An Acyclic Transformation Technique for the Reachability Analysis of Petri Nets

By

Parthasarathy Ramachandran

Bachelor of Engineering
University of Madras
Madras, India
1994

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1995

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
Doctor of Philosophy
December 2002

An Acyclic Transformation Technique for the Reachability Analysis of Petri Nets

Thesis Approved:

Manjunath Kamath

Thesis Advisor

Kenneth J Case

David B Pratt

Carl Conley

Timothy J. Pettitone

Dean of the Graduate College

Acknowledgments

I wish to express my sincere thanks to:

my adviser Dr. Manjunath Kamath for his encouragement, guidance and continued efforts at keeping me focused;

the doctoral committee members, Dr. David B. Pratt, Dr. Kenneth E. Case, and Dr. Hon Shiang Lau for their valuable inputs to this dissertation; and

family and friends for their support and encouragement.

Also, I would like to express my gratitude to the Center for Computer Integrated Manufacturing and the School of Industrial Engineering and Management at Oklahoma State University for the generous financial support they provided during the course of this dissertation. Specifically, I would like to acknowledge the funding from two NSF grants: A Modeling Environment to Support Rapid Reconfiguration of Manufacturing Systems (DMI–9400572) and Curriculum for Integrating Manufacturing Enterprise Decisions (EEC–9527493).

PARTHASARATHY RAMACHANDRAN

Oklahoma State University

December 2002

Table of Contents

List of Tables

List of Figures

Nomenclature

$\mathfrak{I}$	Set of all Integers
$\mathfrak{R}$	Set of all Real numbers
$\mathfrak{N}$	Set of all Natural numbers
PN	A Petri net
m	Number of places
n	Number of transitions
p	A place
t	A transition
${}^\bullet p$, $p^\bullet$	Pre–set and post–set of place p
${}^\bullet t$, $t^\bullet$	Pre–set and post–set of transition t
$\mathcal{M}$	A vector representing the number of tokens in all the places of a Petri net, also referred to as a marking
$\mathcal{M}(p)$	Number of tokens in place p in marking $\mathcal{M}$
$\Gamma(\mathcal{M})$	Set of enabled transitions in marking $\mathcal{M}$
$\mathcal{A}$	Incidence matrix of a Petri net
σ	A valid transition firing sequence
$\mathcal{L}(\mathcal{M})$	Set of all valid transition firing sequences from marking $\mathcal{M}$
$\mathcal{R}(\mathcal{M})$	Set of all reachable markings from marking $\mathcal{M}$ by following all (valid) transition firing sequences in $\mathcal{L}(\mathcal{M})$
π	A place invariant set of a Petri net
Π	The set of all place invariant sets of a Petri net
u	A transition firing count vector
I	An identity matrix
$\mathcal{Z}(m)$	A zero vector of dimension m
x^T	Transpose of vector x
$\|X\|$	Cardinality of set X
$X(x:y)$	Elements of vector X from position x to position y
$\wedge$	Logical AND
$\vee$	Logical OR

Chapter 1

Introduction

In 1962, C. A. Petri developed a modeling tool for the analysis of discrete event dynamic systems (DEDS), which has since then been called *Petri nets*. The utility of Petri nets for describing and studying concurrent, asynchronous, distributed, and stochastic systems has been widely recognized [46, 41]. Petri nets provide a strong mathematical basis for studying these systems, along with a simple yet powerful graphical representation. As Murata [41] says,

> *"... Petri nets can be used by both practitioners and theoreticians. They provide a powerful medium of communication between them ..."*

Petri nets provide an elegant way of representing a DEDS in terms of its events, and the conditions affecting and affected by those events. Petri nets have been successfully used in the analysis of both the structural and behavioral properties of a variety of DEDS. For example, Petri nets have been used to analyze software systems [55], distributed database systems [54], communication protocols [12], manufacturing systems [11], compiler & operating systems [1], formal languages [35], and logic programs [42]. Underlying many of the successful applications is the ability of Petri nets to determine if there exists a sequence of events that would transform the system state from the current one to a desired one, and it is the well

known *reachability problem.* Reachability, though decidable, is still a NP–complete problem [18]. Algorithms with polynomial time complexity to determine reachability are not available for a general Petri net model. In fact, even sufficient conditions for reachability short of enumerating the state space are yet to be determined for a general Petri net. Some special net structures have been identified, which when present facilitate the identification of sufficient conditions for reachability. *Acyclic Petri nets* are a class of nets for which necessary and sufficient conditions for reachability exist. It has been shown that the existence of a non–negative integer solution to the state equations governing the net dynamics is a necessary and sufficient condition for reachability in acyclic nets [25].

