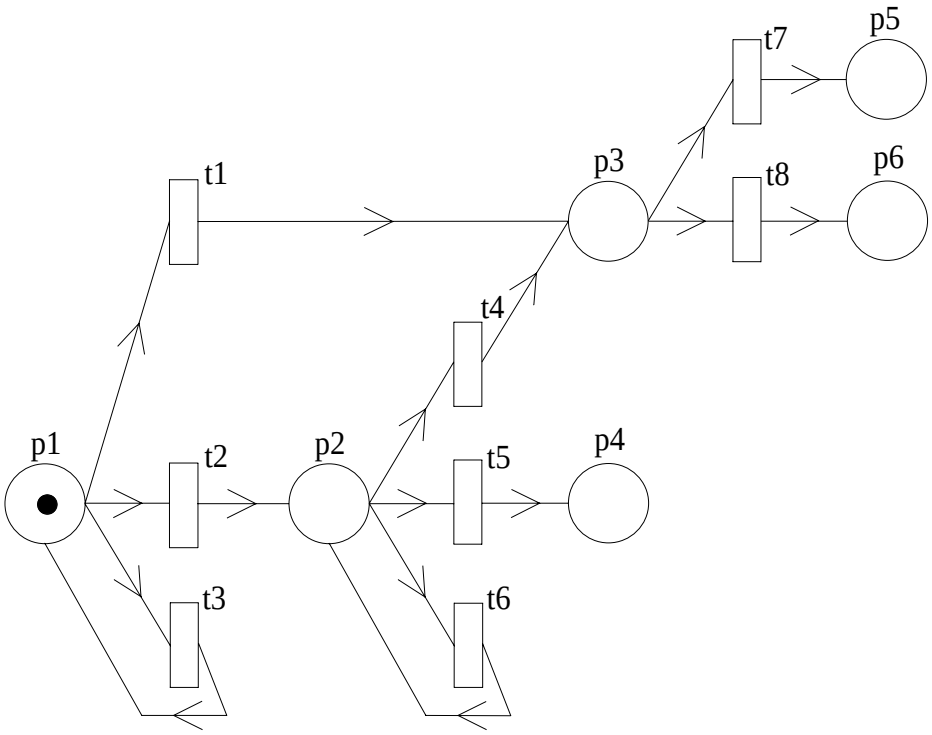


Figure 13.7: STPN with crossover place

Definition 13.9: Crossover place

A *crossover place* is a place that is the output place of more than one non-re-enabling transition.

Crossover places can be identified by considering the number of input transitions associated with a branching place. Given that the circuits of a STPN have been reduced to loops, then a place is a crossover place if there exists t and t'$\in$T, where t$\neq$t', such that

W(t, p) = W(t', p) = 1 and W(p, t) = W(p, t') = 0. Clearly p3 in Figure 13.7 is a crossover place since W(t1, p3) = W(t4, p3) = 1 and W(p3, t1) = W(p3, t4) = 0.

From now on a building block will be referred to by its initially marked place. Thus the building block in Figure 9.3(a) is referred to as the p1 building block.

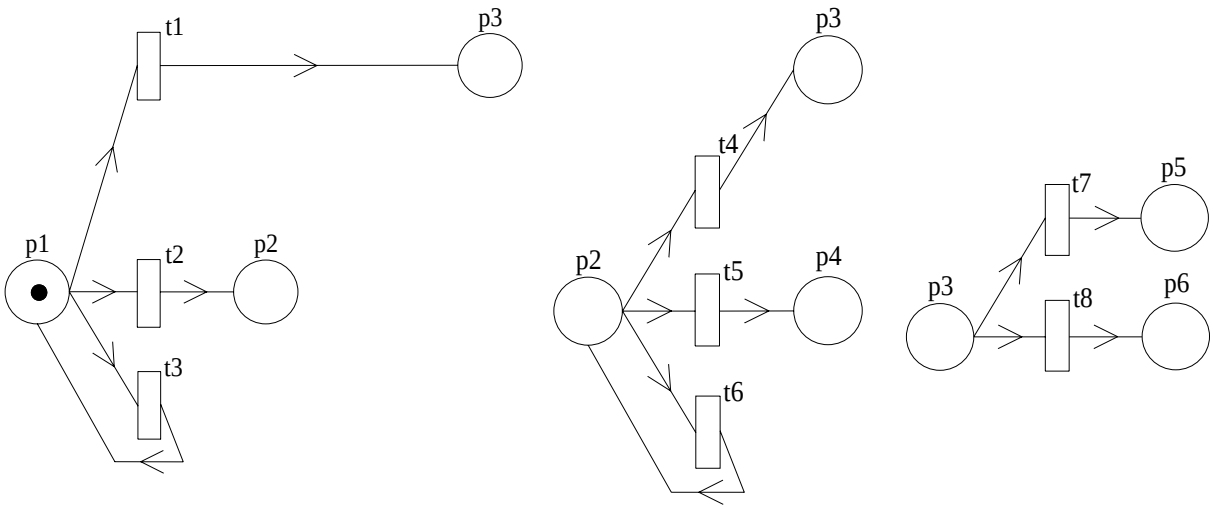
Definition 13.10: Upstream building block

An *upstream building block* with respect to the place p is a building block for which p is an absorbing place.

Also define a relation $\succ$ such that $p \succ p'$ if the building block p is upstream with respect to p'.

This definition holds not only for STPNs with crossover places but for all STPNs. The only difference here is that now there may be more than one upstream building block while previously there was a maximum of one. The STPN in Figure 9.1 has two building blocks, shown in Figure 9.2, and clearly the p1 building block is an upstream building block with respect to place p3. Thus $p1 \succ p3$.

The first step in analysing a decision processes with crossover places is the same as before, to break the STPN into its different building blocks. The building blocks of the STPN in Figure 13.7 are shown in Figure 13.8. Once the building blocks have been determined they can be analysed independently, as was done in Chapter 12, using Corollary 12.1 and Corollary 12.2 to determine the local absorption probabilities and mean conditional absorption rewards.



(a) First building block (b) Second building block (c) Third building block

Figure 13.8: Building blocks for STPN in Figure 13.7

Next the building blocks are ordered. The main aim in this step is to determine which building blocks are upstream of each other building block and thus how to combine analysis of the individual building blocks to give the overall absorption probabilities and mean conditional absorption rewards.

Clearly for the STPN in Figure 13.7 $p1 \succ p2$, $p1 \succ p3$ and $p2 \succ p3$. So to calculate the absorption probabilities and mean conditional absorption rewards of the absorbing places of building block p3 the fact that p3 can be reached via more than one other building block must be considered. That is, to calculate the absorption probability and mean conditional absorption reward of places p5 and p6 all three building blocks must be considered while to calculate the same values for p4 only building blocks p1 and p2 need to be considered.

The absorption probability of place p is given by adding the local absorption probability of p for each of the building blocks that are upstream with respect to p. Thus in the case of the building block in Figure 13.8(c) this means adding the local absorption probability of place p3 from building block p1 and building block p2. In the case of building block p2 this is the product of the absorption probability of place p2 from building block p1 and the absorption probability of place p3 for building block p2. Thus

P(p3) = P(p3 from p1) + P(p2 from p1) * P(p3 from p2),

where P(p from p') is the absorption probability of place p for building block p'.

The mean conditional absorption reward of place p is given by summing the mean conditional absorption rewards of p for each of its upstream building blocks multiplied by the probability of taking that route to p. So for the building block in Figure 13.8(c) this is a combination of the local mean conditional absorption reward of place p3 directly from building block p1 and the mean reward via building block p2. Thus

E[p3] = E[p3 from p1] * P(p3 from p1) +

(E[p3 form p2] + E[p2]) * P(p2 from p1) * P(p3 from p2),

where E[p from p'] is the mean conditional first-hitting reward of p in the building block p'.

Thus to analyse a STPN decision process with crossovers it is a matter of first ordering the building blocks and then combining the local absorbing probabilities and local mean conditional absorption rewards in the appropriate manner to get the overall values.

CHAPTER 14 PART II CONCLUDING REMARKS

Unfortunately the direct analysis techniques developed in this part can be used only for a sub-class of STPNs. When a decision process cannot be modelled using the decision process building blocks, then the direct analysis techniques cannot be used for this part of the model. The best way around this is to aggregate the non-conforming actions into a single action so that the building block structure can be maintained. The aggregated actions must first be analysed separately so that their properties are not lost. Hopefully this analysis can be done directly on the STPN model. If this is not possible then the state space has to be used. To aid in the analysis of such state spaces some useful results have been given. These results define how to calculate the absorption probabilities and mean conditional absorption times of discrete, continuous and semi-Markov chains. The results can also be used to analyse a general STPN provided the process it defines on its state space forms a Markov chain.

