PERFORMANCE AND STATE-SPACE ANALYSES OF SYSTEMS USING PETRI NETS

by

James Francis Watson, III

Rensselaer Polytechnic Institute
Electrical, Computer, and Systems Engineering Department
Troy, New York 12180-3590

November 1992

CIRSSE REPORT #129

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENT

I wish to thank the many people who have helped in the development of this thesis. It is through their support, both technical and non-technical, that not only made this achievement a reality, but also an enjoyable pursuit.

I am grateful to my committee, Dr. Alan Desrochers, Dr. Frank DiCesare, Dr. George List, and Dr. Arthur Sanderson, for their support and encouragement. In particular, I appreciate the efforts of my advisor, Dr. Desrochers, to offer motivation and encouragement, while allowing me the freedom to pursue my own ideas. Additionally, I would like to thank Dr. Goldberg, Dr. Krishnamoorthy, and Dr. McNaughton, for their helpful discussions.

I am deeply appreciative of my family for the emotional support provided during my undergraduate and graduate education. My parents, Merlynn and Jim Watson, and wife, Christina Crionas, have always been a source of motivation, encouragement, and love.

I gratefully acknowledge the financial support provided by the Center for Intelligent Robotic Systems for Space Exploration (CIRSSE), under NASA Grant NAGW-1333.

ABSTRACT

The goal of any modeling methodology is to develop a mathematical description of a system that is accurate in its representation and also permits analysis of structural and/or performance properties. Inherently, trade-offs exist between the level of detail in the model and the ease with which analysis can be performed.

Petri nets (PNs), a highly graphical modeling methodology for Discrete Event Dynamic Systems, permit representation of shared resources, finite capacities, conflict, synchronization, concurrency, and timing between state changes. By restricting the state transition time delays to the family of exponential density functions, Markov Chain analysis of performance problems is possible.

One major drawback of PNs is the tendency for the state-space to grow rapidly (exponential complexity) compared to increases in the PN constructs. It is the state-space, or the Markov Chain obtained from it, that is needed in the solution of many problems. This thesis introduces the novel theory of state-space size estimation for PNs. The problem of state-space size estimation is defined, its complexities are examined, and estimation algorithms are developed. Both top-down and bottom-up approaches are pursued, and the advantages and disadvantages of each are described.

Additionally, this thesis advances the author's research in non-exponential transition modeling for PNs. An algorithm for approximating non-exponential transitions is developed. Since only basic PN constructs are used in the approximation, theory already developed for PNs remains applicable. Comparison to results from Entropy Theory show the transition performance is close to the theoretic optimum. Inclusion of non-exponential transition approximations improves performance results at the expense of increased state-space size. The state-space size estimation theory provides insight and algorithms for evaluating this trade-off.

CHAPTER 1
INTRODUCTION

1.1 Motivation

The goal of any modeling methodology is to develop a mathematical description of a particular system that is accurate in its representation and also permits analysis of structural and/or performance properties. Inherently, trade-offs exist between the level of detail that is modeled and the ease with which the analyses can be performed. These trade-offs may be enforced by the methodology (e.g., stochastic theory does not permit description of a single event), or may be exercised by the modeler (e.g., representing a large queue as having infinite size).

One interesting system subclass is Discrete Event Dynamic Systems (DEDSs). This subclass contains those systems that have an enumerable (as opposed to continuous) state-space. Examples of such systems include birth-death, manufacturing, database, and computer-controlled systems. Petri nets (PNs) and their extensions provide a highly graphical modeling methodology for DEDSs. The methodology permits representation of shared resources, finite capacities, conflict, synchronization, concurrency, and timing between state changes. The modeler has the flexibility to incorporate any level of detail into the model, and methods exist for the systematic addition/reduction of detail. Additionally, the strong mathematical foundation of PNs has led to the development of many analysis tools.

Generalized Stochastic PNs (GSPNs) are a popular extension of PNs that permit time-domain analysis of the model. A random delay function (restricted to fixed, immediate and exponential density functions) is associated with each transition. The transition restriction in GSPNs permits Markov analysis. With the concept of time added to PNs, system models can be used to derive performance results, such as

throughput, resource utilization, and part production rate.

GSPNs provide a compact notation for representing large and complex Markov Chains. One major drawback of GSPNs, however, is the tendency for the state-space to grow rapidly compared to increases in the PN constructs (e.g., tokens, places, transitions, serial lines, switches, etc.). It is the state-space, or the Markov Chain obtained from it, that is used for many problems, including performance analysis and scheduling. The growth of the state-space is known to have exponential complexity, but is otherwise uncharacterized.

Additionally, the exponential transition restriction in GSPNs limits the ability to accurately model certain events. Non-exponential transition approximations have been proposed, each having its modeling limitations. Furthermore, inclusion of the non-exponential approximations aggravates the state-space explosion problem, again in a poorly understood manner.

The focusing goal of this thesis is to develop tools to evaluate the trade-off between model accuracy and computability. Improving model accuracy is accomplished by a unique non-exponential transition approximation, and computability is measured by developing state-space size estimators.

1.2 Problem Statement

Estimating the state-space size of PNs is an open research area; there has been no previously published work on this topic. This thesis introduces and analyzes the problem of state-space size estimation, and develops several estimators based on two different approaches: top-down and bottom-up. The accuracy and sensitivity of the estimators are analyzed, with attention given to PN constructs that weaken the estimator performance.

Non-exponential transition modeling is also an important research area for

GSPNs. A handful of different approaches have been suggested, each with its advantages and disadvantages. S-transitions, which allow non-exponential transitions to be approximated within GSPNs by using a moment-matching approach, are developed here.

While the non-exponential transition approximations improve performance result accuracy, they cause an increase in the state-space size (i.e., decrease computability). The non-exponential transition approximations and the state-space size estimation algorithms are used together to evaluate the trade-off between model accuracy and computability. This trade-off can also be viewed from another perspective: what gains in computability can be expected if performance results are only approximated instead of computed exactly. Drawing upon the current research in performance approximation via state-space reduction, the applicability and benefits of state-space size estimation are illustrated.

1.3 Summary of Contributions

The contributions of this thesis are in areas of performance modeling and state-space size estimation for PNs. A PN construct called an s-transition is developed to approximate non-exponential firing delays within PNs. By focusing on moment-matching and the throughput nature of the approximation, these results supplement the techniques available for non-exponential transition modeling, and provide some important advantages. These advantages include the property of the model to remain within the scope of GSPNs (thereby maintaining the applicability of previously developed theory), the ability to service multiple instances of the density function, and closeness to the theoretic optimal solution, the Maximum Entropy Density Function.

The state-space size estimation theory developed in this thesis opens this research area. The problem is characterized, and the importance and application of its solution is described. Two distinct approaches to the solution are provided, each

with its advantages and disadvantages. Considered together, the algorithms resulting from the two approaches are applicable to a large class of PN models and demonstrate accurate estimation.

The areas of non-exponential transition approximation and state-space size estimation are brought together in several applications, including PN performance modeling. The results in this thesis provide tools to improve the accuracy of performance results and also evaluate the trade-off between this increased accuracy and the solution computability.

The developments contained within this thesis have been recognized by the international research community in [Wats91a, Wats91b, Wats92a, Wats92b, Wats92c].

1.4 Organization of the Thesis

The remainder of this thesis is organized as follows:

Chapter 2 provides a brief survey of relevant literature, including general references and specific research results.

Chapter 3 provides the background definitions and theory that will be utilized in later chapters. The presentation is sectioned into notation, stochastics, combinatorics, and Petri nets.

Chapter 4 discusses the problem of non-exponential transition modeling. A construct called an s-transition is developed, analyzed, and compared to optimality results from entropy theory.

Chapter 5 introduces the state-space size estimation problem and the solution methods proposed in this thesis. Characteristics of the estimation, and advantages and disadvantages of the solutions are discussed.

Chapter 6 considers the problem of state-space size estimation from a top-down perspective. Several estimators are developed and analyzed. The most useful estimator, based on weighted conservative PNs, is given particular attention, and examples are developed to illustrate its application and accuracy.

Chapter 7 considers the problem of state-space size estimation from a bottom-up perspective. Several subnets and interconnections are identified and developed. Examples are given to illustrate the application and accuracy of the estimation method.

Chapter 8 develops two case-studies for the state-space size estimation algorithms. Top-down estimation is used in conjunction with a state-space size reduction algorithm. Bottom-up estimation is applied to a complex manufacturing system, and the impact of including a non-exponential transition approximation is analyzed.

Chapter 9 summarizes the contributions of this thesis and applications to the system modeling and analysis research at the Center for Intelligent Robotic Systems for Space Exploration (CIRSSE). Future research directions are also discussed.

Appendix A provides the derivation of the relationships between the three parameters needed to obtain a Maximum Entropy Density Function meeting mean, variance, and positivity constraints. A line search is derived for finding the values of the parameters.

Appendix B discusses how Maximum Entropy Density Functions can be simulated within a performance model, and derives the stochastic relationships to permit simulation within Matlab.

Appendix C outlines the steps needed to make the bound and consistency proofs for the five top-down state-space size estimators.

Appendices D, E and F present the bound and consistency proofs for the five top-down state-space size estimators. Appendix D considers the two estimators based on place bounds, Appendix E considers the two estimators based on net bounds, and Appendix F considers the estimator for weighted conservative PNs.

CHAPTER 2
LITERATURE SURVEY

2.1 Introduction

This chapter reviews the published literature relevant to this thesis. The literature on Petri Net (PN) theory and its applications is reviewed in the next section. This review focuses on those topics that will be directly utilized within the thesis or provide motivation for the research. In addition to results from PN theory, this thesis draws upon stochastics, combinatorics, and information theory. Literature and general references in these fields having direct relevance to the developments in the thesis are also reviewed.

2.2 Petri Net Theory and Its Applications

PNs were introduced in the Ph.D. thesis of Carl Petri, [Petr62], as a means of analyzing the structural properties of communication systems. Since its initial development, PN theory has been extended in many ways, some of which are so commonplace they are assumed as part of the basic PN model. Other extensions (e.g., Colored PNs, [Jens81, Alla85, Zeni85]), however, are still in their initial development as modeling and analysis tools. Overviews of PN theory are provided in [Pete81, Mura89, Mars89]. The latter describes the notation found in the recent PN literature, and where applicable, in this thesis.

Two prominent software packages are available for general PN analysis. Ciardo *et al.*, [Duga85, Ciar88, Ciar89], developed SPNP by considering the semi-Markovian nature of the firing time distributions and stochastic reward theory. Chiola, [Chio87], developed GreatSPN, a system allowing graphical net description/editing, structural analysis, and performance analysis via simulation. Additional components of this

system permit analysis of PNs containing deterministic transitions, [Lind91]. A software package for PN analysis was developed at Rensselaer by Jongwook Kim. This package, XPN, takes advantage of the network transparent features of the X Windows system, but has yet to be formally documented.

In his doctoral research, Molloy, [Moll82, Moll84], incorporated exponential transitions into PNs, permitting performance analysis. It was shown that these models, called Stochastic PNs (SPNs), are isomorphic to Markov chains.

PNs and SPNs have been used to model a variety of systems and obtain both structural and performance results. Among others, PNs have been developed for manufacturing systems, [Brun85, Visw87, AlJa90a], robotic systems, [Seid89, AlJa90b], hardware/software systems, [Rama80, Holl85], and protocols, [Bill85, Garg85, Gild92]

The state-space explosion behavior of PNs has been noted by many researchers, e.g., [Moll84, Mars84a, Ciar89, AlJa90a]. This observation notwithstanding, there are several applications of PNs that rely on the generation of the state-space.

Extensions to SPNs were designed to decrease the computational complexity of solving the underlying Markov Chain. These extensions include the use of immediate and exponential transitions, and inhibitor arcs, [Mars84a, Balb87], and formed a new model class called Generalized Stochastic PNs (GSPNs).

Any additional reductions in the state-space size will result in approximate performance data. Jungnitz, [Jung92], describes a state-space size reduction algorithm that maintains, in most cases, performance result accuracy. The algorithm does not automatically select the optimal decomposition for the model, because no data on the state-space size reduction was available for different decompositions.

Lee, [Lee92], pursues the PN scheduling problem using various heuristic search strategies. The heuristic functions are designed to give a near optimal schedule without exploration of the entire state-space.

Guo *et al.*, [Guo90], computed performance analysis results of PNs using moment generating functions. This method potentially permits analysis of arbitrary transition density functions, but requires the formation of the reachability graph. Mason's Rule, [Maso53, Maso56], is applied to the state machine described by the reachability graph, and performance measures are obtained from the resulting moment generating functions.

Al-Jaar, [AlJa87, AlJa88a, AlJa88b], focused on the application of GSPNs to manufacturing problems. The performance results obtained were based on modeling the time delay densities with a single exponential transition having the appropriate mean.

PNs provide a useful methodology for system controllers. Bjanes, [Bjan91], and Peck, [Peck91], have developed software to permit system control from PN models. Under nominal operations, the state of the PN describes the state of the real system, and the controller applies the PN firing rules to determine legitimate and appropriate actions. Error recovery is needed whenever the state of the PN no longer reflects the state of the real system. Zhou, [Zhou89, Zhou90] described four error recovery augmentations for PN controllers.

Throughput bounds for GSPNs have been developed by Bruell and Ghanta, [Brue85], whereby a polynomial time algorithm is proposed to estimate upper and lower bounds on transition throughputs. The algorithm requires the PN models to be live, bounded, and conservative.

In [Amma85], Ammar and Liu obtain performance results by segregating the state-space into tangible and vanishing states. The immediate transitions are not approximated by "fast" exponential transitions. and the aggregation produces exact steady-state probability distributions.

Extensions have been made to permit non-exponential transitions to be included in the model description. These extensions fall into two categories: those that

result in a GSPN, and those that result in some form of extended model. Chen *et al.*, [Chen89], considers extensions to phase-type transition functions. The underlying model remains in the class of GSPNs, however, control tokens and marking dependent transitions are required. Additionally, the model can process only a single customer. Dugan *et al.*, [Duga84], and Marsan, [Mars87], consider extended classes of PNs that permit arbitrary stochastic transition functions, and deterministic transition functions, respectively. Both classes require that only one non-exponential transition be enabled at any time. Because the model does not remain a GSPN, various theoretical and analytical results for GSPNs do not apply. The non-exponential transition model presented in this thesis allows the GSPN properties to be maintained, does not require control tokens, and can process multiple customers.

Once the restriction on exclusively permitting exponential transition rates is abandoned, the semantics of the transition firing policy are important. Marsan, [Mars85] describes several different transition firing policies, and demonstrates how they are equivalent for exponential transitions.

Colom, [Colo90], presents a comparison of algorithms to compute P-invariants and the minimal P-invariants from the PN's incidence matrix. P-invariants can be used to compute the weight vector for conservative PNs.

2.3 Other Relevant Results

Stochastic modeling and approximation is needed in any methodology that incorporates non-deterministic qualities. A popular overview of stochastic theory and its applications is provided in [Papo84]. Additionally, we make specific use of some results contained in [Kost73].

Based on ideas and results from Information Theory, Jaynes introduced the Maximum Entropy Method (MEM) as a mechanism to optimally model lack of knowledge about a stochastic process, [Jayn57].

Wragg and Dowson, [Wrag70, Dows73], present some useful results on the existence of maximum entropy density functions given various sets of constraints. In addition to the above results, Kapur, [Kapu90], discusses various models for maximum entropy problems.

Cox, [Cox55], represents all random density functions having rational Laplace transforms with combinations of exponential density functions. Complex probabilities are used in the development.

Combinatoric reasoning will be used in the development of the state-space size estimators. An overview of combinatoric theory and its applications is provided in [Robe84].

In addition to the PN analysis software reviewed above, simulation is required to analyze some systems. For these cases, the simulations in this thesis were performed in Matlab, [Matl90]. The validity of simulation results are only as good as the simulation program. Mitrani, [Mitr82], describes simulation techniques for discrete event systems, and Fishman, [Fish73], additionally considers computer-based random number generation.

An introduction to graph theory is provided in [Bond76]. This is useful for familiarization with the relevant terminology.

2.4 Summary

This chapter has provided general references and briefly reviewed specific research results that are relevant to this thesis. The PN literature provides motivation and applications for this research, and some specific results that can be utilized. Results from the fields of stochastics, combinatorics, and information theory, are also used in the developments of this thesis.

CHAPTER 3
BACKGROUND

3.1 Introduction

This chapter defines the terms and notation used throughout the thesis. Additionally, fundamental theoretical results are presented. When these results are obtained from the literature, an appropriate reference is given. Original developments are also presented, so as not to be an "interruption" in later chapters. This chapter is divided into four sections: notation, stochastics, combinatorics, and Petri nets.

3.2 Notation

The following notation will be used:

- Vectors are columns by default, and will be denoted by a single underline, e.g., $\underline{x}$. Individual elements are denoted as $x_1, x_2, \ldots$. The zero vector (of appropriate size) is denoted by $\underline{0}$, and the vector of all ones is denoted by $\underline{1}$.
- Matrices will be denoted by a double underline, e.g., $\underline{\underline{A}}$. Individual elements are denoted as $A_{1,1}, A_{1,2}, \ldots$.
- Transposes of vectors and arrays are denoted similar to $\underline{\underline{A}}^{\mathsf{T}}$.
- The dot product of two vectors, $\underline{x}$ and $\underline{y}$, is denoted as, $\underline{x} \circ \underline{y}$, thus $\underline{x} \circ \underline{y} = \underline{x}^{\mathsf{T}} \underline{y}$.
- An estimate of a number, x, will be denoted by $\tilde{x}$.
- The floor of a number x is denoted as $\lfloor x \rfloor$, and means the largest integer not exceeding x.

- The ceiling of a number x is denoted as $\lceil x \rceil$, and means the smallest integer not less than x.

- The number of combinations among n objects is denoted by

$$\left(\begin{array}{c} n \\ r \end{array} \right) \stackrel{\text{def}}{=} \frac{n!}{r!(n-r)!} \quad 0 \leq r \leq n < \infty \tag{3.1}$$

$$\left(\begin{array}{c} n \\ r_1, r_2, \ldots, r_m \end{array} \right) \stackrel{\text{def}}{=} \frac{n!}{r_1! r_2! \cdots r_m!} \quad 0 \leq r_i \leq n < \infty; \quad \sum_{i=1}^{m} r_i = n \tag{3.2}$$

- Cardinality of a set M is denoted by $\mid M \mid$.

- The following symbols will also be used: $\ni$ means "such that," $\exists$ means "there exists," $\forall$ means "for all," $\Rightarrow$ means "implies," $\Leftrightarrow$ means "iff," $\stackrel{\text{def}}{=}$ means "equals by definition," $\rightarrow$ denotes an "injection," and $\leftrightarrow$ denotes a "bijection."

We are mainly interested in the following number systems and subsets.

- $\mathbb{R}$ denotes the finite real numbers.

- $\mathbb{R}_+$ denotes the strictly positive finite real numbers.

- $\mathbb{N}$ denotes the finite integers.

- $\mathbb{N}_+$ denotes the strictly positive finite integers.

- $\mathbb{N}_{0+} \stackrel{\text{def}}{=} \mathbb{N}_+ \cup \{0\}$.

- $\mathcal{Q}$ denotes the finite rationals.

- $\mathcal{Q}_+$ denotes the strictly positive finite rationals.

- $\mathcal{Q}_{0+} \stackrel{\text{def}}{=} \mathcal{Q}_+ \cup \{0\}$.

3.3 Stochastics

Stochastic Petri nets (introduced in Section 3.5) permit random state transitions to be modeled. Thus, some background in stochastics is presented. Particular attention is given to the exponential density function, since this is the permitted transition delay function. A comprehensive treatment of stochastics is provided in many standard texts, e.g., [Papo84].

3.3.1 Definitions

3.3.1.1 Probability Density Functions

We are primarily interested in the probability density function (pdf) of a single continuous, one-sided, random variable. Given a random variable, $\mathcal{X}$, its pdf is denoted by $f_{\mathcal{X}}(x)$, and satisfies

$$f_{\mathcal{X}}(x) \geq 0 \qquad x \geq 0 \tag{3.3}$$

$$\int_0^\infty f_{\mathcal{X}}(x)dx = 1 \tag{3.4}$$

3.3.1.2 Moments and Expectation

The first two moments of a given random variable will be used for stochastic calculations. Specifically, for a random variable $\mathcal{X}$,

- its mean is denoted as $\mu_{\mathcal{X}}$:

$$\mu_{\mathcal{X}} \stackrel{\text{def}}{=} \int_0^\infty x f_{\mathcal{X}}(x)dx \tag{3.5}$$

- its variance is denoted as $\sigma^2_{\mathcal{X}}$, and the standard deviation, assumed to be non-negative, as $\sigma_{\mathcal{X}}$:

$$\sigma^2_{\mathcal{X}} \stackrel{\text{def}}{=} \int_0^\infty (x - \mu_{\mathcal{X}})^2 f_{\mathcal{X}}(x)dx \tag{3.6}$$

- The expectation of a function on $\mathcal{X}$, say $g(x)$, is denoted as $E\{g(\mathcal{X})\}$:

$$E\{g(\mathcal{X})\} \stackrel{\text{def}}{=} \int_0^\infty g(x) f_{\mathcal{X}}(x) dx \tag{3.7}$$

- The probability of an event S is denoted as $Pr\{S\}$, and conditional probabilities and expectations are denoted similar to $E\{\mathcal{X} \mid S\}$.

3.3.1.3 Entropy

The entropy of a random variable is a measure of its ability to provide information, or in other words, a measure of its uncertainty. The entropy of $\mathcal{X}$ is computed by

$$H_{\mathcal{X}} \stackrel{\text{def}}{=} -\int_0^\infty f_{\mathcal{X}}(x) \log f_{\mathcal{X}}(x) dx \tag{3.8}$$

Note that $f_{\mathcal{X}}$ may be zero, which is a singularity of the log function. Formally, the integration kernel is evaluated as 0 when $f_{\mathcal{X}}(x) = 0$. This is justified by the fact that the limit of the product is zero, i.e.,

$$\lim_{a \to 0} a \log a = 0 \tag{3.9}$$

3.3.1.4 Independence

Two random variables, $\mathcal{X}$ and $\mathcal{Y}$ are called independent if the outcome of one does not depend on the outcome of the other. Thus, $\mathcal{X}$ and $\mathcal{Y}$ are independent if

$$E\{\mathcal{X} \mid \mathcal{Y} = y\} = E\{\mathcal{X}\} \quad \text{and} \quad E\{\mathcal{Y} \mid \mathcal{X} = x\} = E\{\mathcal{Y}\} \tag{3.10}$$

As a consequence,

$$E\{\mathcal{X}\mathcal{Y}\} = E\{\mathcal{X}\}E\{\mathcal{Y}\}. \tag{3.11}$$

3.3.2 Theory

3.3.2.1 Exponential Random Variables

The family of exponential random variables is parameterized by a single positive value. Formally, if $\mathcal{X} \sim EXP(\lambda)$, then the density function of $\mathcal{X}$ is given by

$$f_{\mathcal{X}}(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

The moments of the density function are

$$\mu_{\mathcal{X}} = 1/\lambda \tag{3.13}$$

$$\sigma^2_{\mathcal{X}} = 1/\lambda^2 \tag{3.14}$$

3.3.2.2 Erlangian Random Variables

The family of Erlangian random variables is parameterized by a rate, $\lambda > 0$, and a number of phases, $n \in \mathbb{N}_+$. Formally, if $\mathcal{X} \sim ERL(\lambda, n)$, then the density function of $\mathcal{X}$ is given by

$$f_{\mathcal{X}}(x) = \begin{cases} \frac{\lambda^n}{(n-1)!} x^{n-1} e^{-\lambda x} & x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

and its moments are

$$\mu_{\mathcal{X}} = n/\lambda \tag{3.16}$$

$$\sigma^2_{\mathcal{X}} = n/\lambda^2 \tag{3.17}$$

Comparing (3.12–3.14) and (3.15–3.17), it can be seen that the exponential family coincides with the one-phase density functions in the Erlangian family.

3.3.2.3 Hyperexponential Random Variables

The family of hyperexponential random variables is parameterized by an order, $n \in \mathbb{N}_+$, and two n-vectors:

- $\underline{\lambda} \in \mathbb{R}_+^n$ describes the exponential parameters.
- $\underline{r} \in \mathbb{R}_+^n$ describes the weights, where it is required that $\sum_{i=1}^n r_i = 1$.

Formally, if $\mathcal{X} \sim HYP(n, \underline{\lambda}, \underline{r})$, then the density function of $\mathcal{X}$ is given by

$$f_{\mathcal{X}}(x) = \begin{cases} \sum_{i=1}^{n} r_i \lambda_i e^{-\lambda_i x} & x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

and its moments are

$$\mu_{\mathcal{X}} = \sum_{i=1}^{n} r_i / \lambda_i \tag{3.19}$$

$$\sigma_{\mathcal{X}}^2 = \sum_{i=1}^{n} r_i / \lambda_i^2 \tag{3.20}$$

Comparing (3.12–3.14) and (3.18–3.20), it can be seen that the exponential family coincides with the first-order density functions in the hyperexponential family.

3.3.2.4 Gaussian Random Variables

The family of Gaussian random variables is parameterized by a mean, μ, and a variance, σ^2. Gaussian density functions are not one-sided, however, they are needed in Chapter 4. Formally, if $\mathcal{X} \sim N(\mu, \sigma^2)$, then the density function of $\mathcal{X}$ is given by

$$f_{\mathcal{X}}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.21}$$

The moments of the density function are given by the parameterization, i.e.,

$$\mu_{\mathcal{X}} = \mu \tag{3.22}$$

$$\sigma_{\mathcal{X}}^2 = \sigma^2 \tag{3.23}$$

3.3.2.5 Summation of Two Independent Random Variables

Given two random variables, $\mathcal{X}$ and $\mathcal{Y}$, and the relationship, $\mathcal{Z} = \mathcal{X} + \mathcal{Y}$, the density function of $\mathcal{Z}$ can be computed. In our context, $\mathcal{X}$ and $\mathcal{Y}$ are independent

and one-sided, thereby producing a one-sided density. Thus,

$$\begin{aligned} f_Z(z) &= (f_X \star f_Y)(z) \\ &= \int_0^z f_X(z-\lambda) f_Y(\lambda) d\lambda \end{aligned} \tag{3.24}$$

where, $\star$ is the convolution operator.

In general,

$$\mu_Z = \mu_X + \mu_Y \tag{3.25}$$

and this clearly holds if $\mathcal{X}$ and $\mathcal{Y}$ are independent. The relationship between the variances is more complicated. We have

$$\begin{aligned} \sigma_Z^2 &= E\{((\mathcal{X}-\mu_X)+(\mathcal{Y}-\mu_Y))^2\} \\ &= \sigma_X^2 + \sigma_Y^2 + 2E\{\mathcal{X}\mathcal{Y}\} - 2\mu_X\mu_Y \end{aligned} \tag{3.26}$$

and for independent random variables,

$$\sigma_Z^2 = \sigma_X^2 + \sigma_Y^2 \tag{3.27}$$

Given that $\mathcal{X}$ and $\mathcal{Y}$ are exponentially distributed with parameters λ_1 and λ_2, respectively, we have

$$\begin{aligned} f_Z(z) &= \int_0^z \left(\lambda_1 e^{-\lambda_1(z-y)}\right)\left(\lambda_2 e^{-\lambda_2 y}\right) dy \\ &= \lambda_1\lambda_2 e^{-\lambda_1 z} \int_0^z e^{(\lambda_1-\lambda_2)y} dy \\ &= \frac{\lambda_1\lambda_2 e^{-\lambda_1 z}}{\lambda_1-\lambda_2}\left[e^{(\lambda_1-\lambda_2)y}\right]\Big|_0^z \\ &= \frac{\lambda_1\lambda_2}{\lambda_1-\lambda_2}\left(e^{-\lambda_2 z} - e^{-\lambda_1 z}\right) \end{aligned} \tag{3.28}$$

where, we have tacitly assumed that $\lambda_1 \neq \lambda_2$. For the case that the two random variables have the same parameter, say λ, the result degenerates to

$$f_Z(z) = \lambda^2 z e^{-\lambda z} \tag{3.29}$$

which is an Erlangian density of 2-phases, (3.15).

3.3.2.6 Minimum of Two Independent Random Variables

Given two random variables, $\mathcal{X}$ and $\mathcal{Y}$, and the relationship, $\mathcal{Z} = \min(\mathcal{X}, \mathcal{Y})$, the density function of $\mathcal{Z}$ can be computed. The minimization can be expressed as

$$Pr\{\mathcal{Z} = z\} = Pr\{\mathcal{X} = z \mid \mathcal{Y} \geq z\} + Pr\{\mathcal{Y} = z \mid \mathcal{X} \geq z\} \tag{3.30}$$

and since $\mathcal{X}$ and $\mathcal{Y}$ are independent, we have,

$$Pr\{\mathcal{Z} = z\} = Pr\{\mathcal{X} = z\}Pr\{\mathcal{Y} \geq z\} + Pr\{\mathcal{Y} = z\}Pr\{\mathcal{X} \geq z\} \tag{3.31}$$

Given that $\mathcal{X}$ and $\mathcal{Y}$ are exponentially distributed with parameters λ_1 and λ_2, respectively, we have

$$\begin{aligned} f_{\mathcal{Z}}(z) &= \lambda_1 e^{-\lambda_1 z} \int_z^\infty \lambda_2 e^{-\lambda_2 t} dt + \lambda_2 e^{-\lambda_2 z} \int_z^\infty \lambda_1 e^{-\lambda_1 t} dt \\ &= \lambda_1 e^{-\lambda_1 z} e^{-\lambda_2 z} + \lambda_2 e^{-\lambda_2 z} e^{-\lambda_1 z} \\ &= (\lambda_1 + \lambda_2) e^{-(\lambda_1 + \lambda_2) z} \end{aligned} \tag{3.32}$$

Thus, the minimum of two exponential random variables, is again an exponential random variable.

3.3.2.7 Choice Among Several Random Variables

Given n independent random variables, $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n$, the density function (i.e., $\mathcal{Z}$) resulting from a choice among these quantities can be computed. Denoting the "weights" of the choice as r_i, $i = 1, \ldots, n$, this choice can be expressed as

$$f_{\mathcal{Z}}(z) = \sum_{i=1}^{n} r_i f_i(z) \tag{3.33}$$

where it is additionally required that

$$\sum_{i=1}^{n} r_i = 1 \tag{3.34}$$

From (3.18), it can be seen that the choice among two or more exponentially distributed random variables results in a hyperexponential density function.

3.3.2.8 Transformation of Variables

Given a random variable, $\mathcal{X}$, and the relationship $\mathcal{Y} = g(\mathcal{X})$, we wish to determine the density function of $\mathcal{Y}$. This density function is given by the following result:

$$f_{\mathcal{Y}}(y) = \sum_{i=1}^{n} \frac{f_{\mathcal{X}}(x_i)}{|g'(x_i)|} \text{ where } x_1, \ldots, x_n = g^{-1}(y) \tag{3.35}$$

3.4 Combinatorics

The state of a Petri net is defined below as a relation between its places (a set of discrete objects) and the number of tokens (non-negative integer quantities) in these places. Thus, the problem of estimating the state-space size of a Petri net will draw on combinatoric reasoning. A comprehensive treatment of combinatorics is provided in many standard texts, e.g., [Robe84].

3.4.1 Definitions

3.4.1.1 Permutations and Combinations

Of primary concern in combinatoric problems is what importance is given to the *order* of the objects of interest. A permutation of n objects considers the order as significant. On the other hand, a combination of n objects ignores the order of the objects. Thus, if the objects can actually be distinguished, an artificial device to make them non-distinct must be considered.

We have,

Definition 3.1 (permutations) *for $n \in N_{0+}$. ${}^nP$ denotes the number of permutations of n distinct objects.*

For example, given the three objects P_1, P_2, and P_3, then 6 permutations exist: (P_1, P_2, P_3), (P_1, P_3, P_2), (P_2, P_1, P_3), (P_2, P_3, P_1), (P_3, P_1, P_2), and (P_3, P_2, P_1).

Definition 3.2 (r-permutations) *for $r, n \in \mathbb{N}_{0+}$, such that $r \leq n$, ${}^nP_r$ denotes the number of r-permutations of n distinct objects.*

For example, six permutations exist when the above objects are considered two at a time: (P_1, P_2), (P_2, P_1), (P_1, P_3), (P_3, P_1), (P_2, P_3), and (P_3, P_2).

Definition 3.3 (combinations) *for $r, n \in \mathbb{N}_{0+}$, such that $r \leq n$, ${}^nC_r$ denotes the number of r-combinations of n distinct objects.*

${}^nC_r$ considers the order of selection to be immaterial. For example, the 2-combinations of the above objects are (P_1, P_2), (P_1, P_3), and (P_2, P_3). As suggested by this example, combinations can be used to select a given number of distinct places in a Petri net model.

3.4.1.2 Placements

Definition 3.4 (placements) *for $r \in \mathbb{N}_{0+}$ and $n \in \mathbb{N}_{+}$, ${}^nX_r$ denotes the number of ways to place r indistinguishable balls into n distinct bins. The minimum number of balls in each bin is zero, and the total among the n bins is r.*

Equating the tokens and places in a Petri net with, respectively, the indistinguishable objects and distinct bins of this definition, ${}^nX_r$ provides a means to count states within the model. An example is given in the next definition.

3.4.1.3 Partitions

Definition 3.5 (partitions) *for $r \in \mathbb{N}_{0+}$ and $n \in \mathbb{N}_{+}$, an n-partition of r is an n-vector of non-negative integers that sum to r. $I_n(r)$ denotes the set of n-partitions of r.*

The size of $I_n(r)$ is given by ${}^nX_r$. For example, the enumeration of $I_3(2)$ is given by

$$\{(2,0,0),(0,2,0),(0,0,2),(1,1,0),(1,0,1),(0,1,1)\}$$

which has a size of ${}^3X_2 = 6$. In a Petri net, partitions can be used to combinatorially "scan" possible distributions of r tokens among n places. The above enumeration of $I_3(2)$ illustrates the possible distributions of exactly two tokens among three places.

3.4.1.4 Occupations

Definition 3.6 (occupations) *for $r, n \in \mathbb{N}_+$, such that $r \geq n$, an n-occupation of r is an n-vector of positive integers that sum to r. $J_n(r)$ denotes the set of n-occupations of r.*

Occupations are similar to partitions, except that zero is not permitted as an addend, i.e., all elements of the n-vector must be occupied. For example, the 2-occupations of 4 are $J_2(4) = \{(1,3),(2,2),(3,1)\}$, and do not include $\{(0,4),(4,0)\}$, which are in the set of partitions, $I_2(4)$. In a Petri net, occupations can be used to consider possible distributions of r tokens among n places for cases where no place is to be empty.

3.4.1.5 Pascal's Triangle

Definition 3.7 (Pascal's Triangle) *Pascal's Triangle results from the following three construction rules:*

1. *The top row consists of the single entry,* 1, *and each subsequent row has one more entry than the row above it.*
2. *The first and last entry of each row are equal to* 1.
3. *An intermediate entry is the sum of the two entries in the previous row that straddle it.*

Figure 3.1 illustrates the upper portion of Pascal's Triangle.

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

Figure 3.1: Top of Pascal's Triangle

3.4.2 Theory

3.4.2.1 Permutations and Combinations

${}^nP_r$ and ${}^nC_r$ denote the most basic of combinatoric quantities. The following are well known formulae:

$$ {}^nP = n! \tag{3.36} $$

$$ {}^nP_r = \frac{n!}{(n-r)!} \tag{3.37} $$

$$ {}^nC_r = \frac{n!}{r!(n-r)!} = \begin{pmatrix} n \\ r \end{pmatrix} \tag{3.38} $$

Three identities are observed directly from the definition of ${}^nC_r$:

$$ {}^nC_r = {}^nC_{n-r} \tag{3.39} $$

$$ {}^nC_0 = 1 \tag{3.40} $$

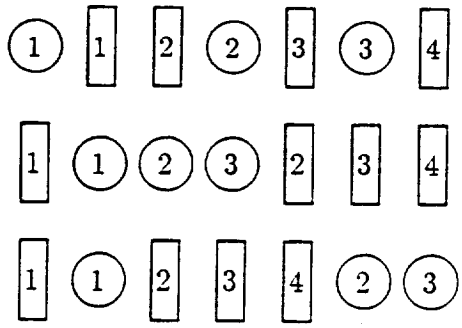
$$ {}^nC_n = 1 \tag{3.41} $$

3.4.2.2 Placements

A lemma resulting in a computable expression for placements, ${}^nX_r$, is proven.

Lemma 3.1 ${}^nX_r = {}^{n+r-1}C_r = \dfrac{(n+r-1)!}{r!(n-1)!}$.

Proof: Consider that we have r distinct balls and $n-1$ distinguishable sticks—later, the balls will be made indistinguishable, and an equivalence between the sticks and the bins will be shown. The balls and sticks total $(n-1+r)$ distinct objects, which can be permuted in ${}^{n-1+r}P$ ways. For example, if $r=3$ and $n=5$ then three of the possible 7! orderings include

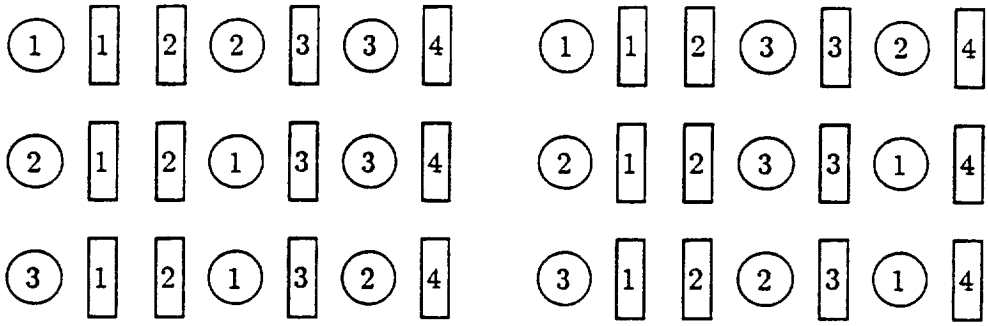


We associate the $n-1$ sticks with the $n-1$ boundaries that occur between n distinct bins when they are stacked side-by-side. Thus, in the first example, ball 1 is in the first bin, ball 2 is between the second and third boundaries, putting it in the third bin, and similarly, ball 3 is in the fourth bin, etc.. The 5-tuples describing the number of balls placed into each bin for the above examples are $(1,0,1,1,0)$, $(0,3,0,0,0)$, and $(0,1,0,0,2)$, respectively.

The above result of ${}^{n-1+r}P$ is based on the distinct balls and distinct sticks. This result needs to be reduced to account for our goal to consider indistinguishable balls and distinct bins.

Given any permutation of the $(n-1+r)$ objects, there exist other permutations that would produce the same result. Thus, without moving the sticks, and without

changing the number of balls between the sticks, the r balls can be permuted ${}^{r}P$ ways. Specifically, for the 5-tuple $(1,0,1,1,0)$ in the example above, the following $3! = 6$ permutations of the balls are possible:



Likewise, the bins are uniquely identified by the *locations* of the sticks only. Independent of the ball permutations, the $n-1$ sticks can be permuted ${}^{n-1}P$ ways. Thus, we conclude that the number of ways to place r indistinguishable balls into n distinct bins is $(n-1+r)!/r!(n-1)!$, precisely, ${}^{n}X_{r}$. □

Two identities are observed directly from the definition of ${}^{n}X_{r}$:

$$ {}^{n}X_{0} = 1 \tag{3.42} $$

$$ {}^{1}X_{r} = 1 \tag{3.43} $$

and, additionally from Lemma 3.1 and (3.39),

$$ {}^{n}X_{r} = {}^{r+1}X_{n-1} \tag{3.44} $$

3.4.2.3 Size of Occupation Set

From Definitions 3.5 and 3.6, it is clear that $|J_n(r)| \leq |I_n(r)|$, and, typically, the number of occupations is much less than the number of partitions. Specifically, since each n-occupation has a minimum of one unit in each of the n elements, $r-n$ units remain to be distributed freely. The ways to distribute these remaining units

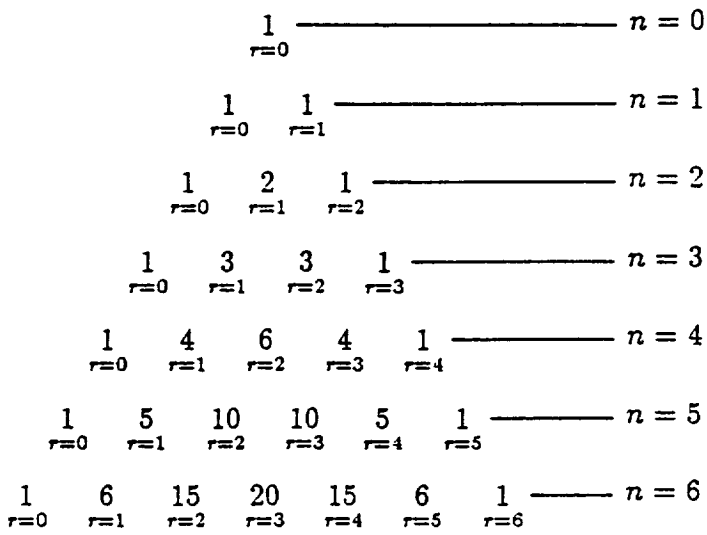


Figure 3.2: Pascal's Triangle as ${}^nC_r$

in the n-vector is given by $I_n(r-n)$. Thus, the size of $J_n(r)$ is given by

$$\begin{aligned} |J_n(r)| &= |I_n(r-n)| \\ &= {}^nX_{r-n} \\ &= {}^{r-1}C_{n-1} \end{aligned} \tag{3.45}$$

3.4.2.4 Pascal's Triangle

Using ${}^nC_r$, an algebraic set of construction rules can be derived from Definition 3.7, (see Figure 3.2). From (3.40) and (3.41), the top entry is given by ${}^0C_0 = 1$, the first entry of a row is given by ${}^nC_0 = 1$, and the last entry of a row is given by ${}^nC_n = 1$. The recurrence relationship in rule 3 can be expressed as

$${}^nC_r = {}^{n-1}C_{r-1} + {}^{n-1}C_r \tag{3.46}$$

which is a well-known combinatorial identity.

3.4.2.5 Binomial Theorem

An expansion of $(a+b)^n$, where $n \in I\!N_+$ is given by the well-known Binomial Theorem.

Theorem 3.1 $(a+b)^n = \sum_{i=0}^{n} {}^nC_i a^{n-i} b^i$

3.4.2.6 Other

The results provided in the following lemmas will be used to develop the state-space size estimators.

Lemma 3.2 *For all* $x, y \in I\!N_{0+}$,

$$\frac{(x+y)!}{x!(y+1)!} \leq (x+1)^y \tag{3.47}$$

Proof: For $y = 0$, we have from (3.47),

$$\frac{x!}{x!} \leq 1 \tag{3.48}$$

which clearly holds. Now that we can assume $y \geq 1$, the remainder of the proof is made by induction on x. For $x = 0$, (3.47) becomes

$$\frac{y!}{(y+1)!} \leq 1$$

which holds for integer $y \geq 1$. We assume (3.47) holds for $x = x_0$, i.e.,

$$\begin{aligned} \frac{(x_0+y)!}{x_0!(y+1)!} &\leq (x_0+1)^y \\ (x_0+y)\cdot(x_0-1+y)\cdots(x_0+1) &\leq (y+1)!(x_0+1)^y \end{aligned} \tag{3.49}$$

and need to show that it holds for the next value of x. Thus, we need to show

$$\frac{(x_0+1+y)!}{(x_0+1)!(y+1)!} \leq (x_0+2)^y \tag{3.50}$$

The validity of (3.50) is proven by contradiction, i.e., we show

$$\begin{aligned}(x_0+1+y)\cdot(x_0+y)\cdots(x_0+2) &> (y+1)!(x_0+2)^y \\ \frac{x_0+1+y}{x_0+1}((x_0+y)\cdots(x_0+1)) &> (y+1)!(x_0+2)^y \end{aligned} \tag{3.51}$$

does not hold. Further simplification of (3.51) results from substituting the assumption (3.49):

$$\begin{aligned}\frac{x_0+1+y}{x_0+1}((y+1)!(x_0+1)^y) &> (y+1)!(x_0+2)^y \\ \frac{x_0+1+y}{x_0+1} &> \left(\frac{x_0+2}{x_0+1}\right)^y\end{aligned}$$

which can be rearranged as

$$1+\frac{y}{x_0+1} > \left(1+\frac{1}{x_0+1}\right)^y \tag{3.52}$$

From the Binomial Theorem, (Theorem 3.1), the righthand side of (3.52) can be expanded to

$$\left(1+\frac{1}{x_0+1}\right)^y = 1+y\frac{1}{x_0+1}+{}^yC_2\left(\frac{1}{x_0+1}\right)^2+\cdots+{}^yC_{y-1}\left(\frac{1}{x_0+1}\right)^{y-1}+\left(\frac{1}{x_0+1}\right)^y$$

which is a sum of positive terms. Since the lefthand side of (3.52) is but the first two terms of this expansion, we conclude that (3.52) does not hold. From this contradiction, (3.50) is proven, which proves the lemma. □

The following lemma states two facts: no matter how x balls are distributed among y boxes (even if the goal is to minimize the number in each box), at least one box will contain at least $\lceil x/y \rceil$ balls; and, similarly, even when trying to maximize the number in each box, at least one box will contain no more than $\lfloor x/y \rfloor$ balls.