The objective of this research is to explore the development of reachability analysis techniques for a general Petri net. Towards this end, a formal transformation procedure is developed, to *expand* a Petri net into an acyclic net, and hence, enable the use of known sufficient conditions for reachability that are available for acyclic Petri nets. Though this result is appealing, it has a restriction in that it requires a priori information about the number of stages of expansion. Hence, this research also explores means to determine the required number stages of expansions for a given reachability problem. These issues would become apparent in later discussions. In light of these new results, this research also explores some problems related to reachability such as sub–marking reachability and deadlock avoidance. Some potential applications of the new results are also presented. The order due date determination problem is studied by developing Petri net modeling constructs and formulating the order due date determination problem as a reachability problem.

Next, Petri nets are introduced with the objective of establishing the

notation used in later chapters. Some of the basic properties of a Petri net needed in later discussions are presented as well.

1.1 Basic Definitions and Notations

A Petri net is a *directed bipartite graph*, with two types of nodes called *places* and *transitions*, and directed arcs connecting the places to transitions and vice versa. An arc is associated with a non–negative integer called its multiplicity. In Petri net models of DEDS, a place represents a condition, and a transition represents an event. Events represented by transitions can be immediate (instantaneous) or timed (see Section K.), the execution of which is controlled by *tokens*. Tokens reside in places, and their movement from one place to another captures the dynamics of the system. The movement of tokens is regulated by transitions and the directed arcs connecting the places and transitions. The *marking* of a Petri net is the number of tokens in each of its places[1]. Graphically, places are represented by circles, transitions by bars and tokens by dots.

Definition 1.1.1 *A Petri net is a 5-tuple,* $PN = (P, T, F, W, \mathcal{M}_0)$, *where*

1. $P = \{p_1, p_2, \ldots, p_m\}$ *is a set of* m *places*
2. $T = \{t_1, t_2, \ldots, t_n\}$ *is a set of* n *transitions*
3. $F \subseteq (P \times T) \cup (T \times P)$ *is a set of directed arcs connecting places and transitions*
4. $W : F \rightarrow \{1, 2, 3, \ldots\}$ *is a weight function*[2] *that assigns a positive integer to all the directed arcs in* F, *and*
5. $\mathcal{M}_0 : P \rightarrow \{0, 1, 2, \ldots\}$ *is the initial marking.*

[1]The terms "marking" and "state" could be used interchangably

[2]The default arc weight is 1, and is often not indicated in graphical representations

The weights of arcs from place p_i to transition t_j and from transition t_j to place p_i is represented by $w(p_i, t_j)$ and $w(t_j, p_i)$, respectively. The *pre* and *post–sets* of transition $t \in T$ are ${}^\bullet t = \{p | w(p,t) > 0\}$ and $t^\bullet = \{p | w(t,p) > 0\}$, respectively. Similarly, the *pre* and *post–sets* of place $p \in P$ are ${}^\bullet p = \{t | w(t,p) > 0\}$ and $p^\bullet = \{t | w(p,t) > 0\}$, respectively. The state of a Petri net is the marking $\mathcal{M} : P \rightarrow \{0, 1, 2, \ldots\}$, a non–negative, integer–valued vector that gives the number of tokens in each place. The i^{th} component of the marking vector, denoted by $\mathcal{M}(p_i)$, represents the number of tokens in place p_i. The dynamic behavior of DEDS is captured in a Petri net model by changing the marking vector in accordance with the *transition enabling and firing rules.*

Example 1.1.1 *Consider the Petri net with four places and three transitions shown in Figure 1.1. The pre–set of place* p_1 *is* ${}^\bullet p_1 = \{t_3\}$, *while its post–set is* $p_1^\bullet = \{t_1\}$. *Similarly, the pre–set of transition* t_1 *is* ${}^\bullet t_1 = \{p_1, p_3\}$, *while its post–set is* $t_1^\bullet = \{p_2\}$. *Note that the arc weights of one are not shown.*

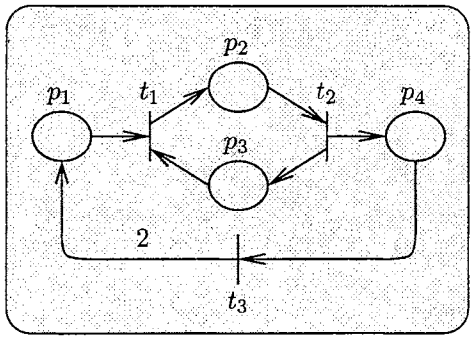

Figure 1.1: A Petri net example

Definition 1.1.2 *A transition t is said to be enabled in marking $\mathcal{M}$, if all the places in its pre–set have at least as many tokens as the weights of the arcs connecting those places to the transition, i.e., if $\forall p \in {}^\bullet t$, $\mathcal{M}(p) \geq w(p,t)$, then the transition t is said to be enabled.*

In a given marking $\mathcal{M}$, the set of all enabled transitions are represented by $\Gamma(\mathcal{M})$. An enabled transition $t_j \in \Gamma(\mathcal{M})$, may or may not fire.