There is one final analysis method that should be mentioned. This is the use of simulation. Although not ideal it should be recognised as a useful way of analysing large models that can not be dealt with any other way.

CHAPTER 15 CONCLUSIONS

The aim of this thesis is to present a way of modelling and analysing the decision processes of command and control systems. To do this required a modelling technique that could represent variable outcomes for the same initial state, timed and non-timed events, modular design, concurrent and sequential activities and resource sharing. The first three of these issues were of particular interest and were the focus of the work presented.

Part I presented a comprehensive study of the ways of representing time in PNs. It presented a unified way of representing time in PNs through so-called STPNs. This representation can represent both HTPNs and ETPNs in a very flexible way. The STPN is then extended to include rewards to allow more general costs to be modelled. The decision process building block was then introduced as a STPN model. This is the basic building block used to build up a complex series of decisions.

The other issue considered in this thesis was determining ways of analysing the decision process building blocks. Two areas of interest were identified: course of action analysis and model validation.

The first of these looks at the problem of determining the best course of action. To do this there needs to be a way of determining the cost of making a certain set of decisions. Thus the average cost of reaching a certain state, given it is reached, for a model of the decision process needs to be calculated. Thus theories and techniques for determining the mean conditional absorption reward were developed.

The other area of interest is the validation of decision process models that are to be used as part of larger simulations and models. In the case of validating models there needs to be a way of comparing the model to the real system. Two good measures for this are the probability of reaching a given state and the mean time it takes to reach a state for the first time, given it is reached. Due to the general approach of using rewards, the theory required for the mean conditional absorption time is simply a special case of rewards. In addition, theories and techniques for calculating the absorption probability for decision processes were developed.

An appealing feature of the results is that they hold in a more general setting. In fact the mean conditional absorption results are actually results for the mean condition first-hitting reward and time. This means that these results are not restricted to absorption states, which is useful when considering ways of extending how decision process building blocks are combined, as covered towards the end of the thesis.

One of the problems with the decision process building block analysis is that it relates only to a restrictive class of STPNs. This is an issue often faced by analysis methods applied directly to a PN instead of its state space. However, this has been covered by the fact that the calculation of the mean conditional first-hitting reward for discrete, continuous and semi-Markov chains has also been developed in this thesis. This appears to be a completely new area of analysis for Markov chains and although not presented here can most likely be extended to more general Markovian processes. The Markov chain results mean that provided the process defined on the state space of a model is a Markov chain it is possible to give equations prescribing the required results. Also due to the modular modelling approach taken it is only those modules of the net that do not conform to the building block that need to be

analysed through their state spaces. Of cause if all else fails then simulation can be used to help determine the desired values.

Clearly there are still some areas of future work. These relate to the extension of the analysis to a wider range of problems.

As stated earlier, this thesis has focused on three of the five modelling capabilities required. The two that have been largely unconsidered are those of concurrency and resource sharing. These issues are certainly catered for in STPNs models but the analysis results presented to date do not allow for them. Thus if one wishes to have these features represented then the analysis needs to be carried out on the state space. It is hoped that more direct analysis techniques can be explored. Also, as alluded to earlier there is potential to extend the Markov chain results to a larger range of Markovian systems. This will also further extend the types of models that can be analysed.

One of the issues with the use of STPNs is that the models can become very large. Management of these large models then becomes an issue. This is partially dealt with through using a modular design approach but can still result in complex and hard-to-manage models. This is a general problem with PNs which is a major motivation for the development of high level PNs. The aim is to extend both the time representation and analysis results in this thesis to high level PNs.

Finally on the validation side of the analysis there is a need to implement these results with a tool such as that described in [28]. This would further enhance the ability of an expert to determine if the model is a valid representation of the process being modelled. Such a tool

would also be an ideal way of implementing the general analysis techniques presented in this thesis.

Although this thesis has focused on the modelling of decision processes the results are not seen to be tied to this application. There are many systems for which the STPN structure would provide a good modelling technique. Also the mean conditional first-hitting reward is applicable not only to decision processes, but also to any system where there is an interest in the mean reward to reach a given state where there is no certainty that the state of interest will be reached.

APPENDIX I FIRING TIME PETRI NETS

I.1 Firing Durations Definition

The use of firing durations to represent time in PNs was introduced by Ramchandani ([143]) and has since been used in [81], [121], [142], [166], [167], [182] and [184] and many other papers. In firing time Petri nets (FTPN) firing durations are associated with the transitions. These values relate to the times taken for the events, represented by the transitions, to occur. To illustrate how FTPNs work, consider the example in Figure I.1, which depicts a simple FTPN with deterministic firing durations (D_d) associated with each of its transitions.

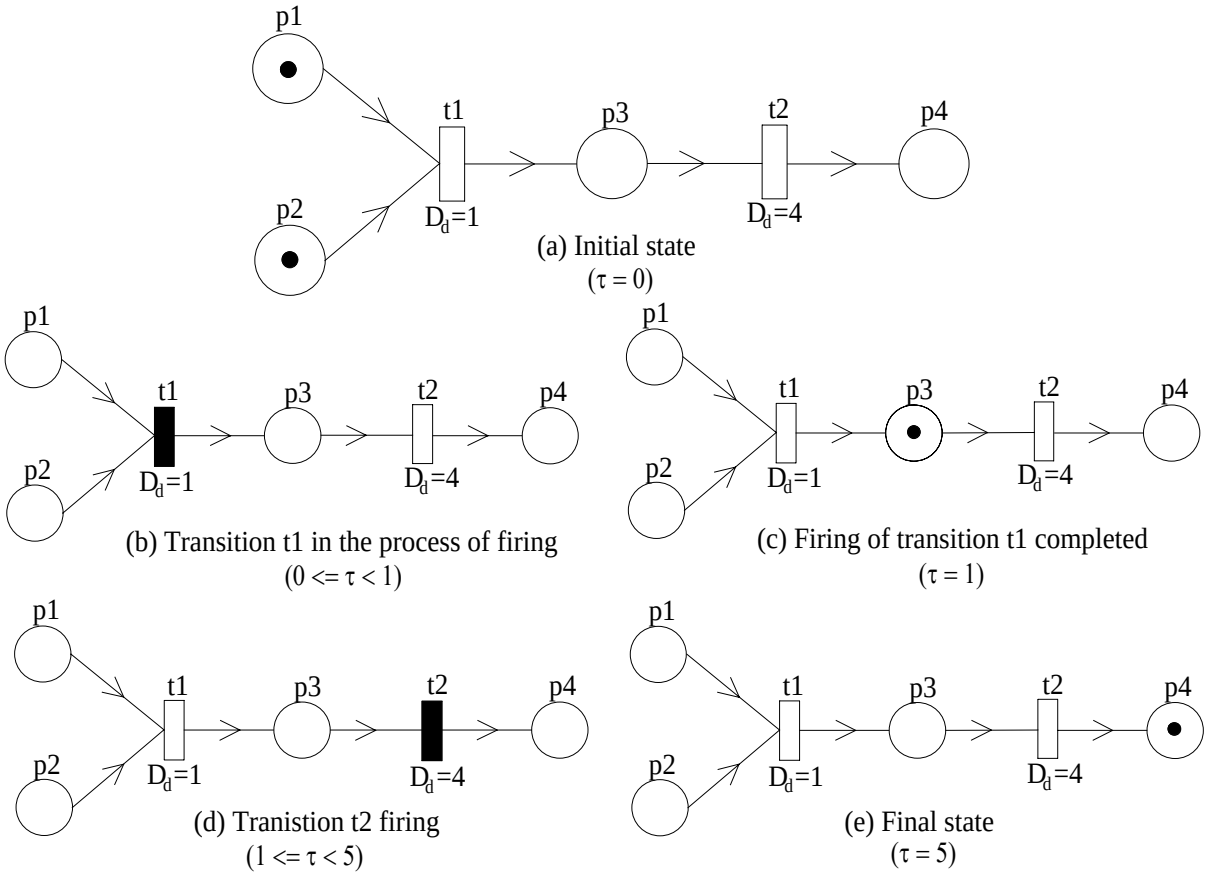


Figure I.1 TPN with firing durations

Figure I.1(a) shows the initial state of the FTPN. The FTPN shown in Figure I.1(a) is structurally the same as the HTPN in Figure 3.1(a). With this initial marking transition t1 is enabled and as expressed above immediately begins to fire, removing the tokens from the input places, p1 and p2. This state is shown in Figure I.1(b). In Figure I.1 a filled transition indicates a transition in the process of firing. Transition t1 completes firing when $\tau = 1$, at which time the output tokens are created, in this case a single token in place p3 (see Figure I.1(c)). Transition t2 is now enabled. In the case of the HTPN this is due to the tokens becoming available at this time while for the FTPN the enabling of t2 was due to the token being created in place p3 at this time. Transition t2 now goes through a similar process as described for the firing of transition t1, as shown in Figure I.1(d) and (e). Thus the final state and the time it is reached are the same as that for the HTPN example in Figure 3.1.