Lemma 3.3 *Putting exactly $x \in \mathbb{N}_{0+}$ balls into $y \in \mathbb{N}_{+}$ boxes, where x_i denotes the number of balls in each box, results in the following relationships:*

$$\max_{i=1\ldots y} x_i \geq \lceil x/y \rceil \tag{3.53}$$

$$\min_{i=1\ldots y} x_i \leq \lfloor x/y \rfloor \tag{3.54}$$

Proof: First, (3.53) is proven by contradiction, followed by (3.54). Assume there are y boxes with x_i balls in each, and

$$\begin{aligned} \max_i x_i &< \lceil x/y \rceil \\ &\leq \lceil x/y \rceil - 1 \end{aligned} \tag{3.55}$$

Then the total number of balls among the y boxes would be

$$\begin{aligned} \sum_{i=1}^{y} x_i &\leq y(\lceil x/y \rceil - 1) \\ &\leq (y\lceil x/y \rceil) - y \\ &\leq (x + y - 1) - y \\ &\leq x - 1 \end{aligned}$$

which is a contradiction. Now, assume

$$\begin{aligned} \min_i x_i &> \lfloor x/y \rfloor \\ &\geq \lfloor x/y \rfloor + 1 \end{aligned} \tag{3.56}$$

Then the total number of balls among the y boxes would be

$$\begin{aligned} \sum_{i=1}^{y} x_i &\geq y(\lfloor x/y \rfloor + 1) \\ &\geq (y\lfloor x/y \rfloor) \dot{+} y \\ &\geq (x - y \dot{+} 1) \dot{+} y \\ &\geq x \dot{+} 1 \end{aligned}$$

which is a contradiction. □

PERFORMANCE AND STATE-SPACE ANALYSES OF SYSTEMS USING PETRI NETS

P.191

by

James Francis Watson, III

Rensselaer Polytechnic Institute
Electrical, Computer, and Systems Engineering Department
Troy, New York 12180-3590

November 1992

CIRSSE REPORT #129

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENT

I wish to thank the many people who have helped in the development of this thesis. It is through their support, both technical and non-technical, that not only made this achievement a reality, but also an enjoyable pursuit.

I am grateful to my committee, Dr. Alan Desrochers, Dr. Frank DiCesare, Dr. George List, and Dr. Arthur Sanderson, for their support and encouragement. In particular, I appreciate the efforts of my advisor, Dr. Desrochers, to offer motivation and encouragement, while allowing me the freedom to pursue my own ideas. Additionally, I would like to thank Dr. Goldberg, Dr. Krishnamoorthy, and Dr. McNaughton, for their helpful discussions.

I am deeply appreciative of my family for the emotional support provided during my undergraduate and graduate education. My parents, Merlynn and Jim Watson, and wife, Christina Crionas, have always been a source of motivation, encouragement, and love.

I gratefully acknowledge the financial support provided by the Center for Intelligent Robotic Systems for Space Exploration (CIRSSE), under NASA Grant NAGW-1333.

ABSTRACT

The goal of any modeling methodology is to develop a mathematical description of a system that is accurate in its representation and also permits analysis of structural and/or performance properties. Inherently, trade-offs exist between the level of detail in the model and the ease with which analysis can be performed.

Petri nets (PNs), a highly graphical modeling methodology for Discrete Event Dynamic Systems, permit representation of shared resources, finite capacities, conflict, synchronization, concurrency, and timing between state changes. By restricting the state transition time delays to the family of exponential density functions, Markov Chain analysis of performance problems is possible.

One major drawback of PNs is the tendency for the state-space to grow rapidly (exponential complexity) compared to increases in the PN constructs. It is the state-space, or the Markov Chain obtained from it, that is needed in the solution of many problems. This thesis introduces the novel theory of state-space size estimation for PNs. The problem of state-space size estimation is defined, its complexities are examined, and estimation algorithms are developed. Both top-down and bottom-up approaches are pursued, and the advantages and disadvantages of each are described.

Additionally, this thesis advances the author's research in non-exponential transition modeling for PNs. An algorithm for approximating non-exponential transitions is developed. Since only basic PN constructs are used in the approximation, theory already developed for PNs remains applicable. Comparison to results from Entropy Theory show the transition performance is close to the theoretic optimum. Inclusion of non-exponential transition approximations improves performance results at the expense of increased state-space size. The state-space size estimation theory provides insight and algorithms for evaluating this trade-off.

CHAPTER 1
INTRODUCTION

1.1 Motivation

The goal of any modeling methodology is to develop a mathematical description of a particular system that is accurate in its representation and also permits analysis of structural and/or performance properties. Inherently, trade-offs exist between the level of detail that is modeled and the ease with which the analyses can be performed. These trade-offs may be enforced by the methodology (e.g., stochastic theory does not permit description of a single event), or may be exercised by the modeler (e.g., representing a large queue as having infinite size).

One interesting system subclass is Discrete Event Dynamic Systems (DEDSs). This subclass contains those systems that have an enumerable (as opposed to continuous) state-space. Examples of such systems include birth-death, manufacturing, database, and computer-controlled systems. Petri nets (PNs) and their extensions provide a highly graphical modeling methodology for DEDSs. The methodology permits representation of shared resources, finite capacities, conflict, synchronization, concurrency, and timing between state changes. The modeler has the flexibility to incorporate any level of detail into the model, and methods exist for the systematic addition/reduction of detail. Additionally, the strong mathematical foundation of PNs has led to the development of many analysis tools.

Generalized Stochastic PNs (GSPNs) are a popular extension of PNs that permit time-domain analysis of the model. A random delay function (restricted to fixed, immediate and exponential density functions) is associated with each transition. The transition restriction in GSPNs permits Markov analysis. With the concept of time added to PNs, system models can be used to derive performance results, such as

throughput, resource utilization, and part production rate.

GSPNs provide a compact notation for representing large and complex Markov Chains. One major drawback of GSPNs, however, is the tendency for the state-space to grow rapidly compared to increases in the PN constructs (e.g., tokens, places, transitions, serial lines, switches, etc.). It is the state-space, or the Markov Chain obtained from it, that is used for many problems, including performance analysis and scheduling. The growth of the state-space is known to have exponential complexity, but is otherwise uncharacterized.

Additionally, the exponential transition restriction in GSPNs limits the ability to accurately model certain events. Non-exponential transition approximations have been proposed, each having its modeling limitations. Furthermore, inclusion of the non-exponential approximations aggravates the state-space explosion problem, again in a poorly understood manner.

The focusing goal of this thesis is to develop tools to evaluate the trade-off between model accuracy and computability. Improving model accuracy is accomplished by a unique non-exponential transition approximation, and computability is measured by developing state-space size estimators.

1.2 Problem Statement

Estimating the state-space size of PNs is an open research area; there has been no previously published work on this topic. This thesis introduces and analyzes the problem of state-space size estimation, and develops several estimators based on two different approaches: top-down and bottom-up. The accuracy and sensitivity of the estimators are analyzed, with attention given to PN constructs that weaken the estimator performance.

Non-exponential transition modeling is also an important research area for

GSPNs. A handful of different approaches have been suggested, each with its advantages and disadvantages. S-transitions, which allow non-exponential transitions to be approximated within GSPNs by using a moment-matching approach, are developed here.

While the non-exponential transition approximations improve performance result accuracy, they cause an increase in the state-space size (i.e., decrease computability). The non-exponential transition approximations and the state-space size estimation algorithms are used together to evaluate the trade-off between model accuracy and computability. This trade-off can also be viewed from another perspective: what gains in computability can be expected if performance results are only approximated instead of computed exactly. Drawing upon the current research in performance approximation via state-space reduction, the applicability and benefits of state-space size estimation are illustrated.

1.3 Summary of Contributions

The contributions of this thesis are in areas of performance modeling and state-space size estimation for PNs. A PN construct called an s-transition is developed to approximate non-exponential firing delays within PNs. By focusing on moment-matching and the throughput nature of the approximation, these results supplement the techniques available for non-exponential transition modeling, and provide some important advantages. These advantages include the property of the model to remain within the scope of GSPNs (thereby maintaining the applicability of previously developed theory), the ability to service multiple instances of the density function, and closeness to the theoretic optimal solution, the Maximum Entropy Density Function.

The state-space size estimation theory developed in this thesis opens this research area. The problem is characterized, and the importance and application of its solution is described. Two distinct approaches to the solution are provided, each

with its advantages and disadvantages. Considered together, the algorithms resulting from the two approaches are applicable to a large class of PN models and demonstrate accurate estimation.

The areas of non-exponential transition approximation and state-space size estimation are brought together in several applications, including PN performance modeling. The results in this thesis provide tools to improve the accuracy of performance results and also evaluate the trade-off between this increased accuracy and the solution computability.

The developments contained within this thesis have been recognized by the international research community in [Wats91a, Wats91b, Wats92a, Wats92b, Wats92c].

1.4 Organization of the Thesis

The remainder of this thesis is organized as follows:

Chapter 2 provides a brief survey of relevant literature, including general references and specific research results.

Chapter 3 provides the background definitions and theory that will be utilized in later chapters. The presentation is sectioned into notation, stochastics, combinatorics, and Petri nets.

Chapter 4 discusses the problem of non-exponential transition modeling. A construct called an s-transition is developed, analyzed, and compared to optimality results from entropy theory.

Chapter 5 introduces the state-space size estimation problem and the solution methods proposed in this thesis. Characteristics of the estimation, and advantages and disadvantages of the solutions are discussed.

Chapter 6 considers the problem of state-space size estimation from a top-down perspective. Several estimators are developed and analyzed. The most useful estimator, based on weighted conservative PNs, is given particular attention, and examples are developed to illustrate its application and accuracy.

Chapter 7 considers the problem of state-space size estimation from a bottom-up perspective. Several subnets and interconnections are identified and developed. Examples are given to illustrate the application and accuracy of the estimation method.

Chapter 8 develops two case-studies for the state-space size estimation algorithms. Top-down estimation is used in conjunction with a state-space size reduction algorithm. Bottom-up estimation is applied to a complex manufacturing system, and the impact of including a non-exponential transition approximation is analyzed.

Chapter 9 summarizes the contributions of this thesis and applications to the system modeling and analysis research at the Center for Intelligent Robotic Systems for Space Exploration (CIRSSE). Future research directions are also discussed.

Appendix A provides the derivation of the relationships between the three parameters needed to obtain a Maximum Entropy Density Function meeting mean, variance, and positivity constraints. A line search is derived for finding the values of the parameters.

Appendix B discusses how Maximum Entropy Density Functions can be simulated within a performance model, and derives the stochastic relationships to permit simulation within Matlab.

Appendix C outlines the steps needed to make the bound and consistency proofs for the five top-down state-space size estimators.

Appendices D, E and F present the bound and consistency proofs for the five top-down state-space size estimators. Appendix D considers the two estimators based on place bounds, Appendix E considers the two estimators based on net bounds, and Appendix F considers the estimator for weighted conservative PNs.

CHAPTER 2
LITERATURE SURVEY

2.1 Introduction

This chapter reviews the published literature relevant to this thesis. The literature on Petri Net (PN) theory and its applications is reviewed in the next section. This review focuses on those topics that will be directly utilized within the thesis or provide motivation for the research. In addition to results from PN theory, this thesis draws upon stochastics, combinatorics, and information theory. Literature and general references in these fields having direct relevance to the developments in the thesis are also reviewed.

2.2 Petri Net Theory and Its Applications

PNs were introduced in the Ph.D. thesis of Carl Petri, [Petr62], as a means of analyzing the structural properties of communication systems. Since its initial development, PN theory has been extended in many ways, some of which are so commonplace they are assumed as part of the basic PN model. Other extensions (e.g., Colored PNs, [Jens81, Alla85, Zeni85]), however, are still in their initial development as modeling and analysis tools. Overviews of PN theory are provided in [Pete81, Mura89, Mars89]. The latter describes the notation found in the recent PN literature, and where applicable, in this thesis.

Two prominent software packages are available for general PN analysis. Ciardo *et al.*, [Duga85, Ciar88, Ciar89], developed SPNP by considering the semi-Markovian nature of the firing time distributions and stochastic reward theory. Chiola, [Chio87], developed GreatSPN, a system allowing graphical net description/editing, structural analysis, and performance analysis via simulation. Additional components of this

system permit analysis of PNs containing deterministic transitions, [Lind91]. A software package for PN analysis was developed at Rensselaer by Jongwook Kim. This package, XPN, takes advantage of the network transparent features of the X Windows system, but has yet to be formally documented.

In his doctoral research, Molloy, [Moll82, Moll84], incorporated exponential transitions into PNs, permitting performance analysis. It was shown that these models, called Stochastic PNs (SPNs), are isomorphic to Markov chains.

PNs and SPNs have been used to model a variety of systems and obtain both structural and performance results. Among others, PNs have been developed for manufacturing systems, [Brun85, Visw87, AlJa90a], robotic systems, [Seid89, AlJa90b], hardware/software systems, [Rama80, Holl85], and protocols, [Bill85, Garg85, Gild92]

The state-space explosion behavior of PNs has been noted by many researchers, e.g., [Moll84, Mars84a, Ciar89, AlJa90a]. This observation notwithstanding, there are several applications of PNs that rely on the generation of the state-space.

Extensions to SPNs were designed to decrease the computational complexity of solving the underlying Markov Chain. These extensions include the use of immediate and exponential transitions, and inhibitor arcs, [Mars84a, Balb87], and formed a new model class called Generalized Stochastic PNs (GSPNs).

Any additional reductions in the state-space size will result in approximate performance data. Jungnitz, [Jung92], describes a state-space size reduction algorithm that maintains, in most cases, performance result accuracy. The algorithm does not automatically select the optimal decomposition for the model, because no data on the state-space size reduction was available for different decompositions.

Lee, [Lee92], pursues the PN scheduling problem using various heuristic search strategies. The heuristic functions are designed to give a near optimal schedule without exploration of the entire state-space.

Guo *et al.*, [Guo90], computed performance analysis results of PNs using moment generating functions. This method potentially permits analysis of arbitrary transition density functions, but requires the formation of the reachability graph. Mason's Rule, [Maso53, Maso56], is applied to the state machine described by the reachability graph, and performance measures are obtained from the resulting moment generating functions.

Al-Jaar, [AlJa87, AlJa88a, AlJa88b], focused on the application of GSPNs to manufacturing problems. The performance results obtained were based on modeling the time delay densities with a single exponential transition having the appropriate mean.

PNs provide a useful methodology for system controllers. Bjanes, [Bjan91], and Peck, [Peck91], have developed software to permit system control from PN models. Under nominal operations, the state of the PN describes the state of the real system, and the controller applies the PN firing rules to determine legitimate and appropriate actions. Error recovery is needed whenever the state of the PN no longer reflects the state of the real system. Zhou, [Zhou89, Zhou90] described four error recovery augmentations for PN controllers.

Throughput bounds for GSPNs have been developed by Bruell and Ghanta, [Brue85], whereby a polynomial time algorithm is proposed to estimate upper and lower bounds on transition throughputs. The algorithm requires the PN models to be live, bounded, and conservative.

In [Amma85], Ammar and Liu obtain performance results by segregating the state-space into tangible and vanishing states. The immediate transitions are not approximated by "fast" exponential transitions, and the aggregation produces exact steady-state probability distributions.

Extensions have been made to permit non-exponential transitions to be included in the model description. These extensions fall into two categories: those that

result in a GSPN, and those that result in some form of extended model. Chen *et al.*, [Chen89], considers extensions to phase-type transition functions. The underlying model remains in the class of GSPNs, however, control tokens and marking dependent transitions are required. Additionally, the model can process only a single customer. Dugan *et al.*, [Duga84], and Marsan, [Mars87], consider extended classes of PNs that permit arbitrary stochastic transition functions, and deterministic transition functions, respectively. Both classes require that only one non-exponential transition be enabled at any time. Because the model does not remain a GSPN, various theoretical and analytical results for GSPNs do not apply. The non-exponential transition model presented in this thesis allows the GSPN properties to be maintained, does not require control tokens, and can process multiple customers.

Once the restriction on exclusively permitting exponential transition rates is abandoned, the semantics of the transition firing policy are important. Marsan, [Mars85] describes several different transition firing policies, and demonstrates how they are equivalent for exponential transitions.

Colom, [Colo90], presents a comparison of algorithms to compute P-invariants and the minimal P-invariants from the PN's incidence matrix. P-invariants can be used to compute the weight vector for conservative PNs.

2.3 Other Relevant Results

Stochastic modeling and approximation is needed in any methodology that incorporates non-deterministic qualities. A popular overview of stochastic theory and its applications is provided in [Papo84]. Additionally, we make specific use of some results contained in [Kost73].

Based on ideas and results from Information Theory, Jaynes introduced the Maximum Entropy Method (MEM) as a mechanism to optimally model lack of knowledge about a stochastic process, [Jayn57].

Wragg and Dowson, [Wrag70, Dows73], present some useful results on the existence of maximum entropy density functions given various sets of constraints. In addition to the above results, Kapur, [Kapu90], discusses various models for maximum entropy problems.

Cox, [Cox55], represents all random density functions having rational Laplace transforms with combinations of exponential density functions. Complex probabilities are used in the development.

Combinatoric reasoning will be used in the development of the state-space size estimators. An overview of combinatoric theory and its applications is provided in [Robe84].

In addition to the PN analysis software reviewed above, simulation is required to analyze some systems. For these cases, the simulations in this thesis were performed in Matlab, [Matl90]. The validity of simulation results are only as good as the simulation program. Mitrani, [Mitr82], describes simulation techniques for discrete event systems, and Fishman, [Fish73], additionally considers computer-based random number generation.

An introduction to graph theory is provided in [Bond76]. This is useful for familiarization with the relevant terminology.

2.4 Summary

This chapter has provided general references and briefly reviewed specific research results that are relevant to this thesis. The PN literature provides motivation and applications for this research, and some specific results that can be utilized. Results from the fields of stochastics, combinatorics, and information theory, are also used in the developments of this thesis.

CHAPTER 3
BACKGROUND

3.1 Introduction

This chapter defines the terms and notation used throughout the thesis. Additionally, fundamental theoretical results are presented. When these results are obtained from the literature, an appropriate reference is given. Original developments are also presented, so as not to be an "interruption" in later chapters. This chapter is divided into four sections: notation, stochastics, combinatorics, and Petri nets.

3.2 Notation

The following notation will be used:

- Vectors are columns by default, and will be denoted by a single underline, e.g., $\underline{x}$. Individual elements are denoted as $x_1, x_2, \ldots$. The zero vector (of appropriate size) is denoted by $\underline{0}$, and the vector of all ones is denoted by $\underline{1}$.
- Matrices will be denoted by a double underline, e.g., $\underline{\underline{A}}$. Individual elements are denoted as $A_{1,1}, A_{1,2}, \ldots$.
- Transposes of vectors and arrays are denoted similar to $\underline{\underline{A}}^{\mathsf{T}}$.
- The dot product of two vectors, $\underline{x}$ and $\underline{y}$, is denoted as, $\underline{x} \circ \underline{y}$, thus $\underline{x} \circ \underline{y} = \underline{x}^{\mathsf{T}} \underline{y}$.
- An estimate of a number, x, will be denoted by $\tilde{x}$.
- The floor of a number x is denoted as $\lfloor x \rfloor$, and means the largest integer not exceeding x.

- The ceiling of a number x is denoted as $\lceil x \rceil$, and means the smallest integer not less than x.

- The number of combinations among n objects is denoted by

$$\left(\begin{array}{c} n \\ r \end{array} \right) \stackrel{\text{def}}{=} \frac{n!}{r!(n-r)!} \quad 0 \leq r \leq n < \infty \tag{3.1}$$

$$\left(\begin{array}{c} n \\ r_1, r_2, \ldots, r_m \end{array} \right) \stackrel{\text{def}}{=} \frac{n!}{r_1! r_2! \cdots r_m!} \quad 0 \leq r_i \leq n < \infty; \quad \sum_{i=1}^{m} r_i = n \tag{3.2}$$

- Cardinality of a set M is denoted by $\mid M \mid$.

- The following symbols will also be used: $\ni$ means "such that," $\exists$ means "there exists," $\forall$ means "for all," $\Rightarrow$ means "implies," $\Leftrightarrow$ means "iff," $\stackrel{\text{def}}{=}$ means "equals by definition," $\rightarrow$ denotes an "injection," and $\leftrightarrow$ denotes a "bijection."

We are mainly interested in the following number systems and subsets.

- $\mathbb{R}$ denotes the finite real numbers.

- $\mathbb{R}_+$ denotes the strictly positive finite real numbers.

- $\mathbb{N}$ denotes the finite integers.

- $\mathbb{N}_+$ denotes the strictly positive finite integers.

- $\mathbb{N}_{0+} \stackrel{\text{def}}{=} \mathbb{N}_+ \cup \{0\}$.

- $\mathcal{Q}$ denotes the finite rationals.

- $\mathcal{Q}_+$ denotes the strictly positive finite rationals.

- $\mathcal{Q}_{0+} \stackrel{\text{def}}{=} \mathcal{Q}_+ \cup \{0\}$.

3.3 Stochastics

Stochastic Petri nets (introduced in Section 3.5) permit random state transitions to be modeled. Thus, some background in stochastics is presented. Particular attention is given to the exponential density function, since this is the permitted transition delay function. A comprehensive treatment of stochastics is provided in many standard texts, e.g., [Papo84].

3.3.1 Definitions

3.3.1.1 Probability Density Functions

We are primarily interested in the probability density function (pdf) of a single continuous, one-sided, random variable. Given a random variable, $\mathcal{X}$, its pdf is denoted by $f_{\mathcal{X}}(x)$, and satisfies

$$f_{\mathcal{X}}(x) \geq 0 \qquad x \geq 0 \tag{3.3}$$

$$\int_0^\infty f_{\mathcal{X}}(x)dx = 1 \tag{3.4}$$

3.3.1.2 Moments and Expectation

The first two moments of a given random variable will be used for stochastic calculations. Specifically, for a random variable $\mathcal{X}$,

- its mean is denoted as $\mu_{\mathcal{X}}$:

$$\mu_{\mathcal{X}} \stackrel{\text{def}}{=} \int_0^\infty x f_{\mathcal{X}}(x)dx \tag{3.5}$$

- its variance is denoted as $\sigma^2_{\mathcal{X}}$, and the standard deviation. assumed to be non-negative, as $\sigma_{\mathcal{X}}$:

$$\sigma^2_{\mathcal{X}} \stackrel{\text{def}}{=} \int_0^\infty (x - \mu_{\mathcal{X}})^2 f_{\mathcal{X}}(x)dx \tag{3.6}$$

- The expectation of a function on $\mathcal{X}$, say $g(x)$, is denoted as $E\{g(\mathcal{X})\}$:

$$E\{g(\mathcal{X})\} \stackrel{\text{def}}{=} \int_0^\infty g(x) f_{\mathcal{X}}(x) dx \tag{3.7}$$

- The probability of an event S is denoted as $Pr\{S\}$, and conditional probabilities and expectations are denoted similar to $E\{\mathcal{X} \mid S\}$.

3.3.1.3 Entropy

The entropy of a random variable is a measure of its ability to provide information, or in other words, a measure of its uncertainty. The entropy of $\mathcal{X}$ is computed by

$$H_{\mathcal{X}} \stackrel{\text{def}}{=} -\int_0^\infty f_{\mathcal{X}}(x) \log f_{\mathcal{X}}(x) dx \tag{3.8}$$

Note that $f_{\mathcal{X}}$ may be zero, which is a singularity of the log function. Formally, the integration kernel is evaluated as 0 when $f_{\mathcal{X}}(x) = 0$. This is justified by the fact that the limit of the product is zero, i.e.,

$$\lim_{a \to 0} a \log a = 0 \tag{3.9}$$

3.3.1.4 Independence

Two random variables, $\mathcal{X}$ and $\mathcal{Y}$ are called independent if the outcome of one does not depend on the outcome of the other. Thus, $\mathcal{X}$ and $\mathcal{Y}$ are independent if

$$E\{\mathcal{X} \mid \mathcal{Y} = y\} = E\{\mathcal{X}\} \quad \text{and} \quad E\{\mathcal{Y} \mid \mathcal{X} = x\} = E\{\mathcal{Y}\} \tag{3.10}$$

As a consequence,

$$E\{\mathcal{X}\mathcal{Y}\} = E\{\mathcal{X}\}E\{\mathcal{Y}\}. \tag{3.11}$$

3.3.2 Theory

3.3.2.1 Exponential Random Variables

The family of exponential random variables is parameterized by a single positive value. Formally, if $\mathcal{X} \sim EXP(\lambda)$, then the density function of $\mathcal{X}$ is given by

$$f_{\mathcal{X}}(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

The moments of the density function are

$$\mu_{\mathcal{X}} = 1/\lambda \tag{3.13}$$

$$\sigma^2_{\mathcal{X}} = 1/\lambda^2 \tag{3.14}$$

3.3.2.2 Erlangian Random Variables

The family of Erlangian random variables is parameterized by a rate, $\lambda > 0$, and a number of phases, $n \in \mathbb{N}_+$. Formally, if $\mathcal{X} \sim ERL(\lambda, n)$, then the density function of $\mathcal{X}$ is given by

$$f_{\mathcal{X}}(x) = \begin{cases} \frac{\lambda^n}{(n-1)!} x^{n-1} e^{-\lambda x} & x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

and its moments are

$$\mu_{\mathcal{X}} = n/\lambda \tag{3.16}$$

$$\sigma^2_{\mathcal{X}} = n/\lambda^2 \tag{3.17}$$

Comparing (3.12–3.14) and (3.15–3.17), it can be seen that the exponential family coincides with the one-phase density functions in the Erlangian family.

3.3.2.3 Hyperexponential Random Variables

The family of hyperexponential random variables is parameterized by an order, $n \in \mathbb{N}_+$, and two n-vectors:

- $\underline{\lambda} \in \mathbb{R}_+^n$ describes the exponential parameters.
- $\underline{r} \in \mathbb{R}_+^n$ describes the weights, where it is required that $\sum_{i=1}^n r_i = 1$.

Formally, if $\mathcal{X} \sim HYP(n, \underline{\lambda}, \underline{r})$, then the density function of $\mathcal{X}$ is given by

$$f_{\mathcal{X}}(x) = \begin{cases} \sum_{i=1}^{n} r_i \lambda_i e^{-\lambda_i x} & x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

and its moments are

$$\mu_{\mathcal{X}} = \sum_{i=1}^{n} r_i/\lambda_i \tag{3.19}$$

$$\sigma_{\mathcal{X}}^2 = \sum_{i=1}^{n} r_i/\lambda_i^2 \tag{3.20}$$

Comparing (3.12–3.14) and (3.18–3.20), it can be seen that the exponential family coincides with the first-order density functions in the hyperexponential family.

3.3.2.4 Gaussian Random Variables

The family of Gaussian random variables is parameterized by a mean, μ, and a variance, σ^2. Gaussian density functions are not one-sided, however, they are needed in Chapter 4. Formally, if $\mathcal{X} \sim N(\mu, \sigma^2)$, then the density function of $\mathcal{X}$ is given by

$$f_{\mathcal{X}}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.21}$$

The moments of the density function are given by the parameterization, i.e.,

$$\mu_{\mathcal{X}} = \mu \tag{3.22}$$

$$\sigma_{\mathcal{X}}^2 = \sigma^2 \tag{3.23}$$

3.3.2.5 Summation of Two Independent Random Variables

Given two random variables, $\mathcal{X}$ and $\mathcal{Y}$, and the relationship, $\mathcal{Z} = \mathcal{X} + \mathcal{Y}$, the density function of $\mathcal{Z}$ can be computed. In our context, $\mathcal{X}$ and $\mathcal{Y}$ are independent

and one-sided, thereby producing a one-sided density. Thus,

$$\begin{aligned} f_Z(z) &= (f_X \star f_Y)(z) \\ &= \int_0^z f_X(z-\lambda) f_Y(\lambda) d\lambda \end{aligned} \tag{3.24}$$

where, $\star$ is the convolution operator.

In general,

$$\mu_Z = \mu_X + \mu_Y \tag{3.25}$$

and this clearly holds if $\mathcal{X}$ and $\mathcal{Y}$ are independent. The relationship between the variances is more complicated. We have

$$\begin{aligned} \sigma_Z^2 &= E\{((\mathcal{X}-\mu_X)+(\mathcal{Y}-\mu_Y))^2\} \\ &= \sigma_X^2 + \sigma_Y^2 + 2E\{\mathcal{X}\mathcal{Y}\} - 2\mu_X\mu_Y \end{aligned} \tag{3.26}$$

and for independent random variables,

$$\sigma_Z^2 = \sigma_X^2 + \sigma_Y^2 \tag{3.27}$$

Given that $\mathcal{X}$ and $\mathcal{Y}$ are exponentially distributed with parameters λ_1 and λ_2, respectively, we have

$$\begin{aligned} f_Z(z) &= \int_0^z \left(\lambda_1 e^{-\lambda_1(z-y)}\right)\left(\lambda_2 e^{-\lambda_2 y}\right) dy \\ &= \lambda_1\lambda_2 e^{-\lambda_1 z} \int_0^z e^{(\lambda_1-\lambda_2)y} dy \\ &= \frac{\lambda_1\lambda_2 e^{-\lambda_1 z}}{\lambda_1-\lambda_2}\left[e^{(\lambda_1-\lambda_2)y}\right]\Big|_0^z \\ &= \frac{\lambda_1\lambda_2}{\lambda_1-\lambda_2}\left(e^{-\lambda_2 z} - e^{-\lambda_1 z}\right) \end{aligned} \tag{3.28}$$

where, we have tacitly assumed that $\lambda_1 \neq \lambda_2$. For the case that the two random variables have the same parameter, say λ, the result degenerates to

$$f_Z(z) = \lambda^2 z e^{-\lambda z} \tag{3.29}$$

which is an Erlangian density of 2-phases, (3.15).

3.3.2.6 Minimum of Two Independent Random Variables

Given two random variables, $\mathcal{X}$ and $\mathcal{Y}$, and the relationship, $\mathcal{Z} = \min(\mathcal{X}, \mathcal{Y})$, the density function of $\mathcal{Z}$ can be computed. The minimization can be expressed as

$$Pr\{\mathcal{Z} = z\} = Pr\{\mathcal{X} = z \mid \mathcal{Y} \geq z\} + Pr\{\mathcal{Y} = z \mid \mathcal{X} \geq z\} \tag{3.30}$$

and since $\mathcal{X}$ and $\mathcal{Y}$ are independent, we have,

$$Pr\{\mathcal{Z} = z\} = Pr\{\mathcal{X} = z\}Pr\{\mathcal{Y} \geq z\} + Pr\{\mathcal{Y} = z\}Pr\{\mathcal{X} \geq z\} \tag{3.31}$$

Given that $\mathcal{X}$ and $\mathcal{Y}$ are exponentially distributed with parameters λ_1 and λ_2, respectively, we have

$$\begin{aligned} f_{\mathcal{Z}}(z) &= \lambda_1 e^{-\lambda_1 z} \int_z^\infty \lambda_2 e^{-\lambda_2 t} dt + \lambda_2 e^{-\lambda_2 z} \int_z^\infty \lambda_1 e^{-\lambda_1 t} dt \\ &= \lambda_1 e^{-\lambda_1 z} e^{-\lambda_2 z} + \lambda_2 e^{-\lambda_2 z} e^{-\lambda_1 z} \\ &= (\lambda_1 + \lambda_2) e^{-(\lambda_1 + \lambda_2) z} \end{aligned} \tag{3.32}$$

Thus, the minimum of two exponential random variables, is again an exponential random variable.

3.3.2.7 Choice Among Several Random Variables

Given n independent random variables, $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n$, the density function (i.e., $\mathcal{Z}$) resulting from a choice among these quantities can be computed. Denoting the "weights" of the choice as r_i, $i = 1, \ldots, n$, this choice can be expressed as

$$f_{\mathcal{Z}}(z) = \sum_{i=1}^{n} r_i f_i(z) \tag{3.33}$$

where it is additionally required that

$$\sum_{i=1}^{n} r_i = 1 \tag{3.34}$$

From (3.18), it can be seen that the choice among two or more exponentially distributed random variables results in a hyperexponential density function.

3.3.2.8 Transformation of Variables

Given a random variable, $\mathcal{X}$, and the relationship $\mathcal{Y} = g(\mathcal{X})$, we wish to determine the density function of $\mathcal{Y}$. This density function is given by the following result:

$$f_{\mathcal{Y}}(y) = \sum_{i=1}^{n} \frac{f_{\mathcal{X}}(x_i)}{|g'(x_i)|} \text{ where } x_1, \ldots, x_n = g^{-1}(y) \tag{3.35}$$

3.4 Combinatorics

The state of a Petri net is defined below as a relation between its places (a set of discrete objects) and the number of tokens (non-negative integer quantities) in these places. Thus, the problem of estimating the state-space size of a Petri net will draw on combinatoric reasoning. A comprehensive treatment of combinatorics is provided in many standard texts, e.g., [Robe84].

3.4.1 Definitions

3.4.1.1 Permutations and Combinations

Of primary concern in combinatoric problems is what importance is given to the *order* of the objects of interest. A permutation of n objects considers the order as significant. On the other hand, a combination of n objects ignores the order of the objects. Thus, if the objects can actually be distinguished, an artificial device to make them non-distinct must be considered.

We have,

Definition 3.1 (permutations) *for $n \in \mathbb{N}_{0+}$, ${}^nP$ denotes the number of permutations of n distinct objects.*

For example, given the three objects P_1, P_2, and P_3, then 6 permutations exist: (P_1, P_2, P_3), (P_1, P_3, P_2), (P_2, P_1, P_3), (P_2, P_3, P_1), (P_3, P_1, P_2), and (P_3, P_2, P_1).

Definition 3.2 (r-permutations) *for $r, n \in \mathbb{N}_{0+}$, such that $r \leq n$, ${}^nP_r$ denotes the number of r-permutations of n distinct objects.*

For example, six permutations exist when the above objects are considered two at a time: (P_1, P_2), (P_2, P_1), (P_1, P_3), (P_3, P_1), (P_2, P_3), and (P_3, P_2).

Definition 3.3 (combinations) *for $r, n \in \mathbb{N}_{0+}$, such that $r \leq n$, ${}^nC_r$ denotes the number of r-combinations of n distinct objects.*

${}^nC_r$ considers the order of selection to be immaterial. For example, the 2-combinations of the above objects are (P_1, P_2), (P_1, P_3), and (P_2, P_3). As suggested by this example, combinations can be used to select a given number of distinct places in a Petri net model.

3.4.1.2 Placements

Definition 3.4 (placements) *for $r \in \mathbb{N}_{0+}$ and $n \in \mathbb{N}_{+}$, ${}^nX_r$ denotes the number of ways to place r indistinguishable balls into n distinct bins. The minimum number of balls in each bin is zero, and the total among the n bins is r.*

Equating the tokens and places in a Petri net with, respectively, the indistinguishable objects and distinct bins of this definition, ${}^nX_r$ provides a means to count states within the model. An example is given in the next definition.

3.4.1.3 Partitions

Definition 3.5 (partitions) *for $r \in \mathbb{N}_{0+}$ and $n \in \mathbb{N}_{+}$, an n-partition of r is an n-vector of non-negative integers that sum to r. $I_n(r)$ denotes the set of n-partitions of r.*

The size of $I_n(r)$ is given by ${}^nX_r$. For example, the enumeration of $I_3(2)$ is given by

$$\{(2,0,0),(0,2,0),(0,0,2),(1,1,0),(1,0,1),(0,1,1)\}$$

which has a size of ${}^3X_2 = 6$. In a Petri net, partitions can be used to combinatorially "scan" possible distributions of r tokens among n places. The above enumeration of $I_3(2)$ illustrates the possible distributions of exactly two tokens among three places.

3.4.1.4 Occupations

Definition 3.6 (occupations) *for $r, n \in \mathbb{N}_+$, such that $r \geq n$, an n-occupation of r is an n-vector of positive integers that sum to r. $J_n(r)$ denotes the set of n-occupations of r.*

Occupations are similar to partitions, except that zero is not permitted as an addend, i.e., all elements of the n-vector must be occupied. For example, the 2-occupations of 4 are $J_2(4) = \{(1,3),(2,2),(3,1)\}$, and do not include $\{(0,4),(4,0)\}$, which are in the set of partitions, $I_2(4)$. In a Petri net, occupations can be used to consider possible distributions of r tokens among n places for cases where no place is to be empty.

3.4.1.5 Pascal's Triangle

Definition 3.7 (Pascal's Triangle) *Pascal's Triangle results from the following three construction rules:*

1. *The top row consists of the single entry,* 1, *and each subsequent row has one more entry than the row above it.*
2. *The first and last entry of each row are equal to* 1.
3. *An intermediate entry is the sum of the two entries in the previous row that straddle it.*

Figure 3.1 illustrates the upper portion of Pascal's Triangle.

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

Figure 3.1: Top of Pascal's Triangle

3.4.2 Theory

3.4.2.1 Permutations and Combinations

${}^nP_r$ and ${}^nC_r$ denote the most basic of combinatoric quantities. The following are well known formulae:

$$ {}^nP = n! \tag{3.36} $$

$$ {}^nP_r = \frac{n!}{(n-r)!} \tag{3.37} $$

$$ {}^nC_r = \frac{n!}{r!(n-r)!} = \begin{pmatrix} n \\ r \end{pmatrix} \tag{3.38} $$

Three identities are observed directly from the definition of ${}^nC_r$:

$$ {}^nC_r = {}^nC_{n-r} \tag{3.39} $$

$$ {}^nC_0 = 1 \tag{3.40} $$

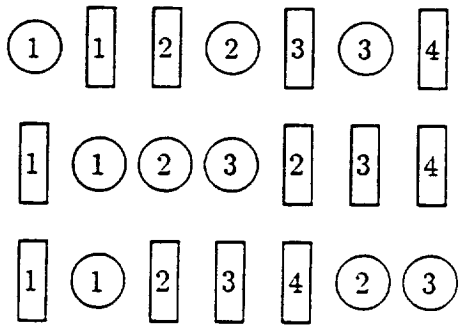
$$ {}^nC_n = 1 \tag{3.41} $$

3.4.2.2 Placements

A lemma resulting in a computable expression for placements, ${}^{n}X_{r}$, is proven.

Lemma 3.1 ${}^{n}X_{r} = {}^{n+r-1}C_{r} = \dfrac{(n+r-1)!}{r!(n-1)!}$.

Proof: Consider that we have r distinct balls and $n-1$ distinguishable sticks—later, the balls will be made indistinguishable, and an equivalence between the sticks and the bins will be shown. The balls and sticks total $(n-1+r)$ distinct objects, which can be permuted in ${}^{n-1+r}P$ ways. For example, if $r=3$ and $n=5$ then three of the possible 7! orderings include

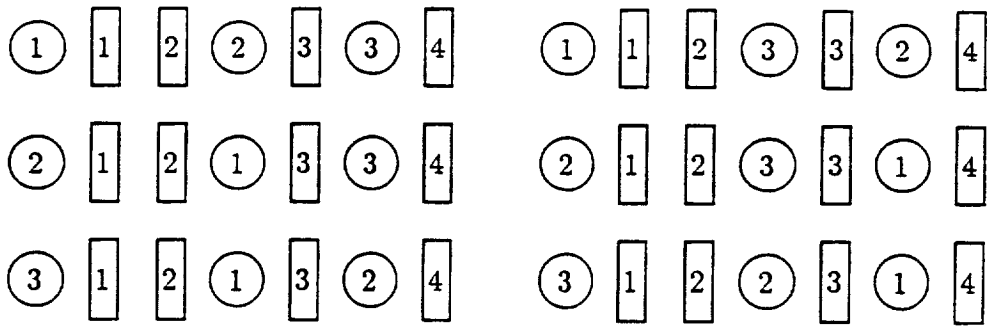


We associate the $n-1$ sticks with the $n-1$ boundaries that occur between n distinct bins when they are stacked side-by-side. Thus, in the first example, ball 1 is in the first bin, ball 2 is between the second and third boundaries, putting it in the third bin, and similarly, ball 3 is in the fourth bin, etc.. The 5-tuples describing the number of balls placed into each bin for the above examples are $(1,0,1,1,0)$, $(0,3,0,0,0)$, and $(0,1,0,0,2)$, respectively.

The above result of ${}^{n-1+r}P$ is based on the distinct balls and distinct sticks. This result needs to be reduced to account for our goal to consider indistinguishable balls and distinct bins.

Given any permutation of the $(n-1+r)$ objects, there exist other permutations that would produce the same result. Thus, without moving the sticks, and without

changing the number of balls between the sticks, the r balls can be permuted ${}^{r}P$ ways. Specifically, for the 5-tuple $(1,0,1,1,0)$ in the example above, the following $3! = 6$ permutations of the balls are possible:



Likewise, the bins are uniquely identified by the *locations* of the sticks only. Independent of the ball permutations, the $n-1$ sticks can be permuted ${}^{n-1}P$ ways. Thus, we conclude that the number of ways to place r indistinguishable balls into n distinct bins is $(n-1+r)!/r!(n-1)!$, precisely, ${}^{n}X_{r}$. □

Two identities are observed directly from the definition of ${}^{n}X_{r}$:

$$ {}^{n}X_{0} = 1 \tag{3.42} $$

$$ {}^{1}X_{r} = 1 \tag{3.43} $$

and, additionally from Lemma 3.1 and (3.39),

$$ {}^{n}X_{r} = {}^{r+1}X_{n-1} \tag{3.44} $$

3.4.2.3 Size of Occupation Set

From Definitions 3.5 and 3.6, it is clear that $|J_n(r)| \leq |I_n(r)|$, and, typically, the number of occupations is much less than the number of partitions. Specifically, since each n-occupation has a minimum of one unit in each of the n elements, $r-n$ units remain to be distributed freely. The ways to distribute these remaining units

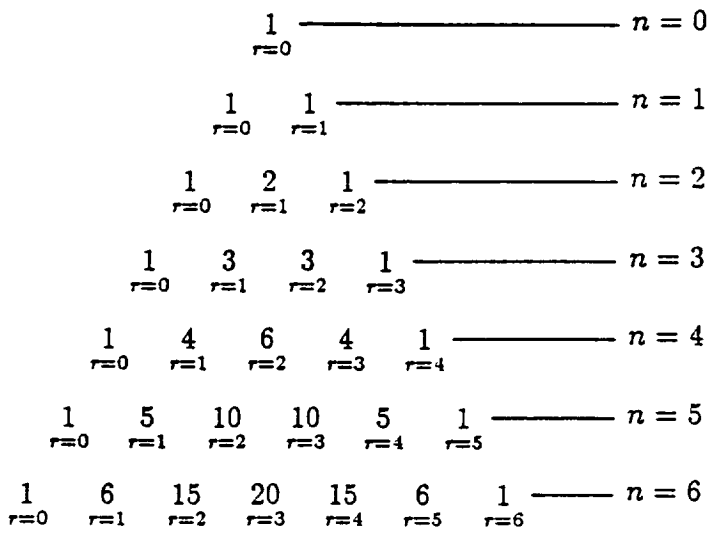


Figure 3.2: Pascal's Triangle as ${}^nC_r$

in the n-vector is given by $I_n(r-n)$. Thus, the size of $J_n(r)$ is given by

$$\begin{aligned} |J_n(r)| &= |I_n(r-n)| \\ &= {}^nX_{r-n} \\ &= {}^{r-1}C_{n-1} \end{aligned} \tag{3.45}$$

3.4.2.4 Pascal's Triangle

Using ${}^nC_r$, an algebraic set of construction rules can be derived from Definition 3.7, (see Figure 3.2). From (3.40) and (3.41), the top entry is given by ${}^0C_0 = 1$, the first entry of a row is given by ${}^nC_0 = 1$, and the last entry of a row is given by ${}^nC_n = 1$. The recurrence relationship in rule 3 can be expressed as

$${}^nC_r = {}^{n-1}C_{r-1} + {}^{n-1}C_r \tag{3.46}$$

which is a well-known combinatorial identity.

3.4.2.5 Binomial Theorem

An expansion of $(a+b)^n$, where $n \in I\!N_+$ is given by the well-known Binomial Theorem.

Theorem 3.1 $(a+b)^n = \sum_{i=0}^{n} {}^nC_i a^{n-i} b^i$

3.4.2.6 Other

The results provided in the following lemmas will be used to develop the state-space size estimators.

Lemma 3.2 *For all* $x, y \in I\!N_{0+}$,

$$\frac{(x+y)!}{x!(y+1)!} \leq (x+1)^y \tag{3.47}$$

Proof: For $y = 0$, we have from (3.47),

$$\frac{x!}{x!} \leq 1 \tag{3.48}$$

which clearly holds. Now that we can assume $y \geq 1$, the remainder of the proof is made by induction on x. For $x = 0$, (3.47) becomes

$$\frac{y!}{(y+1)!} \leq 1$$

which holds for integer $y \geq 1$. We assume (3.47) holds for $x = x_0$, i.e.,

$$\begin{aligned} \frac{(x_0+y)!}{x_0!(y+1)!} &\leq (x_0+1)^y \\ (x_0+y)\cdot(x_0-1+y)\cdots(x_0+1) &\leq (y+1)!(x_0+1)^y \end{aligned} \tag{3.49}$$

and need to show that it holds for the next value of x. Thus, we need to show

$$\frac{(x_0+1+y)!}{(x_0+1)!(y+1)!} \leq (x_0+2)^y \tag{3.50}$$

The validity of (3.50) is proven by contradiction, i.e., we show

$$\begin{aligned}(x_0+1+y)\cdot(x_0+y)\cdots(x_0+2) &> (y+1)!(x_0+2)^y \\ \frac{x_0+1+y}{x_0+1}((x_0+y)\cdots(x_0+1)) &> (y+1)!(x_0+2)^y \end{aligned} \tag{3.51}$$

does not hold. Further simplification of (3.51) results from substituting the assumption (3.49):

$$\begin{aligned}\frac{x_0+1+y}{x_0+1}((y+1)!(x_0+1)^y) &> (y+1)!(x_0+2)^y \\ \frac{x_0+1+y}{x_0+1} &> \left(\frac{x_0+2}{x_0+1}\right)^y\end{aligned}$$

which can be rearranged as

$$1+\frac{y}{x_0+1} > \left(1+\frac{1}{x_0+1}\right)^y \tag{3.52}$$

From the Binomial Theorem, (Theorem 3.1), the righthand side of (3.52) can be expanded to

$$\left(1+\frac{1}{x_0+1}\right)^y = 1+y\frac{1}{x_0+1}+{}^yC_2\left(\frac{1}{x_0+1}\right)^2+\cdots+{}^yC_{y-1}\left(\frac{1}{x_0+1}\right)^{y-1}+\left(\frac{1}{x_0+1}\right)^y$$

which is a sum of positive terms. Since the lefthand side of (3.52) is but the first two terms of this expansion, we conclude that (3.52) does not hold. From this contradiction, (3.50) is proven, which proves the lemma. □

The following lemma states two facts: no matter how x balls are distributed among y boxes (even if the goal is to minimize the number in each box), at least one box will contain at least $\lceil x/y \rceil$ balls; and, similarly, even when trying to maximize the number in each box, at least one box will contain no more than $\lfloor x/y \rfloor$ balls.