Definition 1.1.3 *Firing an enabled transition changes the token distribution by removing as many tokens as the arc weights from all its pre–set places and adding as many tokens as the arc weights to all its post–set places, i.e., by firing $t \in \Gamma(\mathcal{M})$, the change in token distribution from $\mathcal{M}$ to $\widehat{\mathcal{M}}$ is given by,*

$$\widehat{\mathcal{M}}(p) = \begin{cases} \mathcal{M}(p) - w(p,t), & \forall\ p \in {}^\bullet t \\ \mathcal{M}(p) + w(t,p), & \forall\ p \in t^\bullet \\ \mathcal{M}(p), & \forall\ p \notin \{{}^\bullet t \cup t^\bullet\} \end{cases} \tag{1.1}$$

Example 1.1.2 *Consider the chemical reaction: $2H + O \longrightarrow H_2O$. This reaction could be modeled by a Petri net whose places represent the availability of hydrogen, oxygen, and water. The transition represents the chemical reaction. A weight of two for the arc connecting the place representing hydrogen and transition representing the chemical reaction would impose the condition that two atoms of hydrogen are required for the reaction. Figure 1.2(a) shows the Petri net before executing the transition, while Figure 1.2(b) shows the Petri net after executing the transition [41].*

These changes in the state of the system can be represented in an algebraic form using the state equation given by,

$$\mathcal{M}_k \;=\; \mathcal{M}_{k-1} + \mathcal{A}u_k \tag{1.2}$$

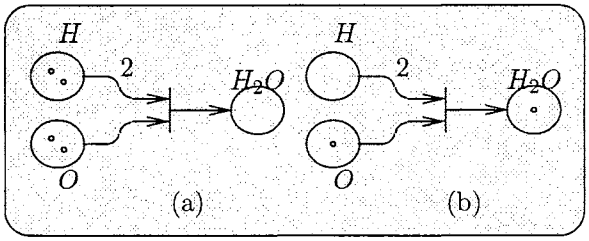

Figure 1.2: A Petri net model of a chemical reaction

where, $\mathcal{A} = [a_{ij}]$ is the $m \times n$ *incidence matrix*, and $u_k \in \mathfrak{N}^n$ is the firing count vector or the control vector. The firing count vector by definition is a non–negative integer vector, with the j^{th} element indicating the number of times transition t_j is to be fired. The incidence matrix describes the place to transition marking relationship in terms of the arc weights connecting them, and the $(i,j)^{th}$ entry is given by $a_{ij} = a_{ij}^- - a_{ij}^+$, where $a_{ij}^+ = w(p_i, t_j)$ and $a_{ij}^- = w(t_j, p_i)$. The control vector u_k is a vector of $(n-1)$ zeroes and an entry of one in the j^{th} position indicating that the transition t_j fires at the k^{th} firing. Note that in the incidence matrix, self–loops[3] would not be reflected.

Definition 1.1.4 *A Petri net is said to be* ***pure*** *if it does not have any self–loops, i.e., in a pure Petri net,* $\forall p \in P$, $\{{}^\bullet p \cap p^\bullet\} = \emptyset$.

Whenever the state equation is discussed in this dissertation, it is assumed that the net model is pure. Note that a net model can be made pure by adding a dummy transition and a dummy place.

[3]a place in both the pre–set and post–set of a transition constitutes a self–loop

1.2 Properties of Petri nets

The analysis of a system modeled by a Petri net is facilitated by two types of properties – those which depend on the initial marking called the *behavioral properties*, and those which are independent of the initial marking called the *structural properties*. A more detailed discussion on these properties can be found in an excellent survey paper by Murata [41].

1.2.1 Behavioral Properties

Behavioral properties depend on the initial marking of the net, implying that for the same Petri net different initial markings could lead to different behavioral properties.

A. Reachability Reachability is one of the basic properties for studying the dynamic behavior of any DEDS. A marking $\mathcal{M}_d$ is said to be *reachable* from a given initial marking $\mathcal{M}_0$, if there exists a sequence of transition firings that transforms $\mathcal{M}_0$ to $\mathcal{M}_d$. A firing sequence is denoted by,

$$\sigma \;=\; \mathcal{M}_0 \xrightarrow{t_1} \mathcal{M}_1 \xrightarrow{t_2} \mathcal{M}_2 \cdots \xrightarrow{t_d} \mathcal{M}_d \tag{1.3}$$

and the marking $\mathcal{M}_d$ is said to reachable from $\mathcal{M}_0$ by σ, which is denoted by $\mathcal{M}_0[\sigma \succ \mathcal{M}_d$. Given an initial marking $\mathcal{M}_0$, the set of all reachable markings is denoted by $\mathcal{R}(\mathcal{M}_0)$, which is the set of all markings generated by all valid transition firing sequences represented by $\mathcal{L}(\mathcal{M}_0)$.

The reachability problem can now be restated as one of finding if $\mathcal{M}_d \in \mathcal{R}(\mathcal{M}_0)$. It has been shown that the reachability problem is

indeed decidable although it takes exponential space and time to verify in the general case [36], [29], [33]. For restricted classes of Petri nets, reachability is decidable in polynomial time, further discussion of which is deferred to Chapter 2.