Changes to the PN semantics resulting from the introduction of firing times can be seen from the simple example in Figure I.1. Not only have the PN execution rules changed from those of a PN, but also new states are created. These new states are shown in Figure I.1(b) and Figure I.1(d). This creates a need to redefine the state space (reachability graph) generated by the PN. Each state of a FTPN is now defined not only by the marking, but also by the condition of its transitions. Thus the marking can be considered to be made up of two parts; the number of tokens in each place, ${}^{P}M$, and the firing time of each transition, ${}^{F}M$. So ${}^{P}M$ is a mapping: $P \rightarrow Z_0^+$. Also ${}^{F}M$ is a mapping: $T \rightarrow \rho$, where ${}^{F}M(t)$ gives the time at which transition t completes its firing.

As with the other TPNs, a FTPN can be represented by a multi-tuple. In this case the 6-tuple

FTPN = (P, T, W, D, M0)

can be used. In this definition P, T, W and D are as given in the main part of the text. In this case the duration functions are associated with the transitions. As with other TPNs, M0 is the initial marking of the FTPN, only now it is the union of the ${}^{P}M0$ and ${}^{F}M0$, where for place $p \in P$, ${}^{P}M0(p)$ is the initial number of tokens in place p and for transition $t \in T$, ${}^{F}M0(t)$ is the time at which the transition t completes firing. In general ${}^{F}M0$ will be infinity for all the transitions, that is, initially none of the transitions are in the process of firing. Thus the TPN in Figure I.1 has the tuple definition:

$P = \{p1, p2, p3, p4\}$,

$T = \{t1, t2\}$,

$W(p1, t1) = 1, W(p2, t1) = 1, W(t1, p3) = 1, W(p3, t2) = 1, W(t2, p4) = 1$,

$D_d(t1) = 1, D_d(t2) = 4$,

${}^{P}M0(p1) = 1, {}^{P}M0(p2) = 1$.

As before weighted flows of zero and the marking of places with no tokens are not included in the definition. Also only the finite ${}^{F}M0$ elements of any marking are given in the FTPN tuple definition.

The semantics of FTPNs can be defined using the above tuple representation. The enabling of the transitions requires two conditions to hold. The first is the same as for PNs, that is,

for all $p \in {}^{\bullet}t$, ${}^{P}M(p) \geq W(p, t)$.

The second condition requires the transition not to be currently firing, that is,

for $t \in T$, ${}^{F}M(t)$ is not defined.

Thus a transition is enabled if there are sufficient tokens in the input place (as with PNs) and it is not currently in the process of firing.

The firing of transition t enabled at time x is done in two stages. First at time x when the transition becomes enabled the marking is changed so that

for all $p \in {}^{\bullet}t$, ${}^{P}M'(p) = {}^{P}M(p) - W(p, t)$ and

${}^{F}M'(t) = x+y$, where y is the firing time of transition t and is generated by the duration function D(t).

So once a transition is enabled it begins to fire by removing the specified tokens from its input places and sets the firing time of the transition to form the new marking. The second stage of firing occurs at time ${}^{F}M'(t)$ and is defined by

for all $p \in t^{\bullet}$, ${}^{P}M''(p) = {}^{P}M'(p) + W(t, p)$ and

${}^{F}M''(t)$ is not defined.

Thus once the firing time has been reached, the output tokens are created in the output places and the transition's firing time is set to infinity. The first step in this process will be referred to as the start of the firing of the transition and the second as the end of the firing of the transition. It should be noted that there is the possibility that the end of the firing of a transition, t say, may not be the event to that directly follows the start of firing transition t. That is, if enabled (firing), another transition may begin (end) firing during the firing process of transition t.

The above definition is the one found in the majority of the literature and assumes that the same transition cannot be enabled while it is in the process of firing, that is, concurrent firings of the same transition cannot occur. Some papers, such as [166] and [167], allow for this but they are in the minority. To allow for concurrent firings of the same transition ${}^{F}M$ must map to multiple firing times. Thus, ${}^{F}M$ is now a mapping from T to a multi-set of firing times. The semantics must also be changed. First the second enabling condition, relating to ${}^{F}M$, is

removed. The firing definition is now such that if transition t is enabled at time x, then the first stage of the transition firing is defined by

for all $p \in {}^{\bullet}t$, ${}^{P}M'(p) = {}^{P}M(p) - W(p, t)$ and

${}^{F}M'(t) = {}^{F}M(t) \cup \{x+y\}$, where y is as defined earlier.

Thus the first part is the same as before except now the transition part of the marking can have multiple entries for a given transition. The second stage of the firing is defined to occur at x+y by

for all $p \in t^{\bullet}$, ${}^{P}M''(p) = {}^{P}M'(p) + W(t, p)$ and

${}^{F}M'(t) = {}^{F}M(t) \backslash \{x+y\}$.

Assigning holding durations to transitions is the same way of representing time as firing durations, the only difference being that in FTPNs the output tokens are 'held' in the transitions, while in HTPNs they are 'held' in the places. From this point on there will be no distinction made between holding and firing durations. Holding durations will be used as they have greater flexibility in their representation allowing the assignment of time to transitions, places or output arcs.

I.2 Comparing Firing and Holding Durations.

The equivalence of these two representations can be shown using the respective state space definitions. Consider the case where there are two TPNs, one a FTPN and the other a HTPN, with equivalent definitions of P, T, W and D. In this case clearly ${}^{P}M$ of the FTPN marking definition is equivalent to ${}^{A}M$ of the HTPN marking definition. The only thing that needs to be shown is that an equivalence transformation exists to transform ${}^{F}M$ of the FTPN definition to ${}^{U}M$ of the HTPN definition and vice versa.

First consider the transformation from ${}^{F}M$ to ${}^{U}M$. Define the transformation function F2H which maps ${}^{F}M$ to ${}^{U}M$. This function is defined such that, for all $t \in T$ and $s \in {}^{F}M(t)$, F2H(t, s) creates the element (W(t, p), s) in ${}^{U}M(p)$ for each $p \in t^{\bullet}$. Thus any FTPN state space can be transformed into the equivalent HTPN state space.

The transformation in the other direction cannot be defined quite as easily. In fact the function from ${}^{U}M$ to ${}^{F}M$, H2F, cannot be defined unless the definition of ${}^{U}M$ is expanded to include the transition that created each entry in ${}^{U}M$. If this is include,d then H2F is defined so for each set $t^{\bullet}$ the set of elements (W(t, $\bullet$), s, t) of ${}^{U}M$, where there is one element from ${}^{U}M$ is removed for each $p \in t^{\bullet}$, is mapped such that the element s is added to ${}^{F}M(t)$. Thus one element is removed from the ${}^{U}M$ for each output place of the transition t. These elements must each have the same time at which their tokens become available, the same creator transition t and a weight given by W(t, $\bullet$).

REFERENCES

[1] W.M.P. van der Aalst, **Interval Timed Coloured Petri Nets and their Analysis**, Lecture Notes in Computer Science Vol. 691, Application and Theory of Petri Nets 1993, Chicago, USA, June 1993.

[2] W.M.P. van der Aalst and M.A. Odijk, **Analysis of Railway Stations by Means of Interval Timed Coloured Petri Nets**, Real-Timed Systems, Vol. 9, 1995, pp. 241-263.

[3] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte and A. Cumani, **On Petri Nets with Stochastic Timing**, Proceedings of the International Workshop on Timed Petri Nets, Torino, Italy, July 1985, pp. 80-87.