Lemma 3.3 *Putting exactly $x \in \mathbb{N}_{0+}$ balls into $y \in \mathbb{N}_{+}$ boxes, where x_i denotes the number of balls in each box, results in the following relationships:*

$$\max_{i=1\ldots y} x_i \geq \lceil x/y \rceil \tag{3.53}$$

$$\min_{i=1\ldots y} x_i \leq \lfloor x/y \rfloor \tag{3.54}$$

Proof: First, (3.53) is proven by contradiction, followed by (3.54). Assume there are y boxes with x_i balls in each, and

$$\begin{aligned} \max_i x_i &< \lceil x/y \rceil \\ &\leq \lceil x/y \rceil - 1 \end{aligned} \tag{3.55}$$

Then the total number of balls among the y boxes would be

$$\begin{aligned} \sum_{i=1}^{y} x_i &\leq y(\lceil x/y \rceil - 1) \\ &\leq (y\lceil x/y \rceil) - y \\ &\leq (x + y - 1) - y \\ &\leq x - 1 \end{aligned}$$

which is a contradiction. Now, assume

$$\begin{aligned} \min_i x_i &> \lfloor x/y \rfloor \\ &\geq \lfloor x/y \rfloor + 1 \end{aligned} \tag{3.56}$$

Then the total number of balls among the y boxes would be

$$\begin{aligned} \sum_{i=1}^{y} x_i &\geq y(\lfloor x/y \rfloor + 1) \\ &\geq (y\lfloor x/y \rfloor) \dot{+} y \\ &\geq (x \dot{-} y \dot{+} 1) \dot{+} y \\ &\geq x \dot{+} 1 \end{aligned}$$

which is a contradiction. □

3.5 Petri Nets

Petri net (PN) theory, [Petr62], is a highly graphical system design and analysis methodology. Additionally, the mathematical foundation of PNs permits structural and performance analyses of models. These two attributes have made PNs an effective modeling tool for Discrete Event Dynamic Systems (DEDSs). In this section, PNs are introduced and some relevant theory is presented. Overviews of PN theory are provided in [Pete81, Mura89].

3.5.1 Definitions

3.5.1.1 Ordinary Petri Nets

Definition 3.8 (OPN) *An Ordinary PN (OPN), $\mathcal{A}$, is: $\mathcal{A} = (P, T, I, O, M^0)$, where*

- $P = \{\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_p\}$, *is a set of p places.*
- $T = \{\mathsf{T}_1, \mathsf{T}_2, \ldots, \mathsf{T}_t\}$, *is a set of t transitions.*
- $I : P \times T \rightarrow \{0,1\}$, *describes the directed connectivity from the places to the transitions.*
- $O : T \times P \rightarrow \{0,1\}$, *describes the directed connectivity from the transitions to the places.*
- $M^0 : P \rightarrow I\!N_{0+}$, *describes the initial marking, i.e., the initial distribution of tokens in the PN.*
- $P \cap T = \emptyset$ *and* $P \cup T \neq \emptyset$.

This definition is preferred to the definition that combines the mappings I and O into a single set $F \subseteq (P \times T) \cup (T \times P)$, called the flow relation.

3.5.1.2 Arc Multiplicity

Ordinary PNs, as defined above, permit only simple connectivity between the places and transitions. An extension of this definition includes arc weights. When arc weights are permitted, the resulting models are simply called Petri nets (PNs).

Definition 3.9 (PN) *A Petri net (PN), $\mathcal{A}$, is: $\mathcal{A} = (P, T, I, O, M^0)$, where P, T, and M^0 are as above and*

- $I : P \times T \rightarrow I\!N_{0+}$, *describes the directed connectivity from the places to the transitions. If $I(\mathsf{P}, \mathsf{T}) = k$, and $k \neq 0$, then place P is an input place to transition T. The arc weight is k.*
- $O : T \times P \rightarrow I\!N_{0+}$, *describes the directed connectivity from the transitions to the places. If $I(\mathsf{T}, \mathsf{P}) = k$, and $k \neq 0$, then place P is an output place to transition T. The arc weight is k.*

3.5.1.3 Place Ordering

The above definition considers the places in the PN as distinct, and enforces no order among them. However, it is natural to label the places with consecutive integers. Mathematically, we associate with every PN a labeling function

- $L : P \leftrightarrow \{1, 2, \ldots, p\}$, describes an ordering to the places.

We often use the place labels $\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_p$, thus making obvious the implicit ordering.

With the implied order among the places, M^0 can be represented as a vector, $\underline{\mu}^0$, having the i^{th} component equal to $M^0(\mathsf{P}_i)$. Thus, the equivalent definition of a PN is $(P, T, I, O, \underline{\mu}^0)$, where any marking of the net can also be represented by a non-negative p-vector.

In a similar manner, an ordering can also be given to the transitions. This permits the matrix notation described below.

3.5.1.4 Incidence Matrix

From the definition of a PN, a matrix notation can be defined that captures the connectivity described in I and O, (or F in the alternative definition).

Definition 3.10 (incidence matrix) *Given a PN with p places and t transitions, then the $p \times t$ incidence matrix, $\underline{\underline{C}}$, is formulated as follows:*

- *$\underline{\underline{C}}^-$ is the matrix of input arc weights, where $C^-_{i,j}$ is the number of input arcs from place P_i to transition T_j, i.e., $C^-_{i,j} = I(\mathsf{P}_i, \mathsf{T}_j)$.*
- *$\underline{\underline{C}}^+$ is the matrix of output arc weights, where $C^+_{i,j}$ is the number of output arcs from transition T_j to place P_i, i.e., $C^+_{i,j} = O(\mathsf{T}_j, \mathsf{P}_i)$.*
- *$\underline{\underline{C}}$ is the incidence matrix for the PN, given by $\underline{\underline{C}} = \underline{\underline{C}}^+ - \underline{\underline{C}}^-$.*

Some researchers use the transpose of this definition of the incidence matrix.

3.5.1.5 Firing Rules and Sequences

The flow of tokens through the net is regulated by the firing rules associated with the transitions. Given a PN $\mathcal{A}$, with a current marking $\underline{\mu}$, then a transition T_j is said to be *enabled* if $\mu_i \geq I(\mathsf{P}_i, \mathsf{T}_j)$, $\forall i = 1, \ldots, p$.

Definition 3.11 (transition enabled set) *The transition enabled set of a PN $\mathcal{A}$ having marking $\underline{\mu}$ is*

$$ES(\mathcal{A}, \underline{\mu}) \stackrel{\text{def}}{=} \{t \in T \mid \mu_i \geq I(\mathsf{P}_i, t);\ \forall i = 1, 2, \ldots, p\} \tag{3.57}$$

From the marking $\underline{\mu}$, any transition in $ES(\mathcal{A}, \underline{\mu})$ may fire, whereby tokens are removed from the transition's input place(s) and deposited into the transition's output place(s). Thus, assuming $\mathsf{T}_j \in ES(\mathcal{A}, \underline{\mu})$, firing T_j, results in the new marking, $\underline{\mu}'$, given by

$$\mu'_i = \mu_i - I(\mathsf{P}_i, \mathsf{T}_j) + O(\mathsf{T}_j, \mathsf{P}_i); \qquad i = 1, 2, \ldots, p \tag{3.58}$$

This firing rule is expressed compactly with the incidence matrix defined above:

$$\underline{\mu}' = \underline{\mu} + \underline{\underline{C}} \cdot \underline{e}_j \tag{3.59}$$

where $\underline{e}_j$ is the j^{th} elementary vector.

Firing a single transition is generalized to firing a sequence of transitions. Define σ to be an ordered sequence of r legal transition firings, i.e., $\sigma = \mathsf{T}_{i_1}\mathsf{T}_{i_2}\ldots\mathsf{T}_{i_r}$. Let $\sigma|_{\mathsf{T}}$ denote the number of occurrences of transition T in σ. Then, we obtain the firing count vector, $\underline{\sigma}$, as follows:

$$\underline{\sigma} \stackrel{\text{def}}{=} \begin{bmatrix} \sigma|_{\mathsf{T}_1} \\ \sigma|_{\mathsf{T}_2} \\ \vdots \\ \sigma|_{\mathsf{T}_t} \end{bmatrix} \tag{3.60}$$

Thus, from the current marking, $\underline{\mu}$, firing the transition sequence σ results in the new marking, $\underline{\mu}'$:

$$\underline{\mu}' = \underline{\mu} + \underline{\underline{C}} \cdot \underline{\sigma} \tag{3.61}$$

The set of all legal firing sequences from a marking $\underline{\mu}$ is denoted as $\Sigma(\mathcal{A}, \underline{\mu})$.

3.5.1.6 Reachability Graph

The states of a PN are associated with the markings that can result from applying the above firing rules to the initial marking. From a PN $\mathcal{A}$, a reachability graph can be derived. This graph is composed of a set of states, i.e., the reachability set $RS(\mathcal{A}, \underline{\mu}^0)$, and a set of directed links that indicate legitimate state changes via specific transition firings. Some PNs have an unbounded reachability graph, whereby, the cardinality of $RS(\mathcal{A}, \underline{\mu}^0)$ is infinite. Since the problem in hand is to estimate the number of states for a PN, we assume that its reachability set is finite. Formally, we have

- $RS(\mathcal{A}, \underline{\mu}^0) \subset I\!N_{0+}^p$
- $\underline{\mu} \in RS(\mathcal{A}, \underline{\mu}^0) \Leftrightarrow \exists \sigma \in \Sigma(\mathcal{A}, \underline{\mu}^0) \ni \underline{\mu} = \underline{\mu}^0 + \underline{\underline{C}} \cdot \underline{\sigma}$
- $|RS(\mathcal{A}, \underline{\mu}^0)| < \infty$.

3.5.1.7 Stochastic Petri Nets

Stochastic Petri nets (SPNs) are defined as an extension of PNs into the time domain. Thus, each transition is associated with an exponential density function that describes the probability of the time required to fire. This is accomplished by appending the definition of PNs with an additional injection that specifies the exponential parameter for each transition.

Definition 3.12 (SPN) *A Stochastic PN (SPN), $\mathcal{A}$, is: $\mathcal{A} = (P, T, I, O, \Lambda, \mu_0)$, with Λ being*

- $\Lambda : T \rightarrow I\!R_+$, *describes the exponential parameter associated with each transition.*

Since the exponential family of density functions has infinite support, the reachability graph for a SPN is formed by considering the underlying PN, [Moll82, Moll84]. The reachability set contains all the states that are possible, and the directed links indicate not only legitimate state transitions, but also the infinitesimal rate of state change.

3.5.1.8 Immediate Transitions

Within SPNs, token flow can occur only after some random time associated with an enabled transition has elapsed. However, parts of the model may be expressing logical connectivity rather than timed events. SPNs can be extended to Generalized

Stochastic Petri nets (GSPNs), whereby "immediate" transitions, i.e., transitions that fire with no delay, are also permitted, [Mars84a, Balb87].

An enabled immediate transition will be fired before any enabled exponential transition. An immediate transition can be signified by the exponential density parameter of ∞, agreeing with the mathematical interpretation of the density function. Thus, for GSPNs the definition of Λ is extended to

- $\Lambda : T \rightarrow I\!R_+ \cup \{\infty\}$.

The resulting reachability set still contains all the states that can be legally reached. However, states exited because of an immediate transition firing are occupied for zero amount of time, and are called "vanishing;" the remaining states will be occupied for some random amount of time, and are called "tangible," [Mars84a, Balb87].

The distinction between vanishing and tangible states permits reduction of the performance analysis problem. It does not, however, permit reduction of the structural analysis problems, nor scheduling problems.

3.5.1.9 Inhibitor Arcs

Another extension to the basic types of PNs permits the use of *inhibitor arcs*. An inhibitor arc from place P_i to transition T_j will cause T_j to be *disabled* if P_i contains a token. A weight can be associated with the inhibitor arc, whereby a weight of k indicates the transition cannot become enabled if the place contains at least k tokens.

This extension does not increase the modeling power of PNs, [Mura89]. The functionality of inhibitor arcs can be accomplished using the places, transitions, and arcs of basic PNs. Furthermore, since using inhibitor arcs prohibits analysis of the PN based on its incidence matrix, this extension will not be considered here.

3.5.1.10 Conservative Petri Nets

A *strictly conservative* PN is one that neither generates nor absorbs tokens. Thus, in this type of PN, the number of tokens in the model is unchanged after any transition firing. An immediate consequence of strict conservatism is that every transition in the PN that can be enabled by some marking must have as many output arcs as it does input arcs (including arc weights). This property often appears in models for resource allocation.

A generalized notion of a conservative PN is that of *weighted conservativeness*—also known as *conservative with respect to (wrt) a weight vector*. In a resource allocation model, this generalization accounts for the fact that a token in a given place may represent a set of resources rather than just a single resource.

Definition 3.13 (conservative wrt a weight vector) *A PN, $\mathcal{A} = (P, T, I, O, \underline{\mu}^0)$, is conservative wrt $\underline{w} \in \mathbb{R}^n_+$, if*

$$\underline{\mu}_1,\ \underline{\mu}_2 \in RS(\mathcal{A}, \underline{\mu}^0) \ \Rightarrow\ \underline{w} \circ \underline{\mu}_1 \ =\ \underline{w} \circ \underline{\mu}_2. \tag{3.62}$$

Definition 3.14 (strictly conservative) *A PN, $\mathcal{A} = (P, T, I, O, \underline{\mu}^0)$, is strictly conservative if it is conservative wrt $\underline{w} = \underline{1}$.*

3.5.1.11 Petri Net Subclasses

Various subclasses of PNs can be defined. These definitions apply only to OPNs, i.e., PNs without arc multiplicity or inhibitor arcs. Given an OPN, $\mathcal{A}$, then for each $\mathsf{T} \in T$ and $\mathsf{P} \in P$, we define

- ${}^\bullet\mathsf{T} \stackrel{\text{def}}{=} \{p \mid I(p, \mathsf{T}) = 1\}$. is the set of input places to T.
- $\mathsf{T}^\bullet \stackrel{\text{def}}{=} \{p \mid O(\mathsf{T}, p) = 1\}$, is the set of output places from T.
- ${}^\bullet\mathsf{P} \stackrel{\text{def}}{=} \{t \mid O(t, \mathsf{P}) = 1\}$, is the set of transitions that output to P, i.e., the set of input transitions to P.

- $\mathsf{P}^{\bullet} \stackrel{\text{def}}{=} \{t \mid I(\mathsf{P}, t) = 1\}$, is the set of transitions to which P is an input place, i.e., the set of output transitions from P.

From the preset and postset definitions above, we define the following PN subclasses:

Definition 3.15 (state machine) *A PN state-machine has exactly one input and output arc for each transition, i.e.,* $t \in T \Rightarrow |{}^{\bullet}t| = |t^{\bullet}| = 1$.

Definition 3.16 (marked graph) *A marked graph has exactly one input and output arc for each place, i.e.,* $p \in P \Rightarrow |{}^{\bullet}p| = |p^{\bullet}| = 1$.

Definition 3.17 (free-choice net) *A free-choice net limits places having common output transitions to having only a single output arc, i.e.,* $p_1, p_2 \in P, p_1^{\bullet} \cap p_2^{\bullet} \neq \emptyset \Rightarrow |p_1^{\bullet}| = |p_2^{\bullet}| = 1$.

Other subclass definitions exist, but have little theory associated with them.

3.5.1.12 Liveness

PN liveness is considered by the liveness of its transitions and is qualified with a level, [Mura89]. If no reachable states enable a specific transition $\mathsf{T} \in T$, the transition is said to be dead. The levels of liveness describe the number of times T can be enabled by various states.

3.5.1.13 Boundedness

Definition 3.18 (boundedness) *A PN,* $\mathcal{A}$, *is bounded if* $|RS(\mathcal{A}, \underline{\mu}^0)| < \infty$.

The reachability set is as dependent on the structure of the model, $\mathcal{A}$, as it is on the initial marking, $\underline{\mu}^0$. *Structural boundedness* means the PN is bounded for *any* finite initial marking.

3.5.1.14 P- and T-Invariants

Two types of invariants are defined for PNs, [Mura89, Colo90]. A P-invariant of $\mathcal{A}$ is a set of places that maintains a constant sum of tokens (in a weighted sense) throughout the evolution of the model. Using the incidence matrix of Definition 3.10, we have:

Definition 3.19 (P-invariant) *A p-vector, $\underline{x} \in \mathbb{N}_{0+}^{p}$ is a P-invariant of $\mathcal{A}$ if*

$$\underline{\underline{C}}^{\mathsf{T}} \underline{x} \;=\; \underline{0} \tag{3.63}$$

The set of P-invariants for $\mathcal{A}$ is denoted as $PI(\mathcal{A})$. Many p-vectors may satisfy this definition. In particular, if $\underline{x}_1$ and $\underline{x}_2$ are P-invariants of $\mathcal{A}$, then so is any vector formed by a linear combination of $\underline{x}_1$ and $\underline{x}_2$:

$$\underline{x}_1, \underline{x}_2 \in PI(\mathcal{A}) \;\Rightarrow\; \alpha\underline{x}_1 + \beta\underline{x}_2 \in PI(\mathcal{A}); \qquad \alpha, \beta \in \mathbb{N}_{0+} \tag{3.64}$$

Additionally, since the zero-vector is a P-invariant of all PNs, $PI(\mathcal{A})$ forms a linear vector-space on the field of integers. This allows the *dimension* of $PI(\mathcal{A})$ to be computed and a suitable basis to be defined.

Likewise, a T-invariant of $\mathcal{A}$ is a set of transitions that when fired (possibly more than once) has the aggregate result of leaving the marking unchanged:

Definition 3.20 (T-invariant) *A t-vector, $\underline{y} \in \mathbb{N}_{0+}^{t}$ is a T-invariant of $\mathcal{A}$ if*

$$\underline{\underline{C}} \cdot \underline{y} \;=\; \underline{0} \tag{3.65}$$

3.5.2 Theory

3.5.2.1 Computing the Conservative Weight Vector

The following method exists to compute the weight vector for a conservative PN, [Pete81]. If $\underline{w}$ is a conservative weight vector for a PN, then (3.62) must be

satisfied by $\underline{\mu}^0$ and any marking resulting from a sequence of legal transition firings. Thus,

$$\begin{aligned} \underline{w} \circ \underline{\mu}^0 &= \underline{w} \circ (\underline{\mu}^0 + \underline{\underline{C}} \cdot \underline{\sigma}) \\ \underline{w} \circ \underline{\mu}^0 &= \underline{w} \circ \underline{\mu}^0 + \underline{w} \circ (\underline{\underline{C}} \cdot \underline{\sigma}) \\ 0 &= (\underline{w}^{\mathsf{T}} \underline{\underline{C}}) \cdot \underline{\sigma} \qquad (3.66) \\ \underline{0}^{\mathsf{T}} &= \underline{w}^{\mathsf{T}} \underline{\underline{C}}. \qquad (3.67) \end{aligned}$$

Peterson argues that (3.67) follows from (3.66) since (3.66) must hold for any $\underline{\sigma}$, and further states "a Petri net is [weighted] conservative if and only if there exists a positive vector $\underline{w}$ such that (3.67) holds." For convenience, in the following, the transpose of (3.67) is used, which is an equivalent relationship:

$$\underline{\underline{C}}^{\mathsf{T}} \underline{w} = \underline{0}. \qquad (3.68)$$

Consider, for example, the PN in Figure 3.3. The incidence matrix of this PN is given by

$$\underline{\underline{C}} = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 1 & -1 \end{bmatrix}. \qquad (3.69)$$

Thus, (3.69) and (3.68) yield

$$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 1 \\ 1 & 0 & -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

which has an infinite number of positive solutions of the form

$$\underline{w} = \begin{bmatrix} \alpha \\ \alpha \\ \beta \\ \alpha - \beta \end{bmatrix} \qquad \alpha \in (0, \infty), \ \beta \in (0, \alpha). \qquad (3.70)$$

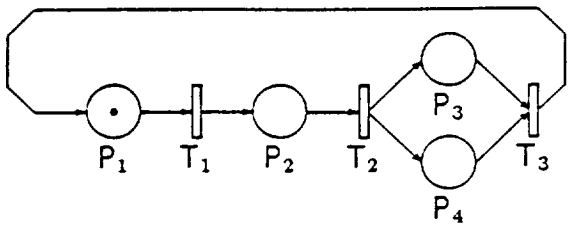


Figure 3.3: Weighted Conservative PN

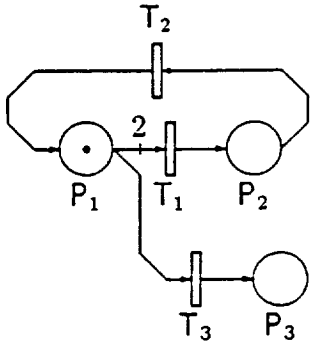


Figure 3.4: Weighted Conservative PN That Does Not Satisfy $\underline{\underline{C}}^{\mathsf{T}}\underline{w} = \underline{0}$

3.5.2.2 Structural Versus Behavioral Conservatism

Application of the above test for weighted conservatism is clarified, [Wats92b]. This is motivated by the differences between *structural* and *behavioral* conservatism, and the fact that there are PNs that demonstrate the property of weighted conservatism without satisfying (3.68).

Consider the PN in Figure 3.4, which has 3 places and 3 transitions. The reachability set consists of the following markings: $[1\ 0\ 0]^{\mathsf{T}}$ and $[0\ 0\ 1]^{\mathsf{T}}$. Thus, by inspection of the reachability set, (3.62) is satisfied by $\underline{w} = [1\ 1\ 1]^{\mathsf{T}}$ (actually, $\underline{w}$ can be any vector of the form $[\alpha\ \beta\ \alpha]^{\mathsf{T}}$, where α and β are positive and finite).

The incidence matrix for this PN is

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

and therefore (3.68) gives

$$\begin{bmatrix} -2 & 1 & 0 \\ 1 & -1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

which does not have a positive solution. Thus, the PN in Figure 3.4 is weighted conservative but does not satisfy (3.68).

Further analysis of the PN in Figure 3.4 illustrates how this situation can occur. Note that the PN is bounded for any initial marking. However, if the initial marking were $[2\ 0\ 0]^T$, the PN is not weighted conservative since after firing transitions T_1 and T_2, place P_1 would contain only a single token. Thus, the conservative nature of this PN is dependent on its initial condition—specifically, the PN is conservative for any initial marking that does not enable transition T_1. The definition of weighted conservatism, (3.62), is based on the PN's reachability set, and therefore dependent on its initial marking. Since only the PN's incidence matrix, and not its initial marking, is considered in the computation of the conservative weight vector, (3.68) might be more accurately termed as an iff relationship for *structural* weighted conservatism. PNs satisfying this relationship are weighted conservative for all bounded initial markings.

The term *behavioral* is used to describe those properties that are a result of the PN's initial marking. Obviously, if a PN satisfies (3.68), then it is also behaviorally weighted conservative. The distinction is that (3.68) is only a sufficient condition for behavioral conservatism, not also a necessary condition. Thus, given a PN that fails this test, then it is conclusive that the PN is not structurally weighted conservative, however, further testing is required to determine whether or not the PN will actually demonstrate the weighted conservative property. No additional tests have been proposed, leaving examination of the PN's reachability set as the only alternative.

3.5.2.3 Equivalent Definition of Conservative Petri Nets

A constraint is added to the definition of weighted conservative to prevent arbitrary scaling.

Definition 3.21 (alternative definition of a conservative PN, [Wats92b]) *A PN $\mathcal{A}$, with $RS(\mathcal{A}, \underline{\mu}^0)$, is conservative wrt weight vector $\underline{w} \in \mathbb{R}_+^p$ if*

$$\mu_1, \mu_2 \in RS(\mathcal{A}, \underline{\mu}^0) \Rightarrow \underline{w} \circ \underline{\mu}_1 = \underline{w} \circ \underline{\mu}_2 \tag{3.71}$$

$$\underline{w} \circ \underline{\mu}^0 = 1 \tag{3.72}$$

This alternative definition is not suitable for a PN that has an empty initial marking, (hence, (3.72) cannot be satisfied). This is an unimportant case, since if the PN is conservative, then an empty initial marking implies that the entire state-space contains only this single marking.

3.5.2.4 Throughput Subnets

Restricting GSPNs to immediate and exponential transition functions limits the internal state changes to immediate and exponential rates. *Throughput subnets* implement an input-output behavior, that allow a variety of non-exponential density functions to be modeled, [Wats89].

A throughput subnet is defined as a live GSPN subnet, having a single entry and exit path, and an empty initial marking, that models a particular transition delay distribution in the system. Thus, a throughput subnet is used for its input-output properties. Since a throughput subnet does not absorb nor generate tokens, and has no initial marking, it may be considered as a single transition in the determination of structural properties (i.e., liveness, boundedness, etc.).

The simplest throughput subnet would be a single exponential transition, (with parameter λ) between an input place, P_1, and an output place, P_2. Note the transition

is assumed to be exponential, because an immediate transition between two places results in only vanishing markings, which do not affect performance data. Whenever a token appears in P_1, it should appear in P_2 after an exponentially distributed delay. If multiple tokens arrived at P_1, we would expect them to appear in P_2 independent of one another. That is, since the throughput subnet is implementing an effective density, the delay experienced by any token in the subnet should not be affected by the presence of another token.

Various combinations of places, exponential transitions, and immediate transitions can be used between input and output places to achieve different delay distributions. Figure 3.5 depicts three throughput subnets formed by series and parallel connections. From the stochastic relationships between the transitions, we have the following results, [Wats89].

The parallel connection in Figure 3.5a results in no additional modeling flexibility with respect to performance analysis:

$$\mathcal{T}_a = \min(\mathcal{T}_1, \mathcal{T}_2) \tag{3.73}$$

$$f_a(t) = (\lambda_1 + \lambda_2)e^{-(\lambda_1+\lambda_2)t}, \qquad t \geq 0. \tag{3.74}$$

The effective delay in Figure 3.5b is the sum of the two serial transitions:

$$\mathcal{T}_b = \mathcal{T}_1 + \mathcal{T}_2 \tag{3.75}$$

$$f_b(t) = \frac{\lambda_1\lambda_2(e^{-\lambda_1 t} - e^{-\lambda_2 t})}{(\lambda_2 - \lambda_1)}, \qquad t \geq 0, \tag{3.76}$$

which can be generalized to more than two transitions, (3.15).

Figure 3.5c depicts a hyperexponential density function between P_1 and P_2:

$$\mathcal{T}_c = r_1\mathcal{T}_1 + (1 - r_1)\mathcal{T}_2 \tag{3.77}$$

$$f_c(t) = r_1\lambda_1 e^{-\lambda_1 t} + (1 - r_1)\lambda_2 e^{-\lambda_2 t}, \qquad t \geq 0. \tag{3.78}$$

where, r_1 and r_2 are the probabilities of taking one path over the other, (3.18).

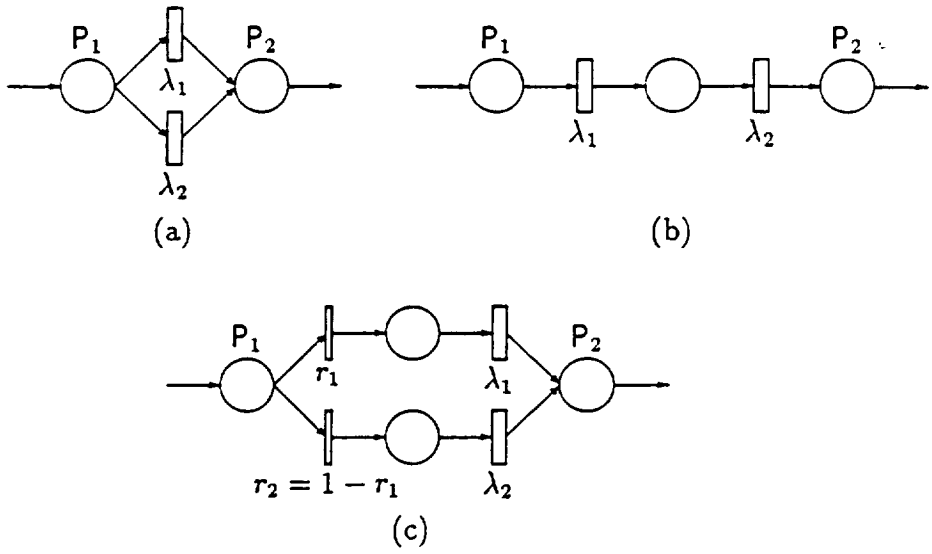


Figure 3.5: Throughput Subnets: Series and Parallel Connections

3.5.2.5 Transition Firing Policies

The semantics associated with enabling and firing a timed transition can be adjusted for a particular application. In [Mars85], several firing policies are analyzed. Once a transition is enabled by a marking, it may eventually fire, or before firing, become disabled by some other transition firing. How the firing delay function for a transition is reset can be done in one of two ways:

- *Age memory*: a transition resamples its firing delay density function only after firing.
- *Enabling memory*: a transition resamples its firing delay density function each occurrence of becoming enabled.

How a particular transition is chosen to fire can be done in one of two ways:

- *Race*: the enabled transition with the shortest delay fires.
- *Preselection*: the transition to be fired is chosen independently of the delay functions.

It was shown that these policies were equivalent for exponential transitions, as used in SPNs. However, for PN extensions that permit non-exponential transitions, the firing policy becomes an important issue.

3.5.2.6 Computation of P-Invariants

Colom, [Colo90], has developed polynomial time algorithms for the computation of P-invariants. These algorithms take advantage of the fact the P-invariants form a linear vector space, and are easily implemented in a matrix based programming language, such as Matlab, [Matl90].

3.6 Summary

This chapter has introduced the notation, definitions, and theory that will be used in later chapters. Specifically the topics of stochastics, combinatorics, and Petri nets (PNs) were presented.

The discussion on stochastics will be utilized in Chapter 4, where an algorithm for approximating non-exponential transitions within GSPNs is developed. Chapters 6 and 7, which present the state-space size estimators, will draw from the discussion on combinatorics. The combinatoric lemmas presented will be used in these chapters. The detailed introduction of PNs will be utilized in the theoretical developments throughout. In particular, the attention given to conservative PNs is needed for one of the state-space size estimators that is based on this type of PN.

CHAPTER 4
NON-EXPONENTIAL TRANSITION MODELING

4.1 Introduction

Generalized Stochastic Petri net (GSPN) models have been used for performance analysis of various systems, for example [Rama80, Moll82, Alla85, Garg85, Holl87, Visw87, AlJa90a]. The restriction of allowing only exponential and immediate transition functions simplifies the problem of computing the performance data, but limits accuracy for systems that actually contain non-exponential state transitions. This chapter introduces and develops a GSPN construct called an s-transition. This construct is based on the throughput subnets (introduced in Section 3.5), and allows non-exponential state transition functions in a system to be approximated within a GSPN.

Other methods for incorporating non-exponential density functions into GSPNs have been proposed (e.g., [Chen89]). Our objective in this work is to provide an abstract transition type with a user-specified transition function. A throughput subnet has the flexibility to be enabled by multiple tokens and serve them simultaneously. This property is useful when modeling systems that contain multiple instances of the same service density function. Conflict for resources, etc., are implemented by other structures in the model, and the throughput subnet models the input-output behavior of the evolution of a user-specified time delay.

4.2 S-transitions

The method used here to approximate a density function within a GSPN is to match moments of a throughput subnet's effective delay with the moments of the specification. The resulting throughput subnet is called an *s-transition*, (for

specified-transition). This algorithm matches first- and second-moments, and therefore approximates the desired density better than achieved by matching only mean (as is done with exponential transitions).

Assume the desired transition delay, $\mathcal{D}$, has density function, $f_{\mathcal{D}}(\cdot)$, with mean, $\mu_{\mathcal{D}}$, and standard deviation, $\sigma_{\mathcal{D}}$. The specified mean and standard deviation can be interpreted as a point in the real plane, i.e., $(\mu_{\mathcal{D}}, \sigma_{\mathcal{D}}) \in I\!R^2$. An algorithm is then needed to map this point into GSPN constructs. The following considerations justify restricting our attention to the *open* first quadrant of $I\!R^2$.

Naturally, we require the $\mathcal{D}$ to take on only non-negative values, (i.e., $f_{\mathcal{D}}(t) = 0, t < 0$). This requirement states the physical reality that the transition delay will never occur in a negative amount of time. As a result of the single-sided density function, $\mu_{\mathcal{D}}$ will be non-negative; $\mu_{\mathcal{D}} = 0$ corresponds to no delay and can be modeled with an immediate transition. Similarly, $\sigma_{\mathcal{D}}$ is non-negative by definition, whereby $\sigma_{\mathcal{D}} = 0$ is equivalent to a non-random transition, which could never be implemented with a finite number of random processes, [Papo84].

Figure 4.1 shows the model for an s-transition that creates the specified delay between places P_1 and P_2. Depending on $f_{\mathcal{D}}(\cdot)$, the s-transition is implemented with a series structure (exponential and Erlangian subnets in series), or a parallel structure (a hyperexponential subnet). This choice for the s-transition model takes advantage of the facts that, in general, Erlangian densities satisfy $\mu \geq \sigma$, and hyperexponential densities satisfy $\mu \leq \sigma$, [Kost73]. The parameters (λ_1, λ_2, and n, on the one hand, or r_1 and λ_h, on the other) are chosen so the mean and standard deviation of the throughput subnet are $\mu_{\mathcal{D}}$ and $\sigma_{\mathcal{D}}$, respectively.

The hyperexponential model has two degrees-of-freedom, namely r_1 and λ_h, that allow any $\sigma \geq \mu > 0$ to be achieved, thereby covering the portion of the open first quadrant above the 45 degree line. Even though two degrees-of-freedom would be provided by two exponential transitions in series, (i.e., λ_1 and λ_2), or by an Erlangian

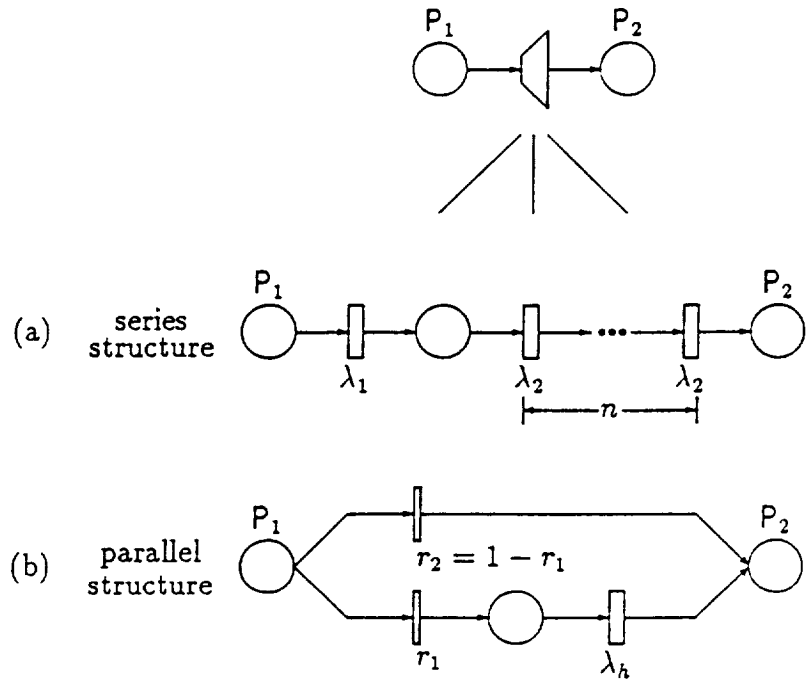


Figure 4.1: S-transition Model

density, (i.e., λ and n), they are, in both cases, not able to cover the remaining portion of the open first quadrant. Putting an exponential and Erlangian subnet in series, however, provides this ability, and additionally a non-unique solution. The following algorithm performs the mapping from the open first quadrant of $\mathbb{R}^2$ to GSPN constructs, and when multiple solutions exist, forms the minimal subnet.

4.3 S-transition Derivation

In this section, the s-transition algorithm is developed. Parameters for the s-transition structures are derived: the series structure applies for $\mu > \sigma > 0$; and the parallel structure applies for $\sigma > \mu > 0$. The remaining case, i.e., $\mu = \sigma > 0$, is shown to be a degeneracy of both structures.

4.3.1 Series Structure

The parameters for the series s-transition model, (Figure 4.1a), are derived so the resulting net has a minimal number of place/transition pairs. Additionally, it is proven that this construct can be used for all $\mu > \sigma > 0$.

The random variable of interest, $\mathcal{T}$, is the s-transition delay. Thus, $\mathcal{T} = \mathcal{T}_1 + \mathcal{T}_2$, where $\mathcal{T}_1$ and $\mathcal{T}_2$ are the random delays of the exponential and Erlangian subnets, respectively. From (3.12–3.14) and (3.15–3.17), the density functions, means, and standard deviations for these two delays are the following:

$$f_1(t) = \lambda_1 e^{-\lambda_1 t} \tag{4.1}$$

$$\mu_1 = 1/\lambda_1 \tag{4.2}$$

$$\sigma_1 = 1/\lambda_1 \tag{4.3}$$

$$\sigma_1^2 = \mu_1^2 \tag{4.4}$$

$$f_2(t) = \frac{\lambda_2^n}{(n-1)!} t^{n-1} e^{-\lambda_2 t} \tag{4.5}$$

$$\mu_2 = n/\lambda_2 \tag{4.6}$$

$$\sigma_2 = \sqrt{n}/\lambda_2 \tag{4.7}$$

$$\sigma_2^2 = \mu_2^2/n \tag{4.8}$$

The exponential and Erlangian subnets are in series and independent, thus, from (3.25,3.27), we have $\mu_{\mathcal{T}} = \mu_1 + \mu_2$ and $\sigma_{\mathcal{T}}^2 = \sigma_1^2 + \sigma_2^2$. The model match involves calculating λ_1, λ_2, and n, such that $\mu_{\mathcal{T}} = \mu_{\mathcal{D}}$ and $\sigma_{\mathcal{T}} = \sigma_{\mathcal{D}}$, the desired mean and standard deviation, respectively.

Substituting (4.4) and (4.8) into $\sigma_{\mathcal{D}}^2 = \sigma_1^2 + \sigma_2^2$, and then eliminating μ_1 using $\mu_1 = \mu_{\mathcal{D}} - \mu_2$, we get

$$\sigma_{\mathcal{D}}^2 = \mu_{\mathcal{D}}^2 - 2\mu_{\mathcal{D}}\mu_2 + \frac{n+1}{n}\mu_2^2, \tag{4.9}$$

which leads to the following quadratic in μ_2

$$\mu_2^2 - \frac{2n\mu_{\mathcal{D}}}{n+1}\mu_2 + \frac{n(\mu_{\mathcal{D}}^2 - \sigma_{\mathcal{D}}^2)}{n+1} = 0. \tag{4.10}$$

The roots of this equation are

$$\mu_2 = \frac{n\mu_{\mathcal{D}} \pm \sqrt{n(n+1)\sigma_{\mathcal{D}}^2 - n\mu_{\mathcal{D}}^2}}{n+1}, \tag{4.11}$$

and since $\mu_1 = \mu_{\mathcal{D}} - \mu_2$, we have

$$\mu_1 = \frac{\mu_{\mathcal{D}} \mp \sqrt{n(n+1)\sigma_{\mathcal{D}}^2 - n\mu_{\mathcal{D}}^2}}{n+1}. \tag{4.12}$$

Using (4.2) and (4.6), the above expressions can be used to determine λ_1 and λ_2, where either the top or bottom signs of (4.11) and (4.12) are used. Since μ_1 and μ_2 must be real, n can be computed from the discriminant of the quadratic expressions:

$$n(n+1)\sigma_{\mathcal{D}}^2 - n\mu_{\mathcal{D}}^2 \geq 0. \tag{4.13}$$

Rearranging to obtain an inequality for n produces

$$n \geq (\mu_{\mathcal{D}}/\sigma_{\mathcal{D}})^2 - 1. \tag{4.14}$$

Since n is an integer and we wish to make the s-transition minimal in the number of place/transition pairs, we choose n to be the unique integer satisfying

$$(\mu_{\mathcal{D}}/\sigma_{\mathcal{D}})^2 - 1 \leq n < (\mu_{\mathcal{D}}/\sigma_{\mathcal{D}})^2. \tag{4.15}$$

We now show that this s-transition structure is valid for all $\mu > \sigma > 0$.

Theorem 4.1 *Given a density function $f_{\mathcal{D}}(\cdot)$ with $\mu_{\mathcal{D}} > \sigma_{\mathcal{D}} > 0$, there exists $n \in \mathbb{N}_+$, and $\lambda_1, \lambda_2 \in \mathbb{R}_+$, such that the series s-transition structure obtains $\mu_{\mathcal{D}}$ and $\sigma_{\mathcal{D}}$.*

Proof: Since $\sigma_{\mathcal{D}} > 0$, by (4.15) n is finite. Additionally, $\mu_{\mathcal{D}} > \sigma_{\mathcal{D}}$, bounds n below by a small positive number. Thus, $n \in \mathbb{N}_+$. By (4.2) and (4.6), $0 < \lambda_1, \lambda_2 < \infty$ is

equivalent to $0 < \mu_1, \mu_2 < \infty$. The top and bottom signs in (4.11) and (4.12) are considered as separate cases.

Considering the top signs, it is clear that $\mu_2 > 0$. Additionally, since the numerator is finite and $n \neq -1$, μ_2 is finite. Furthermore, $\mu_1 > 0$ iff

$$\mu_{\mathcal{D}} > \sqrt{n(n+1)\sigma_{\mathcal{D}}^2 - n\mu_{\mathcal{D}}^2}. \tag{4.16}$$

Squaring both sides and rearranging terms gives

$$(n+1)\mu_{\mathcal{D}}^2 > n(n+1)\sigma_{\mathcal{D}}^2, \tag{4.17}$$

which simplifies to

$$n < (\mu_{\mathcal{D}}/\sigma_{\mathcal{D}})^2. \tag{4.18}$$

This expression is satisfied by the choice of n, (4.15).

Consider, now, the bottom signs. We have $0 < \mu_1 < \infty$ and $\mu_2 < \infty$ immediately. For $\mu_2 > 0$, then

$$n\mu_{\mathcal{D}} > \sqrt{n(n+1)\sigma_{\mathcal{D}}^2 - n\mu_{\mathcal{D}}^2}. \tag{4.19}$$

Rearranging gives

$$n^2(\mu_{\mathcal{D}}^2 - \sigma_{\mathcal{D}}^2) > -n(\mu_{\mathcal{D}}^2 - \sigma_{\mathcal{D}}^2), \tag{4.20}$$

which simplifies to $n^2 > -n$. This inequality holds for all $n > 0$, which is guaranteed by (4.15). □

4.3.2 Parallel Structure

The parameters for the hyperexponential s-transition model, (Figure 4.1b), are derived. Additionally, it is proven that this construct can be used for all $\sigma > \mu > 0$.

From the model in Figure 4.1b, the s-transition delay is $\mathcal{T} = r_1\mathcal{T}_h$, where $\mathcal{T}_h$ is the random delay of the exponential subnet. Thus, from (3.12–3.14) and (3.18–3.20), we have

$$f_{\mathcal{T}}(t) = r_1\lambda_h e^{-\lambda_h t} \tag{4.21}$$

$$\mu_{\mathcal{T}} = r_1/\lambda_h \tag{4.22}$$

$$\sigma_{\mathcal{T}} = \frac{\sqrt{2r_1 - r_1^2}}{\lambda_h} \tag{4.23}$$

The model match involves calculating r_1 and λ_h, (by convention, r_2 is set to $1 - r_1$), such that $\mu_{\mathcal{T}} = \mu_{\mathcal{D}}$ and $\sigma_{\mathcal{T}} = \sigma_{\mathcal{D}}$, the desired mean and standard deviation, respectively. Equating λ_h in (4.22) and (4.23), we have

$$(2r_1 - r_1^2)\mu_{\mathcal{D}}^2 = r_1^2\sigma_{\mathcal{D}}^2, \tag{4.24}$$

which simplifies to

$$r_1 = \frac{2\mu_{\mathcal{D}}^2}{\mu_{\mathcal{D}}^2 + \sigma_{\mathcal{D}}^2}. \tag{4.25}$$

Substituting (4.25) into (4.22) gives

$$\lambda_h = \frac{2\mu_{\mathcal{D}}}{\mu_{\mathcal{D}}^2 + \sigma_{\mathcal{D}}^2}. \tag{4.26}$$

We now show that this s-transition structure is valid for all $\sigma > \mu > 0$.

Theorem 4.2 *Given a density function $f_{\mathcal{D}}(\cdot)$ with $\sigma_{\mathcal{D}} > \mu_{\mathcal{D}} > 0$, there exists $r_1 \in (0,1)$, and $\lambda_h \in \mathbb{R}_+$, such that the parallel s-transition structure obtains $\mu_{\mathcal{D}}$ and $\sigma_{\mathcal{D}}$.*

Proof: The result follows immediately from (4.25), (4.26), and the condition $0 < \mu_{\mathcal{D}} < \sigma_{\mathcal{D}}$. □

4.3.3 Moment Matching Algorithm

Based on the above results, the s-transition moment matching algorithm is given. Consider the following four cases, where the first three refer to the series s-transition model ($\mu_D \geq \sigma_D > 0$), and the fourth refers to the hyperexponential model ($\sigma_D > \mu_D > 0$):

Case 1: If $\mu_D = \sigma_D$, then only the single exponential transition is needed.