B. Coverability A marking of a Petri net is said to be covered if there exists another marking such that, for all places, the number of tokens in a place in the second marking is greater than or equal to the number of tokens in the same place in the covered marking, i.e., $\mathcal{M}$ is said to be covered by $\mathcal{M}'$ if $\forall p \in P,\ \mathcal{M}'(p) \geq \mathcal{M}(p)$.

C. Boundedness A Petri net is said to be *bounded* if the number of tokens in each place does not exceed a finite number for any marking reachable from the initial marking, i.e., if $\forall p \in P,\ \mathcal{M}(p) \leq k$ for every $\mathcal{M} \in \mathcal{R}(\mathcal{M}_0)$. A Petri net which is 1–bounded is said to be *safe.*

D. Liveness A Petri net is said to be *live* (or equivalently $\mathcal{M}_0$ is said to be *live* for the Petri net) if from any marking $\mathcal{M} \in \mathcal{R}(\mathcal{M}_0)$ it is ultimately possible to fire any transition either immediately or after progressing through a sequence of transition firings.

E. Reversibility A Petri net is said to be reversible in an initial marking if from each reachable marking from the initial marking, it is possible to reach the initial marking, i.e., $\forall \mathcal{M} \in \mathcal{R}(\mathcal{M}_0), \mathcal{M}_0 \in \mathcal{R}(\mathcal{M})$.

1.2.2 Structural Properties

Structural properties are those that depend only on the topology of the net, and are independent of the initial marking. Hence, these properties

can often be characterized as a function of the incidence matrix.

F. Structural Boundedness A Petri net is said to be structurally bounded if it is bounded for any finite initial marking, i.e., for any finite initial marking $\mathcal{M}_0$, $\forall \mathcal{M} \in \mathcal{R}(\mathcal{M}_0), \exists k < \infty : \mathcal{M}(p) < k, \ \forall p \in P$.

G. Conservativeness A Petri net is said to be conservative if for any finite initial marking, the weighted of sum of tokens in all the places is a constant in all its reachable markings, i.e., $\exists y \in \mathcal{I}^n : \forall \mathcal{M} \in \mathcal{R}(\mathcal{M}_0), \mathcal{M}^T y = \mathcal{M}_0^T y = k$. However, the weighted sum might be different for different initial markings.

H. Place Invariant A set of places of a Petri net is called its place invariant if the weighted sum of tokens in those places remains a constant in all the reachable markings, i.e., $y \in \mathcal{I}^m$ is a place invariant if $\forall \mathcal{M} \in \mathcal{R}(\mathcal{M}_0), \mathcal{M}^T y = k$.

I. Transition Invariant A set of transitions of a Petri net is called its transition invariant if there exists a sequence of firings of those transitions which would reinitialize the marking, i.e., $y \in \mathcal{I}^n$ is a transition invariant if $\exists \sigma \in \mathcal{L}(\mathcal{M}_0) : (|\sigma| = \sum y) \wedge (\mathcal{M}[\sigma \succ \mathcal{M})$.

J. Structural Liveness A Petri net is said to be structurally live if there exists a live initial marking for the Petri net.

1.3 Modified Petri nets

The power of Petri nets in modeling and analyzing DEDS has motivated researchers to incorporate additional modeling capabilities. Some of the widely used extensions are as follows.

K. Timed Petri nets The notion of time is not explicitly used in the basic definition of a Petri net (see Definition 1.1). The need for modeling time in performance analysis was the motivation for the development of timed Petri nets by Ramchandani [48]. In a timed Petri net, an enabled transition can be fired only after a certain time lag. This time to firing an enabled transition could be either deterministic or stochastic. If the times to fire enabled transitions are exponentially distributed, then the Petri net can be translated into an equivalent Markov chain model [52]. This transformation has been widely used for performance evaluation of DEDS.

L. Colored Petri nets Jensen [26] developed colored Petri nets as a means for developing compact models of large systems. The compactness is achieved by merging all the analogous places/transitions in a model into a single place/transition, and associating colors to places, transitions, and tokens to distinguish among the various elements [52]. The transition enabling and firing rules are now defined with respect to colors. Some of the popular applications of colored Petri nets include automated production systems, communication Petri nets, and workflow analysis [52, 27, 11].

1.4 Document Outline

The rest of this document is structured as follows:

- In Chapter 2, various streams of research activities related to reachability analysis in literature are reviewed. The specific categories covered are (1) the representation of the reachability set of a Petri net, (2) necessary and sufficient conditions for reachability, (3) heuristic procedures for deciding reachability, and (4) unfolding techniques based on a graph branching process. The research conducted as part of this dissertation is related to the necessary and sufficient conditions for reachability.

- In Chapter 3, an acyclic transformation technique is developed that converts a general Petri net into an acyclic Petri net. The relationship between the reachable states of the original net and the transformed net is established. In fact, it is shown that every state in the original Petri net has a corresponding state in the acyclic transformed net. Also, the relationship between the transition firing sequences of the original and transformed Petri nets is established.