[4] M. Ajmone Marsan and G. Chiola, **On Petri Nets with Deterministic and Exponentially Distributed Firing Times**, Lecture Notes in Computer Science 266, Advances in Petri Nets 1986, pp. 132-145.

[5] M. Ajmone Marsan, G. Balbo, G. Chiola and G. Conte, **Generalized Stochastic Petri Nets Revisited: Random Switches and Priorities**, International Workshop on Petri Nets and Performance Models, Madison, Wisconsin, 24-26 August 1987, pp. 44-53.

[6] M. Ajmone Marsan, **Stochastic Petri Nets: An Elementary Introduction**, Lecture Notes in Computer Science Vol. 424, Advances in Petri Nets, 1989, pp. 2-29. Springer Verlag.

[7] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte and A. Cumani, **The Effect of Execution Policies on the Semantics and Analysis of Stochastic Petri Nets**, IEEE Trans. on Software Engineering, Vol. 15, No. 7, July 1989, pp. 832-846.

[8] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, **Modelling with Generalized Stochastic Petri Nets**, Wiley Series in Parallel Computing, John Wiley & Sons Ltd. 1995.

[9] H.H. Ammar and R.W. Liu, **Analysis of the Generalized Stochastic Petri Nets by State Aggregation**, International Workshop on Timed Petri Nets, Torino, Italy, 1-3 July 1985, pp. 88-94.

[10] H.H. Ammar and S. Deng, **Time Warp Simulation of Stochastic Petri Nets**, 4th International Workshop on Petri Nets and Performance Models, December 1991, Melbourne, Australia, pp. 186-195.

[11] S.K. Andreadakis and A.H. Levis, **Design Methodology for Command and Control Organizations**, Proceedings of the Command and Control Research Symposium, National Defence University, 1987, Fort Lesley J. McNair, Washington D.C, pp. 255-261.

[12] M.A. Arslan and I. Fidan, **Modeling and Performance Analysis of an SMD Assembly Station Using Stochastic Petri Nets and Artificial Neural Networks**, Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics: Intelligent Systems for the 21st. Century, pp. 1768-1773.

[13] T. Aura and J. Lilius, **Time Processes for Time Petri Nets**, Proceedings of 18th. International Conference on Application and Theory of Petri Nets, 1997, Toulouse, France, pp. 136-155.

[14] D. Azzopardi and S. Lloyd, **Reduction of Search Space for Scheduling of Multi-Product Batch Process Plant through Petri Net Modelling**, Proceedings of the 2nd. IFAC/IFIP/IFORS International Workshop on Intelligent Manufacturing Systems, 10-13 June 1994, Vienna.

[15] F. Baccelli, **Ergodic Theory of Stochastic Petri Networks**, The Annals of Probability, Vol. 20, No. 1, 1992, pp. 375-396.

[16] F.L. Baccelli, G. Cohen, G.J. Olsder and J. Quadrat, **Synchronization and Linearity: An Algebra for Discrete Event Systems**, John Wiley and Sons, 1992.

[17] M. Baldassari, and G. Bruno, **An Environments for Object-Oriented Conceptual Programming Based on PROT Nets**, Proceedings of the 1988 International Conference on Computer Languages, Miami Beach, Florida, 9 - 13 October 1988 pp 226-33.

[18] J. Bechta Dugan, K.S. Trivedi, R.M. Geist and V.F. Nicola, **Extended Stochastic Petri Nets: Applications and Anaysis**, Proceedings of the Tenth International Symposium on Models of Computer System Performance (Permormance '84), 19-21 December 1984, Paris, France, pp. 507-519.

[19] J. Bechta Dugan, A. Bobbio, C. Ciardo and K. Trivedi, **The design of a Unified Package for the Solution of Stochastic Petri Net Models**, Proceedings of the International Workshop on Timed Petri Nets, July 1985, Torino, Italy, pp. 80-87.

[20] J. Bechta Dugan, K.S. Trivedi, M.K. Smotherman and R.M. Geist, **The Hybrid Automated Reliability Predictor**, Journal Of Guidance Control And Dynamics, 1986, Vol. 9, No. 3, pp. 319-331.

[21] J. Bechta Dugan and K.S. Trivedi, **Coverage Modeling for Dependability Analysis of Fault-Tolerant Systems**, IEEE Transactions On Computers, 1989, Vol. 38, No. 6, pp. 775-787.

[22] J. Berger and L. Lamontagne, **A Colored Petri net model for a naval command and control system**, Lecture Notes in Computer Science Vol. 691, Proceeding of the Conference on Application and Theory of Petri Nets, June 1993, pp. 532-541.

[23] B. Berthomieu and M. Menasche, **A State Enumeration Approach for Analyzing Time Petri Nets**, Proceedings of the 3rd European Workshop on Applications and Theroy of Petri Nets, 27-30 September 1982, Varenna, Italy, pp. 27-65.

[24] B. Berthomieu and M. Menasche, **An Enumerative Approach For Analyzing Time Petri Nets**, Proceedings of the IFIP 9th. World Computer Congress (Information Processing 83), 19-23 September 1983, Paris, France, pp. 41-46.

[25] B. Berthomieu, M. Diaz, **Modeling and Verificaion of Time Dependent Systems Using Time Petri Nets**, IEEE Transactions on Software Engineering, Vol. 17, No. 3, March 1991, pp. 259-273.

[26] A. Bobbio, A. Puliafito and M. Telek, **New Primitives for Interlaced Memory Policies in Markov Regenerative Stochastic Petri Nets**, Proceeding of the 7th International Workshop on Petri Nets and Performance Models, 3-6 June 1997, Saint Malo, France, pp. 70-79.

[27] H. Boucheneb and G. Berthelot, **Towards a Simplified Building of Time Petri Nets Reachability Graph**, Proceedings of the 5th International Workshop on Petri Nets and Performance Models, October 1993, Toulouse, France, pp. 46-55.

[28] F.D.J. Bowden, M. Davies and J.M. Dunn, **Representing Role-Based Agents Using Coloured Petri Nets**, Proceedings of the 5th Conference on Computer Generated Forces and Behavioural Representation, May 1995, Orlando, Florida, pp. 513-520.

[29] F.D.J. Bowden, M. Davies and J.M. Dunn, **Role-based Extended Petri Net Models and their Application**, Proceedings of The International Congress on Modelling and Simulation 1995, 27-30 November 1995, Newcastle, Australia, pp. 296-301.

[30] F.D.J. Bowden, **Modelling Time in Petri Nets**, Proceedings of the 2nd Australia-Japan Workshop on Stochastic Models in Engineering, Technology and Management, Gold Coast, Australia, 17-19 July 1996, pp. 66-80.

[31] F.D.J. Bowden, C. Gabrisch and M. Davies, **Simulation of Air Defence C3I Using the Distributed Interactive C3I Effectiveness (DICE) Simulation**, Proceedings of the 2nd International Simulation Technology and Training Conference (SimTecT '97), Canberra, Australia, 17-20 March 1997, pp. 71-76.

[32] F.D.J. Bowden, C. Gabrisch and M. Davies, **C3I Systems Analysis Using The Distributed Interactive C3I Effectiveness (DICE) Simulation**, Proceedings of the 1997 IEEE International Conference on Systems, Man and Cybernetics, Orlando, Florida, USA, 12-15 October 1997, pp. 4326-4331.

[33] F.D.J. Bowden and M. Davies, **Application of a Role-Based Methodology to Represent Command and Control Processes Using Extended Petri Nets**, Proceedings of the 1997 IEEE International Conference on Systems, Man and Cybernetics, Orlando, Florida, USA, 12-15 October 1997, pp. 4348-4353.

[34] F.D.J. Bowden and M. Davies, **Design Methodology and Implementation Technologies for C3I Artificial Agents**, Proceedings of the 3rd international Simulation Technology and Training Conference (SimTecT '98), Adelaide, Australia, 2-6 March 1998, pp. 111-115.

[35] F.D.J. Bowden and C.E.M. Pearce, **Some Questions Involving First-Passage Problems in Markov Processes**, Proceedings of the 17th Australian Society of Operations Research Conference (ASOR17), 4-7 July 1999, Gold Coast, Australia, pp. 222-231.

[36] F.D.J. Bowden and C.E.M. Pearce, **Decision Process Petri Net Analysis**, Proceedings of the 5th International Command and Control Research and Technology Symposium, 24-26 October 2000, Canberra, Australia.