Case 2: If μ_D/σ_D is an integer, say x, and $x \neq 1$, then only the Erlangian subnet is needed (formally, $\lambda_1 = \infty$). Set n to x^2, and $\lambda_2 = x^2/\mu_D$.

Case 3: For all other $\mu_D > \sigma_D$, both the exponential transition and the Erlangian subnet are needed. Derivations of n, λ_1, and λ_2 given above produced

$$\text{the unique integer } n, \qquad (\mu_D/\sigma_D)^2 - 1 \leq n < (\mu_D/\sigma_D)^2 \tag{4.27}$$

$$\lambda_1 = 1/\mu_1, \quad \text{where} \quad \mu_1 = \frac{\mu_D \mp \sqrt{n(n+1)\sigma_D^2 - n\mu_D^2}}{n+1} \tag{4.28}$$

$$\lambda_2 = n/\mu_2, \quad \text{where} \quad \mu_2 = \frac{n\mu_D \pm \sqrt{n(n+1)\sigma_D^2 - n\mu_D^2}}{n+1} \tag{4.29}$$

Either the top or bottom signs of (4.28) and (4.29) are used.

Case 4: For all $\mu_D < \sigma_D$, the hyperexponential subnet is used. Derivations of r_1, r_2, and λ_h given above produced

$$r_1 = 2\mu_D^2/(\mu_D^2 + \sigma_D^2) \tag{4.30}$$

$$r_2 = 1 - r_1 \tag{4.31}$$

$$\lambda_h = 2\mu_D/(\mu_D^2 + \sigma_D^2) \tag{4.32}$$

As noted in Case 1, when $\mu_D = \sigma_D$, then the basic GSPN exponential transition can be used. This is the degenerate case of both the series and parallel structures. Inspecting (4.27–4.29), shows that the series structure degenerates to the solution

$n = 0$, $\lambda_1 = 1/\mu_D$, and λ_2 arbitrary, which is equivalent to a single exponential transition. Likewise, the results for the parallel structure, (4.30–4.32), also degenerate to a single exponential transition, $r_1 = 1$, $r_2 = 0$, and $\lambda_h = 1/\mu_D$.

4.4 S-transition Density Function

The s-transition density function is derived in this section. The results will be used in an entropy analysis assess the s-transition accuracy.

4.4.1 Series Structure

The following intermediate result is used in the derivation of the s-transition density function. For $a > 0$, it is known that

$$\begin{aligned} \int t^n e^{-at} dt &= -\frac{e^{-at}}{a}\left(\sum_{i=0}^{n} \frac{t^i n!}{a^{n-i} i!}\right) \\ &= -\frac{e^{-at} n!}{a^{n+1}} \sum_{i=0}^{n} \frac{(at)^i}{i!} \end{aligned} \tag{4.33}$$

Thus,

$$\begin{aligned} \int_0^s t^n e^{-at} dt &= -\frac{e^{-as} n!}{a^{n+1}} \sum_{i=0}^{n} \frac{(as)^i}{i!} + \frac{n!}{a^{n+1}} \\ &= \frac{n! e^{-as}}{a^{n+1}}\left(e^{as} - \sum_{i=0}^{n} \frac{(as)^i}{i!}\right) \\ &= \frac{n! e^{-as}}{a^{n+1}} \sum_{i=n+1}^{\infty} \frac{(as)^i}{i!} \end{aligned} \tag{4.34}$$

Since a series connection exists between the exponential and Erlangian random variables, the s-transition density function is given by convolution, (3.24):

$$f_S(s) = (EXP(\alpha) \star ERL(\lambda, n))(s)$$

which simplifies to.

$$\begin{aligned} f_S(s) &= \int_0^s \left(\alpha e^{-\alpha(s-t)}\right)\left(\frac{\lambda^n}{(n-1)!} t^{n-1} e^{-\lambda t}\right) dt \\ &= \frac{\alpha e^{-\alpha s} \lambda^n}{(n-1)!} \int_0^s t^{n-1} e^{-(\lambda-\alpha)t} dt \end{aligned} \tag{4.35}$$

Thus, from (4.34), with $a = \lambda - \alpha$, we have

$$\begin{aligned} f_S(s) &= \frac{\alpha e^{-\alpha s}\lambda^n}{(n-1)!} \cdot \frac{(n-1)!e^{-(\lambda-\alpha)s}}{(\lambda-\alpha)^n}\left(e^{(\lambda-\alpha)s} - \sum_{i=0}^{n-1}\frac{((\lambda-\alpha)s)^i}{i!}\right) \\ &= \frac{\alpha\lambda^n}{(\lambda-\alpha)^n}e^{-\alpha s}\left(1 - e^{-(\lambda-\alpha)s}\sum_{i=0}^{n-1}\frac{((\lambda-\alpha)s)^i}{i!}\right) \\ &= \frac{\alpha\lambda^n}{(\lambda-\alpha)^n}e^{-\alpha s}(1-B_n) \end{aligned} \tag{4.36}$$

where B_n is given by

$$B_n = e^{-(\lambda-\alpha)s}\sum_{i=0}^{n-1}\frac{((\lambda-\alpha)s)^i}{i!} \tag{4.37}$$

The remaining task is to show that $\lambda-\alpha$ is positive. This is demonstrated by revisiting the s-transition design equations, (4.27–4.29). Choosing the bottom signs in (4.28) and (4.29), we have,

$$\begin{aligned} \lambda &\stackrel{?}{>} \alpha \\ \frac{n+1}{\mu_D - \frac{1}{n}\sqrt{n(n+1)\sigma_D^2 - n\mu_D^2}} &\stackrel{?}{>} \frac{n+1}{\mu_D + \sqrt{n(n+1)\sigma_D^2 - n\mu_D^2}} \\ \mu_D + \sqrt{n(n+1)\sigma_D^2 - n\mu_D^2} &\stackrel{?}{>} \mu_D - \frac{1}{n}\sqrt{n(n+1)\sigma_D^2 - n\mu_D^2} \\ 1 &\stackrel{?}{>} -\frac{1}{n} \end{aligned}$$

which is assured by (4.27).

4.4.2 Parallel Structure

For this case, the s-transition density function is easier to derive. From (3.33) and (4.30–4.32), we have

$$\begin{aligned} f_S(s) &= r_1\lambda_h e^{-\lambda_h s} + (1-r_1)\delta(s) \\ \frac{2\mu_D^2}{\mu_D^2+\sigma_D^2}\lambda_h e^{-\lambda_h s} &+ \frac{\sigma_D^2-\mu_D^2}{\sigma_D^2+\mu_D^2}\delta(s) \end{aligned} \tag{4.38}$$

where the delta-function, $\delta(s)$, denotes a unit probability mass at $s = 0$.

4.5 Entropy Analysis

Some processes within a system are well understood and can be accurately described mathematically. However, in other cases, the modeling of stochastic processes is complicated by selection of the correct density function family along with the correct parameterization. Typically, a particular density function within a family is specified by one or two parameters that can be related to the first two moments, i.e., mean and variance. Although these moments may be determined from data obtained via observation of real processes or analysis of specifications, the question may remain as to which family of density functions to select. When analytical determination of the density function family cannot be made, we would want a density function that meets the known constraints and is least presumptive about any unmodeled information. The *Maximum Entropy Method* (MEM) produces a density function having this property, [Jayn57]. To apply the analysis to s-transitions, we focus on the following three constraints: a specified mean, a specified variance, and knowledge that the density is one-sided (i.e., positive domain).

4.5.1 Maximum Entropy Method

With constraints on the mean (μ), variance (σ^2), and positivity, the following optimization problem can be formulated for $f_X(x)$:

$$\text{maximize} \quad -\int_0^\infty f_X(x) \log f_X(x) dx$$

$$\text{such that} \quad \begin{cases} \int_0^\infty x f_X(x) dx &= \mu \\ \int_0^\infty (x-\mu)^2 f_X(x) dx &= \sigma^2 \end{cases}$$

From results of the MEM, [Papo84], we conclude that the density function is of the following family:

$$f_X(x) = \begin{cases} Ae^{\alpha x - \beta^2 x^2} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{4.39}$$

where A (the density normalization), and α and β (the density parameterization), are chosen to satisfy

$$\int_0^\infty f_X(x)dx = 1 \tag{4.40}$$

$$\int_0^\infty x f_X(x)dx = \mu \tag{4.41}$$

$$\int_0^\infty x^2 f_X(x)dx = \sigma^2 + \mu^2 \tag{4.42}$$

Since closed-form solutions for the density parameters have not been found, a gradient search will be used to compute them. This search is developed in Appendix A.

4.5.2 Degeneracy to an Exponential

The MEM concludes that if the density function is known to be one-sided and only constrained by a mean, then the maximum entropy density is from the exponential family:

$$f_X(x) = ae^{-ax} \qquad \text{for } x \geq 0 \tag{4.43}$$

where $\mu = 1/a$ and $\sigma^2 = 1/a^2$.

Comparing (4.39) and (4.43), the latter is a degeneracy for $\beta = 0$. The distraction of this degeneracy is avoided by assuming $\beta \neq 0$. This is justified, since we can test $\mu^2 \stackrel{?}{=} \sigma^2$, and given equality, we use (4.43) with $a = 1/\mu$.

4.5.3 Existence of a MEDF

Otherwise unconstrained, the MEDF with arbitrarily prescribed mean and variance is the Gaussian density. However, on a semi-infinite domain, (e.g., positivity constraint), a MEDF may not exist: an additional condition for the existence of a MEDF is $\mu \geq \sigma$, [Dows73]. Thus, our attention is further focused to the series structure of the s-transition model, where $\mu \geq \sigma$ is satisfied.

Note that there are strictly positive densities with means less than standard deviations—the hyperexponential density is an example. However, since the above result requires $\mu \geq \sigma$, and $\mu = \sigma$ is the degeneracy, we assume $\mu > \sigma$ in the remaining sections.

4.5.4 Line Search for the MEDF

Normally, a 3-D gradient search would be required to find the optimal values of A, α, and β. By showing that the three quantities are loosely related, Appendix A develops a 1-D gradient search, i.e., a line-search, for the MEDF parameters. The optimal value of β is found by the search, and then α and A are computed from non-iterative relationships. By reducing the dimension of the search, the simplicity, numerical accuracy, and convergence rate are improved. In [Kapu90], an alternative method to find the parameters of the MEDF is given based on a tabulation of Bell's Function.

4.6 Example

4.6.1 System Description and Petri Net Model

To illustrate the transition models and the effect they have on performance quantities, a three machine-two buffer (3M2B) transfer line is considered. The first two machines each have mean production times of 25 seconds, and the third machine has mean production time of 25 seconds, with a standard deviation of 6, [Seid89]. The two buffers can each hold one part.

Figure 4.2 depicts the GSPN model of this system, where m_3 will be modeled in turn by an exponential transition, an s-transition, and a maximum entropy transition. In the third case, the model is no longer in the class of GSPNs, and must be simulated.

To model m_3 with an exponential transition, the density parameter is chosen as

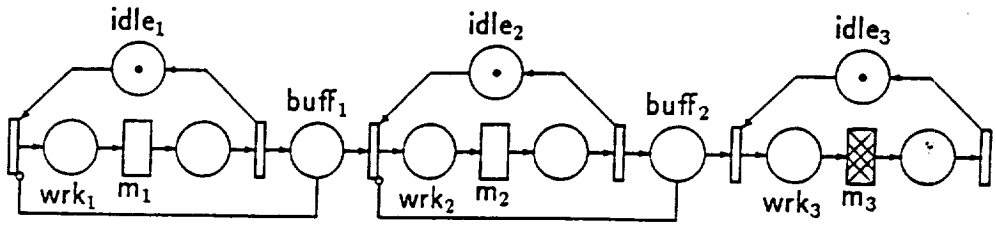


Figure 4.2: Three Machine-Two Buffer (3M2B)

$1/25 = 0.04$; hence, the transition mean matches the specification, [AlJa88b].

Applying the s-transition design equations, (4.27–4.29), to model the density function results in the parameters: $n = 17$; $\lambda_1 \doteq 0.40202$; and $\lambda_2 \doteq 0.75513$. Both the mean and variance of this transition match the specification. The line search of Appendix A is used to compute the parameters for the MEDF model of m_3. With a stopping threshold of 10^{-6}, the search converges in 14 iterations with the parameters: $\beta \doteq 0.11783$; $\alpha \doteq 0.69424$; and $A \doteq 1.1323 \cdot 10^{-5}$. Note, the small size of A is a result of the formulation of the density function. Rewriting (4.39) gives, for $x \geq 0$,

$$f_X(x) \;=\; \left(Ae^{\alpha^2/4\beta^2}\right) e^{-(\beta x - \alpha/2\beta)^2} \tag{4.44}$$

which, for our example yields,

$$f_X(x) \;\doteq\; 0.066529 \cdot e^{-(0.11783x - 2.9459)^2} \tag{4.45}$$

The density functions for the exponential, s-transition, and maximum entropy cases are plotted in Figure 4.3.

4.6.2 Entropy Comparison

While a simple analytic expression exists for the entropy of exponential density functions, its computation is meaningless: the chosen exponential cannot meet all the constraints met by the s-transition and MEDF. However, a direct comparison between the entropies of the s-transition and MEDF produces useful data on the

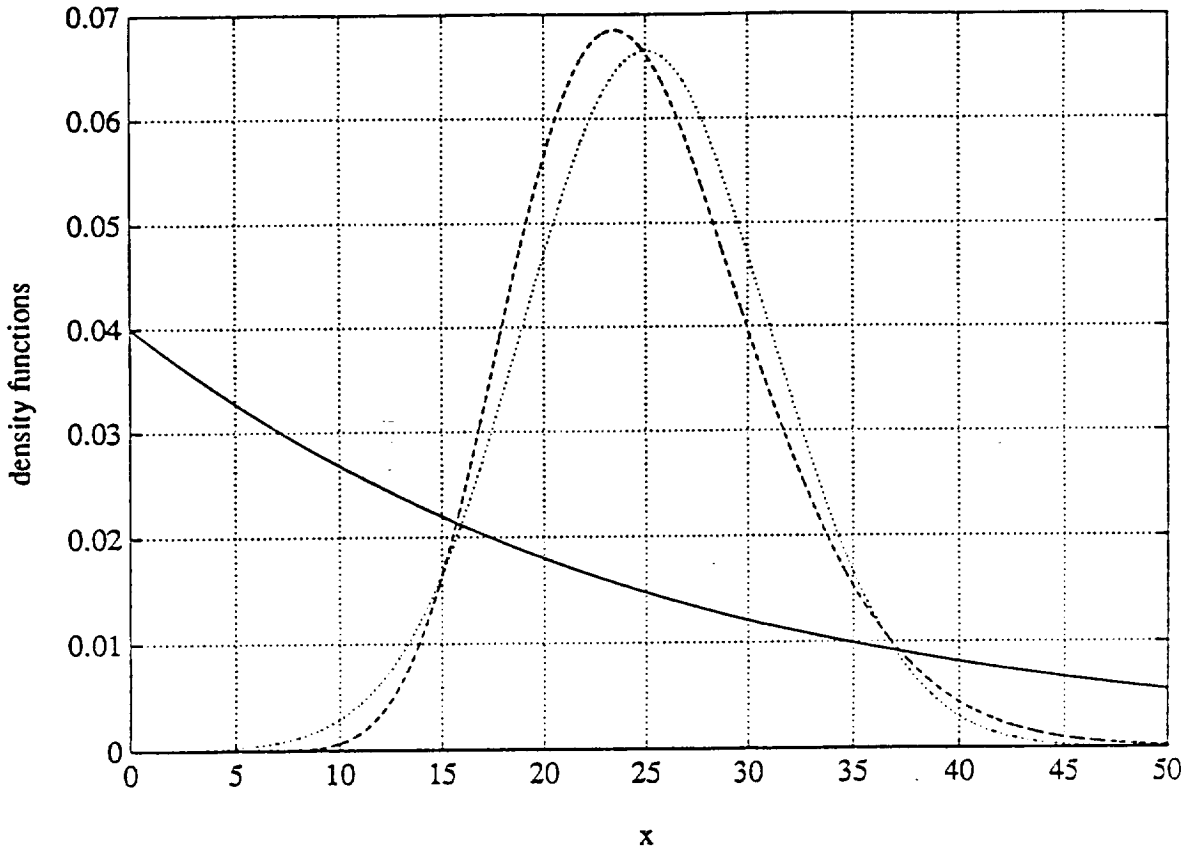


Figure 4.3: Density Function Comparison (solid: exponential; dashed: s-transition; dotted: MEDF)

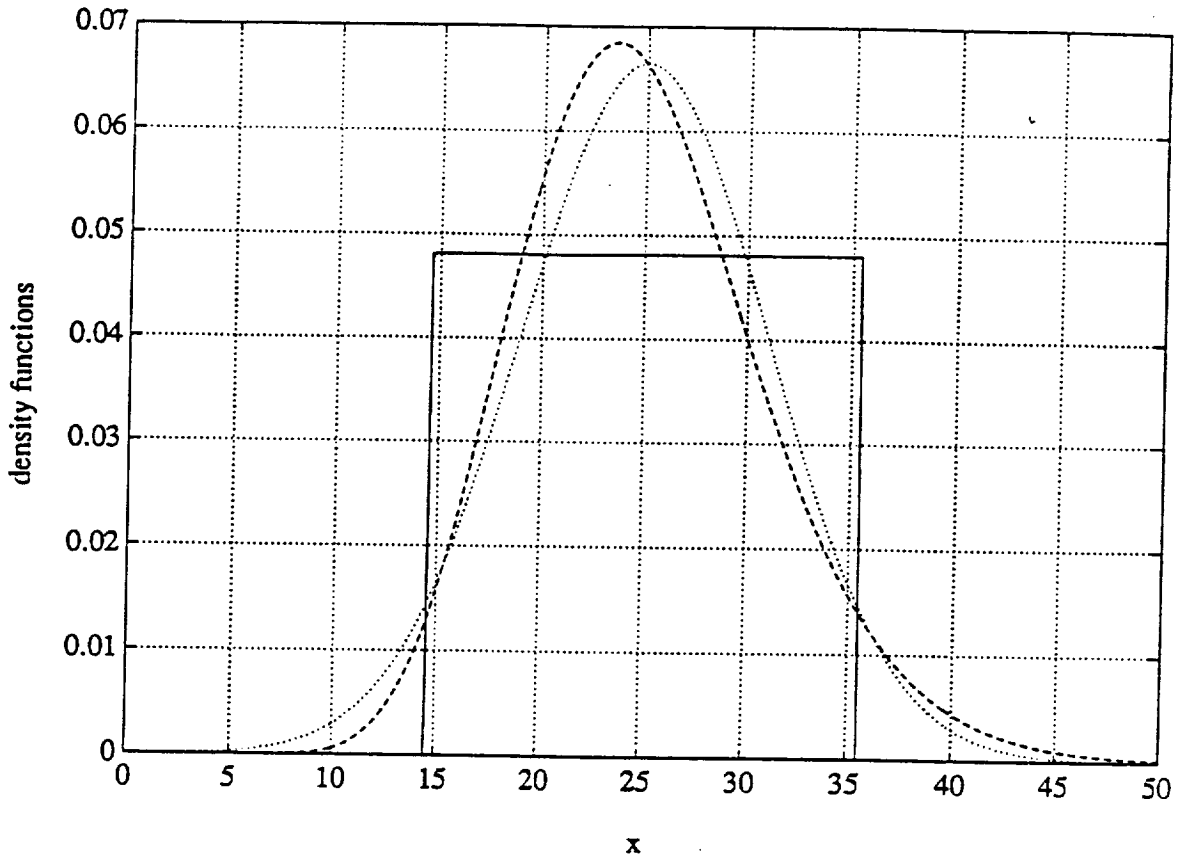


Figure 4.4: Density Function Comparison (solid: uniform; dashed: s-transition; dotted: MEDF)

optimality of the s-transition. To provide additional perspective on this comparison, the entropy of a uniform density function meeting all the design constraints is also computed. Figure 4.4 illustrates the uniform, s-transition, and maximum entropy density functions.

4.6.2.1 Maximum Entropy Density Function

For the MEDF, the entropy can be computed analytically:

$$H_{max} = -\int_0^\infty Ae^{\alpha x-\beta^2 x^2}(\ln A + \alpha x - \beta^2 x^2)dx$$

$$= -\left(\ln A + \alpha\mu - \beta^2(\sigma^2 + \mu^2)\right)$$

which is further simplified by (A.16) to

$$H_{max} = -\left(\ln A + \beta^2(\sigma^2 + \mu^2) - 1\right)$$

For our case $H_{max} \doteq 3.2107$, which represents the theoretical maximum entropy among all density functions satisfying the design constraints.

4.6.2.2 Uniform Density Function

The family of uniform density functions, $UNIF(\mu, c)$, has two parameters that can be used to match a specified mean and variance. For $\mathcal{U} \sim UNIF(\mu, c)$, we have,

$$f_{\mathcal{U}}(u) = \begin{cases} \frac{1}{2c} & \mu - c \leq u \leq \mu + c \\ 0 & \text{otherwise} \end{cases} \tag{4.46}$$

The entropy of a uniform density function is given by

$$\begin{aligned} H_{\mathcal{U}} &= -\int_{\mu-c}^{\mu+c} \frac{1}{2c} \ln \frac{1}{2c} dx \\ &= -\ln \frac{1}{2c} \\ &= \ln 2c \end{aligned}$$

To satisfy the mean, variance, and positivity constraints of the example, we pick $\mu = 25$ and $c = \sigma\sqrt{3} = 6\sqrt{3}$, producing an entropy of $H_{\mathcal{U}} = \ln 2c \doteq 3.0342$.

4.6.2.3 S-transition Density Function

The entropy of an s-transition requires numerical computation. From (4.36) and (4.37), we have,

$$\begin{aligned} H_S &= -\int_0^\infty \left(\frac{\alpha\lambda^n}{(\lambda-\alpha)^n} e^{-\alpha s}(1 - B_n)\right)\left(\ln \frac{\alpha\lambda^n}{(\lambda-\alpha)^n} - \alpha s + \ln(1 - B_n)\right) ds \\ &= -\ln \frac{\alpha\lambda^n}{(\lambda-\alpha)^n} + \alpha\mu_S - \frac{\alpha\lambda^n}{(\lambda-\alpha)^n} \int_0^\infty (1 - B_n) \ln(1 - B_n) ds \end{aligned} \tag{4.47}$$

option	percentage full				average throughput	
	buffer 1		buffer 2		machine 3	
	value	rel error	value	rel error	value	rel error
exponential	43.589	13.14%	23.077	32.31%	0.022564	7.69%
s-transition	38.817	0.75%	17.513	0.41%	0.024473	0.12%
max entropy	38.527	n/a	17.442	n/a	0.024444	n/a

Table 4.1: Comparison of Performance Data

where the last term requires numerical evaluation. For the example case, $H_S \doteq 3.1890$.

The entropy of the s-transition has only one-eighth the relative error that the uniform density does. Thus, in a relative sense, the s-transition is close to the MEDF.

4.6.3 Performance Comparison

This section presents a comparison of performance analysis results using the three options for transition m_3. Since the exponential and s-transition options can be modeled within GSPNs, exact performance results are computed with SPNP, [Ciar88]. However, the MEDF cannot be modeled within a GSPN, necessitating simulation. Appendix B discusses how this case was simulated in Matlab, [Matl90], and illustrates the relationship between the MEDF and the truncated Gaussian density.

Table 4.1 compares the performance quantities of interest: the percentage time that the buffers are full, (i.e., probability that place buff_i, $i = 1, 2$, is marked), and the average throughput of machine 3, (the average throughputs of the three machines are equal). The average throughput of machine 3 is given by the ratio of the number of times transition m_3 is fired and total simulation time. By Little's Law, this is also equal to the product of the probability that wrk_3 is marked and the mean service rate of m_3. The table entries for the maximum entropy transition were obtained from the convergence of the simulation. Figure 4.5 illustrates typical histograms of the simulation values as the net evolved up to one million total transition firings.

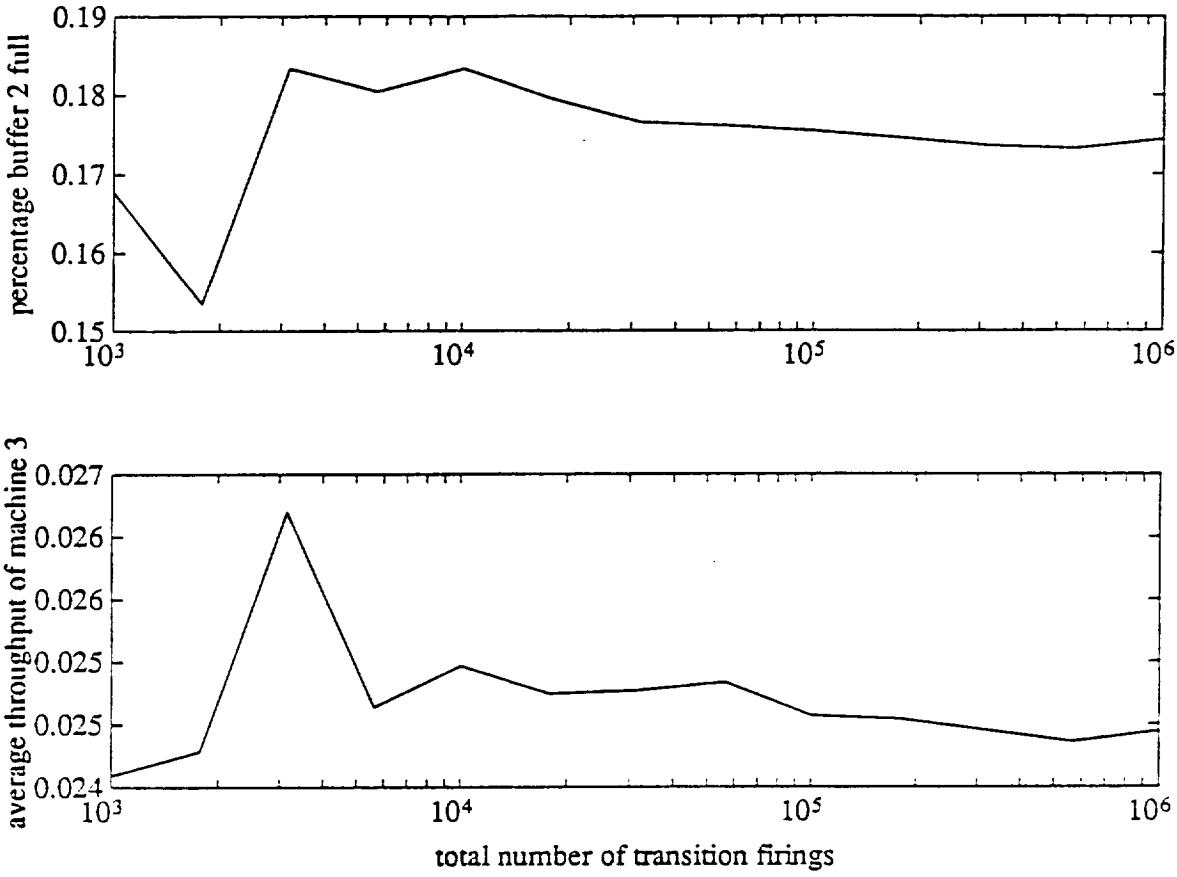


Figure 4.5: Convergence of Performance Data

The simulation used an "age memory" firing policy, [Mars85]. Under this policy, a timed transition samples an instance of its associated random variable upon initial enabling, and fires once this time expires. This is consistent with the physical interpretation of the example. The performance data confirm that the s-transition provides a close representation of the MEDF; the relative error of the performance quantities for the s-transition are between one and two orders of magnitude better than those for the exponential transition.

4.7 Summary

Generalized Stochastic Petri nets (GSPNs) restrict timed transitions to the exponential family. This restriction limits the accuracy of performance results for models of systems not meeting this constraint. The s-transition presented in this chapter provides a mechanism to incorporate non-exponential transitions within GSPNs. By remaining within the class of GSPNs, all the previously developed theory for GSPNs is available for analysis of the model. Other benefits of this method include the ability to service multiple instances of the non-exponential density function, and to analyze the closeness of the results to the theoretic optimal ones provided by the Maximum Entropy Method.

CHAPTER 5
STATE-SPACE SIZE ESTIMATION

5.1 Introduction

Petri nets (PNs) can compactly represent complex systems. The downside of this representational power is that when the underlying state-space needs to be formulated to solve a particular problem, there is no indication of the size of this task. The algorithms developed in this thesis to estimate the state-space size of PNs can be classified into two categories: top-down and bottom-up. This chapter describes the general problem of state-space size estimation and specific issues for the two solution categories. The estimators are developed in Chapters 6 and 7.

5.2 General Problem Characteristics

Since estimating the state-space size of PNs is an open research area, attention is given to the characterization of the problem. As described in Chapter 3, the state-space of a PN is a connected graph of nodes and directed links. The nodes correspond to possible states, and the links describe legal state changes. The size and connectivity of the state-space is a function of all the PN constructs: places, transitions, tokens, and input and output arcs.

The state-space size estimators consider only the number of states (i.e., nodes in the state-space graph), and not the interconnection of the states. It is the number of states that directly impacts the computation of performance and other state-space based analyses. Focusing on the number of nodes is justified by the potential applications of these results: an application for an estimate on the number of links is not apparent.

State-space size estimation, in itself, is not enough to be useful. Additional

theory is needed to determine the accuracy of the estimators. What does the estimate mean: is it *close* to the actual number of states; is the actual number of states always more than, less than, or potentially unconstrained by the estimate; is there any model where the estimator works exactly? The nature of estimation presumes that computed results will not always be exact, but without the answers to the above questions, it is difficult to interpret the estimate.

5.3 General Solution Characteristics

Several statements apply to both categories of estimators developed in the following chapters. These statements answer the above questions and permit the estimation results to be interpreted.

Different estimators are based on different information obtained from the model. If the model has a modular form, then the bottom-up method is best applied. Otherwise, a top-down method is applied that can take advantage of the available model data, such as, number of places, transitions, token bounds, etc.. The given model data applies not only to the PN model being analyzed, but to a whole class of PNs. The estimators determine the PN in this class that has the largest state-space size. and then compute this value.

It is clear, then, that we seek state-space size estimators that are always upper bounds on the number of states, and are consistent, i.e., there always exists a model possessing the appropriate properties that contains exactly the number of estimated states. These facts are proven for each of the top-down estimators, and are of immediate consequence from the construction of the bottom-up estimation method.

The fact that the estimators form a one-sided bound is an extremely useful property. One estimation algorithm can be iterated with different parameterization. and the minimum estimate is simply selected as the most accurate. A lower bound on the number of states is trivially given by one—the initial state. A more accurate

lower bound would require a completely different approach than described above. The class of PNs containing the given model data cannot be used to form a state-space size lower bound, because any dead model will have only a single state. Lower bound estimators are not considered in this thesis.

The fact that the estimators are consistent is also an important property. By assuring that there in no "padding" in the estimate, the estimator performance is maximal for the given class of models.

5.4 Approaches: Advantages, Disadvantages, Goals

The phrase top-down is used to describe approaches that process the model as an indivisible entity, and do not require a modular design. In contrast, bottom-up is used to describe approaches that process pieces (modules) of the model and then aggregate the obtained results. The following discusses the advantages and disadvantages of the two estimation categories, and also lists the goals for estimation accuracy and applicability.

5.4.1 Top-Down

Top-down state-space size estimators make little to no assumptions about how the PN was designed. There are no constraints that the net be constructed from a limited set of subnets and interconnections. Thus, the state-space size estimators consider the model as a whole, indivisible entity.

The advantage of such estimators is their applicability to the most general models. The analysis can be applied to already existing models, or to models being designed that do not happen to have a modular construction. The models can possess all of the representational flexibilities permitted in PNs. However, this general applicability comes at the price of less accurate estimation. Because the design of the PN is unconstrained, the class of PNs containing the same model data becomes

large, hence including PNs of more varied state-space size.

After developing some simple top-down estimators to provide insight into the methodology, an estimator for weighted conservative PNs is obtained. Since every PN is not weighted conservative, this may be viewed as a design constraint. However, it is at most a weak constraint, since it does not impose modularity, and is a very common property in models of real systems. The accuracy of the estimation is significantly improved by the weighted conservativeness of the model.

5.4.2 Bottom-Up

Bottom-up state-space size estimators are based on constraining the PN construction to a limited set of subnets and interconnections. In general, bottom-up analysis is best applied during the construction of the model. Given the limited representation resulting from the design constraints, it may be difficult, or even impossible, to decompose an already existing model. The design constraints permit better estimation than achieved from top-down methods; given a PN with truly modular construction, the bottom-up estimation will be exact. Any lack of accuracy results from modifications made to the model to achieve a modular design.

While there are representational limitations with all bottom-up methods, the state-space size estimation algorithm described achieves broad applicability by being flexible and extendable. The flexibility of the algorithm occurs from its ability to process arbitrary subnets—the representational limitation lies in the interconnections. However, the algorithm is extendable via the development of new interconnections. In addition to the improved estimation accuracy, the modular nature of bottom-up estimation permits the study of the effects of algorithmic model augmentation on state-space size. For example, non-exponential transition approximations improve performance results at the expense of increased numbers of places and transitions (and hence, increased state-space size). Bottom-up state-space size estimation provides a

mechanism to evaluate this trade-off.

5.5 Summary

Characteristics of the state-space size estimation problem and its solutions have been presented here. Two categories of estimators will be developed in the following chapters: top-down and bottom-up. Both estimation methods seek an upper bound on the state-space size. The top-down estimators offer the most general applicability, while the bottom-up estimators offer better accuracy.

CHAPTER 6
TOP-DOWN SIZE ESTIMATION

6.1 Introduction

The problem of estimating the state-space size of Petri nets (PNs) is pursued in two distinct manners: top-down and bottom-up. This chapter focuses on the top-down method. The state-space size estimators developed in this chapter make little to no assumptions about how the PN was designed. There are no constraints that the net be constructed from a limited set of subnets and interconnections. Thus, the state-space size estimators presented in this chapter consider the net as a whole, indivisible model. All the estimators developed are consistent upper bounds on the actual number of states.

Obtaining different data from the PN model leads to different estimators. In general, the better estimates require more information about the model. The symbol $\tilde{s}_i(\cdot)$ denotes the estimator, where the subscript is incremented as each estimator is developed. A functional notation is used to specify that information from the PN model on which a particular estimator depends.

The following sections develop the estimators for various model data. After identifying the appropriate model data, the estimator is formulated and analyzed for sensitivities to the number of places and tokens. Appendix C outlines the proofs for estimator consistency and upper boundedness; Appendices D, E and F contain the actual proofs.

6.2 Simple Estimators

This section presents four simple top-down estimators for the state-space size of a PN. The simplicity of the estimators allows the place and token sensitivities to be

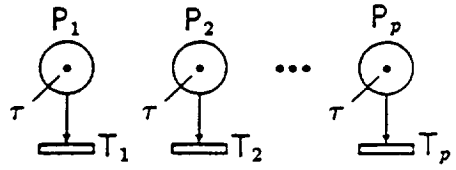


Figure 6.1: PN Model for Estimator 1

analyzed. While, as will be seen, the accuracy of these simple estimators is not very good, they provide insight into the problem and serve as the basis for the development of an accurate estimator for weighted conservative PNs. The consistency and bound proofs for these estimators are given in Appendices D and E.

6.2.1 Place Bounds: Estimator 1

The least amount of information about a PN that is needed to make an estimate on its state-space size is the number of places, p, and the maximum number of tokens that can occupy any place, τ. With this information, an upper bound on the number of states is given by

$$\tilde{s}_1(\tau, p) = (\tau + 1)^p \tag{6.1}$$

This results from the combinatoric argument that the number of tokens in each place is one of the $\tau + 1$ integers between 0 and τ, inclusively, and that the p places have independent allotments of tokens. Figure 6.1 illustrates a PN model that actually achieves $\tilde{s}_1(\tau, p)$ states.

Now consider how $\tilde{s}_1(\tau, p)$ is affected by increases in τ and p. The region in which an additional token results in a larger state-space size increase than an additional place is given by the following derivation:

$$\begin{aligned} \tilde{s}_1(\tau+1, p) - \tilde{s}_1(\tau, p) &> \tilde{s}_1(\tau, p+1) - \tilde{s}_1(\tau, p) \\ (\tau+2)^p &> (\tau+1)^{p+1} \\ p\ln(\tau+2) &> (p+1)\ln(\tau+1) \end{aligned}$$

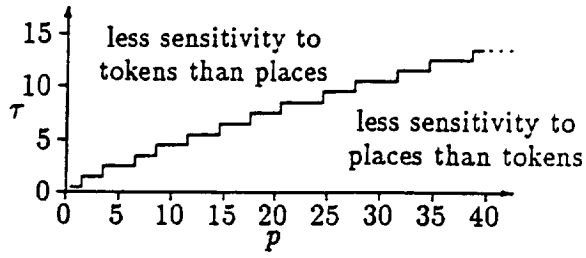


Figure 6.2: $\tilde{s}_1$ Sensitivity

$$\begin{aligned}
(p+1)\ln(\tau+2) - (p+1)\ln(\tau+1) &> \ln(\tau+2) \\
p+1 &> \frac{\ln(\tau+2)}{\ln\frac{\tau+2}{\tau+1}} \\
p &> \frac{\ln(\tau+2) - \ln\frac{\tau+2}{\tau+1}}{\ln\frac{\tau+2}{\tau+1}} \\
p &> \frac{\ln(\tau+1)}{\ln\frac{\tau+2}{\tau+1}} \qquad (6.2)
\end{aligned}$$

Thus, (6.2) indicates that PNs with many places have a greater state-space size sensitivity to increases in the number of tokens. Figure 6.2 plots (6.2) and depicts the regions of state-space size sensitivity.

6.2.2 Place Bounds: Estimator 2

The above estimator can be be improved by the knowledge that each place P_i holds at most τ_i tokens. Denoting the p-vector $(\tau_1, \tau_2, \ldots, \tau_p)^T$ by $\underline{\tau}$, the following estimate results:

$$\tilde{s}_2(\underline{\tau}, p) = \prod_{i=1}^{p}(\tau_i + 1) \qquad (6.3)$$

To evaluate the sensitivity of $\tilde{s}_2(\underline{\tau}, p)$, the meanings of "additional place" and "additional token" have be to clarified. Interpret an additional token as a new vector, $\underline{\tau}'$, defined as

$$\underline{\tau}' \stackrel{\text{def}}{=} \underline{\tau} + k \cdot \underline{e}_j \qquad (6.4)$$

whereby, the number of permissible tokens in place P_j has been increased by k. Furthermore, interpret an additional place as the $(p+1)$-vector,

$$\underline{\tau}'' \stackrel{\text{def}}{=} \left[\begin{array}{c} \underline{\tau} \\ \hline \rho \end{array} \right] \tag{6.5}$$

where ρ indicates the number of permissible tokens in the new place. Then, the region in which an additional token results in a larger state-space size increase than an additional place is given by the following derivation:

$$\begin{aligned} \tilde{s}_2(\underline{\tau}', p) - \tilde{s}_2(\underline{\tau}, p) &> \tilde{s}_2(\underline{\tau}'', p+1) - \tilde{s}_2(\underline{\tau}, p) \\ \left(\prod_{i=1}^{p}(\tau_i+1)\right)\frac{\tau_j+k+1}{\tau_j+1} &> \left(\prod_{i=1}^{p}(\tau_i+1)\right)(\rho+1) \\ \tau_j+k+1 &> (\rho+1)(\tau_j+1) \\ \tau_j+k+1 &> \rho+\rho\tau_j+\tau_j+1 \\ \frac{k}{\tau_j+1} &> \rho \end{aligned} \tag{6.6}$$

It is reasonable to assume that $\tau_j \geq 1$ (i.e., it is not useful to have a place that can never contain tokens), and $\rho \geq 1$ (for the same reason). Thus, if $k \leq \tau_j$, then (6.6) can never be satisfied, indicating that adding a place that can hold even a single token causes a greater state-space size increase than doubling the token holding capacity of any place.

The performances of estimators $\tilde{s}_1$ and $\tilde{s}_2$ can be compared: the following theorem proves that $\tilde{s}_2$ is a better estimator than $\tilde{s}_1$.

Theorem 6.1 $\tilde{s}_2(\underline{\tau}, p) \leq \tilde{s}_1(\tau, p)$

Proof: Equality in Theorem 6.1 holds if $\tau_1 = \tau_2 = \cdots = \tau_p = \tau$, where $\tilde{s}_2(\underline{\tau}, p) = \tilde{s}_1(\tau, p)$. However, in all other cases, $\tilde{s}_1(\tau, p) = \tilde{s}_1(\max(\underline{\tau}), p) > \tilde{s}_2(\underline{\tau}, p)$. $\square$

6.2.3 Net Bounds: Estimator 3

Estimation can be improved by extending the knowledge about place bounds to a bound on the entire PN. Thus, we assume it is known that a PN of p places can contain at most T tokens throughout its entirety. This, of course, implies that no place may contain more than T tokens.

An analogy can be made from the tokens and places of a PN to, respectively, the indistinguishable balls and distinct bins of the combinatorial definition of ${}^{n}X_{r}$. An estimate of the state-space size of a PN containing p places and at most T tokens is given by

$$\tilde{s}_3(T,p) \;=\; \sum_{i=0}^{T} {}^{p}X_i. \tag{6.7}$$

The summation accounts for the *at most* nature in the token limit, T. Since the events containing j tokens are disjoint from the events containing k tokens ($k \neq j$), no duplication results. Analyzing (6.7) directly to obtain sensitivities to p and T is difficult, however, it can be simplified.

Figure 6.3 illustrates an alternative indexing of Pascal's Triangle that will be more useful for the current problem (compare with Figure 3.1). As an immediate consequence of the change in indices (i.e., $i = n - r$; $p = r + 1$), we have ${}^{n}C_{r} = {}^{i+p-1}C_{p-1} = {}^{i+1}X_{p-1}$. Thus, from (3.44),

$${}^{n}C_{r} \;=\; {}^{p}X_i \tag{6.8}$$

and the recurrence relationship in rule 3 of Definition 3.7, (3.46), can be formulated as

$${}^{p}X_i \;=\; {}^{p-1}X_i + {}^{p}X_{i-1}. \tag{6.9}$$

Using (6.9), simplification of (6.7) is provided by the result of the following theorem.

Theorem 6.2 $\displaystyle\sum_{i=0}^{T} {}^{p}X_i \;=\; {}^{p+1}X_T \qquad \forall p \geq 1$

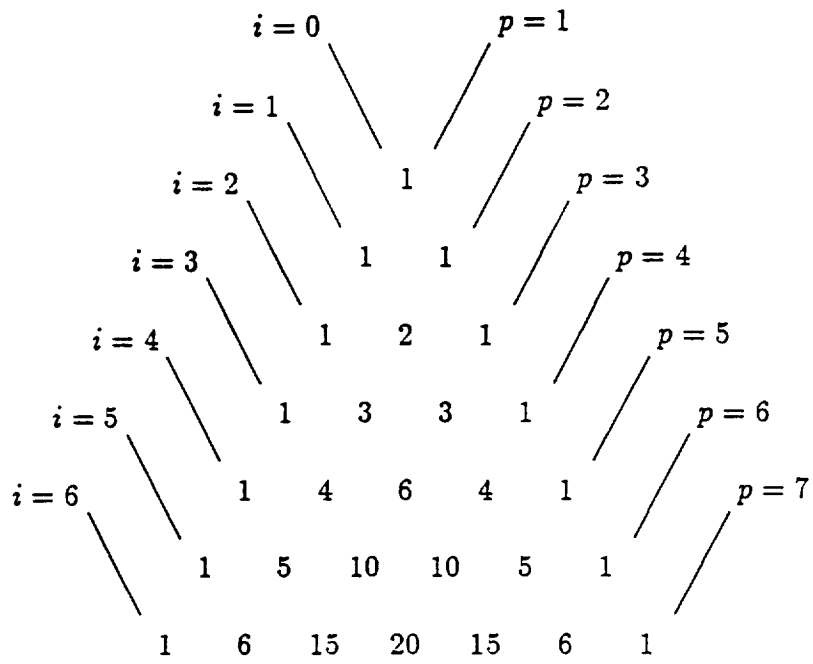


Figure 6.3: Pascal's Triangle (With Alternate Indices)

Proof: The proof is given by induction on T. Fix p. Notice that if $T = 0$, then the sum on the left hand side of Theorem 6.2 contains the single term, ${}^{p}X_0$. Since ${}^{\alpha}X_0 = 1$ for all values of α, Theorem 6.2 holds for $T = 0$. Now assume that Theorem 6.2 holds for $T = T_0 - 1$, i.e.,

$$\sum_{i=0}^{T_0-1} {}^{p}X_i = {}^{p+1}X_{T_0-1}. \tag{6.10}$$

Increasing T to T_0, the left hand side of Theorem 6.2 gives

$$\begin{aligned} \sum_{i=0}^{T_0} {}^{p}X_i &= \sum_{i=0}^{T_0-1} {}^{p}X_i + {}^{p}X_{T_0} \\ &= {}^{p+1}X_{T_0-1} + {}^{p}X_{T_0} \\ &= {}^{p+1}X_{T_0} \end{aligned}$$

where the last step follows from (6.9). □

With the identity in Theorem 6.2, (6.7) is simplified to

$$\tilde{s}_3(T,p) = \sum_{i=0}^{T} {}^{p}X_i = {}^{p+1}X_T = \frac{(T+p)!}{T!p!}. \tag{6.11}$$

This is an interesting result, which can be interpreted as follows. A PN containing p places and at most T tokens is equivalent to a PN containing $p+1$ places and exactly T tokens. The additional place in the larger model holds the "unused" tokens in the original model.