- The acyclic transformation technique developed in Chapter 3 can be used for any arbitrary Petri net. If the transformation is executed for say 10 stages, then all the markings that can be reached by firing transitions at most 10 times in the original Petri net will have a corresponding marking in the transformed Petri net. However, when the required number of transition firings is not known for a given reachability problem, the acyclic transformation pro-

cedure is open ended. Hence, in Chapter 4 issues related to determining the required number of stages of expansion by acyclic transformation are discussed. Specifically, two methods that could potentially help in determining the required number of stages, or minimize the complexity involved in determining the number of stages are proposed.

- Next, in Chapter 5 some of the problems related to the reachability problem are studied. First, the sub–marking reachability problem is discussed. It is shown that the sub–marking reachability problem could be reduced to a full marking reachability problem or simply the reachability problem, which has been the topic of discussion thus far. A given sub–marking reachability problem is reformulated as a reachability problem. The second problem studied is the deadlock avoidance problem; this is done in light of the new reachability results contributed by this research.

- An application of reachability analysis in a discrete part manufacturing environment is presented in Chapter 6. The order due–date quotation problem is formulated as a reachability problem. More importantly, special modeling constructs are developed for this purpose that enable the systnesis of an acyclic Petri net model.

- Finally, a summary of the dissertation research that highlights the major research contributions is presented in Chapter 7. Future research directions are identified in light of the new findings.

Chapter 2

The Reachability Problem

Summary

This chapter reviews literature related to the analysis of the reachability problem. Literature reveals four different strategies: (1) construction of the reachability graph, (2) reachability criteria for special classes of nets, (3) heuristic procedures, and (4) net unfolding. After reviewing the reachability analysis approaches, the objectives of this dissertation research are presented.

2.1 Introduction

Reachability and coverability are very basic Petri net properties that are used in establishing properties such as boundedness and reversibility.

Definition 2.1.1 *The Reachability Problem. Given a Petri net PN and a marking $\mathcal{M}_d$, the reachability problem determines if $\mathcal{M}_d \in \mathcal{R}(\mathcal{M}_0)$.*

Definition 2.1.2 *The Coverability Problem. Given a Petri net PN and a marking $\mathcal{M}$, the coverability problem determines if $\exists \mathcal{M}' \in \mathcal{R}(\mathcal{M}_0)$: $\forall p \in P,\ \mathcal{M}'(p) \geq \mathcal{M}(p)$.*

Another way to reformulate these two problems is to determine if there exists an effective description of the set of all reachable states [36]. It has been shown that there cannot be any reasonable closed and

effective representation for the set of all reachable states for a general Petri net [2, 22]. The implication of these results is that for a general Petri net, the reachability and coverability problems can be solved only by generating the reachability graph and the coverability graph, which involves an exhaustive enumeration of all possible transition firings in each marking.

2.2 Different Approaches to the Reachability Problem

2.2.1 Reachability Graph

Given a Petri net *PN*, this approach generates as many new markings as the number of enabled transitions in the initial marking $\mathcal{M}_0$. At each of these new markings, firing the enabled transitions would result in a different marking, thus resulting in a graph with $\mathcal{M}_0$ as the root. This graph would grow indefinitely if the net is unbounded. Hence, to keep the graph from growing indefinitely, a special symbol 'ω' is introduced, which can be thought of as "pseudo–infinity". This 'ω' has the following special properties for each integer n, $\omega > n$, $\omega \pm n = \omega$, and $\omega \geq \omega$ [28]. Using this short–hand notation a reachability graph can be constructed using Algorithm 1. If the ω symbol is absent in any of the markings generated, then it is called the *reachability graph*, and if present, the graph takes the name of *coverability graph*.

The coverability/reachability graph constructed using Algorithm 1 can be used for studying behavioral properties of Petri nets as follows:

Definition 2.2.1 *A Petri net is bounded with respect to an initial marking $\mathcal{M}_0$, iff the pseudo–infinity symbol does not appear in its coverability graph.*

Definition 2.2.2 *A Petri net is reversible with respect to an initial marking $\mathcal{M}_0$ iff every node in the coverability graph is in a directed circuit containing $\mathcal{M}_0$.*

Require: Petri net PN, initial marking $\mathcal{M}_0$
1: Label the marking $\mathcal{M}_0$ as *new*
2: **while** *new* markings exist **do**
3: Select a new marking $\mathcal{M}$
4: **if** the set of enabled transitions in marking $\mathcal{M}$ is null **then**
5: Label the marking $\mathcal{M}$ as *deadend*
6: **else**
7: **if** the marking $\mathcal{M}$ is identical to any marking generated thus far **then**
8: Label the marking $\mathcal{M}$ as *old*
9: **else**
10: **for all** markings $\mathcal{M}'$ generated by firing an enabled transition at marking $\mathcal{M}$ **do**
11: **if** there exists an already generated marking $\mathcal{M}'' : \mathcal{M}'' \neq \mathcal{M}'$ such that $\forall p \in P, \mathcal{M}'(p) \geq \mathcal{M}''(p)$ **then**
12: set $\mathcal{M}'(p) = \omega$ whenever $\mathcal{M}'(p) > \mathcal{M}''(p)$
13: label $\mathcal{M}'$ as *new* and add it to the graph
14: **end if**
15: **end for**
16: **end if**
17: **end if**
18: **end while**