[37] F.D.J. Bowden, **A Brief Survey and Synthesis of the Roles of Time in Petri Nets**, Mathematical and Computer Modelling, Vol. 31, No. 10-12, May-June 2000, pp. 55-68.

[38] F.D.J. Bowden and C.E.M. Pearce, **Conditional First Hitting Times in Markov Processes**, Submitted.

[39] G. Bucci and E. Vicario, **Compositional Validation of Time-Critical Systems Using Communicating Time Petri Nets**, IEEE Transactions on Software Engineering, Vol. 21, No.12, December 1995, pp 969-991.

[40] P. Buchholz, **Aggregation and Reduction Techniques for Hierarchical GCSPNs**, Proceedings of the 5th International Workshop on Petri Nets and Performance Models, October 1993, Toulouse, France, pp. 216-225.

[41] P. Buchholz, **Hierarchical High Level Petri Nets for Complex System Analysis**, Lecture Notes in Computer Science, Vol. 815, Application and Theory of Petri Nets 1994, June 1994, pp. 119-138.

[42] J. Campos, G. Chiola, J.M. Colom and M. Silva, **Tight Polynomial Bounds for Steady-State Performance of Marked Graphs**, Proceedings of the IEEE 3rd International Workshop on Petri Nets and Performance Models, 11-13 December 1989, Kyoto, Japan, pp. 200-209.

[43] J. Campos, J.M. Colom, H. Jungnitz and M. Silva, **Approximate Throughput computation of Stochastic Marked Graphs**, IEEE Transactions on Software Engineering, Vol. 25, No. 7, July 1994, pp. 526-535.

[44] I.R. Chen, **Stochastic Petri-Net Analysis Of Deadlock Detection Algorithms In Transaction Database-Systems With Dynamic Locking**, Computer Journal, Vol. 38, No. 9, 1995, pp. 717-733.

[45] G. Chiola and G. Fanceschinis, **Colored GSPN Models and Automatic Symmetry Detection**, Proceedings of the 3rd International Workshop on Petri Nets and Performance Models, December 1989, Kyoto, Japan, pp. 50-60.

[46] G. Chiola and C. Dutheillet and G. Franceschinis and S. Haddad, **On Well-Formed Coloured Nets and Their Symbolic Reachability Graph**, High-level Petri Nets, Editors K. Jensen and G. Rozenberg, Springer-Verlag, 1991, pp. 373-396

[47] G. Chiola, M. Ajmone Marsan, G. Balbo and G. Conte, **Generalized Stochastic Petr Nets: A Definition at the Net Level and its Implications**, IEEE Transactions. on Software Engineering, Vol. 19, No. 2, February 1993, pp. 89-107.

[48] G. Chiola, C. Anglano, J. Campos, J.M. Colom and J. Silva, **Operational Analysis of Timed Petri Nets and Application to the Computation of Performance Bounds**, Proceeding of the 5th International Workshop on Petri Nets and Performance Models, October 1993, Toulouse, France, pp. 128-137.

[49] H. Choi and K.S. Trivedi, **Conditional MTTF and its computation in Markov**, Proceedings of the 1993 Annual Reliability and Maintainability Conference, January 1993, Atlanta, GA, pp. 56-63.

[50] H. Choi, V.G. Kulkarni and K.S Trivedi, **Transient Analysis of Deterministic and Stochastic Petri Nets**, Lecture Notes in Computer Science, Vol. 691, Application and Theory of Petri Nets 1993, June 1993, pp. 166-185.

[51] S. Christensen and L. Petrucci, **Modular State Space Analysis of Coloured Petri Nets**, Lecture Notes in Computer Science, Vol. 935, Application and Theory of Petri Nets, 1995. pp 201-217.

[52] G. Ciardo, A. Blakemore, P.F. Chimento, J.K. Muppala and K.S. Trivedi, **Automated Generation and Analysis of Markov Reward Models Using Stochastic Reward Models**, Linear Algebra, Markov Chains, and Queueing Models, Editors C. Meyer and R.J. Plemmons, IMA Volumes in Mathematics and its Applications, Vol. 48, New York: Springer-Verlag, 1993, pp. 145-191.

[53] G. Ciardo and C. Lindemann, **Analysis of Deterministic and Stochastic Petri Nets**, Proceedings of the 5th International Workshop on Petri Nets and Performance Models, October 1993, Toulouse, France, pp. 160-169.

[54] G. Ciardo, R. German and C. Lindermann, **A Characterization of the Stochastic Process Underlying a Stochastic Petri Net**, IEEE Transactions on Software Engineering, Vol. 25, No. 7, July 1994, pp. 506-515.

[55] G. Ciardo, Nicol and K.S. Trivedi, **Discrete-event simulation of fluid stochastic Petri nets**, Proceeding of the 7th International Workshop on Petri Nets and Performance Models, 3-6 June 1997, Saint Malo, France, pp. 217-225.

[56] G. Ciardo and L. Guangzhi, **Approximate transient analysis for subclasses of deterministic and stochastic Petri nets**, Performance Evaluation, Vol.35, No.3-4, 1999, pp. 109-29.

[57] E. Çinlar, **Introduction to Stochastic Process**, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[58] J.L. Coleman, **Stochastic Petri Nets with Product From Equilibrium Distributions**, PhD Thesis, Department of Applied Mathematics, University of Adelaide, 15 December 1993.

[59] J.E. Coolahan(Jr) and N. Roussopoulos, **Timing Requirements for Time-Driven Systems Using Augmented Petri Nets**, IEEE Transactions on Software Engineering, Vol. SE-9, No. 5, September 1983, pp. 603-616.

[60] J.P. Courtiat and M. Diaz, **Time In State Based Formal Description Techniques For Distributed Systems**, Real-Time: Theory in Practice. Proceedings of the REX Workshop, June 3-7 1991, Mook, The Netherlands, pp. 149-175.

[61] R. David, **Modeling of Hybrid Systems Using Continuous and Hybrid Petri Nets**, Proceeding of the 7th International Workshop on Petri Nets and Performance Models, 3-6 June 1997, Saint Malo, France, pp. 47-58.

[62] D.D. Deavours and W.H. Sanders, **"On-The-Fly" Solution Techniques for Stochastic Petri Nets and Extensions**, Proceeding of the 7th International Workshop on Petri Nets and Performance Models, 3-6 June 1997, Saint Malo, France, pp. 132-141.

[63] A.A. Desrochers and R.Y. Al-Jaar, **Applications of Petri Nets in Manufacturing Systems: Modeling, Control, and Performance Analysis**, IEEE Control Systems Society, IEEE Press, 1995.

[64] M. Diaz and P. Sénac, **Time Stream Petri Nets a Model for Timed Multimedia Information**, Lecture Notes in Computer Science, Vol. 815, Application and Theory of Petri Nets, June 1994, pp.219-238.

[65] C. Dutheillet and S. Haddad, **Conflict Sets in Colored Petri Nets**, Proceedings of the 5th International Workshop on Petri Nets and Performance Models, October 1993, Toulouse, France, pp. 76-85.

[66] J.C.A. de Figueiredo and A. Perkusich, **Faults and timing analysis in real-time distributed systems: A fuzzy time Petri-net-based approach**, Fuzzy Sets and Systems, International Journal of Soft Computing and Intelligence, Vol 83, No. 2, October 1996, pp. 143-168.

[67] G. Franceschinis and R.R. Muntz, **Computing Bounds for the Performance Indices of Quasi-Lumpable Stochastic Well-Formed Nets**, IEEE Transactions on Software Engineering, Vol. 25, No. 7, July 1994, pp. 516-525.

[68] C. Gabrisch F.D.J. Bowden, M. Davies, N.A. Haydon and J. Winkles, **Synthetic Environment Support to Air Asset Visualisation Tool Development**, Proceedings of the 4th International Simulation Technology and Training Conference (SimTect '99), 1999, Melbourne, Victoria, Australia, pp. 281-288.

[69] R.G. Gallager, **Discrete Stochastic Process**, Kluwer Academic Publishers, Boston, Dordrecht, London, 3rd edition, 1999.

[70] P. Garske and V.L. Narasimhan, **Petri Net Modelling and Analysis of Centralised Timeout and Batching Arbitration Units**, Proceedings of the 4th International Parallel Processing Syposium, 1992, Seattle, U.S.A.

[71] R. Geist and K. Trivedi, **Reliability Estimation Of Fault-Tolerant Systems - Tools And Techniques**, Computer, 1990, Vol. 23, No. 7, pp. 52-61.