The sensitivity of $\tilde{s}_3(T,p)$ can now be computed. The region in which an additional token results in a larger state-space size increase than an additional place is given by the following derivation:

$$\begin{aligned}
\tilde{s}_3(T+1,p) - \tilde{s}_3(T,p) &> \tilde{s}_3(T,p+1) - \tilde{s}_3(T,p) \\
\frac{(T+1+p)!}{(T+1)!p!} &> \frac{(T+p+1)!}{T!(p+1)!} \\
T!(p+1)! &> (T+1)!p! \\
(p+1)p!T! &> (T+1)T!p! \\
p &> T
\end{aligned} \tag{6.12}$$

6.2.4 Net Bounds: Estimator 4

The above estimator can be improved with additional knowledge that there are exactly T tokens (as opposed to at most T tokens) in the PN. This would mean that the PN is strictly conservative, i.e., the number of tokens in every marking is constant.

Since strict conservatism is not a property of every PN, the applicability of this estimator is restricted. However, every PN can be converted into an equivalent model (i.e., having the same reachability graph, hence the same state-space size) that is strictly conservative: one mechanism for doing this is given by the interpretation of (6.11). Additionally, the development of this estimator maintains the lines of reasoning from the previous estimators to the forthcoming estimation of weighted conservative PNs.

From the derivation of $\tilde{s}_3(T,p)$, a PN with p places and exactly T tokens, has

a state-space size estimate of

$$\tilde{s}_4(T,p) = {}^{p}X_T \tag{6.13}$$

The sensitivity of $\tilde{s}_4(T,p)$ is now computed. The region in which an additional token results in a larger state-space size increase than an additional place is given by the following derivation:

$$\begin{aligned} \tilde{s}_4(T+1,p) - \tilde{s}_4(T,p) &> \tilde{s}_4(T,p+1) - \tilde{s}_4(T,p) \\ \frac{(T+p)!}{(T+1)!(p-1)!} &> \frac{(T+p)!}{T!p!} \\ \frac{p!}{(p-1)!} &> \frac{(T+1)!}{T!} \\ p &> T+1 \end{aligned} \tag{6.14}$$

Interestingly, (6.14) is very similar to (6.12), even though the estimation is based on exactly T tokens versus at most T tokens.

We show that $\tilde{s}_4$ is a better estimator than $\tilde{s}_3$.

Theorem 6.3 $\tilde{s}_4(T,p) \leq \tilde{s}_3(T,p)$

Proof: From (6.13), $\tilde{s}_4$ is but one of the positive terms added to obtain $\tilde{s}_3$, (6.7). Thus, the theorem clearly holds. □

6.3 Comparing Net and Place Bounds

The first two estimators, $\tilde{s}_1$ and $\tilde{s}_2$, were developed based on place bounds. On the other hand, $\tilde{s}_3$ and $\tilde{s}_4$ were developed based on net bounds. Demonstrating relationships between the two sets of estimators requires consideration be given to how place bounds and net bounds are equated.

Maintaining the above notation, let $\underline{r}$ be a p-vector describing the token bounds in each of the places, and let T denote the token bound for the entire model. Then,

the following is a logical restriction:

$$\max_i \tau_i \;\leq\; T \;\leq\; \sum_{j=1}^{p} \tau_j \tag{6.15}$$

The lefthand inequality states that T, the bound for the net, should be at least as large as the maximum bound permitted in any of the places. If this were not the case, then some place, say P_k, would have $\tau_k > T$, and could not reach its local bound. The righthand inequality states that T should be realizable, i.e., if all the places are filled to their bounds, the bound for the net should be met or exceeded. For the case that a uniform place bound is known, (6.15) becomes

$$\tau \;\leq\; T \;\leq\; \sum_{j=1}^{p} \tau \;=\; p\tau \tag{6.16}$$

From (6.16) and Lemma 3.3, the following inequalities can be expressed for τ:

$$\lceil T/p \rceil \;\leq\; \tau \;\leq\; T \tag{6.17}$$

Thus, (6.15–6.17) provide a method to form place bounds from net bounds, and vice-versa. Specifically, given place bounds, (6.15) or (6.16) can be used to determine the range on the net bound; likewise, given a net bound, (6.17) can be used to determine the range on the place bounds.

Consider, for example, a PN that has $p = 5$ places. Figure 6.4 plots the results of $\tilde{s}_1$, $\tilde{s}_3$, and $\tilde{s}_4$ for various number of tokens, T. The least conservative net bound is used for $\tilde{s}_1$, i.e., $\tau = \lceil T/p \rceil$. As can been seen, the performance of $\tilde{s}_1$ is not consistently better or worse than $\tilde{s}_3$ or $\tilde{s}_4$.

The above discussion notwithstanding, the following proof shows that $\tilde{s}_3$ is a better estimator than $\tilde{s}_1$ when the most conservative net bound is used.

Theorem 6.4 *Given p and T, and using the bound conversion $\tau = T$, (6.17), then*

$$\tilde{s}_3(T,p) \;\leq\; \tilde{s}_1(\tau,p)$$

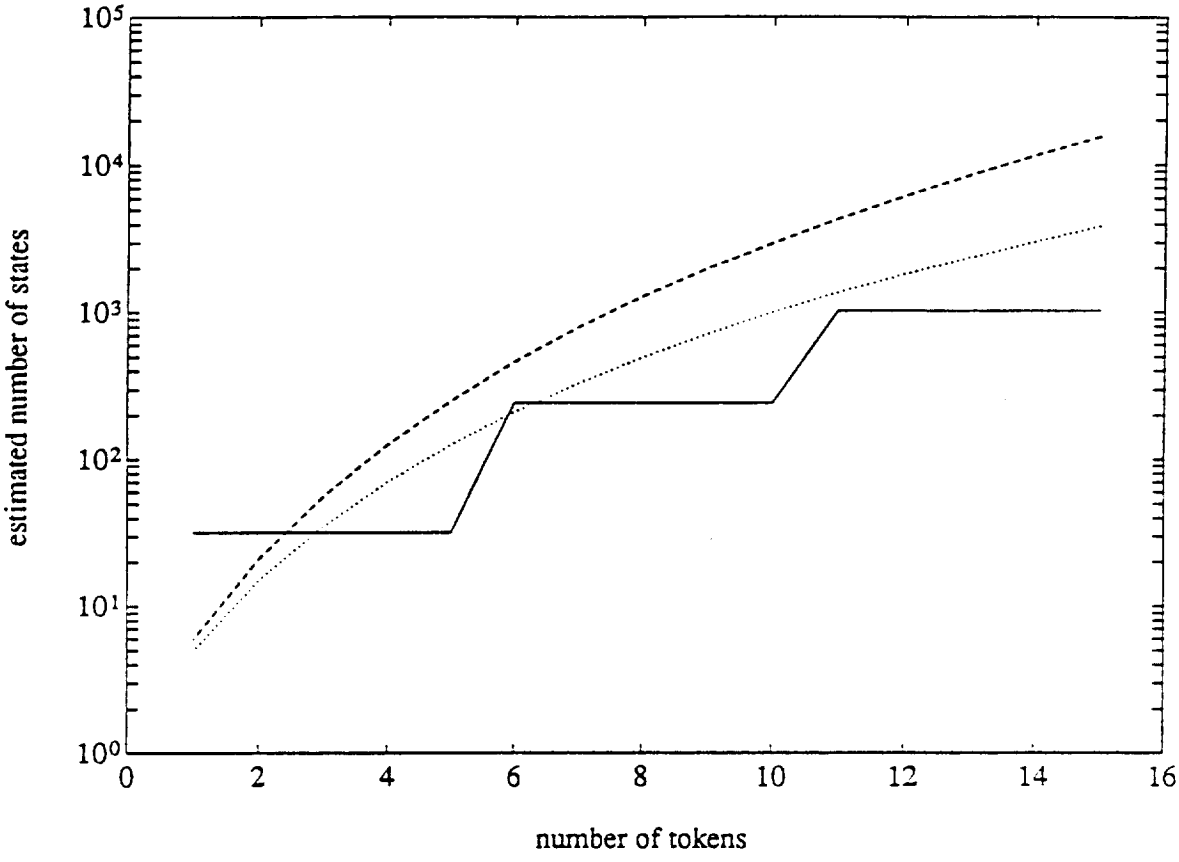


Figure 6.4: Comparison of $\tilde{s}_1$ (solid), $\tilde{s}_3$ (dashed), $\tilde{s}_4$ (dotted) for $p = 5$

Proof: Since $\tau = T$, we have

$$\tilde{s}_1(\tau, p) = (T+1)^p \tag{6.18}$$

The derivation of $\tilde{s}_3(T,p)$ was based on a summation over T. From (6.11) and (3.44), a summation over p is formulated:

$$\begin{aligned} \tilde{s}_3(T,p) &= {}^{p+1}X_T \\ &= {}^{T+1}X_p \\ &= \sum_{i=0}^{p} {}^{T}X_i \\ &= \sum_{i=0}^{p} \frac{(T+i-1)!}{i!(T-1)!} \end{aligned} \tag{6.19}$$

where the last step is a result of Lemma 3.1. Using (6.18) and (6.19), the theorem is proven by induction on p.

For $p = 1$, we have $\tilde{s}_1(\tau,p) = \tilde{s}_3(T,p) = T+1$. Thus, we assume the theorem holds for $p = p_0$, i.e.,

$$\tilde{s}_3(T,p_0) \leq \tilde{s}_1(\tau,p_0) \tag{6.20}$$

and need to show that it is valid for the next value of p, i.e.,

$$\tilde{s}_3(T,p_0+1) \leq \tilde{s}_1(\tau,p_0+1) \tag{6.21}$$

The validity of (6.21) is proven by contradiction, i.e., we show

$$\tilde{s}_3(T,p_0+1) > \tilde{s}_1(\tau,p_0+1) \tag{6.22}$$

does not hold. From (6.18) and (6.19), we simplify (6.22) to

$$\tilde{s}_3(T,p_0) + \frac{(T+p_0)!}{(p_0+1)!(T-1)!} > \tilde{s}_1(\tau,p_0)\cdot(T+1)$$

and upon substituting (6.20),

$$\begin{aligned}
\tilde{s}_1(\tau, p_0) + \frac{(T+p_0)!}{(p_0+1)!(T-1)!} &> \tilde{s}_1(\tau, p_0) \cdot (T+1) \\
\frac{(T+p_0)!}{(p_0+1)!(T-1)!} &> \tilde{s}_1(\tau, p_0) \cdot T \\
\frac{(T+p_0)!}{(p_0+1)!T!} &> \tilde{s}_1(\tau, p_0) \\
\frac{(T+p_0)!}{(p_0+1)!T!} &> (T+1)^{p_0} \qquad (6.23)
\end{aligned}$$

Identifying, respectively, T and p_0 in (6.23) with x and y in Lemma 3.2, we conclude that (6.23) never holds. This contradiction proves (6.21), which proves the theorem. □

6.4 Combining Net and Place Bounds

The above comparison of estimators based on net bounds and place bounds demonstrated that each have a region of effectiveness. Given this, estimators that combine net and place bounds should provide consistently better results. Estimation of weighted conservative PNs, which results in a combined estimator, is presented below.

6.5 Estimation for Weighted Conservative Petri Nets: Estimator 5

Our goal is develop a state-space size estimation algorithm for weighted conservative PNs. The information obtained from the PN will include a net bound (in the weighted sense) and place bounds. During the development, several points of discussion arise, resulting in the following outline for this section. Ambiguity inherent to a weight vector solution is discussed first. Then, the PN properties that affect the weight vector solution are characterized. Lastly, an algorithm for estimating a conservative PN's state-space size is given. In the following, "conservative PN" is used to mean "weighted conservative PN."

6.5.1 Weight Vector Ambiguity

From Definition 3.13, a (weighted) conservative PN, $\mathcal{A}$, with initial marking $\underline{\mu}^0$, satisfies

$$\underline{\mu}_1,\ \underline{\mu}_2 \in RS(\mathcal{A}, \underline{\mu}^0) \ \Rightarrow\ \underline{w} \circ \underline{\mu}_1 \ = \ \underline{w} \circ \underline{\mu}_2. \tag{6.24}$$

The choice of the weight vector, $\underline{w}$, is, in general, not unique. There are two reasons for the ambiguity:

Type 1: If $\underline{w}$ satisfies (6.24), then so does $\alpha\underline{w}$, where $\alpha \in \mathbb{R}_+$.

Type 2: The markings in the reachability set *may* permit an ambiguity in $\underline{w}$ due to the summing operation of the dot product. Consider, for example, that $RS(\mathcal{A}, \underline{\mu}^0)$ consists of two markings, $[1\ 0\ 0]^T$ and $[0\ 1\ 1]^T$. Possible weight vectors are $[1\ \frac{1}{2}\ \frac{1}{2}]^T$ and $[1\ \frac{1}{4}\ \frac{3}{4}]^T$.

The first type of ambiguity is removed by adding a constraint to the weight vector computation so as to remove arbitrary scaling. Constraints that force the norm of the weight vector to some constant are disadvantageous because they will tend to make the vector irrational. As shown below, it is beneficial to the application of using a weight vector for estimating a conservative PN's state-space size to have a rational vector. Using Definition 3.21, (6.24) is constrained by

$$\underline{w} \circ \underline{\mu}^0 \ = \ 1 \tag{6.25}$$

For the example PN in Figure 3.3, the constraint (6.25) yields

$$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 1 \\ 1 & 0 & -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \ = \ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

C-2

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 1,$$

which has positive solutions of the form

$$\underline{w} = \begin{bmatrix} 1 \\ 1 \\ \beta \\ 1-\beta \end{bmatrix} \qquad \beta \in (0, 1), \tag{6.26}$$

where the remaining ambiguity is of the second type.

The second type of ambiguity can be removed by choosing $\underline{w}$ to optimize some criterion. One such criterion is considered after the following discussion of PN structures that affect the selection of a conservative weight vector.

6.5.2 Structures Affecting the Weight Vector

6.5.2.1 Boundedness

Given the definition for a weight vector (Definition 3.21), PN boundedness is a necessary condition for structural and behavioral conservatism. This fact is obtained directly from the requirement of non-zero components in the weight vector.

6.5.2.2 Liveness

PN liveness distinguishes structural and behavioral conservatism. A transition that is dead can never be fired, thereby essentially removing parts of the model that may be nonconservative. The PN in Figure 3.4, although not structurally conservative, was behaviorally conservative. This resulted because T_1 was dead, thereby removing the nonconservative loop of the PN from consideration.

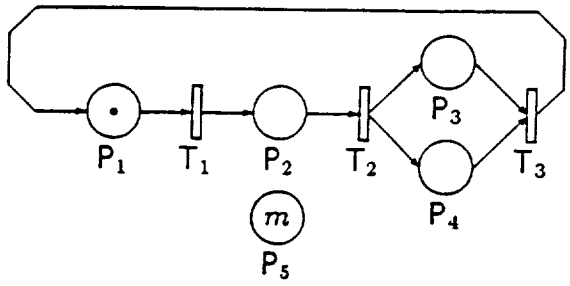


Figure 6.5: PN With an Isolated Place

6.5.2.3 Isolated Places and Self-loops

Places in PNs are useful for holding a variable number of tokens during the model's evolution. Two constructs, isolated places and self-loops, result in places that do not readily change their marking. Isolated places are considered first, and then it is shown that self-loops are an extension.

Consider the PN in Figure 6.5, where P_5 has initial marking m. Place P_5 is called isolated because it is neither the input to or output of any transition. The incidence matrix for this case will have all zeros corresponding to the row for the isolated place. Thus, the weight corresponding to the isolated place is completely unconstrained. For Figure 6.5, (3.68) produces

$$\begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 1 & 0 & -1 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

where w_5 does not appear in any of the weight vector component equations. Thus, w_5 can be chosen as any positive number, and then the term $w_5 m$ will appear in the weight vector normalization, (6.25).

The arbitrary value that is chosen for the weight vector component corresponding to the isolated place, and the subsequent normalization, result in a complicated

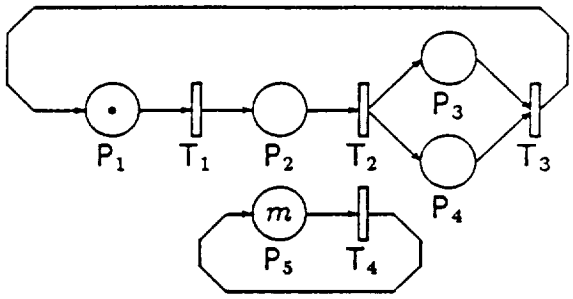


Figure 6.6: PN With an Isolated Self-loop

influence in the choice of the weight vector. Since isolated places and their tokens have no bearing on the model evolution, it can be assumed, without loss of generality, that a PN does not contain these places.

Self-loops can be considered as a generalization of isolated places. Figure 6.6 contains a self-loop on P_5—it is called an isolated self-loop since the place involved is not connected to any other transitions. Obviously, a PN with a self-loop can only be conservative if the weights for the input and output arcs are equal. As a consequence, the incidence matrix will have the structure similar to an isolated place. This is because the incidence matrix only considers the difference between the input and output arcs, thus resulting in an all zero row. The transition is also isolated, resulting in all zeros in the corresponding column, which is equivalent to a trivial equation in (3.68). Thus, the place and transition of an isolated self-loop can be removed from a PN. Figure 6.7 illustrates a non-isolated self-loop. In this structure, the place is functional for the remainder of the PN, and thus, only the transition can be removed.

6.5.2.4 Dead Tokens

The PN in Figure 3.3 is structurally conservative. Therefore, this PN is conservative for any initial marking, including $\underline{\mu}^0 = [1\ 0\ 1\ 0]^T$, which is illustrated in

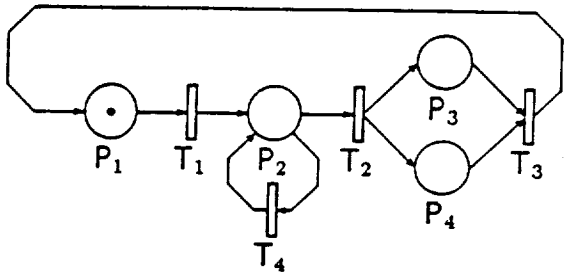


Figure 6.7: PN With a Self-loop

Figure 6.8. Note, however, that every marking in the reachability set for this PN contains at least one token in P_3. This is due to the fact that the initial token in P_3 is *dead* in the sense that the transition firing sequences, σ, for the PN in Figure 6.8 are not different from those for the PN having P_3 initially empty, (i.e., Figure 3.3). However, since the initial marking is considered in the constraint on $\underline{w}$, this dead token will affect the weight vector. Specifically, with this additional token, we have

$$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 1 \\ 1 & 0 & -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = 1$$

which has positive solutions of the form

$$\underline{w} = \begin{bmatrix} 1-\beta \\ 1-\beta \\ \beta \\ 1-2\beta \end{bmatrix} \qquad \beta \in (0,\ 0.5)$$

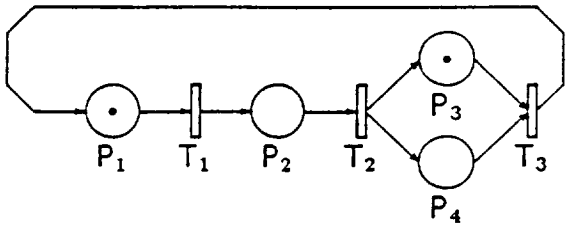


Figure 6.8: PN With a Dead Token

6.5.2.5 Dead Places

Given a conservative PN, it is possible that one or more places may never become marked, i.e., they are dead. This implies that the transitions that output to these places are dead.

Consider the PN in Figure 6.9, which has weight vector solutions of the form

$$\begin{bmatrix} -1 & 2 & 0 & 0 \\ 0 & -1 & 1 & 1 \\ 0 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 2\alpha \\ \alpha \\ \beta \\ \alpha - \beta \end{bmatrix}, \quad \begin{matrix} \alpha \in (0, \infty) \\ \beta \in (0, \alpha) \end{matrix}$$

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = 1 \Rightarrow \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ \beta \\ 1 - \beta \end{bmatrix}, \quad \beta \in (0, 1)$$

Notice that if P_1 had a non-zero number of tokens as part of a marking $\underline{\mu}$, then $\underline{w} \circ \underline{\mu} = 1$ cannot be satisfied. Thus, from this result, it can be determined that P_1 can never be marked during the evolution of the model, and is therefore a dead place. Like isolated places and self-loops, dead places do not influence the evolution of the model.

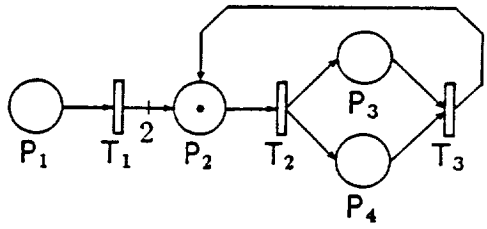


Figure 6.9: PN With a Dead Place

6.5.3 Estimation Algorithm

Assume that we are given a PN, $\mathcal{A}$, that is structurally weighted conservative, and $\underline{\mu}^0$ is not the empty marking. Then it is possible to find a weight vector, $\underline{w}$, for $\mathcal{A}$ that satisfies

$$\begin{aligned} \underline{C}^T \underline{w} &= \underline{0}, \qquad \underline{w} > \underline{0} \\ \underline{w} \circ \underline{\mu}^0 &= 1 \end{aligned}$$

The above set of equations, in general, is still under defined for $\underline{w}$. Within the flexibility for $\underline{w}$, a choice can be made that does not aggravate algorithms for estimating a PN's state-space size. The algorithm presented here requires only the PN's weight vector and computes an upper bound for the number of states in its reachability set.

For a conservative PN, $\mathcal{A} = (P, T, I, O, \underline{\mu}^0)$, this algorithm assumes the weight vector, $\underline{w}$, has been computed, that its components, w_1, w_2, ..., w_p, are rational, and the equation

$$\underline{w} \circ \underline{\mu}^0 = 1 \tag{6.27}$$

is satisfied. The size of $\underline{w}$ implies the number of places in $\mathcal{A}$. The set M is defined as

$$M \stackrel{\text{def}}{=} \{\underline{x} \mid \underline{w} \circ \underline{x} = 1\}, \tag{6.28}$$

where $\underline{x}$ is a non-negative integer p-vector. Thus, an upper bound on the number of

states in a conservative PN with weight vector $\underline{w}$ is given by the number of vectors in M. The following is an algorithm to count these state vectors.

Given $\underline{w}$ satisfying the above assumptions, and assuming $\underline{w}$ contains no components larger than 1 (by removing dead places from consideration, which can be done without loss of generality), a vector $\underline{\omega}$ is formed in the following manner: $\underline{w}$ is scaled by a common multiple of its components' denominators, λ, and the result is then sorted from highest to lowest. Repeated elements are grouped, and $\underline{\omega}$ is formed, consisting of unique integers, each associated with a repetition count. Properties of $\underline{\omega}$ include

- $\underline{\omega}$ is a vector of positive integers, each with an associated repetition count.
- $\omega_1 > \omega_2 > \cdots > \omega_p$.
- $\lambda \geq \omega_1$.

Note that $\underline{\omega}$ would not necessarily be a vector of integers if $\underline{w}$ was not rational.

A non-negative integer vector $\underline{x}$ satisfying $\underline{x} \circ \underline{\omega} = \lambda$ indicates that $\underline{x}'$ (which maps to $\underline{x}$ via the above sort) is in M. Because of the properties of $\underline{\omega}$, counting the number of vectors can be done with the following algorithm:

[A.] Call [B.] with parameters $\underline{\omega}$ and λ. The result is the number of solutions.

[B.] Routine with arguments $\underline{x}$ and s.

[1.] $r := 0$

[2.] $\eta :=$ number of unique elements in $\underline{x}$.

[3.] $v :=$ first value in $\underline{x}$.

[4.] $c :=$ repetition count for the first value in $\underline{x}$.

[5.] $\phi := s/c$.

[6.] If $\eta = 1$ then

[6.1.] If ϕ is an integer then $r := {}^{c}X_{\phi}$.

[6.2.] If ϕ is not an integer then $r := 0$.

[7.] If $\eta > 1$ then

[7.1.] Loop for $i = 0, 1, \ldots, \lfloor \phi \rfloor$

[7.1.1.] $r' :=$ the result of [2.] called with parameters $\underline{x}_{2\ldots\eta}$ and $s - i\eta$.

[7.1.2.] $r := r + {}^{c}X_{i}r'$.

6.5.3.1 Algorithmic Complexity Analysis

The performance of the above algorithm is obtained by analyzing its complexity. This analysis is not easily applied to a recursive algorithm that depends on more than one input. However, since the above algorithm involves only tail-recursion, i.e., no processing occurs after reaching the recursive call to itself, the algorithm can be converted to an iterative form, [Inde87].

An iterative version of the algorithm for $\tilde{s}_5$, with input parameters $\underline{\omega}$ and λ is given below. The algorithm has polynomial complexity.

[1.] $\eta :=$ number of unique elements in $\underline{\omega}$.

[2.] $v :=$ first value in $\underline{\omega}$.

[3.] $c :=$ repetition count for the first value in $\underline{\omega}$.

[4.] $\phi := \lfloor s/c \rfloor$.

[5.] $\underline{\underline{\rho}} :=$ a 2-row matrix with a variable number of columns.

[6.] Compute the first ϕ columns of $\underline{\underline{\rho}}$ by $\rho_{1,j} = (j-1)v + 1$ and $\rho_{2,j} = {}^{c}X_{j}$.

[7.] Loop for $m = 2, 3, \ldots, \eta$

[7.1.] $v := m^{\text{th}}$ value in $\underline{\omega}$.

[7.2.] $c :=$ repetition count for the m^{th} value in $\underline{\omega}$.

[7.3.] $\underline{\underline{l}} :=$ a matrix with 2 rows and $\lfloor \lambda / v \rfloor$ columns.

[7.4.] $l_{1,j} := jv$ and $l_{2,j} := {}^{c}X_j$.

[7.5.] Loop for $\underline{n} =$ columns of $\underline{\underline{\rho}}$

[7.5.1.] $\underline{\underline{ll}} :=$ a 2-row matrix with a variable number of columns.

[7.5.2.] Loop for $p = 1, 2, \ldots, \lfloor (\lambda - n_1)/v \rfloor$

[7.5.2.1.] Append $\underline{\underline{ll}}$ with a column $\begin{bmatrix} n_1 + pv \\ n_2 l_{2,p} \end{bmatrix}$.

[7.5.3.] Append $\underline{\underline{ll}}$ to $\underline{\underline{l}}$. For repetitions in the first row element, sum the corresponding second row elements.

[7.6.] Append $\underline{\underline{l}}$ to $\underline{\underline{\rho}}$. For repetitions in the first row element, sum the corresponding second row elements.

[8.] The solution is given by $\rho_{2,\phi}$.

The iterative algorithm was useful for performing the complexity analysis, but the recursive version is easier to implement (and follow). The inefficiencies associated with recursive algorithms typically occur because no global data is maintained on the sub-problems solved at each recursion step. Potentially, the same sub-problem appears at different recursion steps and is solved several times. This inefficiency is eliminated by tracking the sub-problems encountered and only solving them once.

6.5.3.2 Weight Vector Formulation and Parameterization

The above algorithm takes the weight vector as input. The weight vector can be formulated in a variety of ways. Colom, [Colo90], describes polynomial time

algorithms for computing the P-invariants of a PN, (see Section 3.5). For a PN with n P-invariants, $\underline{p}_i, i = 1, 2, \ldots, n$, the weight vector, $\underline{w}$, is given by

$$\underline{w} = \sum_{i=1}^{n} \alpha_i \underline{p}_i \tag{6.29}$$

where, α_i is an arbitrary positive rational coefficient. Alternatively, formulation of the weight vector can be posed as a constrained graph labeling problem. Solutions to these types have been presented using the back-chaining capabilities of Prolog, [Cloc84].

Regardless of the method used to formulate the weight vector, after it is normalized, any remaining ambiguity will be of type 2 ambiguity. This ambiguity can be removed by selecting any solution of the weight vector parameterization that results in rational components. It is clear that such a solution always exists; in fact many solutions result from different choices of the parameterization values. Experience indicates that the accuracy of the estimation algorithm benefits from components that are relatively prime. For example, from Figure 3.3, which has weight vector solutions (3.70) that are reduced to (6.26), permissible solutions of (6.26) include: $\underline{w}_1 = [1\ 1\ \frac{1}{2}\ \frac{1}{2}]^\mathsf{T}$, $\underline{w}_2 = [1\ 1\ \frac{2}{3}\ \frac{1}{3}]^\mathsf{T}$, and $\underline{w}_3 = [1\ 1\ \frac{3}{5}\ \frac{2}{5}]^\mathsf{T}$. The estimated number of states for $\underline{w}_i$ are 5, 4, and 3, respectively. There are actually 3 states in this PN. The errors from $\underline{w}_1$ come from 3 extraneous states picked up by distributing 2 tokens between places P_3 and P_4. For $\underline{w}_2$, the only extraneous state is the possibility of 3 tokens in P_4 that comes from using a weight of $\frac{1}{3}$.

It is difficult to provide additional insight into the weight vector parameterization. However, the upper bound property of the estimation and the speed of the algorithm make it possible to try several parameterizations and simply pick the lowest state-space size estimate. Typically, only a few trials are needed to achieve good accuracy.

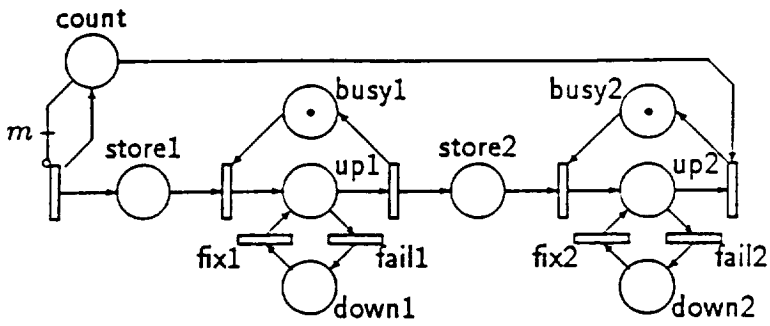


Figure 6.10: PN Model of 2M1B Transfer Line With Machine Failures

6.6 Example: Comparing the Top-Down Estimators

In this section a two machine-one buffer (2M1B) transfer line is considered. A similar example has been considered for studying variations in production rates given changes in machining performance, [AlJa88b, AlJa90a].

The PN in Figure 6.10 represents a 2M1B transfer line. Tokens, representing parts, enter on the left to fill a local store, store1, are processed by machine 1 and then by machine 2. Failure and repair times of the machines are modeled by transitions fail1, fix1, fail2, and fix2. The place count counts parts and inhibits arrivals if m parts occupy the system. This PN, because of the arrival and exit of tokens, is nonconservative. Figure 6.11 illustrates an equivalent conservative PN, whereby the arrival process inhibitor arc weight m is now considered as the initial marking of P_1. This PN is used to demonstrate the four estimators.

The conservative PN has 8 places and $m + 2$ tokens total, with a maximum of m tokens in P_1 and P_5 and a maximum of 1 token in the remaining places. Table 6.1 summarizes the actual and estimated number of states. GreatSPN, [Chio87], was used to obtain the actual number of states (however for $m = 1000$, the computation would not complete due to software capacities). While the first three estimators provide some insight into the sensitivity issue, as expected, they quickly result in

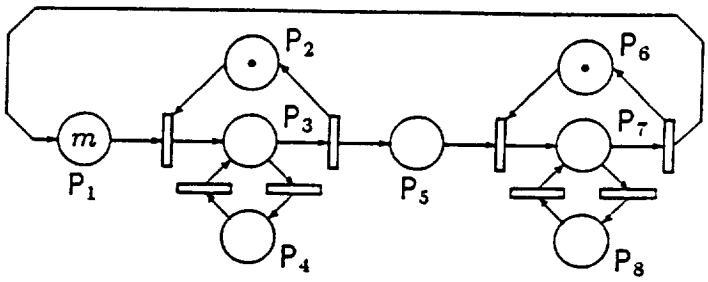


Figure 6.11: Conservative PN Model

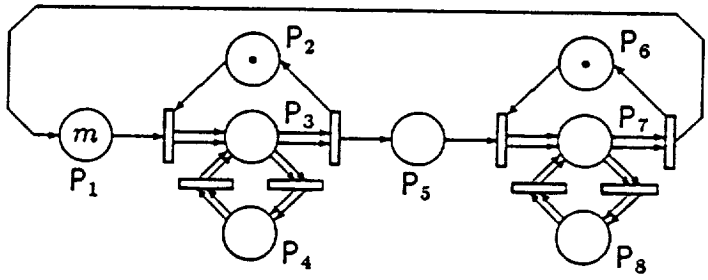


Figure 6.12: Strictly Conservative PN Model

astronomical estimates.

Estimator $\tilde{s}_5$ demonstrated excellent results: the estimated number of states was always 2 more than the actual number of states, thus making the relative error tend to zero as the number of tokens was increased. The actual number of states approaches $9m$, which supports the conjecture that the number of states of a given PN will asymptotically approach a simple relationship. This quality of estimation is not achieved with all models, however experience indicates that the estimation is well within a half-order of magnitude. Switches on places in the model tend to weaken the estimator, indicating an opportunity to improve accuracy by applying the estimator in conjunction with a subnet analysis. This is done in Chapter 7, which considers the bottom-up estimation method.

m	actual	$\tilde{s}_1$	$\tilde{s}_2$	$\tilde{s}_3$	$\tilde{s}_4$	$\tilde{s}_5$
1	6	256	256	165	120	8
2	15	65536	576	495	330	17
3	24		1024	1287	792	26
4	33		1600	3003	1716	35
10	87		7744	125970	50388	89
100	897					899
1000	n/a					8999

Table 6.1: Comparison of Estimators

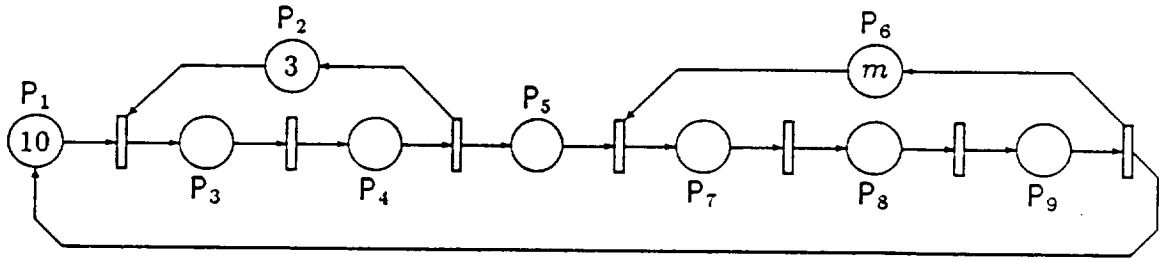


Figure 6.13: PN Model of Delivery-Machining System

6.7 Example: Estimation of a Weighted Conservative Petri Net

As a more complex example, consider the PN in Figure 6.13, which is a manufacturing system model, consisting of a parts delivery system (with a second-order Erlangian service time) followed by a machining cell (with a third-order Erlangian service time). There are 3 servers in the parts delivery system, a maximum of 10 parts that will occupy the system, and m servers in the machining cell.

The weight vector for this PN is formulated as follows. P_1 is arbitrarily assigned weight α, and P_2 a weight of β. Then from the network connectivity, P_3 must have weight $\alpha+\beta$, as must P_4. The sum of the weights of P_5 and P_2 must be $\alpha+\beta$, leading to a weight of α for P_5. This result agrees with our intuition that places P_1 and P_5 should be of equal weight. Furthermore, P_6 is arbitrarily assigned weight γ leading

to a weight of $\alpha + \gamma$ for places P_7, P_8, and P_9. This gives the normalization equation

$$\begin{bmatrix} \alpha & \beta & \alpha+\beta & \alpha+\beta & \alpha & \gamma & \alpha+\gamma & \alpha+\gamma & \alpha+\gamma \end{bmatrix} \begin{bmatrix} 10 \\ 3 \\ 0 \\ 0 \\ 0 \\ m \\ 0 \\ 0 \\ 0 \end{bmatrix} = 1$$

resulting in the constraint of

$$10\alpha + 3\beta + m\gamma = 1$$

or, equivalently,

$$\alpha = \frac{1 - 3\beta - m\gamma}{10} \tag{6.30}$$

Substituting (6.30) into the weight vector solution gives

$$\underline{w} = \begin{bmatrix} (1-3\beta-m\gamma)/10 \\ \beta \\ (1+7\beta-m\gamma)/10 \\ (1+7\beta-m\gamma)/10 \\ (1-3\beta-m\gamma)/10 \\ \gamma \\ (1-3\beta+(10-m)\gamma)/10 \\ (1-3\beta+(10-m)\gamma)/10 \\ (1-3\beta+(10-m)\gamma)/10 \end{bmatrix} \tag{6.31}$$

Estimating the number of states is accomplished by choosing values for β and γ, and applying the above algorithm. Table 6.2 summarizes the number of actual and estimated states for various m. Observe that since the number of parts in the model is limited to 10, there is no increase in the number of states beyond that for 10 servers. Additional tokens in P_6 are dead. The values for the actual number of states were obtained from GreatSPN, [Chio87].

m	number of states		
	actual	$\beta = \frac{1}{9}; \gamma = \frac{1}{11}$	$\beta = \frac{1}{11}; \gamma = \frac{1}{31}$
1	330	332	332
2	750	1188	913
3	1350	1352	2267
4	2100	13978	2383
5	2940	3922	2942
6	3780	26590	4136
7	4500	272063	4502
8	4950	—	12935
9	5170	—	5172
10	5236	—	5700
11	5236	—	5238
12	5236	—	5700
13	5236	—	34337
14	5236	—	10250
15	5236	—	5238

Table 6.2: Number of States With Varying Number of Servers, m

To illustrate the estimator's behavior, two sets of values are considered for β and γ, each with relatively prime denominators. With $\beta = \frac{1}{9}$ and $\gamma = \frac{1}{11}$, the weight vector becomes

$$\underline{w} = \frac{1}{990} \cdot \begin{bmatrix} 66 - 9m \\ 110 \\ 176 - 9m \\ 176 - 9m \\ 66 - 9m \\ 90 \\ 156 - 9m \\ 156 - 9m \\ 156 - 9m \end{bmatrix}, \tag{6.32}$$

where λ is taken as 990. This choice is only acceptable for $m \leq 7$, since larger m makes the first and fifth elements negative. The estimation results are given in Table 6.2. For $m = 4$, the accuracy of the estimator becomes poorer; (6.32) in this

case is

$$\begin{bmatrix} 30 & 110 & 140 & 140 & 30 & 90 & 120 & 120 & 120 \end{bmatrix}^T,$$

and many extraneous states are estimated from the fact that the weight vector elements are all multiples of ten.

Experience has shown that larger relatively prime denominators tend to be more effective. For this example, wanting to estimate the number of states for larger m is also motivation for choosing a smaller γ. For $\beta = \frac{1}{11}$ and $\gamma = \frac{1}{31}$, we get a weight vector of

$$\underline{w} = \frac{1}{3410} \cdot \begin{bmatrix} 246 - 11m \\ 310 \\ 556 - 11m \\ 556 - 11m \\ 246 - 11m \\ 110 \\ 356 - 11m \\ 356 - 11m \\ 356 - 11m \end{bmatrix},$$

where λ is taken as 3410. The performance for this choice of weight vector is also summarized in Table 6.2. Note the affect of the dead tokens can cause a momentary decrease in the estimator's performance (e.g., $m = 13$).

6.8 Example: Poor Estimator Performance

Section 6.5.2 described several constructs that interfere with the performance of the estimation algorithm for weighted conservative PNs. These constructs, while permissible, can be considered poor design. For example, there is no value or purpose provided by a dead token. In this section, a more illustrative example of poor estimation performance is presented. The PN in Figure 6.14 represents a potential model that does not contain any poor design. The model contains $n + 1$ places, n of

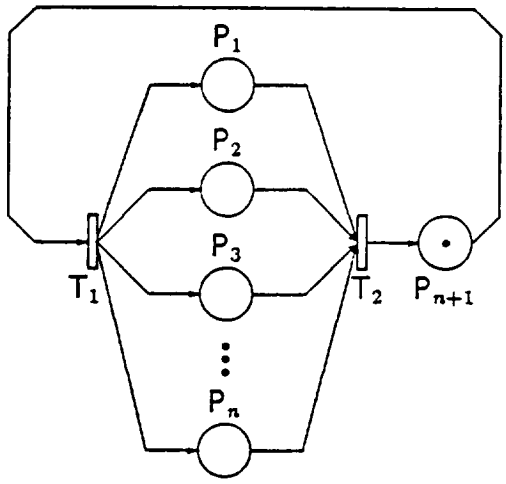


Figure 6.14: Example PN Model Having Poor Top-Down Estimation

which are involved in a choice. It is easy to verify that the model has only two states for any value of n.

The weight vector for this model has form

$$\underline{w} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_n \\ \sum_{i=1}^{n} \alpha_i \end{bmatrix} \tag{6.33}$$

which upon normalization with the initial marking requires

$$\sum_{i=1}^{n} \alpha_i = 1 \tag{6.34}$$

Solving (6.34) for α_1 results in the constraint

$$\alpha_1 = 1 - \sum_{i=2}^{n} \alpha_i \tag{6.35}$$

which is substituted into (6.33) to produce

$$\underline{w} = \begin{bmatrix} 1 - \sum_{i=2}^{n} \alpha_i \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_n \\ 1 \end{bmatrix} \tag{6.36}$$

Notice that the weight vector contains $n-1$ free parameters. The estimation algorithm is hampered by the need to search over permissible values of these parameters for a good result. When there are only a few parameters, this search space is small—however, for large n, the problem is much more complex.

All models having choices among a large number of alternative paths will demonstrate this behavior. It is important to iterate, however, that these are the types of models best handled by the bottom-up estimation method presented in the next chapter. The alternate paths represent modules, and the modular construction of models similar to that in Figure 6.14 can be used to produce accurate estimation in the bottom-up method.

6.9 Summary

This chapter has developed the area of top-down state-space size estimation. Simple estimators introduce the problem and provide insight into its solution. An estimation algorithm for weighted conservative PNs was developed. This estimator provides accurate results, and additionally places no constraints on the PN design. The algorithm was analyzed for complexity and PN constructs that interfere with effective estimation were identified.

CHAPTER 7
BOTTOM-UP SIZE ESTIMATION

7.1 Introduction

The problem of estimating the state-space size of Petri nets (PNs) is pursued in two distinct manners: top-down and bottom-up. This chapter focuses on the bottom-up method.

The state-space size estimators developed in this chapter are based on constraining the PN construction to a limited set of subnets and interconnections. In general, bottom-up analysis is best applied during the construction of the model. Given the limited representation resulting from the design constraints, it may be difficult, or even impossible, to decompose an already existing model. However, the design constraints permit better estimation than achieved from top-down methods.

The state-space size estimation algorithm described here allows relatively arbitrary subnets, thereby permitting analysis of a large class of PN models. Furthermore, the technique is extendable via the development of additional interconnections. Focusing on the interconnections is important, because it is the interconnection transitions that control the token flow throughout the aggregate model. Included in this chapter are interconnections for sequential activity, parallel activity, choice, blocking, resource sharing, and failure/repair. Other interconnections are possible, and their development would increase the applicability of the state-space size analysis. In addition to the interconnections, the effect of algorithmic model augmentation on the state-space size is investigated. Specifically, non-exponential transition models, [Wats91b], and automated error recovery models, [Zhou89, Zhou90], are considered. By studying the state-space size growth resulting from such augmentations, trade-offs between model accuracy and computational complexity can be evaluated.

Estimators are formulated for each of the subnets, and the effects of combining the estimators via the permitted interconnections are determined. The symbol $\tilde{s}_{\text{subnet}}(\cdot)$ denotes the estimator for a particular subnet.

The remainder of this chapter is organized as follows. Section 7.2 provides a general background on the bottom-up algorithm. State counting functions are discussed in Section 7.3. Sections 7.4 and 7.5 describe the interconnections and augmentations, and Section 7.6 describes the subnets.

7.2 Background

The bottom-up algorithm considers the PN model to consist of basic subnets and interconnections. The subnets are themselves distinct PNs, (i.e., the subnets cannot share places); the interconnections are not restricted. PN extensions, such as transition priorities, inhibitor arcs, and transition classes (i.e., immediate and timed), [Mars84a, Balb87], are permitted, but add to the complexity of the analysis.

Analogous to the state of a PN, a subnet's state is given by the token distribution among the places in the subnet. The estimation technique works with the state-spaces of the individual subnets. Subnet states might be termed "substates" of the aggregate PN model, however such a distinction is not required. Since the subnets do not share places, i.e., clear boundaries can be made, a state of a subnet unambiguously corresponds to a set of nodes in the state-space graph of the aggregate PN.

For each subnet, our goal is to develop a state counting function, (abbreviated as sc-function), that describes the subnet's state-space size when it contains exactly τ "flow" tokens. We distinguish "flow" tokens (those that enter and leave the subnet from its entry and exit paths) from "control" tokens (those that are part of the subnet's initial marking, remain within the subnet, and are used to control its behavior, e.g., by enabling/disabling certain transitions under various circumstances). If the subnet's input path begins with a transition, e.g., T_{in}, and its output path ends

with another transition, e.g., T_{out}, then, in most models, τ is equivalent to the firing deviation, [Mura89], between T_{in} and T_{out}.

The computation of an sc-function depends on the complexity of the subnet. However, in general, subnets will fall into one of the following categories:

1. The subnet has an sc-function with a known closed-form solution. We will illustrate some of these subnets in this chapter, concentrating on the serial line.

2. The subnet can be modified so that the new subnet, which approximates the original, has a closed-form sc-function.

3. A top-down state-space size estimation algorithm, such as presented in Chapter 6, can be used to compute an estimate of the sc-function. The errors in the sc-function estimation will propagate to the bottom-up state-space size estimation.

4. The sc-function can be computed by hand. While it is an option for the user to supply data, we emphasize automated computation (or estimation) of sc-functions.