Algorithm 1: Reachability graph construction [11]

In the process of developing a bounded graph of the set of all reachable states by using 'ω', valuable information on the actual markings would have been lost, and hence, the graph cannot be used for solving the reachability problem. However, even if the pseudo–infinity symbol

'ω' is not present, the graph construction approach is not of much practical use for solving the reachability problem because of the computational burden. Algorithm 1 essentially enumerates all possible transition firings in all reachable states and this contributes to the computational burden. It has been shown that the reachability problem is indeed decidable although it takes exponential space and time to verify in the general case [36, 29, 33].

2.2.2 Reachability Criteria

The objective of this approach is to develop reachability criteria that are independent of the reachability graph. The state equation representation of Petri nets (equation 1.2) is especially useful for this purpose. Suppose $\mathcal{M}_d$ is reachable from $\mathcal{M}_0$ by firing a sequence of transitions, then

$$\begin{aligned} \mathcal{M}_1 &= \mathcal{M}_0 + \mathcal{A}u_1 \\ \mathcal{M}_2 &= \mathcal{M}_1 + \mathcal{A}u_2 \\ &\vdots \\ \mathcal{M}_d &= \mathcal{M}_{n-1} + \mathcal{A}u_n \end{aligned} \tag{2.1}$$

Adding the above equations,

$$\mathcal{M}_d = \mathcal{M}_0 + \mathcal{A}u \tag{2.2}$$

where $u = \sum_{j=1}^{n} u_i$, and the j^{th} entry denotes the number of times transition t_j must fire. Note that the firing count vector u does not contain any information about the firing sequence. This equation can be rewritten as $\mathcal{A}u = \Delta\mathcal{M}$, where $\Delta\mathcal{M} = \mathcal{M}_d - \mathcal{M}_0$. Given a Petri net PN, and a marking $\mathcal{M}_d$, if $\mathcal{M}_d \in \mathcal{R}(\mathcal{M}_0)$, then the set of linear equations (equation 2.2) must have a non–negative integer solution. This gives rise

to a necessary condition for reachability (Theorem 2.2.1). This is only a necessary condition because, in solving the state equation for a given pair of $\mathcal{M}_d$ and $\mathcal{M}_0$, their actual identities are lost in the expression $\Delta\mathcal{M} = \mathcal{M}_d - \mathcal{M}_0$. Note that multiple pairs of $\mathcal{M}_d$ and $\mathcal{M}_0$ can give rise to the same $\Delta\mathcal{M}$. However, the contrapositive of the necessary condition gives a sufficient condition for non–reachability (Corollary 2.2.1).

Theorem 2.2.1 *If $\mathcal{M}_d$ is reachable from $\mathcal{M}_0$ in a Petri net PN, then there exists $u \in \mathcal{I}^n : \mathcal{A}u = \Delta\mathcal{M}$.*

Corollary 2.2.1 *In a Petri net PN, the marking $\mathcal{M}_d$ is not reachable from $\mathcal{M}_0$ if $\Delta\mathcal{M}$ does not span the columns of the incidence matrix $\mathcal{A}$.*

Proof: If $\mathcal{A}u = \Delta\mathcal{M}$ holds, then $rank[\mathcal{A}] = rank[\mathcal{A} \mid \Delta\mathcal{M}]$. This implies that $\Delta\mathcal{M}$ is a linear combination of the columns of the incidence matrix $\mathcal{A}$. Therefore if $\Delta\mathcal{M}$ is not a linear combination of the columns of $\mathcal{A}$, $rank[\mathcal{A}] \neq rank[\mathcal{A} \mid \Delta\mathcal{M}]$. Hence $\nexists u \in \mathcal{I}^n : \mathcal{A}u = \Delta\mathcal{M}$, and $\mathcal{M}_d \notin \mathcal{R}(\mathcal{M}_0)$. □

Though a sufficient condition for reachability in a general Petri net is yet to be realized, sufficient conditions exist for restricted sub–classes of Petri nets. Next, some of the known sufficient conditions for reachability are presented.

2.2.2.1 Marked Graphs

Definition 2.2.3 *A **marked graph** is a Petri net in which each place has exactly one input transition and one output transition, i.e., in a marked graph $|{}^\bullet p| = |p^\bullet| = 1$.*

Marked graphs represent conflict–free concurrent systems and are very amenable to analysis. Algorithms are available for determining

the liveness and safeness of marked graphs, and for solving the reachability problem. However, by their nature they have limited modeling power [26].