[72] R. German and C. Lindemann, **Analysis Of Stochastic Petri Nets By The Method Of Supplementary Variables**, Performance Evaluation, May 1994, Vol. 20, No. 1-3, pp. 317-335.

[73] R. German, **Transient Analysis of Deterministic and Stochastic Petri Nets by the Method of Supplementary Variables**, Quality of Communication-Based Systems, Kluwer Academic Publishers, 1995, pp. 105-121.

[74] L. Gomes and A. Steiger-Garcao, **Programmable Controller Design Based on a Sychronized Colored Petri Net Model and Integrating Fuzzy Reasoning,** Lecture Notes in Computer Science, Vol. 935, Application and Theory of Petri Nets, 1995, pp. 218-237.

[75] P.J. Haas and G.S. Shedler, **Regenerative Stochastic Petri Nets**, Performance Evaluation, Vol. 6, 1986, pp. 189-204.

[76] B.R. Haverkort, **Approximate Performability Analysis using Generalized Stochastic Petri Nets**, Proceedings of the 4th International Workshop on Petri Nets and Performance Models, December 1991, Melbourne, Australia, pp. 300-307.

[77] A. Heindl and R. German, **A Fourth-Order Algorithm with Automatic Stepsize Control for the Transient Analysis of DSPNs**, Proceedings of the 7th International Workshop on Petri Nets and Performance Models, 3-6 June 1997, Saint Malo, France, pp. 60-69.

[78] W. Henderson and P.G. Taylor, **Embedded Processes in Stochastic Petri Nets**, IEEE Transactions on Software Engineering, Vol. 17, No.2, February 1991, pp. 108-116.

[79] W. Henderson, **Finding and Using Exact Equilibrium Distributions for Stochastic Petri Nets**, Computer Networks and ISDN Systems, Vol. 25, 1993, pp. 1143-1153.

[80] A.J. van der Hoeven, A.A.J. de Lange, E.F. Deprettere and P.M. Dewilde, **A Model for the High-level Description and Simulation of VLSI Networks**, IEEE Micro, Vol. 10, No. 4, August 1990, pp.41-48.

[81] M.A. Holliday and M.K. Vernon, **A Generalized Timed Petri Net Model for Performance Analysis**, IEEE Transactions on Software Engineering, Vol. SE-13, No. 12, December 1987, pp. 1297-1310.

[82] O.C. Ibe and K.S. Trivedi, **Stochastic Petri Net Models of Polling Systems**, IEEE Journal on Selected Areas in Communications, Vol. 8, No. 9, December 1990, pp1649-1657.

[83] D.L. Isaacson and R.W. Madsen, **Markov chains theory and applications**, Wiley, New York, 1977.

[84] M.D. Jeng, X.L. Xie and S.W. Chou, **Modeling, qualitative analysis, and performance evaluation of the etching area in an IC wafer fabrication system using Petri nets**, IEEE Transactions On Semiconductor Manufacturing, Vol. 11, No. 3, 1998, pp. 358-373.

[85] K. Jensen, **Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Vol.1: Basic Concepts**, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1992.

[86] K. Jensen, **Condensed State Spaces for Symmetrical Coloured Petri Nets**, Formal Methods in System Design, Vol. 9, August 1996, pp. 7-40.

[87] Q. Jin, Y. Yano and Y. Sugasawa, **Non-Regenerative Stochastic Petri Nets - Modeling And Analysis**, IEICE Transactions On Fundamentals Of Electronics Communications And Computer Sciences, 1996, Vol. E79A, No. 11, pp. 1781-1790.

[88] A.H. Jones, M. Uzam and N. Ajlouni, **Design of Discrete Event Control Systems For Programmable Logic Controllers Using T-Timed Petri Nets**, Proceedings of the 1996 IEEE International Symposium on Computer-Aided Control System Design, September 1996, Dearborn, Michigan, USA, pp. 212-217.

[89] G. Juanole and Y. Atamna, **Dealing with arbitrary time distributions with the Stochastic Timed Petri Net model - Application to queueing systems**, Proceedings of the 4th International Workshop on Petri Nets and Performance Models, December 1991, Melbourne, Australia, pp. 32-41.

[90] S. Karlin, **A First Course in Stochastic Processes**, Academic Press, New York, 1966.

[91] J.G. Kemeny, J.L. Snell and A.W. Knapp, **Denumerable Markov chains**, Springer-Verlag, New York, 1975.

[92] P. Kemper, **Transient Analysis of Superposed GSPNs**, Proceeding of the 7th International Workshop on Petri Nets and Performance Models, 3-6 June 1997, Saint Malo, France, pp. 101-110.

[93] J. Kim, A.A. Desrochers and A.C. Sanderson, **Task Planning and Project Management using Petri Nets**, Proceedings of the 1995 IEEE International Symposium on Assembly and Task Planning, pp265-271.

[94] T.S. Kim and K.C. Chae, **A note on the age of the absorbing semi-Markov process**, Journal of the Operational Research Society, 1997, Vol. 48, pp. 843-845.

[95] I. Koh, F. DiCesare and A. Rubenstein, **Modeling and Conrol Infromation Flow for a CIM System Using Colored Petri Nets**, Proceedings of the 1991 Symposium on Command and Control Research, June 1991, US National Defence University, Fort McNair, Washington D.C., pp. 343-348.

[96] I.M. Kyratzoglou, **Computer Aided PN Design for Decision-Making Organizations**, Proceeding of the 1988 Winter Simulation Conference, 12-14 December 1988, San Diego, California, Editors M Abrams, P. Haigh and J. Comfort, pp. 394-401.

[97] J-K. Lee, **A Formal Definition Methods For Time Petri Nets Reduction**, SAMS, 1995, Vol. 18-19, pp. 249-252.

[98] P.E. Lehner, **Towards a Prescriptive Theory of Team Design**, Proceedings of the 1991 IEEE International Conference on Systems, Man and Cybernetics, Fairfax, Va., USA, 13-16 October 1991, pp. 2029-2034.

[99] A.H. Levis and K.L. Boettcher, **Decisionmaking Organizations with Acyclical Information Structures**, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 13, No. 3, 1983, pp 384-390.

[100] A.H. Levis and S.L. Skulsky, **Migration of control in decision making organisations**, Information and Decision Technologies, Vol 15, 1990, pp. 147-159.

[101] A.H. Levis, **Human Interaction with Decision Aids: a Mathematical Approach**, C3I Centre, George Mason University, Fairfax, Virginia, February 1993.

[102] A.H. Levis, **Modeling and Design of Distributed Intelligence Systems**, An Introduction to Intelligent and Autonomous Control, Eds P.J. Antosaklis and K.M. Passino, Kluwer Academic, 1993, pp. 109-127.

[103] C. Lin and D.C. Marinescu, **On Stochastic High-Level Petri Nets**, Proceedings of the International Workshop on Petri Nets and Performance Models, 24-26 August 1987, Madison, Wisconsin, pp. 34-43.

[104] C. Lindemann and G.S. Shedler, **Numerical Analysis of Deterministic and Stochastic Petri Nets with Concurrent Deterministic Transitions**, Performance Evaluation Journal, Vol. 27-28, 1996, pp. 565-582.

[105] C. Lindemann and A. Thummler, **Transient Analysis of Deterministic and Stochastic Petri Nets with Concurrent Deterministic Transitions**, Performance Evaluation, Vol. 36-37, 1999, pp. 35-54.

[106] Z. Liu, **Performance Bounds for Stochastic Timed Petri Nets**, Lecture Notes in Computer Science, Vol. 935, Application and Theory of Petri Nets, 1995, pp.316-334.

[107] Z. Liu, **Performance Analysis of Stochastic Timed Petri Nets using Linear Programming Approach**, IEEE Transactions On Software Engineering, 1998, Vol. 24, No. 11, pp. 1014-1030.

[108] C.G. Looney, **Fuzzy Petri Nets for Rule-Based Decisionmaking**, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 18, No. 1, Jan/Feb 1988, pp. 178-183.

[109] E. Lopez-Mellado, **Simulation of Timed Petri Net Models**, Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, October 1995, Vancouver, Canada, Vol. 3 of 5, pp. 2270-2273.

[110] E. Lopez-Mellado and S. Vallejo-Gonzalez, **An Algorithm for Timed Petri Net-Based Simulation**, International Journal of Robotics and Automation, Vol. 11, No. 2, 1996, pp. 83-90.