7.3 State Counting Functions

A description of the state-space size for each subnet is used in the estimation algorithm. Based on the number of flow tokens in the subnet, an sc-function, provides this information. In general, an sc-function, $\tilde{s}(\cdot)$, possesses the following properties:

- It is a function of τ only, i.e.,

$$\tilde{s} \; : \; I\!N_{0+} \rightarrow I\!N_{0+}$$

Thus, the influence of any control tokens in a subnet must be "invariant" with respect to the history of τ. In other words, $\tilde{s}$ cannot depend on time. The

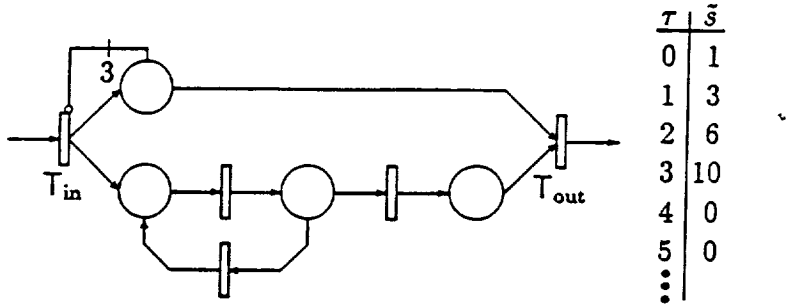


Figure 7.1: Example Subnet and SC-Function

exclusion of any transient behavior in the model produces a state-space size estimate for the steady-state model.

- For subnets without control tokens, $\tilde{s}(0) = 1$, that is, there is a single state in the presence of zero flow tokens, namely, the null-state. For subnets that contain control tokens, the above relationship may not hold. If the control tokens can create states even in the absence of flow tokens, then $\tilde{s}(0) \neq 1$.

- $\tilde{s}(\tau) = 0$ implies the subnet cannot possibly contain τ flow tokens. This result occurs regardless of the presence of control tokens.

Figure 7.1 illustrates a subnet and its sc-function, which demonstrates these properties.

7.4 Interconnections

By accommodating arbitrary subnets, the representational power of this algorithm is dictated by the permissible interconnections. Interconnections for basic mechanisms of execution, resource utilization, and failure/repair are presented here. Other interconnections are possible, and their development extends the applicability of the state-space size analysis. Individual SISO subnets are denoted by SN_1, SN_2, and so forth, having, respectively, sc-functions, $\tilde{s}_1$, $\tilde{s}_2$, etc.

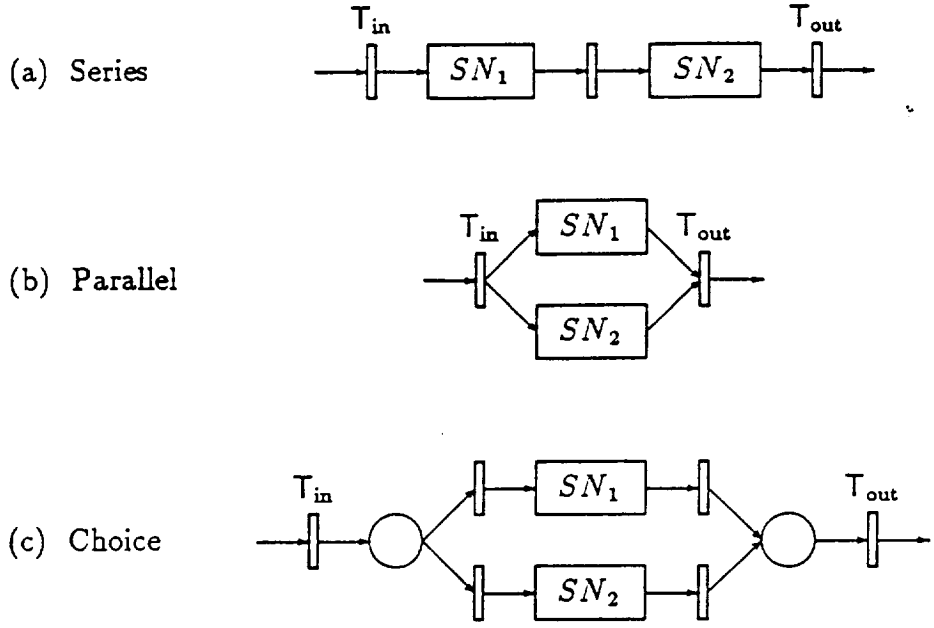


Figure 7.2: Basic Interconnections

7.4.1 Basic Mechanisms of Execution

PNs can model various mechanisms of execution, including sequential execution of operations, parallel execution of operations, and choice among several alternative operations. Operations are abstracted to be SISO subnets, and the effects of these interconnections on the sc-functions are determined.

Figure 7.2a illustrates two subnets in series, whereby tokens pass from subnet SN_1 to SN_2. The sc-function of the interconnection is determined from the fact that the τ tokens between T_{in} and T_{out} can be distributed between the two subnets according to the partitions in $I_2(\tau)$. Thus,

$$\tilde{s}_{\text{series}}(\tau) = \sum_{\underline{v} \in I_2(\tau)} \tilde{s}_1(v_1)\tilde{s}_2(v_2)$$

which is equivalent to

$$\tilde{s}_{\text{series}}(\tau) = \sum_{i=0}^{\tau} \tilde{s}_1(i)\tilde{s}_2(\tau - i) \qquad (7.1)$$

Extending this interconnection model to more than two subnets is done iteratively: the connection of the first two subnets is a SISO subnet itself in series with SN_3, which, again is a SISO subnet in series with SN_4, and so forth. Thus, for n subnets, the sc-function is easily expressed by the partition notation,

$$\tilde{s}_{\text{series}}(\tau) = \sum_{\underline{v} \in I_n(\tau)} \tilde{s}_1(v_1)\tilde{s}_2(v_2)\cdots\tilde{s}_n(v_n) \tag{7.2}$$

Parallel execution of operations is depicted in Figure 7.2b. For every token entering through T_{in}, there is one in each subnet. Thus, the sc-function is given by

$$\tilde{s}_{\text{parallel}}(\tau) = \tilde{s}_1(\tau)\tilde{s}_2(\tau) \tag{7.3}$$

Since grouping two subnets in this interconnection does not result in a SISO subnet, (7.3) cannot be extended using the iterative technique employed for the series interconnection. However, regardless of how many subnets are in parallel, the individual state-spaces develop independent of one another, producing for n parallel subnets,

$$\tilde{s}_{\text{parallel}}(\tau) = \prod_{i=1}^{n} \tilde{s}_i(\tau) \tag{7.4}$$

Figure 7.2c models the execution of a choice among two subnets. There are three constructs to consider: SN_1, SN_2, and, as a group, the two places that commence and terminate the choice. Although there is no difference combinatorially between considering the places individually or as a group, a more concise sc-function results from the latter. With τ flow tokens, any 3-partition of τ describes a potential token distribution among the two subnets and the group of places. Having k tokens in the group of two places generates ${}^2X_k = k + 1$ states. Thus,

$$\tilde{s}_{\text{choice}}(\tau) = \sum_{\underline{v} \in I_3(\tau)} \tilde{s}_1(v_1)\tilde{s}_2(v_2)\cdot(v_3 + 1) \tag{7.5}$$

Extending to n subnets, we have

$$\tilde{s}_{\text{choice}}(\tau) = \sum_{\underline{v} \in I_{n+1}(\tau)} \tilde{s}_1(v_1)\tilde{s}_2(v_2)\cdots\tilde{s}_n(v_n)\cdot(v_{n+1} + 1) \tag{7.6}$$

7.4.2 Resource Constrained Operations

PNs are useful for modeling resource constrained operations. Two resource interconnections are considered: resource blocking of a single subnet; and resource sharing between two subnets. Figures 7.3 and 7.4 illustrate two variations, one with queuing and one without, for each of the interconnections. The tokens in place $\mathsf{P}_{\mathsf{res}}$ represent the availability of the resource—these are control tokens and should not be confused with the flow tokens that enter and exit the subnet. The generality of the resource interconnection models would be improved by substituting a generic subnet, say SN_{res}, for the single place, $\mathsf{P}_{\mathsf{res}}$ that holds the resources. The sc-functions would then contain the term $\tilde{s}_{\mathsf{res}}(r-\tau)$. For the purposes of easier discussion, we choose to develop the simpler models.

Figure 7.3 illustrates a resource blocking interconnection. First, we consider the case without queuing. The initial marking of $\mathsf{P}_{\mathsf{res}}$ indicates the number of tokens that SN_1 can process simultaneously. For $\tau \leq r$, the execution, and therefore the states that result from it, are unaffected by the resource limitation. Thus,

$$\tilde{s}_{\mathsf{blocking}}(\tau) = \begin{cases} \tilde{s}_1(\tau) & \tau \leq r \\ 0 & \text{otherwise} \end{cases} \tag{7.7}$$

where the sc-function is zero for $\tau > r$ since this represents an impossible situation. With a queuing subnet included in the model, the ability to handle more than r flow tokens is possible. If $\tau \leq r$, then SN_Q and SN_1 behave as two subnets in series; and, for $\tau > r$, at most r tokens can enter SN_1, leaving the remainder in SN_Q. Thus, the sc-function is given by

$$\tilde{s}_{\mathsf{Qblocking}}(\tau) = \begin{cases} \sum_{i=0}^{\tau} \tilde{s}_1(i)\tilde{s}_Q(\tau-i) & \tau \leq r \\ \sum_{i=0}^{r} \tilde{s}_1(i)\tilde{s}_Q(\tau-i) & \tau > r \end{cases} \tag{7.8}$$

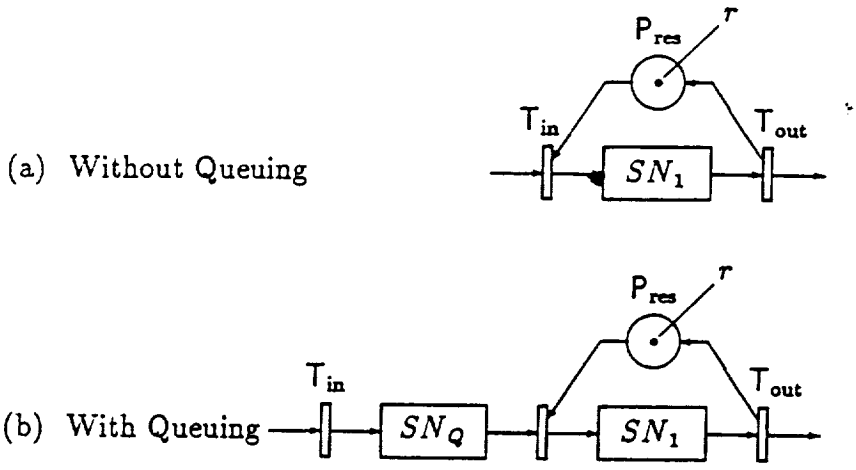


Figure 7.3: Blocking Interconnections

Sharing a resource among two subnets is modeled in Figure 7.4. This interconnection does not result in a SISO subnet. It is included to illustrate the extendibility of the sc-functions and the additional representational flexibility possible within the method. For this interconnection, the sc-function is based on two variables, τ_1 and τ_2, that describe the flow tokens in upper and lower branches, respectively. We have, for the non-queuing case,

$$\tilde{s}_{\text{share}}(\tau_1, \tau_2) = \begin{cases} \tilde{s}_1(\tau_1)\tilde{s}_2(\tau_2) & \tau_1 + \tau_2 \leq r \\ 0 & \tau_1 + \tau_2 > r \end{cases} \tag{7.9}$$

and for the queuing case,

$$\tilde{s}_{\text{Qshare}}(\tau_1, \tau_2) = \begin{cases} \sum_{i=0}^{\tau_1} \sum_{j=0}^{\tau_2} \tilde{s}_1(i)\tilde{s}_{Q1}(\tau_1 - i)\tilde{s}_2(j)\tilde{s}_{Q2}(\tau_2 - j) & \tau_1 + \tau_2 \leq r \\ \sum_{i=0}^{\min(\tau_1, r)} \sum_{j=0}^{\min(\tau_2, r-i)} \tilde{s}_1(i)\tilde{s}_{Q1}(\tau_1 - i)\tilde{s}_2(j)\tilde{s}_{Q2}(\tau_2 - j) & \tau_1 + \tau_2 > r \end{cases} \tag{7.10}$$

7.4.3 Failure/Repair

PNs of manufacturing systems and other systems containing unreliable components often include models for the failure and repair of these components. Figure 7.5 depicts a common model for these operations.

(a) Without Queuing

(b) With Queuing

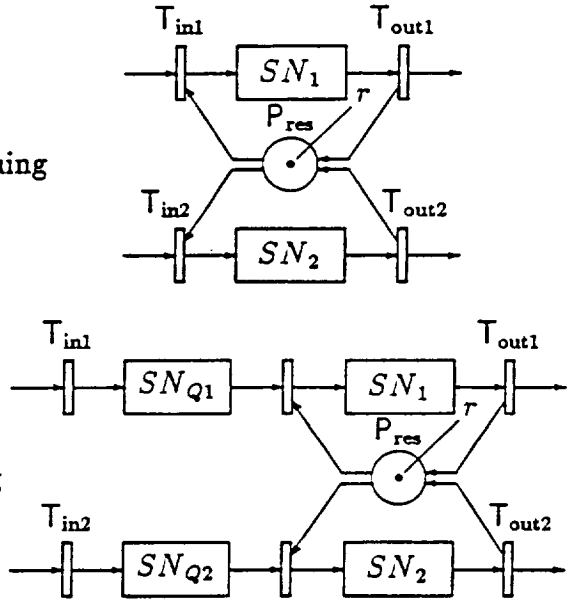


Figure 7.4: Resource Sharing Interconnection

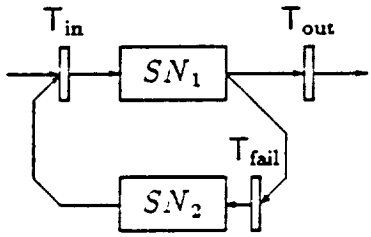


Figure 7.5: Failure/Repair Interconnection

Under normal circumstances, T_{out} fires instead of T_{fail}, leaving SN_2 out of consideration. However, this choice between T_{out} and T_{fail} permits the possibility that SN_2 will be used by all τ tokens. Thus, SN_1 and SN_2 are essentially in series, producing

$$\tilde{s}_{\text{failrep}}(\tau) = \sum_{\underline{v} \in I_2(\tau)} \tilde{s}_1(v_1)\tilde{s}_2(v_2) \tag{7.11}$$

The fact that SN_2 is connected, serially, back to SN_1 does not introduce more states, but only increases the number of ways to move between these states.

7.5 Augmentations

7.5.1 Non-exponential Transition Approximations

The author has described a method to approximate non-exponential transitions within GSPNs, (see Chapter 4 and [Wats91b]). When conditions permit such an approximation, the appropriate exponential transition can be replaced by a serial line of transitions and places. This replacement can be considered an augmentation to the nominal performance model to provide improved accuracy in the results. This improvement in accuracy comes at the price of increased state-space size. As with the error recovery augmentations, it is this state-space size growth we wish to study.

Serial lines are discussed in the section on subnets, thus a discussion here is not needed. We note that in addition to being a potential subnet, the serial line is also a potential augmentation.

7.5.2 Error Recovery

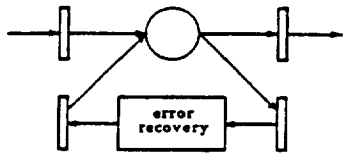
The modeling properties of PNs make them attractive for use as controllers. In such systems, a PN models the evolution of actions and states in a real system. The controller decides what actions are permissible by using the basic PN firing

rules, and the model is updated to reflect the state of the system. Error recovery is needed whenever the state of the PN does not reflect the state of the real system. Zhou, [Zhou89, Zhou90], identified four error recovery augmentations for PNs: input conditioning, alternate path, backward recovery, and forward recovery. Figure 7.6 illustrates the models for each of the cases. If an operation cannot be executed because an error has caused a required pre-condition to no longer exist, input conditioning can be used to return the system, locally, to a state containing that precondition. The alternate path error recovery is appropriate if the state resulting from the recovery operation leaves the system in the same state as the nominal operation. Feedforward recovery is a variation of alternate path that handles errors in the middle of a sequence of nominal operations. Finally, feedback recovery is used to return the system to some previous state, from which the nominal operations will be tried again.

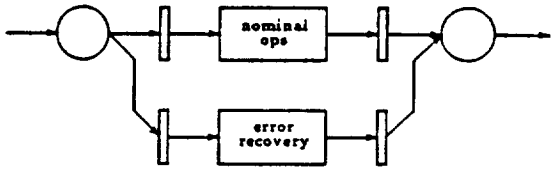
The error recovery augmentations can be represented by the interconnection models, or combinations thereof, discussed in Section 7.4. Input conditioning can be modeled as a failure/repair interconnection—in fact the failure subnet is performing operations to return the system to the state necessary for nominal operations. The choice interconnection can describe the decision found in both the alternate path and feedforward recovery models, with the latter also utilizing the series interconnection. Feedback recovery, as depicted, does not match any of the interconnection models. Note, however, that the re-execution of nominal operations following the execution of error recovery operations is illustrated in Figure 7.5. Thus, the essence of this error recovery augmentation, particularly from a state-space size perspective, is represented by the interconnections.

7.6 Subnets

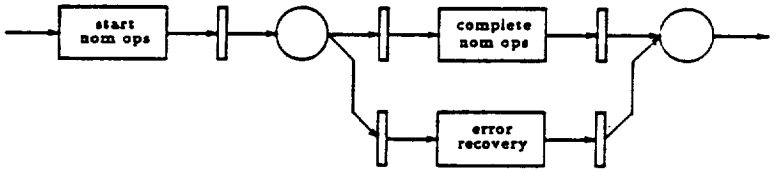
With the above development of the interconnections, we now focus on the subnets and their sc-functions. Any configuration of places, transitions, arcs, and control



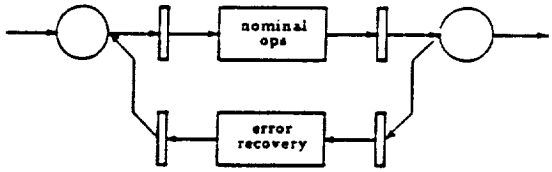
(a) Input Conditioning



(b) Alternate Path



(c) Feedforward Recovery



(d) Feedback Recovery

Figure 7.6: Error Recovery Augmentations

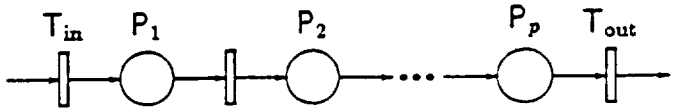


Figure 7.7: Serial Subnet

tokens, resulting in a SISO subnet is valid. To perform the state-space size estimation, an sc-function is associated with each of the subnets. For subnets that do not possess the "invariance" property described in Section 7.3, the accuracy of the estimation will be weakened.

We focus on the serial subnet and its variations, which possess several useful properties. Foremost, while the state-space size estimation is not limited to automatic formulation of sc-functions, the serial subnets provide convenient closed-form solutions that can be included in automated estimation tools. Additionally, the serial subnet degenerates to a single place, thereby, producing a basic construct. Also, the serial subnet can be used in performance models to approximate non-exponential time delays.

7.6.1 Serial Subnet

Figure 7.7 illustrates a p-place serial subnet having input and output paths designated by, respectively, transitions T_{in} and T_{out}. With τ tokens between these two transitions, the sc-function is described by Definition 3.4:

$$\tilde{s}_{serial}(p;\tau) \;=\; {}^{p}X_{\tau} \tag{7.12}$$

where p parameterizes $\tilde{s}_{serial}(\tau)$. (Do not confuse $\tilde{s}_{series}$, the interconnection, with $\tilde{s}_{serial}$, the subnet.)

7.6.2 Single Place

The serial line degenerates into a single place for $p = 1$. In this case, the sc-function is

$$\tilde{s}_{\text{serial}}(1;\tau) \;=\; {}^{1}X_{\tau} \;=\; 1 \tag{7.13}$$

Note that regardless of the number of tokens, there is only a single state. This property has already shown itself in the derivations of some of the interconnection sc-functions. For example, for the resource interconnections, the influence of the single-place resource subnet, P_{res}, is unity.

In many models, it is typical to use a single place as a queuing subnet. This results in a significant simplification of the sc-functions of interconnections containing queues. Reconsider Figure 7.3b, with SN_Q as a single place. Then (7.8), repeated here,

$$\tilde{s}(\tau) \;=\; \begin{cases} \sum_{i=0}^{\tau} \tilde{s}_1(i)\tilde{s}_Q(\tau - i) & \tau \le r \\ \sum_{i=0}^{\tau} \tilde{s}_1(i)\tilde{s}_Q(\tau - i) & \tau > r \end{cases}$$

simplifies to

$$\tilde{s}(\tau) \;=\; \begin{cases} \sum_{i=0}^{\tau} \tilde{s}_1(i) & \tau \le r \\ \sum_{i=0}^{\tau} \tilde{s}_1(i) & \tau > r \end{cases} \tag{7.14}$$

Note the property of (7.14) to remain constant for $\tau \ge r$.

7.6.3 Choice Subnet

Figure 7.8 models the choice between n serial subnets of length p_i, $i = 1, 2, \ldots, n$. Interestingly, the sc-function does not depend on the individual lengths of the subnets, but only the total number of places. Including the two places that commence and

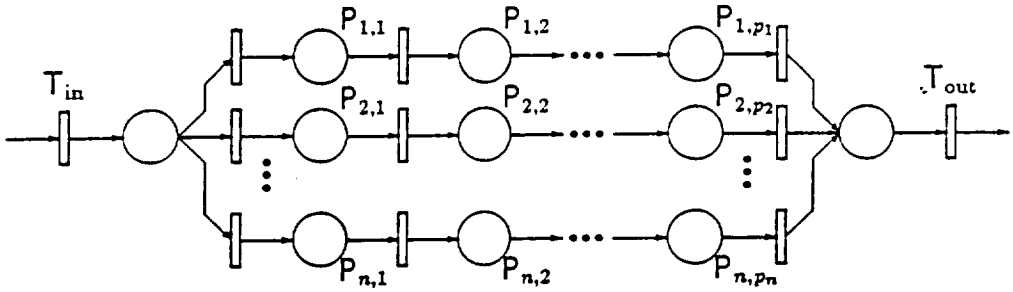


Figure 7.8: Choice Subnet

terminate the choice, we define,

$$k \stackrel{\text{def}}{=} 2 + \sum_{i=1}^{n} p_i \tag{7.15}$$

For $\tau = 0$, the sc-function evaluates to unity. Consider $\tau = 1$. In this case, there are ${}^{k}C_1 = k$ ways to select a single place (containing the single token), thereby resulting in k states. For $\tau = 2$, there are two possibilities, namely, both tokens are in the same place, or each token is in a different place. The former, (as for $\tau = 1$), produces k states, and the latter produces ${}^{k}C_2 = k(k-1)/2$ states. These two sets of states are distinct, so the combined state-space is given by $k + k(k-1)/2$.

As a final example, we consider $\tau = 3$, in full detail. There are τ possibilities: one place contains all τ tokens; two places, together, contain all τ tokens; and so forth, until the alternative to select τ places is considered. Note that the selection of more than τ places is not feasible, since at least one of the places will be empty. For the case that i places have been selected, $1 \leq i \leq \tau$, the distribution of the tokens is given by the set of occupations, $J_i(\tau)$. Thus, there are ${}^{k}C_i$ ways to select i places from the total set of places, and there are $|J_i(\tau)|$ ways to then distribute the tokens among these places.

The above logic assumed $\tau \leq k$, permitting τ possibilities in the number of

places selected. However, for $\tau > k$, there are only k possibilities. Thus, the sc-function is given by

$$\tilde{s}_{\text{choice-serial}}(k;\tau) = \begin{cases} 1 & \tau = 0 \\ \sum_{i=1}^{\min(\tau,k)} {}^{k}C_i \cdot {}^{\tau-1}C_{i-1} & \tau \geq 1 \end{cases} \tag{7.16}$$

Comparing (7.6) and (7.16) reveals a reduction in the number of computations required to evaluate the sc-function. For the general choice interconnection, ${}^{n+1}X_\tau$ additions are required, compared with at most τ additions for the choice composed of serial subnets.

7.6.4 Parallel Subnet

Similar to Figure 7.8, we can place n serial subnets of length $p_i, i = 1, 2, \ldots, n$, in parallel. This is offered for completeness, since there is no simplification of (7.4); the sc-function is

$$\tilde{s}_{\text{par-serial}}(n, \underline{p}; \tau) = \prod_{i=1}^{n} {}^{p_i}X_\tau$$

7.7 Example: Composition of Interconnections and Subnets

The bottom-up state-space size estimation method is applied to the delivery-machining system considered in Section 6.7. The PN model, which is repeated in Figure 7.9, consists of a parts delivery system (with a second-order Erlangian service time) in series with a machining cell (with a third-order Erlangian service time). Each subsystem is resource limited: three servers in the delivery cell, and five servers in the machining cell (from Figure 6.13, we take $m = 5$). We are interested in obtaining the state-space size when at most 10 parts are offered to the system.

The subnets and interconnections for the model are considered in turn. One subnet is the part delivery system, which consists of places P_1 through P_4. This

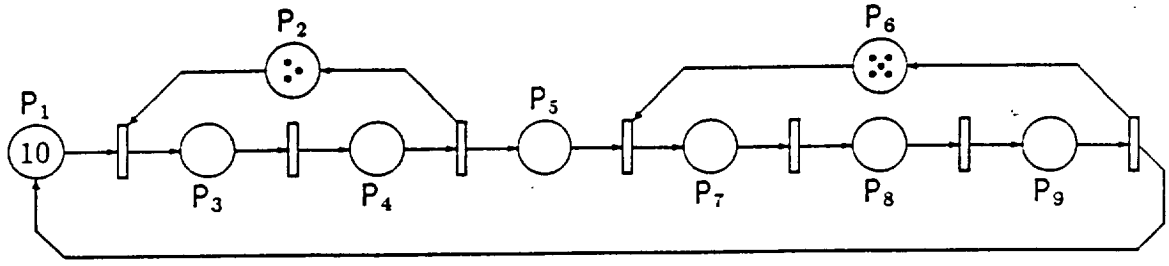


Figure 7.9: PN Model of Delivery-Machining System

subnet is a resource block with a single-place queue, i.e., P_1. Place P_2 holds the resources that are utilized by the internal subnet, i.e., a 2-place serial line formed by places P_3 and P_4. From (7.12), the sc-function for the 2-place serial line internal to the delivery subnet is

$$\begin{aligned} \tilde{s}_{\text{del-int}}(\tau) &= {}^{2}X_{\tau} \\ &= \tau + 1 \end{aligned} \tag{7.17}$$

Using (7.17) as the sc-function for the subnet internal to the resource block interconnection, (7.8) or, equivalently, (7.14), the sc-function for the entire delivery system is given by

$$\tilde{s}_{\text{del-sys}}(\tau) = \begin{cases} \sum_{i=0}^{\tau}(i+1) & \tau \leq 3 \\ \sum_{i=0}^{3}(i+1) & \tau > 3 \end{cases} \tag{7.18}$$

It is unnecessary to carry the baggage of an analytical solution for a subnet sc-function. Indeed, the implementation of an automated estimation tool would be complicated by such a practice. Instead, a tabulation of the sc-function can be made

quickly and used efficiently. Thus, equivalent to (7.18), we have,

$$\tilde{s}_{\text{del-sys}}(\tau) = \begin{cases} 1 & \tau = 0 \\ 3 & \tau = 1 \\ 6 & \tau = 2 \\ 10 & \tau = 3 \\ 10 & \tau > 3 \end{cases} \tag{7.19}$$

Note the resource interconnection provides information as to where to terminate the tabulation. This is due to the single-place queue. In general, the extent of the tabulation necessary is not known. A heuristic or user input could be used to judge where to terminate the tabulation—although, excess tabulation does not present a time or memory concern in an automated tool. However, since the interconnections are described by closed-formed functions, they can be evaluated for various τ whenever needed.

The state-space size estimation is continued by finding the sc-function for the machining cell. This module is also a resource block with queue, but contains a different internal subnet. The sc-function for this internal subnet is easily obtained from (7.12), as above. However, this analysis method can also accept user input or estimated sc-functions. Once the tabulation is obtained, from whatever source, the analysis can proceed. Thus, we have for the internal subnet sc-function,

$$\begin{aligned} \tilde{s}_{\text{mch-int}}(\tau) &= {}^{3}X_{\tau} \\ &= \frac{(\tau+2)(\tau+1)}{2} \\ &= \begin{cases} 1 & \tau = 0 \\ 3 & \tau = 1 \\ 6 & \tau = 2 \\ 10 & \tau = 3 \\ 15 & \tau = 4 \\ 21 & \tau = 5 \\ 28 & \tau = 6 \\ 36 & \tau = 7 \\ \vdots \end{cases} \end{aligned} \tag{7.20}$$

The extent of the tabulation will be determined when the resource interconnection is applied:

$$\tilde{s}_{\text{mch-sys}}(\tau) = \begin{cases} \sum_{i=0}^{\tau} \tilde{s}_{\text{mch-int}}(i) & \tau \leq 5 \\ \sum_{i=0}^{5} \tilde{s}_{\text{mch-int}}(i) & \tau > 5 \end{cases} \tag{7.21}$$

which is equivalent to

$$\tilde{s}_{\text{mch-sys}}(\tau) = \begin{cases} 1 & \tau = 0 \\ 4 & \tau = 1 \\ 10 & \tau = 2 \\ 20 & \tau = 3 \\ 35 & \tau = 4 \\ 56 & \tau = 5 \\ 56 & \tau > 5 \end{cases} \tag{7.22}$$

The delivery and machining subsystems are in series, so their sc-functions are combined in accordance with (7.1), producing

$$\begin{aligned} \tilde{s}_{\text{system}}(\tau) &= \sum_{i=0}^{\tau} \tilde{s}_{\text{del-sys}}(i)\tilde{s}_{\text{mch-sys}}(\tau - i) \\ &= \begin{cases} 1 & \tau = 0 \\ 7 & \tau = 1 \\ 28 & \tau = 2 \\ 84 & \tau = 3 \\ 205 & \tau = 4 \\ 431 & \tau = 5 \\ 784 & \tau = 6 \\ 1260 & \tau = 7 \\ 1820 & \tau = 8 \\ 2380 & \tau = 9 \\ 2940 & \tau = 10 \\ 3500 & \tau = 11 \\ 4060 & \tau = 12 \\ \vdots & \end{cases} \end{aligned} \tag{7.23}$$

A state-space size estimate for the model is given by $\tilde{s}_{\text{system}}(10)$, which is 2940.

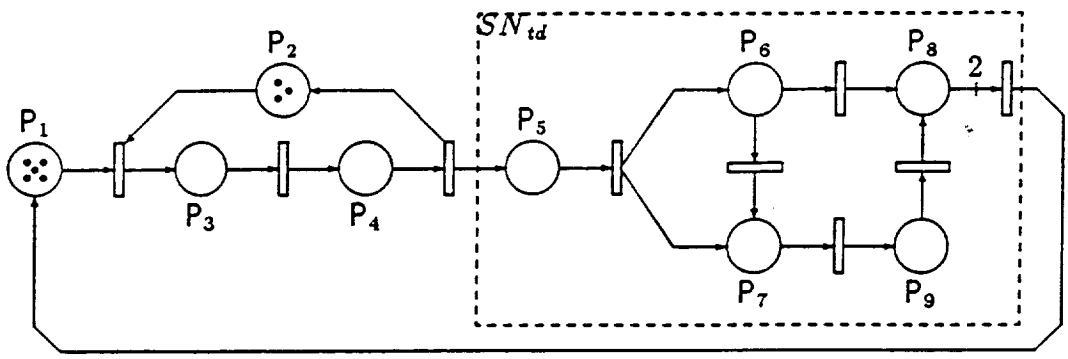


Figure 7.10: Example PN Model Having Poor Bottom-Up Estimation

agreeing with the GreatSPN, [Chio87], results computed in Section 6.7. The state-space size estimation is exact because no approximated or estimated sc-functions were needed.

7.8 Example: Poor Estimator Performance

Provided the PN to be estimated has a modular design, the bottom-up estimation is potentially exact. This potential is realized whenever the sc-functions for the needed subnets and interconnections have been developed. Estimation errors will occur if, for example, the sc-function for a subnet is approximated, either by hand or by top-down state-space size estimation. Figure 7.10 illustrates the PN that will be considered in this section. Top-down estimation will be used to approximate the sc-function for subnet SN_{td}, and the effects of these approximation errors on the total state-space size estimation will be analyzed.

The lefthand module is the same as in the previous example. Thus, from (7.19), the sc-function is given by

$$\tilde{s}_{\text{del-sys}}(\tau) = \begin{cases} 1 & \tau = 0 \\ 3 & \tau = 1 \\ 6 & \tau = 2 \\ 10 & \tau = 3 \\ 10 & \tau > 3 \end{cases} \qquad (7.24)$$

τ	estimated	actual
0	1	1
1	11	10
2	46	41
3	130	115
4	295	260
5	581	511

Table 7.1: Comparison of Actual and Approximated SC-Functions

The sc-function for SN_{td} is approximated with top-down state-space size estimation, as described in Chapter 6. Formulating the weight vector and normalizing it for τ tokens in place P_5 produces

$$\underline{w} = \begin{bmatrix} 1/\tau \\ 1/2\tau \\ 1/2\tau \\ 1/2\tau \\ 1/2\tau \end{bmatrix} \tag{7.25}$$

Using (7.25), the sc-function $\tilde{s}_{\text{td}}(\tau)$ is approximated with the weighted conservative PN state-space size estimation algorithm. Table 7.1 compares the values of the approximated and exact sc-functions.

As in the previous example, $\tilde{s}_{\text{del-sys}}$ and $\tilde{s}_{\text{td}}$ are in series, producing, from (7.1), an aggregate sc-function of

$$\begin{aligned} \tilde{s}_{\text{system}}(\tau) &= \sum_{i=0}^{\tau} \tilde{s}_{\text{del-sys}}(i)\tilde{s}_{\text{td}}(\tau - i) \\ &= \begin{cases} 1 & \tau = 0 \\ 14 & \tau = 1 \\ 85 & \tau = 2 \\ 344 & \tau = 3 \\ 1081 & \tau = 4 \\ 2826 & \tau = 5 \\ \vdots \end{cases} \end{aligned} \tag{7.26}$$

Thus, for the five parts entering the system, the estimated number of states is 2826.

This compares to the actual number of states of 2501, a 13% over estimation error.

7.9 Summary

A bottom-up state-space size estimation method for Petri nets (PNs) has been described. The method relies on the modularity of the PN design, and forms state counting functions, (sc-functions), that can be combined to generate the size estimates. Provided the PN model can be described with subnets and interconnections, the computation of state-space size is exact. Otherwise, errors in the estimation may result from changes in the PN model to produce modularity or inaccuracies in the sc-functions that are approximated or estimated.

CHAPTER 8
CASE STUDIES

8.1 Introduction

In this chapter, two case studies are developed. The PN models considered are taken from the published literature. Since a detailed description of the formulation of the models is prohibitive, a brief, high-level description is provided, and the reader is referred to the appropriate references.

Top-down estimation is applied to a problem in state-space size reduction. This case-study not only demonstrates the size estimation algorithm as a stand-alone tool, but how it can be used in conjunction with other analysis methods. Bottom-up estimation is applied to a performance model for a complex multi-server Flexible Manufacturing System (FMS). A non-exponential transition approximation is used to illustrate the estimation method's ability to analyze model augmentation.

8.2 Top-Down Estimation For State-Space Size Reduction Algorithm

One of the motivations for developing state-space size estimation theory is the recognition of the fact that models of real systems typically have enormous state-space sizes. Performance results are obtained by generating and analyzing the state-space of a model. Since both the generation and analysis of a large state-space can be very time consuming, research has been conducted in generating approximate performance results based on a reduced state model.

Jungnitz. [Jung92], develops an algorithm for approximating performance results by applying an iterative and hierarchical state-space size reduction algorithm. At each step. the algorithm divides the model into two subnets. $\mathcal{S}_1$ and $\mathcal{S}_2$. (which both have reduced state-space sizes), and abstracts the throughput interaction of one

subnet to another by a single transition. Closed-loop performance of $\mathcal{S}_1$ is obtained, and the data is transferred to $\mathcal{S}_2$. The performance of $\mathcal{S}_2$ is then obtained, data is transferred to $\mathcal{S}_1$, and the process is repeated until convergence occurs.

The reduction/approximation algorithm is automated except for the decision as to where to divide the model. State-space size estimation can be used in several ways:

- Data can be obtained to determine if it is even necessary to pursue the state-space size reduction algorithm. Approximation of performance results at the expense of reduced solution time represents a trade-off that is better evaluated with state-space size data.

- Even with the division step remaining unautomated, the user's decision would benefit from information on the state-space size of the potential subnet choices.

- Without information on the state-space size of the potential subnets, there would be no systematic way to automate the division step.

- The state-space size estimates of the before- and after-reduction models can be compared to determine what computational savings can be expected.

Figure 8.1 depicts a dataflow graph model considered in [Jung92]. The original model contained four pairs of immediate transitions, that are now indistinguishable from the timed transitions. The dotted line represents the place where the model was divided during the reduction/approximation algorithm.

First we apply state-space size estimation to the model to determine if the reduction algorithm is warranted. Since the model was already designed and has no clear modularity, the top-down estimation strategy is used. The model is weighted conservative, and from the methods developed in Chapter 6, the following general

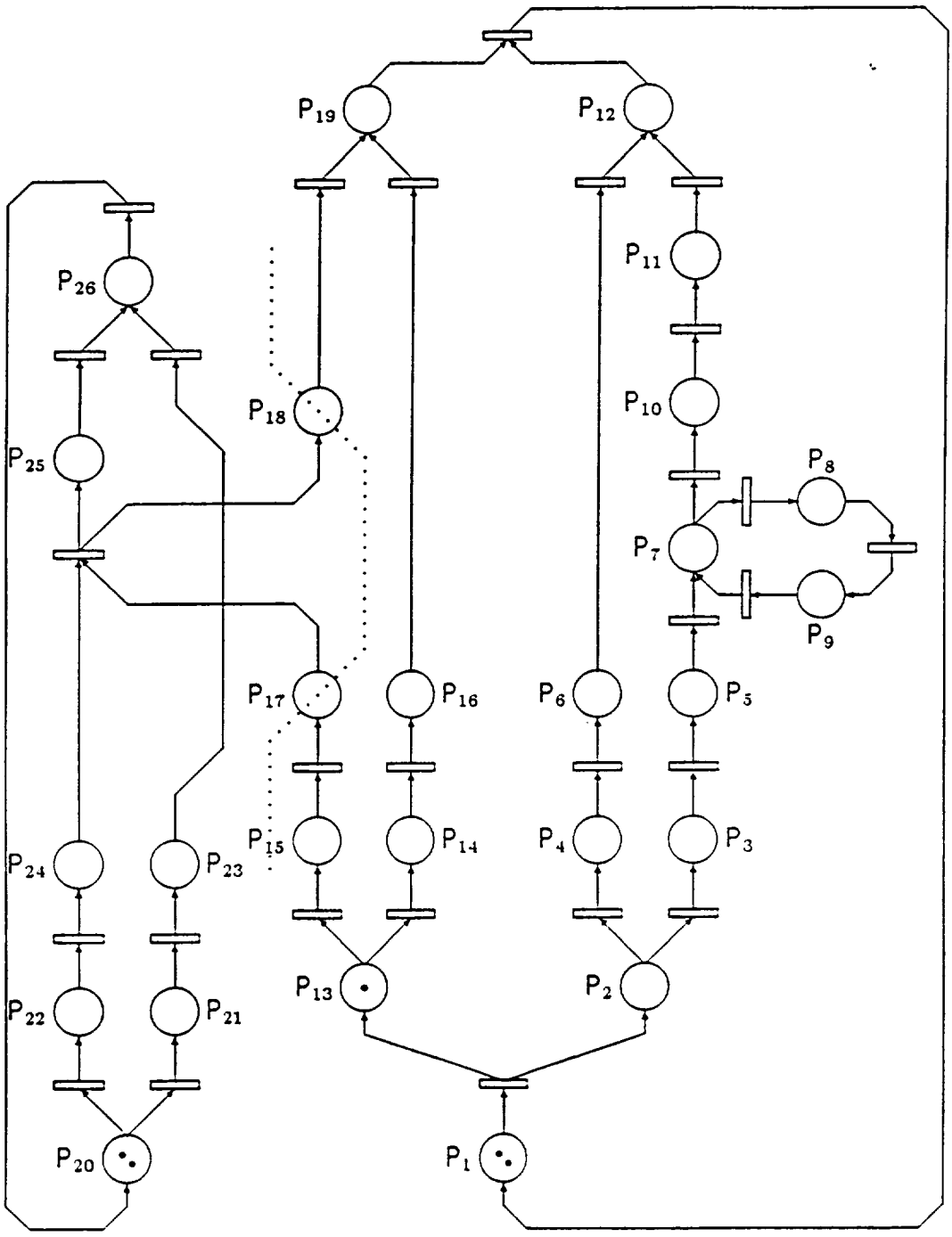


Figure 8.1: PN Model of Dataflow Graph

weight vector is obtained:

$$\underline{w} = \begin{bmatrix} \alpha \\ \alpha-\beta \\ \alpha-\beta \\ \alpha-\beta \\ \alpha-\beta \\ \alpha-\beta \\ \alpha-\beta \\ \alpha-\beta \\ \alpha-\beta \\ \alpha-\beta \\ \alpha-\beta \\ \beta \\ \beta \\ \beta \\ \beta \\ \beta \\ \beta \\ \beta \\ \gamma \\ \gamma \\ \gamma \\ \gamma \\ \gamma \\ \gamma \\ \gamma \end{bmatrix} \tag{8.1}$$

Using (8.1), the weight vector normalization equation, (6.25), and the initial marking of the model, the following constraint is obtained:

$$\begin{aligned} 2\alpha + 1\beta + 3\gamma &= 1 \\ \gamma &= \frac{1}{3}(1 - 2\alpha - \beta) \end{aligned} \tag{8.2}$$

The normalized weight vector is given by substituting (8.2) into (8.1) and choosing values of α and β that result in strictly positive elements. For each trial (set of

α	β											
	$\frac{1}{11}$	$\frac{2}{11}$	$\frac{3}{11}$	$\frac{1}{13}$	$\frac{2}{13}$	$\frac{3}{13}$	$\frac{1}{17}$	$\frac{2}{17}$	$\frac{3}{17}$	$\frac{1}{19}$	$\frac{2}{19}$	$\frac{3}{19}$
$\frac{1}{11}$	–	–	–	5.16e13	–	–	4.99e10	–	–	1.38e10	–	–
$\frac{2}{11}$	3.40e07	–	–	4.34e05	2.42e08	–	9.16e05	1.64e05	7.49e16	1.35e06	1.64e05	7.35e09
$\frac{3}{11}$	1.70e06	7.09e06	–	4.34e05	1.64e05	2.52e08	9.16e05	1.64e05	1.64e05	1.35e06	1.64e05	1.64e05
$\frac{1}{13}$	–	–	–	–	–	–	8.91e12	–	–	8.14e11	–	–
$\frac{2}{13}$	3.72e07	–	–	1.69e08	–	–	9.16e05	1.64e05	–	1.35e06	1.64e05	–
$\frac{3}{13}$	3.15e05	3.70e07	–	3.88e06	1.34e07	–	9.16e05	1.64e05	1.64e05	1.35e06	1.64e05	1.64e05
$\frac{1}{17}$	–	–	–	–	–	–	–	–	–	6.35e17	–	–
$\frac{2}{17}$	3.34e10	–	–	2.32e09	–	–	3.21e09	–	–	1.35e06	4.29e09	–
$\frac{3}{17}$	3.15e05	–	–	4.34e05	2.32e09	–	3.28e07	1.48e08	–	1.35e06	1.64e05	4.29e09
$\frac{1}{19}$	–	–	–	–	–	–	–	–	–	–	–	–
$\frac{2}{19}$	–	–	–	2.85e10	–	–	6.62e08	–	–	1.20e10	–	–
$\frac{3}{19}$	3.15e05	–	–	4.34e05	1.12e18	–	9.16e05	6.61e08	–	9.52e07	4.87e08	–

Table 8.1: Dataflow Model State-Space Size

values for α and β), the vector is processed by the state-space size estimation algorithm described in Section 6.5. Table 8.1 shows the results for various α and β. Missing entries indicate illegal combinations of α and β (i.e., the weight vector produced was non-positive). The speed of the algorithm allowed the table to be computed in approximately 30 seconds. The upper bound nature of the estimation allows us to simply choose the smallest value in the table as the state-space size estimate, i.e., 1.64e05 (the exact value is 164052). Out of the 144 trials, 65 were legal, and this minimum state-space size resulted from 14 of them (another 7 estimates were within a factor of three, and an additional 11 estimates were within an order of magnitude). From SPNP, [Ciar88], the model actually contains 164052 states.

The state-space size of this problem is typical, and may or may not warrant reduction to obtain approximate performance results. However, to continue the case study, we decide to apply the reduction algorithm. We note that the size estimation

is the key to a completely automated reduction algorithm, and continue using the size estimation to support a user-selected model division.

Figures 8.2 and 8.3 illustrate the subnets, $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively, obtained from applying the division identified in Figure 8.1. The redistribution of tokens is necessary to match the subnet properties with that of the original model (this process is explained in [Jung92]). Notice that one area of $\mathcal{S}_1$ has been contracted into a single transition, T_{fea}. This was done in [Jung92] to illustrate another performance approximation method. The two subnets are aware of one another via the throughput performances of transitions T_1 and T_2. To determine if this will be an effective division, the state-space sizes of the two subnets are estimated.