2.2.2.2 Free Choice Nets

Definition 2.2.4 *A **free choice net** is a Petri net in which every arc from a place is either a unique outgoing arc or a unique incoming arc to a transition, i.e., in a free choice net* $\forall t, t' \in T, t \neq t', {}^\bullet t \cap {}^\bullet t' \neq \emptyset \Longrightarrow |{}^\bullet t| = 1 = |{}^\bullet t'|$.

This class of Petri nets are said to allow controlled conflicts [11]. By definition, in a free choice net, all the enabled transitions would have the same input places. This retains the control over the transition that fires next. Hence, they are called free choice Petri nets. Free choice nets that are live, bounded, and acyclic are known to have polynomial time algorithms to decide reachability [10]. However, for live and safe free choice nets, it has been shown that reachability problem is NP–complete [16].

2.2.2.3 Acyclic Petri nets

Definition 2.2.5 *A Petri net having no directed circuit is called an **acyclic Petri net.***

For an acyclic Petri net it can be shown that the existence of a non–negative integer solution to the state equation (equation 2.2) is necessary and sufficient for reachability of $\mathcal{M}_d$ from $\mathcal{M}_0$ [25].

Theorem 2.2.2 *In an acyclic Petri net PN,* $\mathcal{M}_d \in \mathcal{R}(\mathcal{M}_0)$ *iff there exists a non–negative integer solution* u *satisfying* $\Delta\mathcal{M} = \mathcal{A}u$.

2.2.3 Heuristic Procedures

In the absence of reachability criteria for a general Petri net, the construction of the reachability graph and searching through the nodes of the graph seems to be the only approach for deciding reachability. Heuristic techniques have been developed that try to tackle the time complexity of the search process. These heuristic procedures can be divided into ones that generate portions of the reachability graph and those that do not. Search procedures such as genetic algorithms are examples of the latter [50], while graph search procedures are examples of the former [32].

2.2.3.1 Genetic Algorithm Approach

Genetic algorithms were formally introduced in the 1970s by John Holland at the University of Michigan. They are inspired by the mechanism of natural selection – a biological process which prefers stronger individuals in a competing environment. To use a genetic algorithm, a solution to a problem is represented as a genome (or chromosome). The genetic algorithm then creates a population of solutions and applies genetic operators such as mutation and crossover to evolve the solutions in order to find the best one [34].

The three most important aspects of using genetic algorithms are: (1) definition of the objective function, (2) definition and implementation of the genetic representation, and (3) definition and implementation of the genetic operators (mutation and crossover).

Takahashi et al. [50] have proposed a genetic algorithm based reachability analysis procedure in which the three aspects mentioned above are represented as follows:

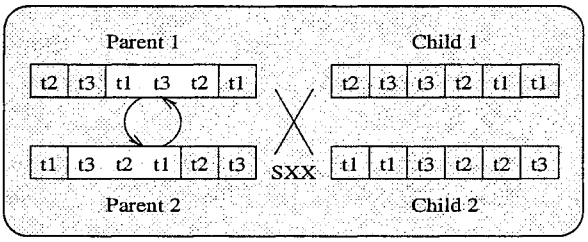

Figure 2.1: Subsequence exchange crossover (SXX)

Genetic representation: Genetic representation for a given firing count vector $u \in \mathcal{I}^n : \mathcal{A}u = \Delta\mathcal{M}$ is a transition sequence $\sigma : \|\sigma\| = \sum_{i=1}^{n} u_i$. The set of all transition sequences for a given firing count vector is denoted by C. Note that $\Gamma(\mathcal{M}) \subseteq C$.

Objective function: The objective function, also termed as the fitness function, is defined as $f : C \rightarrow \mathcal{I}$. For a transition sequence $\sigma \in C$, the value of $f(\sigma)$ defines the number of transitions fired successively from an initial marking. The value of the fitness function is maximized iff $\mathcal{M}_d$ is reachable from $\mathcal{M}_0$ [50].

Genetic operators: The subsequence exchange crossover (SXX) genetic operator is used to create new genes (transition sequences), which exchanges a subsequence in one parent string with a subsequence in another parent string. As shown in Figure 2.1, SXX allows the exchange of a subsequence of Parent 1 for a subsequence of Parent 2 subject to the condition that the same number of each unique transition is exchanged.

Having defined the three components of the genetic algorithm, the

reachability is decided using Algorithm 2.

Require: $u \in \mathcal{I}^n : \mathcal{A}u = \Delta\mathcal{M}$

1: *Initialization:* Create an initial population of transition sequences. Calculate the fitness values of those strings.
2: *Selection:* Choose a pair of strings randomly.
3: **repeat**
4: *Crossover:* Apply SXX to the chosen pair of strings(parents) and make new strings(children). Calculate the fitness values of the children.
5: *Selection:* According to the fitness value, select the top two strings.
6: **until** Fitness value for one of the strings is maximized or time allocated has expired.