[111] Z. Lu and A.H. Levis, **A Colored Peti Net Model of Distributed Tactical Decision Model**, Command and Control Research Symposium, June 1991, US National Defence University, Fort McNair, Washington D.C., pp 70-80.

[112] D. Lucic, **On Exact Equilibrium Distributions of Stochastic Petri Nets**, PhD Thesis, Department of Applied Mathematics, University of Adelaide, November 1990.

[113] V. Mainkar, H. Choi and K. Trivedi, **Sensitivity Analysis of Markov Regenerative Stochastic Petri Nets**, Proceedings of the 5th International Workshop on Petri Nets and Performance Models, October 1993, Toulouse, France, pp. 180-189.

[114] V. Mainkar and K.S. Trivedi, **Transient Analysis of Real-Time Systems Using Deterministic and Stochastic Petri Nets**, Quality of Communication-Based Systems, Editor G. Hommel, 1995, Kluwer Academic Publishers, pp. 65-84.

[115] P. Marenzoni, S. Caselli and G. Conte, **Analysis of Large GSPN Models: A Distributed Solution Tool**, Proceeding of the 7th International Workshop on Petri Nets and Performance Models, 3-6 June 1997, Saint Malo, France, pp. 122-131.

[116] M. Menasche and B. Berthomieu, **Time Petri Nets for Analyzing and Verifying Time Dependent Communications Protocols**, Protocol Specification, Testing and Verification III, Elsevier Science Publishers V.V. (North-Holland), IFIP, 1983, pp.161-172.

[117] M. Menasche, **PAREDE: An Automated Tool for the Analysis of Time(d) Petri Nets**, International Workshop on Timed Petri Nets, 1985, Torino, Italy, pp. 162-169.

[118] P.M. Merlin, **A Study of Recoverability of Computer Systems**, PhD. Thesis, Computer Science University of California IRVINE, 1974.

[119] P.M. Merlin and D.J. Farber, **Recoverability of Communication Protocols-Implications of a Theoretical Study**, IEEE Transactions on Communications, September 1976, pp. 1036-1043.

[120] M.K. Molloy, **Performance Analysis Using Stochastic Petri Nets**, IEEE Transactions on Computers, Vol. C-31, No. 9, September 1982, pp. 913-917.

[121] K.E. Moore and J.P. Lynch(III), **Stochastic, Timed, Attributed Petri Net (STAPN) Modeling of Antisubmarine Warfare C3 Architectures**, Proceedings of the Symposium on Command & Control Research, June 12-14, 1990, Monterey, CA.

[122] K.E. Moore, M.A. Gerber, A.L. Stutsman and D.A. Brager, **Air Defense Network (ADNET) Tool for Air Surveillance Analysis**, Proceedings of The 1992 Summer Computer Simulation Conference, July 27-30 1992, Reno, Nevada, pp. 519-524.

[123] K.E. Moore and J.E. Brennan, **Alpha/Sim Simulation Software Tutorial**, Proceedings of the 1995 Winter Simulation Conference, 3-6 December 1995, Crystal City, VA, pp387-394.

[124] E.T. Morgan and R.R. Razouk, **Interactive State-Space Analysis of Concurrent Systems**, IEEE Transactions on Software Engineering, Vol. SE-13, No. 10, October 1987, pp.1080-1091.

[125] T. Murata, **Petri Nets: Properties, Analysis and Applications**, Proceedings of IEEE, Vol. 77, No. 4, April 1989, pp. 541-580.

[126] T. Murata, **Temporal Uncertainty and Fuzzy-Timing High-Level Petri Nets**, Lecture Notes in Computer Science, Vol. 1091, Application and Theory of Petri Nets, 1996, pp. 11-28.

[127] U. Narayan Bhat, **Elements of Applied Stochastic Processes**, John Wiley & Sons, New York, Chichester, Brisbane, Toronto, Singapore, 2nd edition 1984.

[128] S. Natkin, **Reseaux de Petri stochastiques**, These de docteur ingeneur, CNAM-Paris, Paris, France, June 1980.

[129] J.R. Norris, **Markov Chains**, Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press, 1997.

[130] T. Nowicki and E. Wilczkowiak, **Study of Local Area Networks Using Petri Nets**, Microelectronics and Reliability, Vol. 29, No. 3, Pergamon Press, 1989, pp.433-437.

[131] R.R. Paulussen and L.J. Somers, **Simulation of Distributed Simulation with Timed Colored Petri Nets**, Proceedings of Eurosim '95, pp547-552.

[132] D. Peled, **All from One, One for All: On Model Checking Using Representatives**, Proceedings of the fifth international conference on Computer Aided Verification, CAV'93, Lecture Notes in Computer Science, Vol. 697, Springer-Verlag, pp. 409-423.

[133] D.M. Perdu and A.H. Levis, **Requirements specification using the cube tool methodology**, Laboratory for Information and Decision Systems, MIT, Cambridge, Massachusetts, July 1989, LIDS-R-1890.

[134] D.M. Perdu, **Requiremetns Specification with PN's using the Cube Tool Methodology**, Laboratory for Information and Decision Systems, MIT, Cambridge, Massachusetts, August 1989, LIDS-R-1901.

[135] D.M. Perdu and A.H Levis, **Evaluation of Expert Systems in Decision Making Organizations**, Science of Command and Control, Fairfax, Va. AFCEA International Press, 1994, pp. 76-90.

[136] A. Perkusich, J.C.A. de Figueiredo and S.K. Chang, **Embedding Fault-Tolerant Properties in the Design of Complex Software Systems**, Journal of Systems and Software, Vol. 25, No. 1, April 1994, pp. 23-37.

[137] J.L. Peterson, **Petri Nets**, Computing Surveys, Vol. 9, No. 3, September 1977, pp.223-252.

[138] J.L. Peterson, **Petri Net Theory and the Modeling of Systems**, Prentice-Hall, Inc., Enlgewood Cliffs, 1981.

[139] Petri Nets Standard, http://www.daimi.aau.dk/~petrinet/standard/, 13 September 2000.

[140] C.A. Petri, Kommunikaiton mit Automaten, Ph. D. dissertation, University of Bonn, West Germany, 1962. Also: M.I.T Memorandum MAC-M-212, Project MAC, Massachusetts Institute of Technology, Cambridge, Massachusetts: Also: C. F. Greene Jr. (translator), 'Communication with Automata'. Supplement.

[141] A. Puliafito, M. Scarpa and K.S. Trivedi, **Petri nets with k Simultaneously Enabled Generally Distributed Timed Transitions**, Performance Evaluation, Vol. 32, No. 1, 1998, pp. 1-34.

[142] C.V. Ramamoorthy and G.S. Ho, **Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets**, IEEE Transactions on Software Engineering, Vol. SE-6, No. 5, September 1980, pp. 440-449.

[143] C. Ramchandani, **Analysis of Asynchronous Concurrent Systems by Petri Nets**, PhD., Thesis, Massachusetts Institute of Technology, July 1973.

[144] M. Rauterberg, S. Schluep and M. Fjeld, **How to Model Behavioural and Cognitive Complexity in Human-Computer Interaction with Petri Nets**, Proceedings of the 6th IEEE International Workshop on Robot and Human Communication, RO-MAN '97, 29 September - 1 October 1997, Sendai, Japan, pp. 320-325.

[145] R.R Razouk, **The Derivation of Performance Expressions for Communication Protocols from Timed Petri Net Models**, Proceedings of SIGCOMM-84, 1984, 6-8 June 1984, Montreal, Canada, pp. 210-217.

[146] R.R Razouk and C.V. Phelps, **Performance Analysis Using Timed Petri Nets**, Protocol Specification, Testing, and Verification IV, Elsevier Science Publishers B.V. IFIP, Proceedings of the IFIP WG 6.1 Fourth International Workshop on Protocol Specification, Testing, and Verification, June 11-14 1984, Columbia University, pp. 561-576.

[147] W. Reisig, **Petri Nets, An Introduction**, EATCS, Monographs on Theoretical Computer Science, Springer Verlag, Berlin, 1985, Translation of the original German edition, “Petrinetze", 1982.

[148] P.A. Remy, A.H. Levis and B.Y.-Y. Jin, **On the Design of Distributed Organizational Structures**, Automatica, Vol. 24, No. 1, 1988, pp. 81-86.