Considering subnet $\mathcal{S}_1$ in Figure 8.2, with the places in ascending order, a weight vector of

$$\underline{w} = \begin{bmatrix} \alpha \\ \alpha-\beta \\ \alpha-\beta \\ \alpha-\beta \\ \alpha-\beta \\ \beta \\ \beta \\ \beta \\ \beta \\ \beta \\ \beta \\ \beta \end{bmatrix} \tag{8.3}$$

is obtained, along with the following normalization

$$\begin{aligned} 2(\alpha-\beta)+3\beta &= 1 \\ \beta &= 1-2\alpha \end{aligned} \tag{8.4}$$

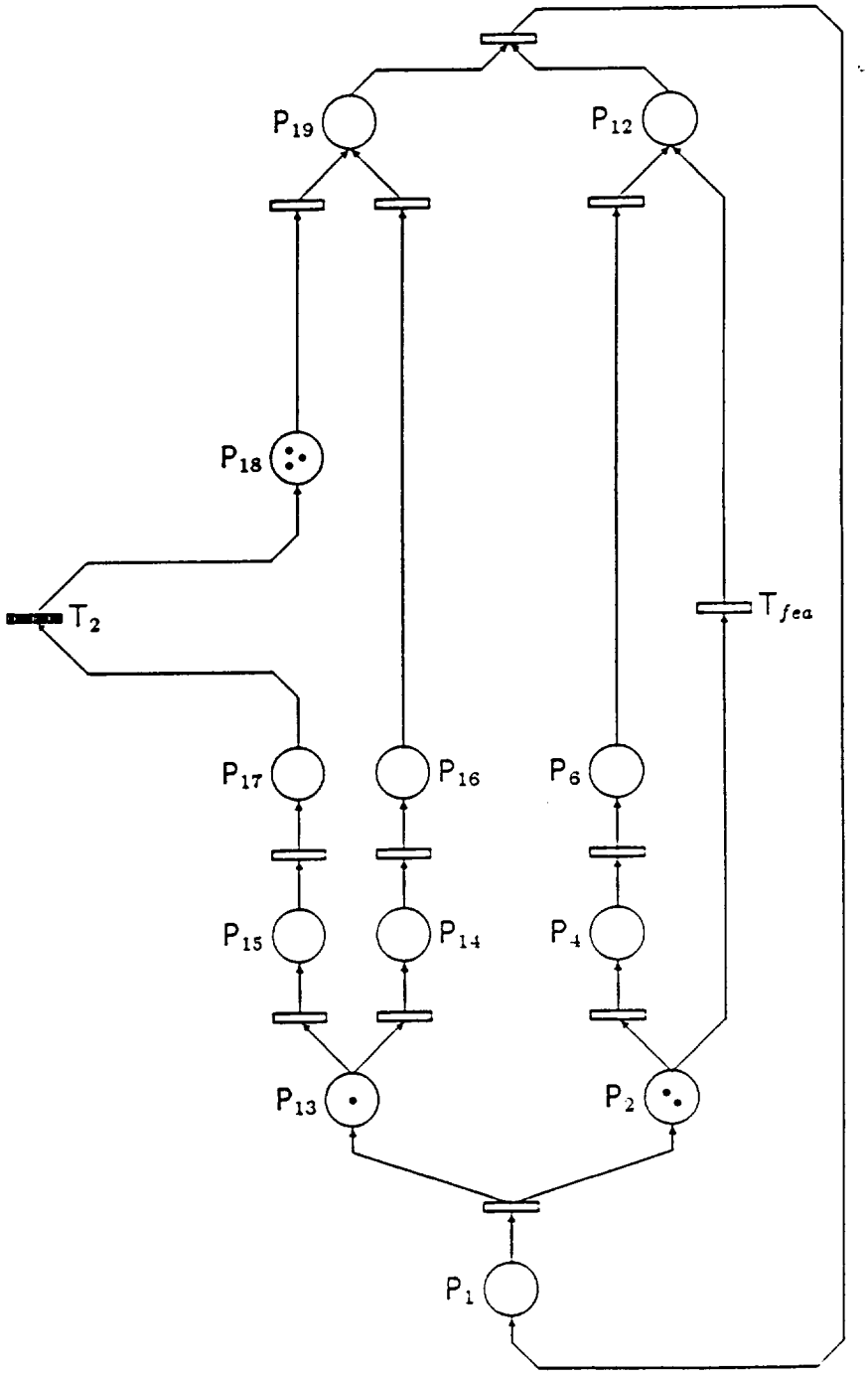


Figure 8.2: PN Model of Subnet 1

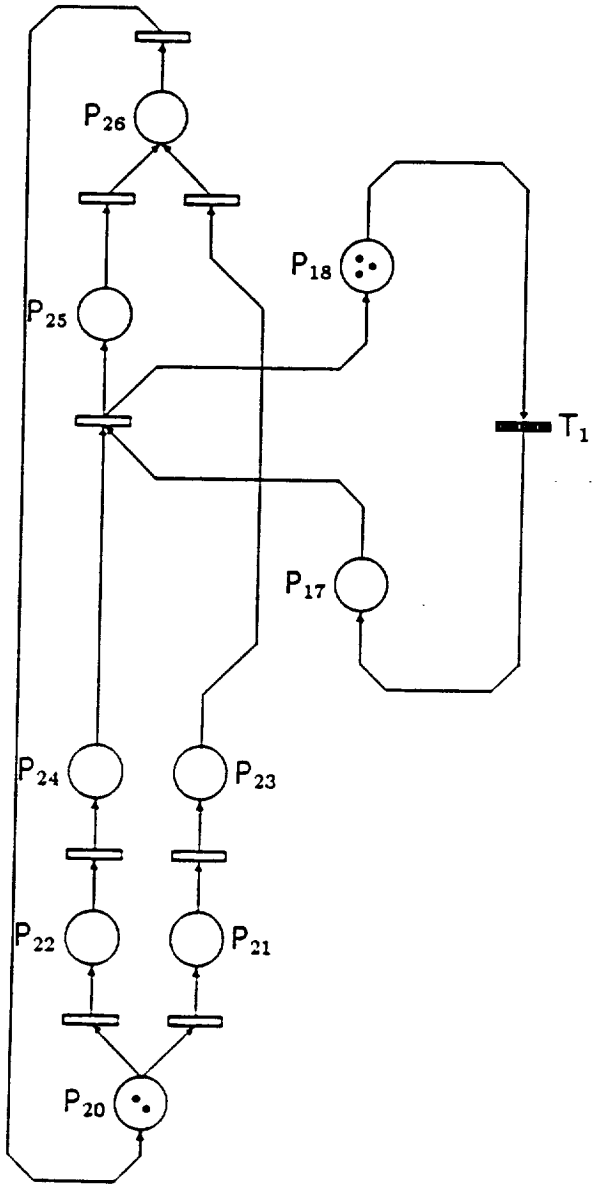


Figure 8.3: PN Model of Subnet 2

α											
$\frac{2}{5}$	$\frac{3}{7}$	$\frac{3}{8}$	$\frac{4}{9}$	$\frac{4}{10}$	$\frac{4}{11}$	$\frac{5}{11}$	$\frac{5}{12}$	$\frac{5}{13}$	$\frac{6}{13}$	$\frac{5}{14}$	$\frac{6}{14}$
1461	3196	760	7634	1461	605	17047	1244	362	35169	660	3196

Table 8.2: Subnet 1 State-Space Size

which substituted into (8.3) produces

$$\underline{w} = \begin{bmatrix} \alpha \\ 3\alpha - 1 \\ 3\alpha - 1 \\ 3\alpha - 1 \\ 3\alpha - 1 \\ 1 - 2\alpha \\ 1 - 2\alpha \\ 1 - 2\alpha \\ 1 - 2\alpha \\ 1 - 2\alpha \\ 1 - 2\alpha \\ 1 - 2\alpha \end{bmatrix} \tag{8.5}$$

Elements two through five require $\alpha > 1/3$, while the bottom elements require $\alpha < 1/2$. The first dozen rational fractions meeting these inequalities are used for the trials, and the state-space size estimation results are given in Table 8.2. The minimum estimate is 362 states, a 14.9 percent error above the actual state-space size of 315.

Similarly, we consider $\mathcal{S}_2$. With the places in Figure 8.3 taken in ascending order, a weight vector of

$$\underline{w} = \begin{bmatrix} \alpha \\ \alpha \\ \beta \\ \beta \\ \beta \\ \beta \\ \beta \\ \beta \\ \beta \\ \beta \end{bmatrix} \tag{8.6}$$

β											
$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{2}{5}$	$\frac{1}{6}$	$\frac{2}{6}$	$\frac{1}{7}$	$\frac{2}{7}$	$\frac{3}{7}$	$\frac{1}{8}$	$\frac{2}{8}$	$\frac{3}{8}$
255	329	1287	198	1036	255	1828	330	225	5598	329	125

Table 8.3: Subnet 2 State-Space Size

is obtained. The normalization of

$$\begin{aligned} 3\alpha + 2\beta &= 1 \\ \alpha &= \frac{1}{3}(1 - 2\beta) \end{aligned} \tag{8.7}$$

is substituted into (8.6), producing

$$\underline{w} = \begin{bmatrix} \frac{1}{3}(1-2\beta) \\ \frac{1}{3}(1-2\beta) \\ \beta \\ \beta \\ \beta \\ \beta \\ \beta \\ \beta \\ \beta \end{bmatrix} \tag{8.8}$$

The first two elements require $\beta < 1/2$, while the remainder of the vector requires only the positivity of β. Table 8.3 shows the estimation results for twelve values of β satisfying this constraint. The minimum estimate is 125 states, a 11.6 percent error above the SPNP computed state-space size of 112.

Thus, state-space size estimation was useful in determining the applicability of a particular model to reduction/approximation, and was utilized at the decision point of the analysis.

8.3 Bottom-Up Estimation of a Flexible Manufacturing System

Figure 8.4 depicts a PN model used to evaluate the performance of a Flexible Manufacturing System (FMS). The FMS consists of a Material Handling System

(MHS) and three workcells, (A, B, and C), to process the parts. Parts enter the FMS via an arrival process and wait to be processed by one of the three servers in the MHS. A resource place in the MHS indicates the availability of these servers. Once serviced by the MHS, a part is routed to one of the three workcells.

The capacity of each workcell is limited by its number of palates. Provided one of the four palates is available in a particular workcell, a part can be routed to this workcell and is mounted for machining. The part then waits for the availability of one of the workcell's two servers, which can operate simultaneously. The service time of each workcell is modeled by the remaining places and transitions, (although workcells A and B have the same structure, in the performance model they contain different transition rates, and therefore have different service times).

Once the part is machined, it is routed to a common exit process, and the palate and workcell are again available for utilization. The steady-state behavior of the FMS is provided by closing the part exit and entry paths, and limiting the number of parts in the system to a maximum of ρ. Without this limit on the number of parts, the model would be unbounded. A similar model was analyzed in [Yao85]. The subnets and interconnections for this model are readily apparent, and are discussed in turn.

The MHS is a resource blocking subnet. With a single-place queue having an sc-function of unity, (7.13)) the sc-function for the MHS subnet is obtained from (7.8):

$$\tilde{s}_{\mathrm{mhs}}(\tau) = \begin{cases} 1 & \tau = 0 \\ 3 & \tau = 1 \\ 6 & \tau = 2 \\ 10 & \tau = 3 \\ 10 & \tau > 3 \end{cases} \tag{8.9}$$

Workcell A is composed of three interconnections: the innermost is a choice between two subnets (each of one place); the middle interconnection is a resource block with queue; and the outermost interconnection is a resource block without

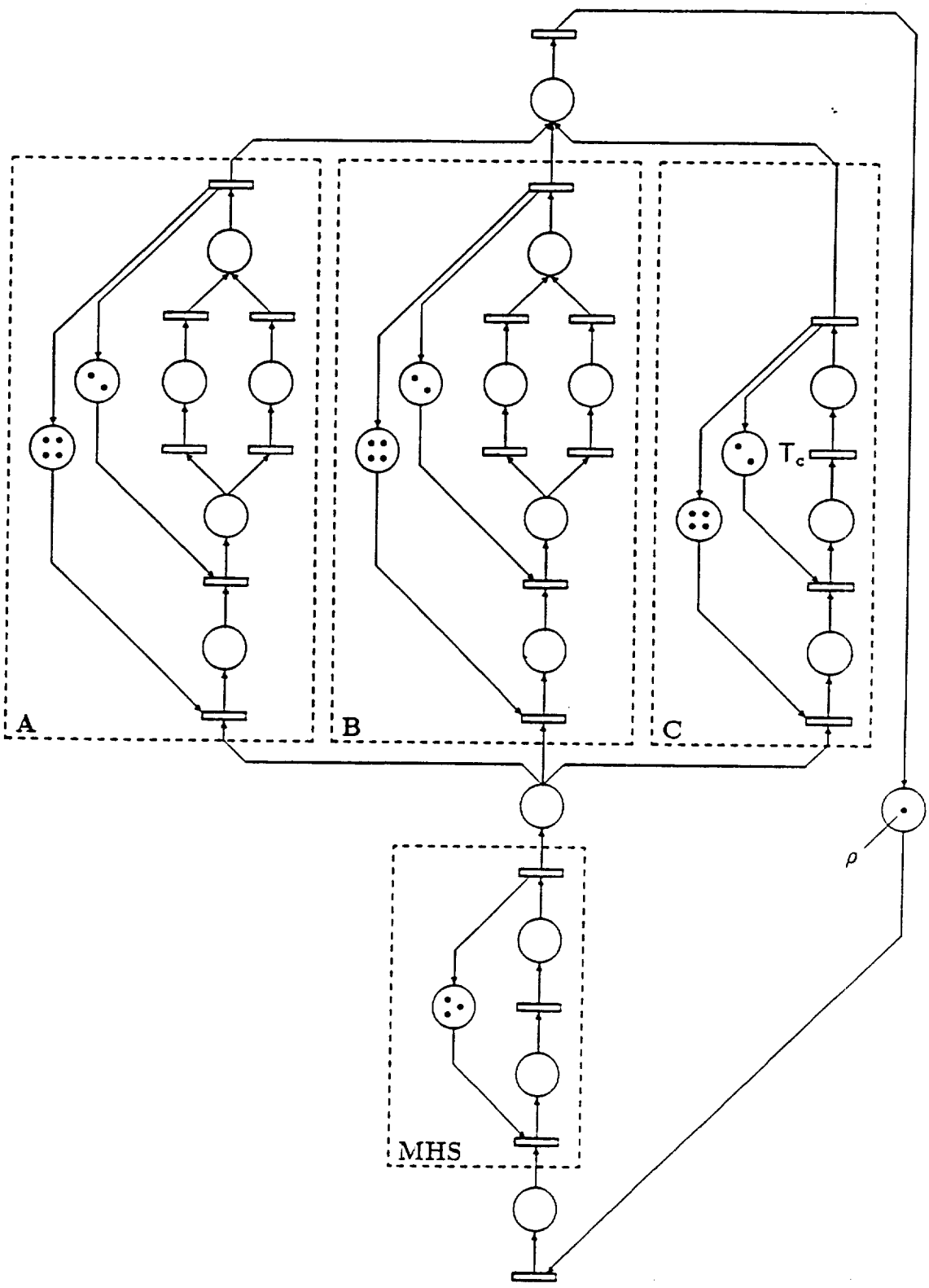


Figure 8.4: PN Model of a Flexible Manufacturing System

queue. From (7.5) or (7.16), the following sc-function is obtained for the inner choice:

$$\tilde{s}_{\text{a-in}}(\tau) = \begin{cases} 1 & \tau = 0 \\ 4 & \tau = 1 \\ 10 & \tau = 2 \\ 20 & \tau = 3 \\ 35 & \tau = 4 \\ 56 & \tau = 5 \\ 84 & \tau = 6 \\ \vdots & \end{cases} \tag{8.10}$$

Interconnecting (8.10) with the middle resource block, (7.8), produces

$$\tilde{s}_{\text{a-mid}}(\tau) = \begin{cases} 1 & \tau = 0 \\ 5 & \tau = 1 \\ 15 & \tau = 2 \\ 15 & \tau > 2 \end{cases} \tag{8.11}$$

Finally, the outer resource block, (7.7), produces an sc-function of

$$\tilde{s}_{a}(\tau) = \begin{cases} 1 & \tau = 0 \\ 5 & \tau = 1 \\ 15 & \tau = 2 \\ 15 & \tau = 3 \\ 15 & \tau = 4 \\ 0 & \tau > 4 \end{cases} \tag{8.12}$$

The sc-function for Workcell B is the same as that for A. For Workcell C, there are two interconnections: inside, a resource block with queue contains a two-place serial line; outside, a resource block without queue contains the inner subnet. From (7.12), the sc-function for the two-place serial line is

$$\tilde{s}_{\text{c-serial}}(\tau) = {}^{2}X_{\tau} = \tau + 1 \tag{8.13}$$

which, connected with the inner resource block, (7.8), produces,

$$\tilde{s}_{\text{c-in}}(\tau) = \begin{cases} 1 & \tau = 0 \\ 3 & \tau = 1 \\ 6 & \tau = 2 \\ 6 & \tau > 2 \end{cases} \tag{8.14}$$

With the outer resource block, (7.7), the sc-function for Workcell C is

$$\tilde{s}_c(\tau) = \begin{cases} 1 & \tau = 0 \\ 3 & \tau = 1 \\ 6 & \tau = 2 \\ 6 & \tau = 3 \\ 6 & \tau = 4 \\ 0 & \tau > 4 \end{cases} \tag{8.15}$$

The three-way choice among the workcells requires the following function to produce an aggregate sc-function, from (7.6),

$$\tilde{s}_{abc}(\tau) = \sum_{\underline{v} \in I_4(\tau)} \tilde{s}_a(v_1)\tilde{s}_b(v_2)\tilde{s}_c(v_3)(v_4 + 1) \tag{8.16}$$

Using (8.16) and the sc-functions given in (8.12), which serves for both Workcells A and B, and (8.15), an sc-function for the aggregation of the three workcells is produced:

$$\tilde{s}_{abc}(\tau) = \begin{cases} 1 & \tau = 0 \\ 15 & \tau = 1 \\ 120 & \tau = 2 \\ 636 & \tau = 3 \\ 2\,493 & \tau = 4 \\ 7\,635 & \tau = 5 \\ 19\,092 & \tau = 6 \\ 40\,134 & \tau = 7 \\ 72\,831 & \tau = 8 \\ 116\,553 & \tau = 9 \\ 168\,375 & \tau = 10 \\ 224\,247 & \tau = 11 \\ 281\,469 & \tau = 12 \\ 338\,691 & \tau = 13 \\ 395\,913 & \tau = 14 \\ 453\,135 & \tau = 15 \\ 510\,357 & \tau = 16 \\ 567\,579 & \tau = 17 \\ 624\,801 & \tau = 18 \\ 682\,023 & \tau = 19 \\ 739\,245 & \tau = 20 \\ \vdots & \end{cases} \tag{8.17}$$

Note the rapid growth in the state-space size until $\tau = 12$, after which incrementing τ

by one consistently increments the state-space size by 57222. The change in the state-space growth rate is due to the saturation of the combined resources in the three workcells.

The MHS and the workcells are connected in series. Thus, from (7.1), we have

$$\tilde{s}_{\mathbf{mhs-abc}}(\tau) = \sum_{i=0}^{\tau} \tilde{s}_{\mathbf{mhs}}(i)\tilde{s}_{\mathbf{abc}}(\tau - i) \tag{8.18}$$

which results in

$$\tilde{s}_{\mathbf{mhs-abc}}(\tau) = \begin{cases} 1 & \tau = 0 \\ 18 & \tau = 1 \\ 171 & \tau = 2 \\ 1\,096 & \tau = 3 \\ 5\,281 & \tau = 4 \\ 20\,290 & \tau = 5 \\ 64\,675 & \tau = 6 \\ 175\,870 & \tau = 7 \\ 416\,785 & \tau = 8 \\ 875\,770 & \tau = 9 \\ 1\,656\,280 & \tau = 10 \\ 2\,858\,260 & \tau = 11 \\ 4\,559\,560 & \tau = 12 \\ 6\,807\,430 & \tau = 13 \\ 9\,622\,120 & \tau = 14 \\ 13\,009\,030 & \tau = 15 \\ 16\,968\,160 & \tau = 16 \\ 21\,499\,510 & \tau = 17 \\ 26\,603\,080 & \tau = 18 \\ 32\,278\,870 & \tau = 19 \\ 38\,526\,880 & \tau = 20 \\ \vdots & \end{cases} \tag{8.19}$$

A state-space size estimate for the entire model, $S(\rho)$, is obtained by considering the ρ parts offered:

$$S(\rho) = \sum_{i=0}^{\rho} \tilde{s}_{\mathbf{mhs-abc}}(i) \tag{8.20}$$

ρ	$S(\rho)$	$S_{\text{aug}}(\rho)$
0	1	1
1	19	23
2	190	276
3	1 286	2 204
4	6 567	13 037
5	26 857	60 557
6	91 532	229 962
7	267 402	735 332
8	684 187	2 027 437
9	1 559 957	4 913 997
10	3 216 237	10 644 947
11	6 074 497	20 916 007
12	10 634 057	37 779 897
13	17 441 487	63 509 837
14	27 063 607	100 473 547
15	40 072 637	151 063 947
16	57 040 797	217 673 957
17	78 540 307	302 696 497
18	105 143 387	408 524 487
19	137 422 257	537 550 847
⋮		

Table 8.4: Comparison of State-Space Sizes for the Nominal and Augmented Models

which is shown in Table 8.4. For $\rho \leq 6$, the SPNP results agreed exactly with $S(\rho)$; larger ρ required lengthy (exponential complexity) SPNP run-times. The state-space size estimation is exact because no approximated or estimated sc-functions were needed.

8.4 Estimation With Non-Exponential Transition Augmentation

The effect of model augmentation on state-space size is illustrated by replacing a transition in Figure 8.4 by a non-exponential transition approximation (see Chapter 4). Specifically, transition T_c in Workcell C is replaced by a s-transition containing a 5-place serial line. With no other information, one would conclude a

worst case state-space size increase: the number of states doubles for each place in the serial line, i.e., a 32-fold increase. Such a dramatic increase would indeed occur if every state in the original model enabled T_c, thereby permitting tokens to flow through the serial line in the augmented model. Often, however, only a small fraction of the states enable any given transition.

To determine the impact of this augmentation on the state-space size, the above analysis is re-entered at (8.13), where the sc-function for Workcell C is being computed. Instead of a two-place serial line, the internal portion of Workcell C is now a six-place serial line, producing

$$\tilde{s}_{c-serial}(\tau) \;=\; {}^{6}X_{\tau} \;=\; \frac{(\tau+5)!}{\tau!5!}$$

Connected with the inner and outer resource blocks, the new sc-function for Workcell C, (8.15), is

$$\tilde{s}_c(\tau) \;=\; \begin{cases} 1 & \tau = 0 \\ 7 & \tau = 1 \\ 28 & \tau = 2 \\ 28 & \tau = 3 \\ 28 & \tau = 4 \\ 0 & \tau > 4 \end{cases} \tag{8.21}$$

Continuing the analysis with (8.16), (8.18), and (8.20), the estimate for the entire augmented model is obtained, $S_{aug}(\rho)$, which is shown in Table 8.4. Comparing $S(\rho)$ and $S_{aug}(\rho)$, the state-space size increase is about 3-fold for small ρ, and 4-fold for large ρ. Augmenting the model with the non-exponential transition approximation would provide more accurate performance results at the expense of increased computation time. State-space size estimation provides data to evaluate this trade-off.

8.5 Summary

This chapter has presented two systems to illustrate the application of the state-space size estimation methods. Each of the systems considered has appeared in the published literature.

The applicability of top-down state-space size estimation to size reduction algorithms was demonstrated by considering the work of Jungnitz, [Jung92], and the system used for the case study of his research. Size estimation provided information for determining the necessity of the reduction algorithm and for confirming the effectiveness of the model division. Automation of the size reduction algorithm is made possible by the state-space size estimation capabilities.

Modular systems permit highly accurate bottom-up state-space size estimation. A multi-cell Flexible Manufacturing System (FMS) illustrated the complexity of model that can be analyzed. A non-exponential transition approximation was used to demonstrate the usefulness of the state-space size estimation to evaluate trade-offs between model accuracy and solution complexity.

CHAPTER 9
CONCLUSIONS

This chapter concludes the thesis with the following three sections. First, the contributions of this research are summarized and specific publications the author has had accepted in transactions and refereed conferences are identified. The second section describes the application of this research to the robotic Testbed at the Center for Intelligent Robotic Systems for Space Exploration (CIRSSE). Finally, future research directions are identified and briefly discussed.

9.1 Contributions

The focusing goal of this research was to provide theory and algorithms for estimating the state-space size of PNs and improving performance analysis capabilities. The following paragraphs summarize the contributions made in these areas.

Chapter 4 describes a mechanism to approximate non-exponential transitions by a moment matching algorithm. The construction of the approximation model draws exclusively on legal Generalized Stochastic PN (GSPN) components, thereby permitting standard analysis of the non-exponential performances. Analysis of the entropy and performance of the approximation model shows that it is close to the theoretic optimal, i.e., the Maximum Entropy Density Function.

State-space size estimation theory and algorithms are developed in Chapters 5 through 7. Since no prior work had been published in this area, the problem is characterized and analyzed, producing two distinct solution methods. The top-down method benefits from broad applicability, while the bottom-up method benefits from exact estimation at the sacrifice of requiring modular model construction. Several examples demonstrate the advantages and short-comings of each method.

As an indication of recognition from the international research community, various topics and results of this thesis have been accepted into publication:

- *IEEE Transactions on Systems, Man, and Cybernetics*: "Applying Generalized Stochastic Petri Nets to Manufacturing Systems Containing Non-Exponential Transitions," [Wats91b].
- *IEEE Conference on Decision and Control*: "Entropy Analysis of Generalized Stochastic Petri Net S-Transitions," [Wats91b].
- *IEEE Robotics and Automation Conference*: "State-Space Size Estimation of Conservative Petri Nets," [Wats91b].
- *IEEE Robotics and Automation Conference*: "Methods for Estimating the State-Space Size of Petri Nets," [Wats92a].
- *IEEE Robotics and Automation Conference*: "Applying Generalized Stochastic Petri Nets to Manufacturing Systems Containing Non-Exponential Transitions," [Wats91a]

Additionally, the following submissions have been made, but as of this writing, their acceptance for publication has not been confirmed:

- *IEEE Transactions on Robotics and Automation*: "State-Space Size Estimation of Petri Nets: A Top-Down Perspective," [Wats92d].
- *IEEE Transactions on Robotics and Automation*: "State-Space Size Estimation of Petri Nets: A Bottom-Up Perspective," [Wats92e].
- *IEEE Robotics and Automation Conference*: "A Bottom-Up Algorithm for State-Space Size Estimation of Petri Nets," [Wats93].

9.2 Application to a Robotic Testbed

The theoretical results produced by this research will aid in the design, development, analysis, and control of large scale Discrete Event Dynamic Systems (DEDSs). These systems typically undergo the following life-cycle:

1. Planning
2. Modeling
3. Analysis
4. Evaluation
5. Implementation
6. Control of execution
7. Augmentation/modification

where, the first three stages can be iterated depending on the evaluation that takes place. The Center for Intelligent Robotic Systems for Space Exploration (CIRSSE) houses a dual arm robotic testbed. This testbed, which contains several cameras and other sensing devices, two PUMA robots, a custom-built platform, and a dozen computer workstations, represents a large scale DEDS. In addition to serving as a facility for research in robotic control, vision and sensing, and machine intelligence, [Sari88a], the CIRSSE Testbed also is used for research in systems modeling and integration. The individual subsystems of the testbed are at different stages of existence. Some subsystems in the machine hierarchy, [Sari88b], including the coordination level, [Wang90], and the representation and planning level, [Cao91], have already utilized PN modeling, analysis, and control, making them candidate subsystems for this research. Other subsystems, including the client interface, and the supervised autonomy

workstation, are in their early design stages, and are potentially developed and/or controlled with PN theory.

The most direct applications of this theory to the CIRSSE Testbed are for the analysis, evaluation, control, and augmentation of the PN based subsystems. The aggregation of the PN models for the various subsystems will undoubtedly result in an enormous state-space. Figure 9.1 illustrates a PN model for the coordinator used in a strut insertion experiment at CIRSSE. Because of its size, the model is divided into three pieces; the repeated transitions, T_1, T_2, and T_3, denote where the pieces are joined. The coordinator manages the resources needed to run the experiment, (e.g., camera, robot, gripper), and sequences various actions depending on sensor values (e.g., move robot, locate strut with camera, plan path). The experiment is started by firing T_1. A more complete description of the model is provided in [Jung92].

Figure 9.1 represents a portion of the testbed system that may be modeled by PNs. State-space size estimation can be applied to this subsystem (and others, as they become available) to determine the size of scheduling, [Lee92], and performance analysis problems. Given the estimated size of the state-space, a reduced model may be needed to compute the performance data, [Jung92]. Automated (or semi-automated) error recovery for the CIRSSE Testbed is still in its early design stages. This functionality will reside (at least in some part) at the Coordination level, where PN control is being used. Since designing a system to handle all possible errors is a formidable task, an adaptable system is needed to handle errors as they occur, [Zhou90]. The examples and case studies presented throughout this thesis have illustrated the application of state-space size estimation to these problems.

As a final example, state-space size estimation is applied to the PN in Figure 9.1. The model possesses some modularity: clearly the pieces are modules connected in series, and additionally piece (a) has internal modularity. Although the model was designed for a single robot experiment, the PN can be applied to dual arm experiments

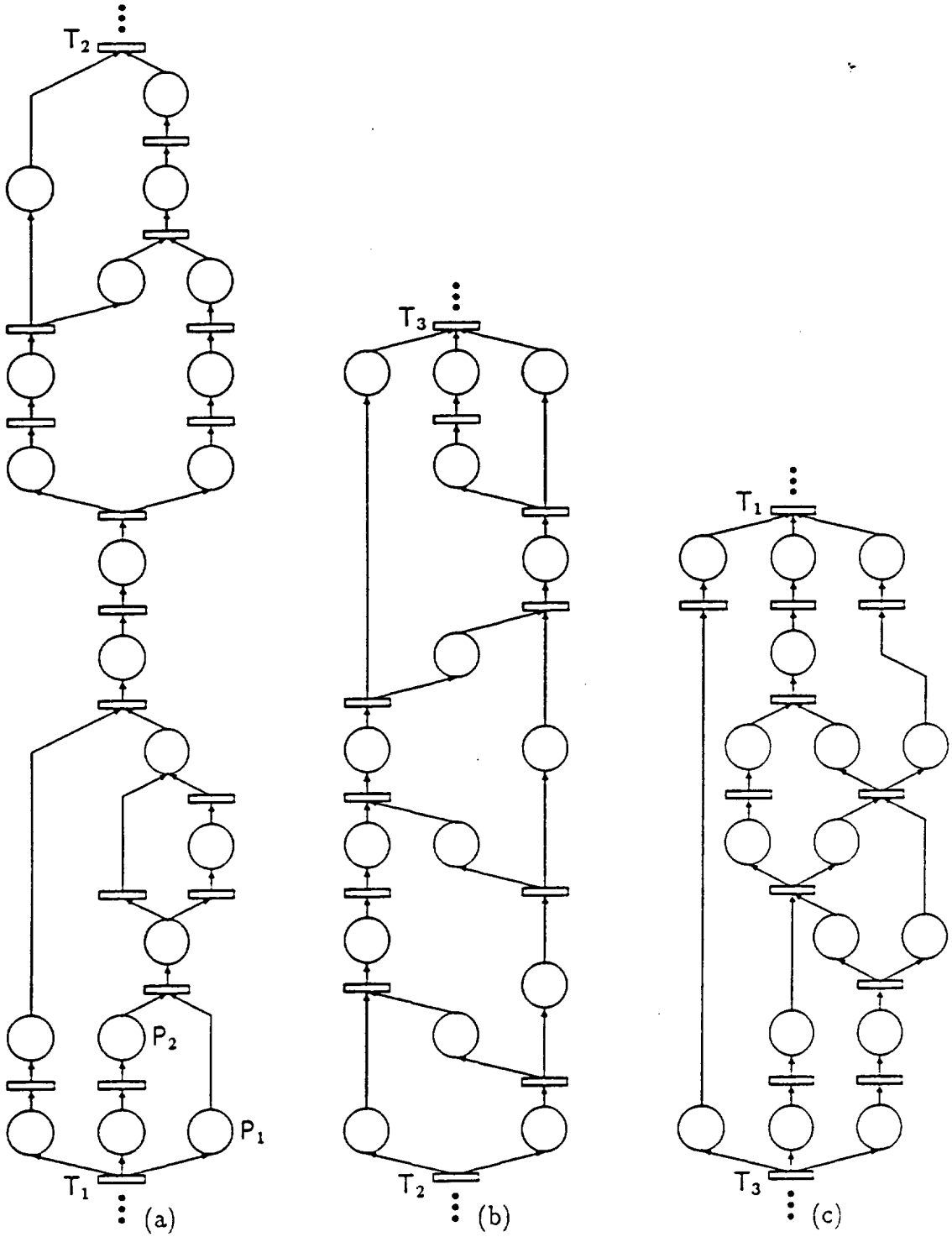


Figure 9.1: PN Model for CIRSSE Testbed Strut Insertion Experiment

in the testbed. Similar to the analysis in earlier chapters, top-down state-space size estimation is used to determine the sc-functions for model pieces (b) and (c), and bottom-up estimation is applied to the aggregate model. For the single and dual robot cases, the number of estimated states are 134 and 6093 respectively. Using SPNP, [Ciar88], the actual number of states for the two cases are 67 and 2103, indicating that the estimation errors are within a half-order of magnitude. Improved estimation can be achieved by removing from the model constructs that adversely affect the estimation algorithm (see Sections 6.5.2 and 6.8). Consider place P_1 in Figure 9.1, which is redundant in the sense that its removal does not affect the state evolution, hence, the state-space size. Since a token will arrive at P_1 prior to one arriving at P_2, the output transition of P_2 is enabled immediately—this is also the case if P_1 was not present in the model. Removing P_1, and other places possessing this same property, produces state-space size estimates of 97 and 3217 states, for the single and dual arm cases, respectively. The model for the strut insertion was different than those considered in the development of the state-space size estimators (i.e., instead of having a relatively few number of places and many states, it has a relatively large number of places for the few states it generates). The above results indicate the applicability of the state-space size estimation algorithm to such models requires consideration be given to places that potentially do not contribute to the state evolution. The state-space size estimation will probably be more useful to the CIRSSE Testbed when additional subsystems are modeled and integrated.

9.3 Future Research Directions

This research has opened the area of state-space size estimation and contributed to the techniques available for non-exponential transition modeling. Several research directions can be identified:

- The non-exponential transition model can be extended to include binary choices among non-exponential densities. This would improve the applicability of the approximation to performance problems.

- Now that the state-space size estimation area has been opened, other methods and algorithms can be developed. In particular, focusing on PN subclasses may provide useful results. The subclass models have representational limitations, but could be combined modularly.

- Development of a software tool for state-space size estimation would provide a needed capability to modelers. This can be done as a stand-alone system, or incorporated into existing popular PN analysis software.

- The bottom-up state-space size estimation approach is inherently extendable via the development of additional interconnections. Additional interconnections for specific problem domains, e.g., protocols, should be considered.

- Combining bottom-up and top-down size estimation has been described to some extent by using top-down estimation to approximate the sc-functions used in the bottom-up approach. A more thorough combination will probably increase applicability to models that are neither modular or conservative.

- The development of tight lower bound estimates of the state-space size may be useful for some applications. This type of estimation would require a completely different approach than presented here.

LITERATURE CITED

[AlJa87] R. Y. Al-Jaar and A. A. Desrochers. Petri nets in automation and manufacturing. RAL Report 99, Rensselaer Polytechnic Institute, Nov 1987.

[AlJa88a] R. Y. Al-Jaar and A. A. Desrochers. A modular approach for the performance analysis of automated manufacturing systems using generalized stochastic Petri nets. RAL Report 116, Rensselaer Polytechnic Institute, July 1988.

[AlJa88b] R. Y. Al-Jaar. *Performance Evaluation of Automated Manufacturing Systems Using Generalized Stochastic Petri Nets.* PhD thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, 1988.

[AlJa90a] R. Y. Al-Jaar and A. A. Desrochers. Performance evaluation of automated manufacturing systems using generalized stochastic Petri nets. *IEEE Transactions on Robotics and Automation*, 6(6):621–639, Dec 1990.

[AlJa90b] R. Y. Al-Jaar and A. A. Desrochers. Petri nets in automation and manufacturing. In G. N. Saridis, editor, *Advances in Robotics and Automation*, volume 2. JAI Press, 1990.

[Alla85] H. Alla, P. Ladet, J. Martinez, and M. Silva-Suarez. Modelling and validation of complex systems by coloured Petri nets; application to a flexible manufacturing system. In *Lecture Notes in Computer Science*, volume 188, pages 15–31. New York: Springer-Verlag, 1985.

[Amma85] H. H. Ammar and R. W. Liu. Analysis of generalized Petri nets by state aggregation. In *Proceedings of the 1985 Workshop on Timed Petri Nets*, pages 88–95, Torino, Italy, July 1985. IEEE Computer Society Press.

[Balb87] G. Balbo, G. Chiola, and M. G. Roet. On the efficient construction of the tangible reachability graph of generalized stochastic Petri nets. In *Proceedings International Workshop on Petri Nets and Performance Models*, pages 136–145, Madison, WI, 1987. IEEE Computer Society Press.

[Bill85] J. Billington. On specifying performance aspects of protocol services. In *Proceedings of the 1985 Workshop on Timed Petri Nets*, pages 288–295, Torino, Italy, July 1985. IEEE Computer Society Press.

[Bjan91] A. Bjanes. A distributed Petri net controller for a dual arm testbed. Master's thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, 1991.

[Bond76] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications.* Elsevier, New York, 1976.

[Brue85] S. C. Bruell and S. Ghanta. Throughput bounds for generalized Petri net models. In *Proceedings of the 1985 Workshop on Timed Petri Nets*, pages 250–261, Torino, Italy, July 1985.

[Brun85] G. Bruno and P. Bigla. Performance evaluation and validation of tool handling in flexible manufacturing systems using Petri nets. In *Proceedings of the 1985 Workshop on Timed Petri Nets*, pages 64–71, Torino, Italy, July 1985. IEEE Computer Society Press.

[Cao91] T. Cao and A. C. Sanderson. Task sequence planning in a robot workcell using AND/OR nets. CIRSSE Report 94, Rensselaer Polytechnic Institute, Troy, NY, June 1991.

[Chen89] P. Chen, S. Bruell, and G. Balbo. Alternative methods for incorporating non-exponential distributions into stochastic timed Petri nets. In *International Workshop on Petri Nets and Performance Models*, pages 187–197, 1989. IEEE Computer Society Press.

[Chio87] G. Chiola. *GreatSPN Users' Manual.* Universita degli Studi di Torino. Sep 1987.

[Ciar88] G. Ciardo. *Manual for the SPN Package.* Duke University, July 1988.

[Ciar89] G. Ciardo. *Analysis of Large Stochastic Petri Net Models.* PhD thesis, Computer Science Department, Duke University, Durham, NC, 1989.

[Ciar90] G. Ciardo and K. S. Trivedi. Solution of large GSPN models. In *First International Workshop on the Numerical Solution of Markov Chains.* Jan 1990.

[Cloc84] W. F. Clocksin and C. S. Mellish. *Programming in Prolog.* New York: Springer-Verlag, 1984.

[Colo90] J. M. Colom and M. Silva. Convex geometry and semiflows in P/T nets: A comparative study of algorithms for computation of minimal p-semiflows. In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1990*, pages 79–112. Springer-Verlag, 1990.

[Cox55] D. R. Cox. A use of complex probabilities in the theory of stochastic processes. *Proceedings Cambridge Philosophical Society*, (51):313–319, 1955.

[Dows73] D. C. Dowson and A. Wragg. Maximum-entropy distributions having prescribed first and second moments. *IEEE Transactions on Information Theory*, 19(9):689–693, Sep 1973.

[Duga84] J. B. Dugan, K. Trivedi, R. Geist, and V. Nicola. Extended stochastic Petri nets: Applications and analysis. In E. Gelenbe, editor, *PERFORMANCE '84, Models of Computer System Performance*, pages 507–519, 1984.

[Duga85] J. B. Dugan, A. Bobbio, G. Ciardo, and K. Trivedi. The design of a unified package for the solution of stochastic Petri net models. In *Proceedings of the 1985 Workshop on Timed Petri Nets*, pages 6–13, Torino, Italy, July 1985. IEEE Computer Society Press.

[Fish73] G. S. Fishman. *Concepts and Methods in Discrete Event Digital Simulation.* New York: Wiley, 1973.

[Garg85] K. Garg. An approach to performance specification of communications protocols using timed Petri nets. *IEEE Transactions on Software Engineering*, SE-11(10):1216–1225, Oct 1985.

[Gild92] K. Gildea. *A Building Block Approach for Computer Communication Protocols.* PhD thesis, Rensselaer Polytechnic Institute, Troy, NY, 1992.

[Guo90] D. Guo, F. DiCesare, and M. C. Zhou. A moment generating function based approach for evaluating arbitrary stochastic Petri nets, 1990. Working Paper, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, 1990.

[Holl85] M. A. Holliday and M. K. Vernon. A generalized Petri net model for performance analysis. In *Proceedings of the 1985 Workshop on Timed Petri Nets*, pages 181–190, Torino, Italy, July 1985.

[Holl87] M. A. Holliday and M. K. Vernon. A generalized timed Petri net model for performance analysis. *IEEE Transactions on Software Engineering*, SE-13(12):1297–1310, Dec 1987. IEEE Computer Society Press.

[Inde87] K. Indermark and H. Klaeren. Compiling Fibonacci-Like recursion. *ACM SIGPLAN Notices*, 22(6):101–108, June 1987.

[Jayn57] E. T. Jaynes. Information theory and statistical mechanics. *Physics Review*, 106(4):620–630, 1957.

[Jens81] K. Jensen. Colored Petri nets and the invariant method. *Theoretical Computer Science*, 14, 1981.

[Jung92] H. Jungnitz. *Approximation Methods for Stochastic Petri Nets.* PhD thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, 1992.

[Kapu90] J. N. Kapur. *Maximum Entropy Models in Science and Engineering.* Wiley, New York, 1990.

[Kost73] L. Kosten. *Stochastic Theory of Service Systems.* Pergamon Press, 1973.

[Lee92] D. Y. Lee. A scheduling method for flexible and automated manufacturing systems using Petri nets and hueristic search. Proposal for Doctoral Research, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, 1992.

[Lind91] C. Lindemann. An improved numerical algorithm for claculating steady-state solutions of deterministic and stochastic Petri net models. In *Proceedings of the Fourth International Workshop on Petri Nets and Performance Models*, pages 176–185, Melbourne, Australia, Dec 1991. IEEE Computer Society Press.

[Mars84a] M. A. Marsan, G. Balbo, and G. Conte. Generalized stochastic Petri nets revisited: Random switches and priorities. *ACM Transactions on Computer Systems*, 2(1):93–122, May 1984.

[Mars84b] M. A. Marsan, G. Conte, and G. Balbo. A class of generalized Petri nets for the performance evaluation of multiprocessor systems. *IEEE Transactions on Computer Science*, 2(2):93–122, May 1984.

[Mars85] M. A. Marsan, G Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. On Petri nets with stochastic timing. In *Proceedings of the 1985 Workshop on Timed Petri Nets*, pages 80–87, Torino, Italy, July 1985. IEEE Computer Society Press.

[Mars87] M. A. Marsan and G. Chiola. On Petri nets with deterministic and exponentially distributed firing times. In Grzegorz Rozenberg, editor. *Advances in Petri Nets 1987*, pages 132–145. Springer-Verlag, 1987.

[Mars89] M. A. Marsan. Stochastic Petri nets: An elementary introduction. In Grzegorz Rozenberg, editor. *Advances in Petri Nets 1989*, pages 1–29. Springer-Verlag, 1989.

[Maso53] S. J. Mason. Feedback theory—some properties of signal flow graphs. *Proceedings IRE*, 41(9):1144–1156, Sep 1953.

[Maso56] S. J. Mason. Feedback theory—further properties of signal flow graphs. *Proceedings IRE*, 44(7):920–926, July 1956.

[Matl90] *Matlab User's Guide.* The MathWorks, Inc., Massachusetts, 1990.

[Mitr82] I. Mitrani. *Simulation Techniques for Discrete Event Systems.* Cambridge University Press, 1982.

[Moll82] M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, C-31(9):913–917, 1982.

[Moll84] M. K. Molloy. *On the Integration of Delay and Throughput Measures in Distributed Operating Models.* PhD thesis, University of California, Los Angeles, CA, 1984.

[Mura89] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[Papo84] A. Papoulis. *Probability, Random Variables, and Stochastic Processes.* New York: McGraw-Hill, 1984.

[Peck91] J. Peck. A Petri net controller for distributed hierarchical systems. Master's thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, 1991.

[Pete81] J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Englewood Cliff: Prentice-Hall, 1981.

[Petr62] C. A. Petri. *Kommunikation mit Automaten.* PhD thesis, University of Bonn, West Germany, 1962.

[Rama80] C. V. Ramamoorthy and G. S. Ho. Performance evaluation of asynchronous concurrent systems using Petri nets. *IEEE Transactions on Software Engineering*, SE-6(5):440–449, Sep 1980.

[Robe84] F. S. Roberts. *Applied Combinatorics.* New York: Prentice-Hall, 1984.

[Sari88a] G. N. Saridis. Intelligent machines: Distributed versus hierarchical intelligence. CIRSSE Report 2, Rensselaer Polytechnic Institute, Troy, NY, Oct 1988.

[Sari88b] G. N. Saridis and K. P. Valavanis. Analytic design of intelligent machines. *Automatica: The Journal of IFAC*, 24(2):123–133, 1988.

[Seid89] A. Seidmann and S. Nof. Operational analysis of an autonomous assembly robotic station. *IEEE Transactions on Robotics and Automation*, 5(1):1–15, Feb 1989.

[Visw87] N. Viswanadham and Y. Narahari. Coloured Petri net models for automated manufacturing systems. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1985–1990, 1987.

[Wang90] F. Wang. *A Coordination Theory for Intelligent Machines.* PhD thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, 1990.

[Wats89] J. F. Watson, III. A comparison of performance evaluation methodologies for manufacturing systems. Master's thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, 1989.

[Wats91a] J. F. Watson, III and A. A. Desrochers. Applying generalized stochastic Petri nets to manufacturing systems containing non-exponential transition functions. In *1991 IEEE Robotics and Automation Conference*, pages 366–371, Sacramento, CA, Apr 1991.

[Wats91b] J. F. Watson, III and A. A. Desrochers. Applying generalized stochastic Petri nets to manufacturing systems containing non-exponential transition functions. *IEEE Transactions on Systems, Man, and Cybernetics*, 6:1008–1017, Sep/Oct 1991.

[Wats92a] J. F. Watson, III and A. A. Desrochers. Methods for estimating state-space size of Petri nets. In *1992 IEEE Robotics and Automation Conference*, pages 1031–1036, Nice, France, May 1992.

[Wats92b] J. F. Watson, III and A. A. Desrochers. State-space size estimation of conservative Petri nets. In *1992 IEEE Robotics and Automation Conference*, pages 1037–1042, Nice, France, May 1992.

[Wats92c] J. F. Watson, III and A. A. Desrochers. Entropy analysis of generalized stochastic Petri net s-transitions. In *Proceedings of the 31st IEEE Conference on Decision and Control*, Tuscon, AZ, Dec 1992.

[Wats92d] J. F. Watson, III and A. A. Desrochers. State-space size estimation of Petri nets: A top-down perspective. *Submitted to the IEEE Transactions on Robotics and Automation*, Aug 1992.

[Wats92e] J. F. Watson, III and A. A. Desrochers. State-space size estimation of Petri nets: A bottom-up perspective. *Submitted to the IEEE Transactions on Robotics and Automation*, Oct 1992.

[Wats93] J. F. Watson, III and A. A. Desrochers. A bottom-up algorithm for state-space size estimation of Petri nets. *Submitted to the 1993 IEEE Robotics and Automation Conference*, Atlanta, GA, May 1993.

[Wrag70] A. Wragg and D. C. Dowson. Fitting continuous probability density functions over $[0, \infty)$ using information theory ideas. *IEEE Transactions on Information Theory*, 16:226–230, Mar 1970.

[Yao85] D. D. Yao and J. A. Buzacott. Modeling the performance of flexible manufacturing systems. *International Journal on Production Research*, 23(5):945–959, 1985.

[Zeni85] A. Zenie. Colored stochastic Petri nets. In *Proceedings of the 1985 Workshop on Timed Petri Nets*, pages 262–271, Torino, Italy, 1985. IEEE Computer Society Press.

[Zhou89] M. C. Zhou and F. DiCesare. Adaptive design of Petri net controllers for error recovery in automated manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):963–973, 1989.

[Zhou90] M. C. Zhou. *A Theory for the Synthesis and Augmentation of Petri Nets in Automation.* PhD thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, 1990.

APPENDIX A
LINE SEARCH FOR MEDF PARAMETERS

This appendix derives the line search for the maximum entropy density function (MEDF) parameters. The results are used in Chapter 4 for a comparison between s-transitions and MEDF transitions.

The MEDF is given by (4.39), which is repeated here:

$$f_x(x) = \begin{cases} Ae^{\alpha x - \beta^2 x^2} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{A.1}$$

Provided the coefficient of x^2 is negative, (A.1) is proper. This is explicitly guaranteed by $-\beta^2$. Thus, without loss of generality, we assume $\beta > 0$. During the derivation, the following definitions and intermediate results will be useful. We define the error function and its complement as

$$\text{erf}(x) \stackrel{\text{def}}{=} \frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} du \tag{A.2}$$

$$\text{erfc}(x) \stackrel{\text{def}}{=} \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-u^2} du \tag{A.3}$$

and.

$$\begin{aligned} \xi &\stackrel{\text{def}}{=} \int_0^\infty e^{-(x-\alpha/2\beta)^2} dx \\ &= \int_{-\alpha/2\beta}^\infty e^{-u^2} du \\ &= \frac{\sqrt{\pi}\,\text{erfc}(-\alpha/2\beta)}{2} \end{aligned} \tag{A.4}$$

Thus. the following intermediate results are obtained:

$$\begin{aligned} \int_0^\infty e^{-(\beta x - \alpha/2\beta)^2} dx &= \frac{1}{\beta} \int_{-\alpha/2\beta}^\infty e^{-u^2} du \\ &= \xi/\beta \end{aligned} \tag{A.5}$$

and,

$$\begin{aligned}\int_0^\infty e^{\alpha x-\beta^2 x^2}dx &= \int_0^\infty e^{-(\beta x-\alpha/2\beta)^2+(\alpha/2\beta)^2}dx \\ &= e^{\alpha^2/4\beta^2}\int_0^\infty e^{-(\beta x-\alpha/2\beta)^2}dx \\ &= \frac{e^{\alpha^2/4\beta^2}\xi}{\beta} \end{aligned} \tag{A.6}$$

and furthermore,

$$\begin{aligned}\int_0^\infty xe^{-(x-\alpha/2\beta)^2}dx &= \int_0^\infty (x-\alpha/2\beta)\,e^{-(x-\alpha/2\beta)^2}dx + \int_0^\infty (\alpha/2\beta)\,e^{-(x-\alpha/2\beta)^2}dx \\ &= \int_{-\alpha/2\beta}^\infty ue^{-u^2}du \;+\; \frac{\alpha}{2\beta}\int_0^\infty e^{-(x-\alpha/2\beta)^2}dx \\ &= -\frac{1}{2}e^{-u^2}\Big|_{-\alpha/2\beta}^\infty + \frac{\alpha}{2\beta}\xi \\ &= \frac{e^{-\alpha^2/4\beta^2}}{2} + \frac{\alpha}{2\beta}\xi \\ &= \frac{\beta e^{-\alpha^2/4\beta^2}+\alpha\xi}{2\beta} \end{aligned} \tag{A.7}$$

As a first step in the line search derivation, a relationship between the density normalization, A, and the parameters α and β, is obtained from (4.40) and (A.4):

$$A = \frac{\beta e^{-\alpha^2/4\beta^2}}{\xi} \tag{A.8}$$

The mean constraint (4.41) leads to the following relationship between μ and the parameters:

$$\begin{aligned}\mu &= \int_0^\infty Axe^{\alpha x-\beta^2x^2}dx \\ \mu\xi &= \int_0^\infty (\beta x)e^{-\alpha^2/4\beta^2}e^{\alpha x-\beta^2x^2}dx \\ &= \int_0^\infty (\beta x)e^{-(\beta x-\alpha/2\beta)^2}dx \\ &= \frac{1}{\beta}\int_0^\infty ue^{-(u-\alpha/2\beta)^2}du \\ &= \frac{1}{\beta}\cdot\frac{\beta e^{-\alpha^2/4\beta^2}+\alpha\xi}{2\beta} \\ \mu &= \frac{\beta e^{-\alpha^2/4\beta^2}+\alpha\xi}{2\beta^2\xi} \end{aligned} \tag{A.9}$$

To satisfy the variance constraint, the relationship

$$\sigma^2 = E\{x^2\} - \mu^2 \tag{A.10}$$

is used. Finding a relationship for $E\{x^2\}$ produces,

$$\begin{aligned}
E\{x^2\} &= \int_0^\infty x^2 A e^{\alpha x - \beta^2 x^2} dx \\
&= \frac{1}{\xi}\int_0^\infty (\beta x^2) e^{-\alpha^2/4\beta^2} e^{\alpha x - \beta^2 x^2} dx \\
&= \frac{1}{\xi\beta}\int_0^\infty (\beta x)^2 e^{-(\beta x - \alpha/2\beta)^2} dx \\
&= \frac{1}{\xi\beta^2}\int_0^\infty u^2 e^{-(u-\alpha/2\beta)^2} du \\
&= \frac{1}{\xi\beta^2}\int_0^\infty \left[\left(-\frac{1}{2}u\right)\left(-2ue^{-(u-\alpha/2\beta)^2} + \frac{\alpha}{\beta}e^{-(u-\alpha/2\beta)^2}\right) + \frac{\alpha}{2\beta}ue^{-(u-\alpha/2\beta)^2}\right] du \\
&= \frac{1}{\xi\beta^2}\left[\left(-\frac{1}{2}ue^{-(u-\alpha/2\beta)^2}\Big|_0^\infty + \frac{1}{2}\int_0^\infty e^{-(u-\alpha/2\beta)^2} du\right) + \right. \\
&\qquad \left. \frac{\alpha}{2\beta}\int_0^\infty ue^{-(u-\alpha/2\beta)^2} du\right] \\
&= \frac{1}{\xi\beta^2}\left[\left(0 + \frac{1}{2}\xi\right) + \frac{\alpha}{2\beta}\cdot\frac{\beta e^{-\alpha^2/4\beta^2} + \alpha\xi}{2\beta}\right] \\
&= \frac{(2\beta^2 + \alpha^2)\xi + \alpha\beta e^{-\alpha^2/4\beta^2}}{4\beta^4\xi}
\end{aligned} \tag{A.11}$$

From (A.9), (A.10) and (A.11), we have

$$\begin{aligned}
\sigma^2 &= \frac{(2\beta^2 + \alpha^2)\,\xi + \alpha\beta e^{-\alpha^2/4\beta^2}}{4\beta^4\xi} - \left(\frac{\beta e^{-\alpha^2/4\beta^2} + \alpha\xi}{2\beta^2\xi}\right)^2 \\
&= \frac{(2\beta^2 + \alpha^2)\,\xi^2 + \alpha\beta\xi e^{-\alpha^2/4\beta^2}}{4\beta^4\xi^2} - \frac{\beta^2 e^{-\alpha^2/2\beta^2} + 2\alpha\beta\xi e^{-\alpha^2/4\beta^2} + \alpha^2\xi^2}{4\beta^4\xi^2} \\
&= \frac{2\beta^2\xi^2 - \beta^2 e^{-\alpha^2/2\beta^2} - \alpha\beta\xi e^{-\alpha^2/4\beta^2}}{4\beta^4\xi^2} \\
&= \frac{2\beta\xi^2 - \beta e^{-\alpha^2/2\beta^2} - \alpha\xi e^{-\alpha^2/4\beta^2}}{4\beta^3\xi^2}
\end{aligned} \tag{A.12}$$

From (A.9), we have

$$e^{-\alpha^2/4\beta^2} = \frac{\xi}{\beta}\left(2\mu\beta^2 - \alpha\right) \tag{A.13}$$

which, upon squaring, results in

$$e^{-\alpha^2/2\beta^2} = \frac{\xi^2}{\beta^2}\left(2\mu\beta^2 - \alpha\right)^2 \tag{A.14}$$

Substituting (A.13) and (A.14) into (A.12) produces

$$\begin{aligned}
\sigma^2 &= \frac{2\beta\xi^2 - \frac{\xi^2}{\beta}\left(2\mu\beta^2 - \alpha\right)^2 - \frac{\alpha\xi^2}{\beta}\left(2\mu\beta^2 - \alpha\right)}{4\beta^3\xi^2} \\
&= \frac{2\beta^2 - \left(2\mu\beta^2 - \alpha\right)^2 - \alpha\left(2\mu\beta^2 - \alpha\right)}{4\beta^4} \\
&= \frac{2\beta^2 - 2\mu\beta^2\left(2\mu\beta^2 - \alpha\right)}{4\beta^4} \\
&= \frac{1 - \mu\left(2\mu\beta^2 - \alpha\right)}{2\beta^2} \\
&= \frac{1 - 2\mu^2\beta^2 + \mu\alpha}{2\beta^2}
\end{aligned} \tag{A.15}$$

which can be solved for α:

$$\alpha = \frac{2\left(\sigma^2 + \mu^2\right)\beta^2 - 1}{\mu} \tag{A.16}$$

Substituting (A.16) into (A.4) results in

$$\xi = \frac{\sqrt{\pi}}{2}\mathrm{erfc}\left(\frac{-2\left(\sigma^2 + \mu^2\right)\beta^2 + 1}{2\beta\mu}\right) \tag{A.17}$$

which is an evaluation of ξ that depends only on β and the knowns, i.e., μ and σ. Defining the following re-occurring expression,

$$c \stackrel{\text{def}}{=} -\frac{\alpha}{2\beta} = \frac{-2\left(\sigma^2 + \mu^2\right)\beta^2 + 1}{2\beta\mu} \tag{A.18}$$

we have

$$\xi = \frac{\sqrt{\pi}}{2}\mathrm{erfc}(c) \tag{A.19}$$

Substituting (A.18) into (A.9), we have

$$\mu = \frac{e^{-c^2} - 2\xi c}{2\beta\xi} \tag{A.20}$$

which is strictly in β, and the knowns, μ and σ.

We formulate $v(\beta) = 0$ from (A.20):

$$\begin{aligned} \mu &= \frac{e^{-c^2}}{2\beta\xi} - \frac{c}{\beta} \\ \frac{\mu\beta + c}{\beta} &= \frac{e^{-c^2}}{2\beta\xi} \\ v(\beta) \stackrel{\text{def}}{=} 2(\mu\beta + c)\xi - e^{-c^2} &= 0 \end{aligned} \tag{A.21}$$

To set up a line search for the value of β solving this equation, $\nabla_\beta v$ is computed:

$$\nabla_\beta v = 2(\mu + \nabla_\beta c)\xi + 2(\mu\beta + c)\nabla_\beta \xi + 2c\nabla_\beta c \cdot e^{-c^2} \tag{A.22}$$

From (A.18), we have

$$\nabla_\beta c = -\frac{2(\sigma^2 + \mu^2)\beta^2 + 1}{2\beta^2\mu} \tag{A.23}$$

and from Leibnitz's Rule,

$$\nabla_x \left(\int_{h(x)}^{g(x)} f(x, \epsilon) d\epsilon \right) = g'(x) f(x, g(x)) - h'(x) f(x, h(x)) + \int_{h(x)}^{g(x)} \nabla_x f(x, \epsilon) d\epsilon$$

we have

$$\nabla_\beta \xi = -\nabla_\beta c \cdot e^{-c^2} \tag{A.24}$$

Thus, from (A.22–A.24), we solve the gradient of v:

$$\begin{aligned} \nabla_\beta v &= -\frac{2\sigma^2\beta^2 + 1}{\beta^2\mu}\xi - \frac{-2\sigma^2\beta^2 + 1}{\beta\mu}\nabla_\beta c e^{-c^2} + 2c\nabla_\beta c e^{-c^2} \\ &= -\frac{2\sigma^2\beta^2 + 1}{\beta^2\mu}\xi - \left(\frac{-\sigma^2\beta^2 + 1}{\beta\mu} - 2c \right) \nabla_\beta c e^{-c^2} \\ &= -\frac{2\sigma^2\beta^2 + 1}{\beta^2\mu}\xi - (2\mu\beta\nabla_\beta c)\, e^{-c^2} \\ &= -\frac{2\sigma^2\beta^2 + 1}{\beta^2\mu}\xi + \frac{2(\sigma^2 + \mu^2)\beta^2 + 1}{\beta} e^{-c^2} \end{aligned} \tag{A.25}$$

The line search for the value of β satisfying $v(\beta) = 0$ is given by the following update equation:

$$\beta_i = \beta_{i-1} - \kappa \frac{v(\beta_{i-1})}{\nabla_\beta v(\beta_{i-1})} \tag{A.26}$$

We now formulate the initial guess for β. From (A.21), the fact that 2ξ is positive, and $-e^{-c^2}$ is negative, we conclude $\mu\beta + c$ must be positive. Thus,

$$\begin{aligned} \mu\beta + c &> 0 \\ 2\mu^2\beta^2 - 2\left(\sigma^2 + \mu^2\right)\beta^2 + 1 &> 0 \\ 2\sigma^2\beta^2 &< 1 \\ \beta &< 1/\sigma\sqrt{2} \end{aligned} \tag{A.27}$$

and we conclude that the solution satisfies

$$0 < \beta < 1/\sigma\sqrt{2} \tag{A.28}$$

making $\beta = 1/\sigma\sqrt{2}$ the initial guess for the search.

APPENDIX B
SIMULATION OF MEDF TRANSITIONS

This appendix derives the equations needed to simulate maximum entropy density function (MEDF) transitions in Matlab, [Matl90]. The derivation illustrates the relationship between the MEDF and the Gaussian density. The results are used in Chapter 4 for a performance comparison example.

Matlab's built-in random number generators compute either uniform or Gaussian distributed sequences. Thus, exponential and maximum entropy densities will be obtained from transformations, (3.35). Referring to (3.12) and (4.46), an exponential density, $\mathcal{Y} \sim EXP(\lambda)$, is obtained from a uniform density, $\mathcal{X} \sim UNIF(0,1)$, as follows:

$$\begin{aligned} g(x) &\stackrel{\text{def}}{=} -\frac{\ln x}{\lambda} \quad \text{where } \lambda > 0 \\ g'(x) &= -\frac{1}{\lambda x} \\ g^{-1}(y) &= e^{-\lambda x} \\ f_{\mathcal{Y}}(y) &= \frac{f_{\mathcal{X}}\left(e^{-\lambda y}\right)}{1/\lambda x} = \lambda e^{-\lambda y} \end{aligned} \tag{B.1}$$

A variant of the Gaussian density function, (3.21), is used to obtain an MEDF. Denoting the family of truncated Gaussian density functions as $N_+(\mu, \sigma^2)$, where μ and σ^2 refer to the moments of the untruncated function. $\mathcal{X} \sim N_+(\mu, \sigma^2)$ has a density of

$$f_{\mathcal{X}}(x) = \begin{cases} \frac{\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}{h} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{B.2}$$

where h is the density normalization.

$$h = \int_0^\infty \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \tag{B.3}$$

Simplifying h with the transformation $v = x - \mu$,

$$h = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\mu}^{\infty} e^{-\frac{v^2}{2\sigma^2}} dv$$

followed by the transformation $u = \frac{v}{\sqrt{2}\sigma}$, gives

$$h = \frac{1}{\sqrt{\pi}} \int_{-\mu/\sqrt{2}\sigma}^{\infty} e^{-u^2} du \tag{B.4}$$

Thus, a Gaussian sequence can be truncated and re-normalized to obtain the density function (B.2, B.4). From the family of MEDFs, (4.39), substituting (A.8) and (A.4), produces

$$\begin{aligned} f_Y(y) &= \frac{\beta e^{-(\beta y - \alpha/2\beta)^2}}{\xi} \\ &= \frac{\frac{1}{(1/\sqrt{2}\beta)\sqrt{2\pi}} \exp\left(-\frac{\left(y-\alpha/2\beta^2\right)^2}{2(1/\sqrt{2}\beta)^2}\right)}{\frac{1}{\sqrt{\pi}} \int_{-\alpha/2\beta}^{\infty} e^{-u^2} du} \end{aligned} \tag{B.5}$$

Comparing (B.2) with (B.5), we choose

$$\mathcal{X} \sim N_+\left((\alpha/2\beta^2), (1/\sqrt{2}\beta)^2\right) \tag{B.6}$$

APPENDIX C
BOUND AND CONSISTENCY PROOFS: OUTLINE

In the following three appendices, the upper bound nature and consistency of the five top-down estimators are proven. Each proof follows the same outline, described below.

For each estimator, we restrict our attention to PNs that fall within the estimator's domain. Denoting the set of p-place PNs as $\mathcal{C}^p$, we define the set of PNs having state property Z as the net class $\mathcal{C}(p; Z)$:

$$\mathcal{C}(p; Z) \stackrel{\text{def}}{=} \left\{ \mathcal{A} \in \mathcal{C}^p \;\middle|\; \underline{\mu} \in RS(\mathcal{A}, \underline{\mu}^0) \Rightarrow \underline{\mu} \text{ has property } Z \right\} \tag{C.1}$$

With this definition, the proof for estimator $\tilde{s}_i(\cdot)$ is made by demonstrating the following two relationships:

$$\exists \mathcal{A} \in \mathcal{C}(p; Z) \ni \left| RS(A, \underline{\mu}^0) \right| = \tilde{s}_i(\cdot) \tag{C.2}$$

$$\forall \mathcal{A} \in \mathcal{C}(p; Z), \left| RS(A, \underline{\mu}^0) \right| \leq \tilde{s}_i(\cdot) \tag{C.3}$$

The upper bound nature of the estimator is given by (C.3), while (C.2) demonstrates that there is no slack in the estimation, i.e., there exists a PN for which the estimator gives exact results. Typically, (C.2) is proven by induction based on the construction of an appropriate PN model, and (C.3) is proven by contradiction based on combinatoric reasoning. To illustrate the idea of inductive proofs on PN models, the proofs for estimator 1 in Appendix D includes more figures.

APPENDIX D
BOUND AND CONSISTENCY PROOFS: ESTIMATORS 1 AND 2

Estimators 1 and 2, based on place bounds, are defined for the following net classes:

$$\mathcal{C}_1 \stackrel{\text{def}}{=} \mathcal{C}(p;\mu_i \le \tau) = \left\{ \mathcal{A} \in \mathcal{C}^p \,\middle|\, \underline{\mu} \in RS(\mathcal{A}, \underline{\mu}^0) \Rightarrow \mu_i \le \tau \right\} \quad \text{(D.1)}$$

$$\mathcal{C}_2 \stackrel{\text{def}}{=} \mathcal{C}(p;\mu_i \le \tau_i) = \left\{ \mathcal{A} \in \mathcal{C}^p \,\middle|\, \underline{\mu} \in RS(\mathcal{A}, \underline{\mu}^0) \Rightarrow \mu_i \le \tau_i \right\} \quad \text{(D.2)}$$

First we consider estimator 1, and prove that the PN in Figure D.1 contains exactly $\tilde{s}_1(\tau, p)$ states.

Lemma D.1 *The PN in Figure D.1 contains exactly* $\tilde{s}_1(\tau, p) = (\tau + 1)^p$ *states.*

Proof: Inspection of Figure D.1 confirms that the PN is a member of $\mathcal{C}_1$. Assume that $p = 1$, i.e., the PN consists of a single branch. Clearly the number of tokens in P_1 ranges from τ to 0, inclusively, resulting in $\tau + 1$ states. Thus, the lemma holds for $p = 1$.

Now, assume the lemma holds for $p = p_0$, i.e., assume

$$\tilde{s}_1(\tau, p_0) = (\tau + 1)^{p_0} \quad \text{(D.3)}$$

We need to show that the lemma holds for $p = p_0 + 1$. The outlined region in Figure D.2 corresponds to the assumption, (D.3). Thus, for each of the $\tilde{s}_1(\tau, p_0)$ states in the outlined PN, there are $\tau + 1$ states in the single branch. We have,

$$\tilde{s}_1(\tau, p_0 + 1) = (\tau + 1)^{p_0} \cdot (\tau + 1) \quad \text{(D.4)}$$

$$= (\tau + 1)^{p_0 + 1} \quad \text{(D.5)}$$

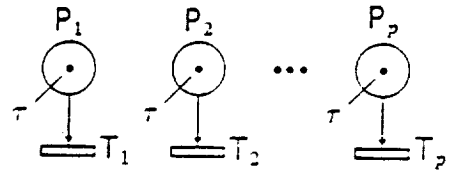


Figure D.1: PN Model for Estimator 1

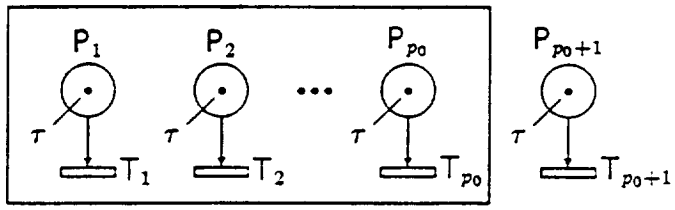


Figure D.2: PN Model for Estimator 1: Induction Step

which proves the lemma. □

In addition to the lemma above, we need to prove (C.3).

Lemma D.2 *Given C_1 as defined above, $\forall \mathcal{A} \in C_1, \left|RS(A, \underline{\mu}^0)\right| \leq \tilde{s}_1(\tau, p)$*

Proof: Assume there exists a PN, $\mathcal{P} \in C_1$, containing more states than estimated by $\tilde{s}_1(\tau, p)$. Then there must be *at least* one state, $\underline{\mu}$, not counted by the estimator. Lemma D.1 proves a correspondence between the states of the PN in Figure D.1 and those counted by $\tilde{s}_1(\tau, p)$, thus $\underline{\mu}$ cannot be a state of Figure D.1. Since $\mu_i \leq \tau$, (from (D.1)), we can form the non-negative integers,

$$x_i \stackrel{\text{def}}{=} \tau - \mu_i \qquad i = 1, 2, \ldots, p \tag{D.6}$$

After x_i firings of transition T_i in Figure D.1, we would reach $\underline{\mu}$, a contradiction. □

Theorem D.1 *Estimator 1 is a consistent upper bound estimator.*

Proof: This result follows from (C.2-C.3) and Lemmas D.1 and D.2. □

We now consider estimator 2, and show that the PN in Figure D.3 contains exactly $\tilde{s}_2(\underline{\tau}, p)$ states.

Lemma D.3 *The PN in Figure D.3 contains exactly $\tilde{s}_2(\underline{\tau}, p) = \prod_{i=1}^{p} \tau_i + 1$ states.*

Proof: Inspection of Figure D.3 confirms that the PN is a member of C_2. Assume that $p = 1$, i.e., the PN consists of a single branch. Clearly the number of tokens in

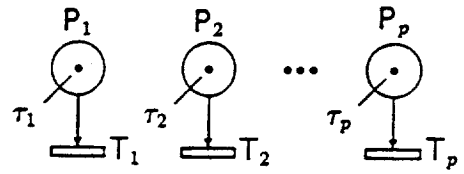


Figure D.3: PN Model for Estimator 2

P_1 ranges from τ_1 to 0, inclusively, resulting in $\tau_1 + 1$ states. Thus, the lemma holds for $p = 1$.

Now, assume the lemma holds for $p = p_0$, i.e., assume

$$\tilde{s}_2(\underline{\tau}, p_0) = \prod_{i=1}^{p_0} \tau_i + 1 \tag{D.7}$$

We need to show that the lemma holds for $p = p_0 + 1$. For each of the $\tilde{s}_2(\underline{\tau}, p_0)$ states in the p_0-place PN, there are $\tau_{p_0+1} + 1$ states in the branch containing P_{p_0+1}. Thus, we have,

$$\tilde{s}_2(\underline{\tau}, p_0 + 1) = \left(\prod_{i=1}^{p_0} \tau_i + 1\right) \cdot (\tau_{p_0+1} + 1) \tag{D.8}$$

$$= \prod_{i=1}^{p_0+1} \tau_i + 1 \tag{D.9}$$

which proves the lemma. □

Lemma D.4 *Given $\mathcal{C}_2$ as defined above, $\forall \mathcal{A} \in \mathcal{C}_2, \left|RS(\mathcal{A}, \underline{\mu}^0)\right| \leq \tilde{s}_2(\underline{\tau}, p)$*

Proof: Assume there exists a PN, $\mathcal{P} \in \mathcal{C}_2$, containing more states than estimated by $\tilde{s}_2(\underline{\tau}, p)$. Then there must be *at least* one state, $\underline{\mu}$, not counted by the estimator. Lemma D.3 proves a correspondence between the states of the PN in Figure D.3 and those counted by $\tilde{s}_2(\underline{\tau}, p)$, thus $\underline{\mu}$ cannot be a state of Figure D.3. Since $\mu_i \leq \tau_i$, (from (D.2)), we can form the non-negative integers,

$$x_i \stackrel{\text{def}}{=} \tau_i - \mu_i \qquad i = 1, 2, \ldots, p \tag{D.10}$$

After x_i firings of transition T_i in Figure D.3, we would reach $\underline{\mu}$, a contradiction. □

Theorem D.2 *Estimator 2 is a consistent upper bound estimator.*

Proof: This result follows from (C.2-C.3) and Lemmas D.3 and D.4. □

APPENDIX E
BOUND AND CONSISTENCY PROOFS: ESTIMATORS 3 AND 4

Estimators 3 and 4, based on net bounds, are defined for the following net classes:

$$\mathcal{C}_3 \stackrel{\text{def}}{=} \mathcal{C}(p; \underline{\mu} \circ \underline{1} \leq T) = \left\{ \mathcal{A} \in \mathcal{C}^p \;\middle|\; \underline{\mu} \in RS(\mathcal{A}, \underline{\mu}^0) \Rightarrow \underline{\mu} \circ \underline{1} \leq T \right\} \quad \text{(E.1)}$$

$$\mathcal{C}_4 \stackrel{\text{def}}{=} \mathcal{C}(p; \underline{\mu} \circ \underline{1} = T) = \left\{ \mathcal{A} \in \mathcal{C}^p \;\middle|\; \underline{\mu} \in RS(\mathcal{A}, \underline{\mu}^0) \Rightarrow \underline{\mu} \circ \underline{1} = T \right\} \quad \text{(E.2)}$$

Since, as will be seen, the proofs for estimator 3 can be developed from those for estimator 4, we will start with the latter. There are several PN models that have exactly $\tilde{s}_4(T,p)$ states. One such model, (although no proof is offered, it can be inferred from the proof below), is given in Figure E.1. A slightly more complicated PN model, Figure E.2, is used for the proof, however, because it is more amenable to induction. First we show that the PN in Figure E.2 contains exactly $\tilde{s}_4(T,p)$ states.

Lemma E.1 *The PN in Figure E.2 contains exactly* $\tilde{s}_4(T,p) = {}^pX_T$ *states.*

Proof: Inspection of Figure E.2 confirms that the PN is a member of $\mathcal{C}_4$. Assuming $p = 1$, i.e., the PN consists of a single place, then a single state containing T tokens results. Since ${}^1X_T = 1$, the lemma holds for $p = 1$.

To better illustrate the inductive step, we continue the proof to $p = 2$. In this case, the PN can trade tokens between places P_1 and P_2, resulting in $T + 1$ states. Since ${}^2X_T = T + 1$, the lemma holds for $p = 2$.

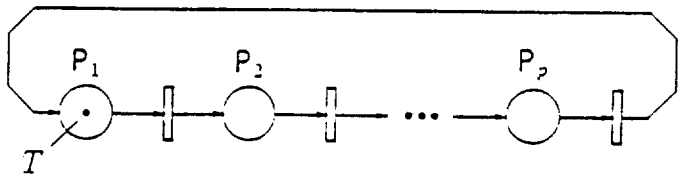


Figure E.1: Potential PN Model for Estimator 4

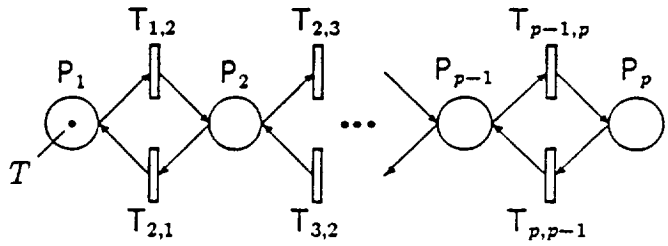


Figure E.2: PN Model for Estimator 4

Now, assume the lemma holds for $p = p_0$, i.e., assume

$$\tilde{s}_4(T, p_0) = {}^{p_0}X_T \tag{E.3}$$

We need to show that the lemma holds for $p = p_0 + 1$. Each time a token enters place P_{p_0}, it may cause another set of states by entering P_{p_0+1}. Thus, the number of states is given by

$$\sum_{i=0}^{T} ({}^{p_0}X_i)\left({}^{1}X_{T-i}\right) = \sum_{i=0}^{T} ({}^{p_0}X_i) \cdot 1 \tag{E.4}$$

$$= {}^{p_0+1}X_T \tag{E.5}$$

where, the simplification is a result of Theorem 6.2. □

Lemma E.2 *Given $\mathcal{C}_4$ as defined above, $\forall \mathcal{A} \in \mathcal{C}_4$, $\left|RS(\mathcal{A}, \underline{\mu}^0)\right| \leq \tilde{s}_4(T, p)$*

Proof: Assume there exists a PN, $\mathcal{P} \in \mathcal{C}_4$, containing more states than estimated by $\tilde{s}_4(T, p)$. Then there must be *at least* one state, $\underline{\mu}$, not counted by the estimator. Lemma E.1 proves a correspondence between the states of the PN in Figure E.2 and those counted by $\tilde{s}_4(T, p)$, thus $\underline{\mu}$ cannot be a state of Figure E.2. Since $0 \leq \mu_i \leq T$ and $\underline{\mu} \circ \underline{1} = T$, (from (E.2)), we can form the non-negative integers,

$$x_i \stackrel{\text{def}}{=} \sum_{j=i-1}^{p} \mu_j \qquad i = 1, 2, \ldots, p-1 \tag{E.6}$$

After x_1 firings of transition $\mathsf{T}_{1,2}$, followed by x_2 firings of transition $\mathsf{T}_{2,3}$, and so forth, the PN in Figure E.2 would reach $\underline{\mu}$, a contradiction. □

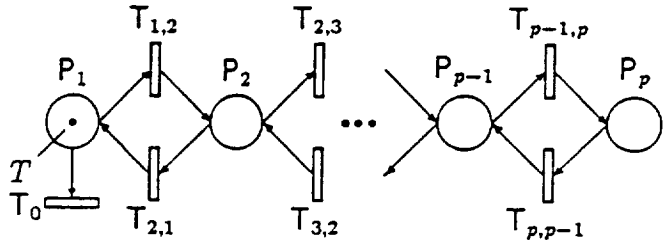


Figure E.3: PN Model for Estimator 3

Theorem E.1 *Estimator 4 is a consistent upper bound estimator.*

Proof: This result follows from (C.2-C.3) and Lemmas E.1 and E.2. □

Now we consider estimator 3, and show that the PN in Figure E.3 contains exactly $\tilde{s}_3(T,p)$ states.

Lemma E.3 *The PN in Figure E.3 contains exactly* $\tilde{s}_3(T,p) = {}^{p+1}X_T$ *states.*

Proof: Inspection of Figure E.3 confirms that the PN is a member of $\mathcal{C}_3$. Assuming $p = 1$, the PN consists of a single place that can contain between T and 0 tokens, inclusively. Thus, $T+1$ states result, and since ${}^2X_T = T+1$, the lemma holds for $p = 1$.

Now, assume the lemma holds for $p = p_0$, i.e., assume

$$\tilde{s}_3(T,p_0) = {}^{p_0+1}X_T \tag{E.7}$$

We need to show that the lemma holds for $p = p_0 + 1$. Each time a token enters place P_{p_0}, it may cause another set of states by entering P_{p_0+1}. Thus, the number of states is given by

$$\sum_{i=0}^{T} \left({}^{p_0+1}X_i\right)\left({}^{1}X_{T-i}\right) = \sum_{i=0}^{T} \left({}^{p_0+1}X_i\right) \cdot 1 \tag{E.8}$$

$$= {}^{p_0+2}X_T \tag{E.9}$$

where, the simplification is a result of Theorem 6.2. □

Lemma E.4 *Given $\mathcal{C}_3$ as defined above, $\forall \mathcal{A} \in \mathcal{C}_3, \left|RS(A, \underline{\mu}^0)\right| \leq \tilde{s}_3(T,p)$*

Proof: Assume there exists a PN, $\mathcal{P} \in \mathcal{C}_3$, containing more states than estimated by $\tilde{s}_3(T,p)$. Then there must be *at least* one state, $\underline{\mu}$, not counted by the estimator. Lemma E.3 proves a correspondence between the states of the PN in Figure E.3 and those counted by $\tilde{s}_3(T,p)$, thus $\underline{\mu}$ cannot be a state of Figure E.3. Since $0 \leq \mu_i \leq T$ and $\underline{\mu} \circ \underline{1} \leq T$, (from (E.1)), we can form the non-negative integers,

$$x_i \stackrel{\text{def}}{=} \sum_{j=i+1}^{p} \mu_i \qquad i = 1, 2, \ldots, p-1 \tag{E.10}$$

$$x_0 \stackrel{\text{def}}{=} T - x_1 \tag{E.11}$$

After x_0 firings of transition T_0, followed by x_1 firings of $\mathsf{T}_{1,2}$, then by x_2 firings of $\mathsf{T}_{2,3}$, and so forth, the PN in Figure E.3 would reach $\underline{\mu}$, a contradiction. □

Theorem E.2 *Estimator 3 is a consistent upper bound estimator.*

Proof: This result follows from (C.2-C.3) and Lemmas E.3 and E.4. □

APPENDIX F
BOUND AND CONSISTENCY PROOFS: ESTIMATOR 5

Estimator 5, based on the weighted conservative property, is defined for the following net class:

$$\mathcal{C}_5 \stackrel{\text{def}}{=} \mathcal{C}(p; \underline{\mu} \circ \underline{w} = 1) = \left\{ \mathcal{A} \in \mathcal{C}^p \;\middle|\; \underline{\mu} \in RS(\mathcal{A}, \underline{\mu}^0) \Rightarrow \underline{\mu} \circ \underline{w} = 1 \right\} \quad \text{(F.1)}$$

Unlike the simple estimators, the proofs for estimator 5 require additional background development. We start by introducing a compact notation that describes the input-output behavior of a given transition, (see Figure F.1). Specifically, we will refer to the i^{th} transition as $\mathsf{T}_i(\underline{w}^-, \underline{w}^+)$, where (recall Definition 3.8),

$$w_j^- \stackrel{\text{def}}{=} I(\mathsf{P}_j, \mathsf{T}_i) \quad \text{(F.2)}$$

$$w_j^+ \stackrel{\text{def}}{=} O(\mathsf{T}_i, \mathsf{P}_j) \quad \text{(F.3)}$$

Thus, $\underline{w}^-$ describes the input arc weights from the places to the transition of interest, and $\underline{w}^+$ describes the output arc weights.

With this notation, we can define a transition that has input and output arc weights that preserve the aggregate token weight within the PN.

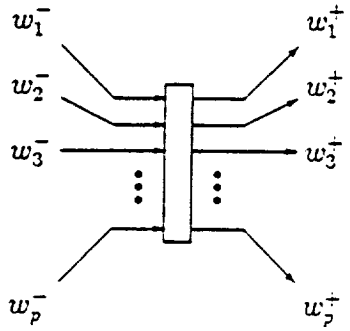


Figure F.1: Transition Model for Estimator 5

Definition F.1 (c-transition) *A transition* $\mathsf{T}(\underline{w}^-, \underline{w}^+)$ *is called a conservative transition (i.e., c-transition) wrt to a PN* $\mathcal{A}$ *with weight vector* $\underline{w}$*, if*

$$\underline{w}^- \circ \underline{w} \;=\; \underline{w}^+ \circ \underline{w} \tag{F.4}$$

Lemma F.1 *The firing of a c-transition does not change the aggregate token weight within a conservative PN.*

Proof: Consider a conservative PN, $\mathcal{A}$, with weight vector $\underline{w}$. Assume the given c-transition, $\mathsf{T}(\underline{w}^-, \underline{w}^+)$, is enabled by state $\underline{x}$. Denoting the resulting state as $\underline{y}$, we have

$$\underline{y} \;=\; \underline{x} - \underline{w}^- + \underline{w}^+ \tag{F.5}$$

The aggregate token weight in the new state is

$$\underline{y} \circ \underline{w} \;=\; \left(\underline{x} - \underline{w}^- + \underline{w}^+\right) \circ \underline{w} \tag{F.6}$$

$$\;=\; \underline{x} \circ \underline{w} - \underline{w}^- \circ \underline{w} + \underline{w}^+ \circ \underline{w} \tag{F.7}$$

$$\underline{y} \circ \underline{w} \;=\; \underline{x} \circ \underline{w} \tag{F.8}$$

where the final simplification is a result of (F.4). □

From the result in Lemma F.1, we can conclude that the constraint (6.25), i.e., $\underline{\mu} \circ \underline{w} = 1$, is also maintained after firing a c-transition. Extending this line of reasoning further, we define a *consistent* conservative transition:

Definition F.2 (cc-transition) *A transition* $\mathsf{T}(\underline{w}^-, \underline{w}^+)$ *is called a consistent conservative transition (i.e., cc-transition) wrt to a PN* $\mathcal{A}$ *with weight vector* $\underline{w}$*, if*

$$\underline{w}^- \circ \underline{w} \;=\; \underline{w}^+ \circ \underline{w} \;=\; 1 \tag{F.9}$$

While maintaining the properties of a c-transition, a cc-transition also has the property if being enabled by at most one state.

Lemma F.2 *Given a PN, $\mathcal{P} \in C_5$, a cc-transition $\mathsf{T}(\underline{w}^-, \underline{w}^+)$ designed for $\mathcal{P}$ can be enabled by at most one state, which is specified by $\underline{w}^-$.*

Proof: Consider a conservative PN, $\mathcal{A} \in C_5$, with weight vector $\underline{w}$. Assume the given cc-transition, $\mathsf{T}(\underline{w}^-, \underline{w}^+)$, is enabled by state $\underline{\mu}$. We will show that $\underline{\mu}$ is unique.

Since $\underline{\mu} \in RS(\mathcal{A}, \underline{\mu}^0)$, we have

$$\underline{\mu} \circ \underline{w} = 1 \tag{F.10}$$

and, from the definition of a cc-transition, we have

$$\underline{w}^- \circ \underline{w} = 1 \tag{F.11}$$

Subtracting (F.11) from (F.10) produces

$$\begin{aligned} \underline{\mu} \circ \underline{w} - \underline{w}^- \circ \underline{w} &= 0 \\ \underline{\mu} - \underline{w}^- \circ \underline{w} &= 0 \\ \sum_{i=1}^{p} \left(\mu_i - w_i^-\right) w_i &= 0 \end{aligned} \tag{F.12}$$

Since w_i is strictly positive, and $\mu_i \geq w_i^-$ to enable the transition, (F.12) can only be satisfied if $\mu_i = w_i^-$. Thus, a cc-transition is enabled by a unique state. □

We are almost ready for prove the consistency and upper bound nature of estimator 5. The final missing piece, namely, control of the output state resulting from firing a cc-transition, is provided by the next lemma.

Lemma F.3 *Given a PN, $\mathcal{P} \in C_5$, the firing of a cc-transition $\mathsf{T}(\underline{w}^-, \underline{w}^+)$ designed for $\mathcal{P}$ results in a single unique state, which is specified by $\underline{w}^+$.*

Proof: Consider a conservative PN, $\mathcal{A} \in C_5$, with weight vector $\underline{w}$, and a given cc-transition, $\mathsf{T}(\underline{w}^-, \underline{w}^+)$. Denote the state resulting from firing this transition as $\underline{y}$. We will show that $\underline{y}$ is unique.

After firing the transition from any state, $\underline{x}$ that enables it, we have,

$$\underline{y} = \underline{x} - \underline{w}^- + \underline{w}^+ \qquad \text{(F.13)}$$

From Lemma F.2, we know $\underline{x} = \underline{w}^-$, thus, (F.13) becomes,

$$\underline{y} = \underline{w}^+ \qquad \text{(F.14)}$$

which proves that the resulting output state is uniquely given by $\underline{w}^+$. □

Alas, with all the machinery in place, we are prepared to give a straight-forward proof of the consistency of $\tilde{s}_5(\underline{w}, \underline{\mu}^0)$.

Lemma F.4 *There exists a conservative PN with initial marking $\underline{\mu}^0$ and weight vector $\underline{w}$ satisfying $\underline{\mu}^0 \circ \underline{w} = 1$ that has exactly $\tilde{s}_5(\underline{w}, \underline{\mu}^0)$ states.*

Proof: Given a PN $\mathcal{P} \in \mathcal{C}_5$ with initial marking $\underline{\mu}^0$ and weight vector $\underline{w}$, estimator 5 determines the number of states in the set M, given by (see (6.28)),

$$M = \{\underline{x} \mid \underline{w} \circ \underline{x} = 1\} \qquad \text{(F.15)}$$

We construct a PN that has all the states contained in M.

Since all conservative PNs have bounded state-spaces, $|M|$ must be finite. We denote this bound by $m = |M|$. Therefore, the states in M can be labeled by the integers $1, 2, \ldots, m$, and can be referred to as $s_1, s_2, \ldots, s_m$. The labeling is arbitrary, except that s_1 is to be initial marking, $\underline{\mu}^0$ (since the initial marking is unique, contained in M, and M is countable, this requirement can always be satisfied).

Starting with a PN of p places labeled P_1 through P_p, and no transitions, we initially distribute tokens in the places as described by $\underline{\mu}^0$. We construct a cc-transition, $\mathrm{T}_1(s_1, s_2)$, and add it to the PN. From Lemma F.2, this transition can be enabled only by s_1 (which is in the state-space as the initial marking), and from Lemma F.3, the only possible output state is s_2. From Lemma F.1, s_2 is contained within M, (F.15).

The state-space of our PN now consists of two states, namely, s_1 and s_2. We construct another cc-transition, $T_2(s_2, s_3)$, and add it to the PN. Reasoning as above, this transition will only be enabled by s_2 (which is in the state-space via firing T_1 from the initial marking), and produces state s_3. Continuing this construction to transition $T_m(s_{m-1}, s_m)$ we have a PN that contains exactly $\tilde{s}_5(\underline{w}, \underline{\mu}^0)$ states. □

We now prove the upper bound nature of estimator 5.

Lemma F.5 *Given C_5 as defined above, $\forall \mathcal{A} \in C_5, \left|RS(A, \underline{\mu}^0)\right| \leq \tilde{s}_5(\underline{w}, \underline{\mu}^0)$*

Proof: Assume there exists a PN, $\mathcal{P} \in C_5$, containing more states than estimated by $\tilde{s}_5(\underline{w}, \underline{\mu}^0)$. Then there must be *at least* one state, $\underline{\mu}$, not counted by the estimator. Lemma F.4 proves a correspondence between the states contained in M and those counted by $\tilde{s}_5(\underline{w}, \underline{\mu}^0)$, thus $\underline{\mu}$ cannot be in M. From (F.1), this is a contradiction. □

Theorem F.1 *Estimator 5 is a consistent upper bound estimator.*

Proof: This result follows from (C.2-C.3) and Lemmas F.4 and F.5. □