[149] S. Ribaric and B.D. Basic, **Temporal Knowledge Representation and Reasoning Model Based on Petri Nets with Time Tokens**, Proceedings of the 8th Mediterranean Electrotechnical Conference (MELECON '96), May 13-16 1996, Vol. 1, Bari, Italy, pp. 131-135.

[150] S.M. Ross, **Applied Probability Models with Optimization Applications**, Holden-Day, 1970.

[151] J-L Roux and G. Juanole, **Functional and Performance Analysis using Extended Time Petri Nets**, Proceedings of the International Workshop on Petri Nets and Performance Models, August 1987, Madison, Wisconsin, pp. 14-23.

[152] P. Senac, P. de Saqui-Sannes and R. Willrich, **Hierarchical Time Stream Petri Nets: a Model for Hypermedia Systems**, Lecture Notes in Computer Science, Vol. 935, Application and Theory of Petri Nets, June 26-30 1995, pp.451-470.

[153] P. Senac, M. Diaz, A. Leger, and P. de Saqui-Sannes, **Modeling Logical and Temporal Synchronization in Hypermedia Systems**, IEEE Journal on Selected Areas in Communications, Vol. 14, No. 1, January 1996, pp. 84-103.

[154] J. Sifakis, **Use of Petri Nets for Performance Evaluation, Measuring**, Modelling and Evaluating Computer Systems, editors H. Beilner and E. Gelenbe, North Holand, 1977, pp. 75-93.

[155] J. Sifakis, **Performance Evaluation of Systems Using Nets**, Lecture Notes in Computer Science, Vol. 84, Advanced Course on General Net Theory of Processes and Systems. Hamburg, October 8-19 1979, Springer-Verlag, pp. 307-319.

[156] R.H. Sloan and U. Buy, **Reduction Rules for Time Petri Nets,** Acta Informatica, Vol. 33, 1996, pp. 687-706.

[157] A. Song and D.L. Kleinman, **Petri Net Simulation and Parameter Optimization for Designing Distributed Bynamic Decisionmaking Experiments**, Proceedings of The 1993 Summer Computer Simulation Conference, July 19-21, Boston, Massachusetts, pp. 948-953.

[158] T. Suzuki, S.M. Shatz and T. Murata, **A Protocol Modeling and Verification Approach Based on a Specification Language and Petri Nets**, IEEE Transactions on Software Engineering, Vol. 16, No. 5, May 1990. pp.523-536.

[159] F.J.W. Symons, **Modeling and Analysis of Communications Protocols using Numerical Petri Nets**, Ph.D. Thesis, University of Essex, 1978.

[160] D. Tabak and A.H. Levis, **Petri Net Representation of Decision Models**, IEEE Transaction on Systems, Man, and Cybernetics, Vol. SMC-15, No. 6, November/December 1985, pp. 812-818.

[161] M. Telek, A. Bobbio and A. Puliafito, **Steady State Solution of MRSPN with Mixed Preemption Policies**, Proceedings of IEEE International Computer Performance and Dependability Conference (IPDS '95), 4-6 September 1996, IL, pp. 106-115.

[162] P.S. Thiagarajan, **Elementary Net Systems**, Lecture Notes in Computer Science, Vol. 254, Advanced Course on Advances in Petri Nets, Part I, Petri Nets: Central Models and Their Properties, September 1986, pp. 26-59.

[163] C. Tournes, **Cube Tool - A C3I Specification Oriented Tool**, Proceeding of the 9th AFCEA European Symposium and Exposition, October 1988, Brussels, pp. 24-30.

[164] J.J.P. Tsai, S.J. Yang and Y. Chang, **Timing Constraint Petri Nets and Their Application to Schedulability Analysis of Real-Time System Specifications**, IEEE Transactions on Software Engineering, Vol. 21, No. 1, January 1995, pp. 32-49.

[165] K.P. Valavanis, **On the Hierarchical Modeling Analysis and Simulation of Flexible Manufacturing Systems with Extend**, IEEE Transactions on Systems, Man and Cybernetics, Vol. 20, No. 1, January/February 1990, pp. 94-110.

[166] V. Valero Ruiz, D. de Frutos Escrig and F. Cuartero Gomez, **Simulaion of Timed Petri Nets by Ordinary Petri Nets and Applications to Decidability of the Timed Reachability Problem and Other Related Problems**, Proceedings of the 4th International Workshop on Petri Nets and Performance Models, December 1991, Melbourne, Australia, pp. 154-163.

[167] V. Valero, D. de Frutos and F. Cuartero, **Timed Processes of Timed Petri Nets**, Lecture Notes in Computer Science, Vol. 935, Applications and Theory of Petri Nets 1995, pp. 490-509.

[168] A. Valmari, **Stubborn Sets for Reduced State Space Generation**, Advances in Petri Nets '90, Lecture Notes in Computer Science, Vol. 483, Editor G. Rozenberg, 1991, pp. 491-515.

[169] A. Valmari, **The State Explosion Problem**, Lectures on Petri Nets I: Basic Models, Lecture Notes in Computer Science, Vol. 1491, 1998, Springer-Verlag, pp. 429-528.

[170] D.M. Walker, **The Expected Time Until Absorption When Absorption is Not Certain**, Journal of Applied Probability, Vol. 35, 1998, pp. 812-823.

[171] B. Walter, **Timed Petri-Nets for Modelling and Analyzing Protocols with Real-Time Characteristics**, Protocol Specification, Testing, and Verification III, Elsevier Science Publishers B.V., IFIP, 1983, pp.149-159.

[172] J. Wang and M. Zhou, **On Information Processing Capacity of Anti-Aircraft Tactic Command and Control Systems**, Proceedings of the 1997 IEEE International Conference on Systems, Man and Cybernetics, October 12-15 1997, Orlando, Florida, USA, pp. 4337-4341.

[173] J.F Watson and A.A. Desrochers, **Applying Generalized Stochastic Petri Nets To Manufacturing Systems Containing Nonexponential Transition Functions**, IEEE Transactions On Systems Man And Cybernetics, 1991, Vol. 21, No. 5, pp. 1008-1017.

[174] S.T. Weingaertner and A.H. Levis, **Analysis of Decision Aiding in Submarine Emergency Decisionmaking**, Automatica, Vol. 25, No. 3, 1989, pp.349-358.

[175] C.Y. Wong, T.S. Dillon and K.E. Forward, **Timed Places Petri Nets with Stochastic Representation of Place Time**, Proceedings of the International Workshop on Timed Petri Nets, July 1-3 1985, Torino, Italy, pp. 96-103.

[176] C.Y. Wong and W. Henderson, **Time-Extended Petri Net Models-An Overview**, Second Australian Teletraffic Research Seminar, organsied by Teletraffic Research Centre, University of Adelaide, 1987.

[177] W. Feng, H. Biqing and L. Wenhuang, **Optimal Solutions to Job Planning Problems in the Assembly Shop by Timed Petri Net**, Journal of Systems Engineering and Electronics, Vol. 9, No. 1, pp. 8-14.

[178] X. Xie, **A Class of Petri Nets for Modular Modeling**, Proceedings of 1995 IEEE International Conference on Robotics and Automation, New York, NY, 21-27 May 1995, volume 2, pp. 2066-71.

[179] A.K. Zaidi and A.H. Levis, **Rule Decomposition and Validation for Distributed Decision Making**, Proceedings of the 1st International Symposium on Command and Control Research and Technology, National Defence University, Washington, D.C., June 1995, pp. 210-217.

[180] A.K. Zaidi and A.H. Levis, **Validation and Verification of Decision Making Rules**, Proceedings of the 1995 IFAC Man Machine Systems, June 1995, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, pp. 45-52.

[181] M. Zhou and I. Mayk, **PNs and C2 Reference model for C2 Systems**, Proceeding of the 2nd Regional Conference on Control Systems, New Jersy Institue of Technology, New Jersy, August 1993, pp. 81-83.

[182] W.M. Zuberek, **M-Timed Petri Nets, Priorities, Preemptions, and Performance Evaluation of Systems**, Lecture Notes in Computer Science, Vol. 222, Advances in Petri Nets, 1985, pp. 478-499.

[183] W.M. Zuberek, **Performance Evaluation Using Extended Timed Petri Nets**, Proceedings of the International Workshop on Timed Petri Nets, Turin, Italy, 1-3 July 1985.

[184] W.M. Zuberek, **Timed Petri Nets Definitions, Properties and Applications**, Microelectronics and Reliability, Vol. 31, No. 4, 1991, pp. 627-644.