Algorithm 2: Genetic algorithm for the reachability problem [50]

2.2.3.2 Graph Search Algorithm

A significant amount of work on the reachability problem appears in relation to scheduling [32]. The sample scheme discussed in this section formulates a scheduling problem using a Petri net model, and employs global search by limiting the search space with the use of heuristic functions. Optimal or near optimal feasible schedules are generated in terms of transition firing sequences. The Petri net model of a system captures all possible evolutions of the system through its markings, which is represented in the reachability graph. Hence, theoretically it should be possible to determine the optimal path by searching the reachability graph.

The heuristic procedure for searching the reachability graph without generating it completely is based upon a graph search algorithm called A^* [44, 45]. This procedure uses an *evaluation function* to expand the search along those sectors of the graph which are most promising. The evaluation function defined in terms of a marking $\mathcal{M}$ is $f(\mathcal{M}) =$

$g(\mathcal{M}) + h(\mathcal{M})$, where $g(\mathcal{M})$ is the current lowest cost obtained from the initial marking ($\mathcal{M}_0$) to the current marking ($\mathcal{M}$), and $h(\mathcal{M})$ is an estimate of the cost from the current marking ($\mathcal{M}$) to the final marking ($\mathcal{M}_d$). Hence, $f(\mathcal{M})$ is an estimate of the total cost from the initial marking($\mathcal{M}_0$) to the final marking($\mathcal{M}_d$),

```
1: Put the initial marking M_0 on the list OPEN
2: while OPEN list is populated do
3:   Remove the first marking M from OPEN and put M on the list
     CLOSED
4:   if M is the final marking then
5:     construct the optimal path from the initial marking to the final
       marking and terminate
6:   else
7:     Find the enabled transitions of marking M
8:     Generate the successor marking M' for each enabled transition
9:     Compute g(M') for each successor marking M'
10:    for all successor M' of M do
11:      if M' is already in OPEN list then
12:        direct it along the path yielding the smallest g(M')
13:      else if M' is already in CLOSED list then
14:        direct it along the path yielding the smallest g(M')
15:      else
16:        calculate h(M') and f(M') and add M' to the OPEN list
17:      end if
18:      Reorder OPEN list in the increasing magnitude of f
19:    end for
20:  end if
21: end while
```

Algorithm 3: Graph search algorithm [32]

2.2.4 Net Unfolding

This unfolding technique has its origins in the unfolding of a rooted graph into a tree. The unfolding technique is a partial order semantics and is

also referred to as the *branching process* [15, 43]. This approach received a boost with the work done by McMillan [37]. The application of this technique has been extensively explored in the literature [17, 18, 39]. Partial order semantics describes the behavior of the net by its maximal branching process, also called the maximal unfolding of the system. If the net is unbounded, its maximal unfolding is also unbounded. McMillan [37] shows a way for computing a finite initial part of the maximal branching, in which every reachable marking of the system is represented. This is called the *finite complete prefix*. This was later made more efficient by Esparza, Römer and Vogler [17]. Though the finite complete prefix has complete information on the set of all reachable states, they are deeply embedded in the unfolding. This can be a source of significant computational burden. However, when the net is bounded these unfoldings can be effectively used for analysis. Once the finite complete prefix has been generated, multiple methods can been applied for determining reachability: branch and bound techniques [37], linear programming algorithms [38], and graph theoretic methods [39] are some examples.

A branching process of a net is a special kind of net called the *occurrence net*. The occurrence net is an acyclic directed graph, and hence, the transitive closure of the flow relation defines the partial order between nodes. An occurrence net has the following properties:

- The net is acyclic,
- The net is free of forward conflict, meaning that the size of the pre–set of all the places is either zero or one,
- No event in the net is in conflict with itself, and

- Every node in the net is finitely preceded.

An occurrence net could be constructed for a Petri net using Algorithm 4. For an unbounded net this branching process could proceed without any natural termination. Also, for a cyclic Petri net the branching process could extend infinitely. However, for bounded Petri nets, even when they are cyclic, it is possible to terminate the branching process called the *prefix*, and preserve all the information about reachable markings. This prefix is called the net *unfolding*. There are a number of techniques for determining these cut-off points [37, 17], and they are evaluated by their ability to restrict the size of the unfolding without losing information on the reachable markings.

2.3 Research Goals

The main goal of this research is to develop the ability to efficiently decide reachability in a general Petri net. The reachability graph with the ω notation (Section 2.2.1) provides a means to efficiently describe the set of all reachable states of a system. The heuristic techniques (Section 2.2.3) provide a means to efficiently search through the reachability graph. These search procedures, though efficient, are not conclusive in deciding reachability. The necessary and sufficient conditions (Section 2.2.2) provide a means for deciding reachability conclusively, but only for certain sub-classes of Petri nets. The unfolding technique (Section 2.2.4) can been efficiently used for only bounded Petri nets. Hence, the major objective of this research is to develop a technique for determining reachability in a general Petri net by building on the existing approaches.

```
Require: Petri net PN
 1: Copy all the marked places of the Petri net into the occurrence net
 2: repeat
 3:    Choose a transition t ∈ T
 4:    Find the pre–set of the chosen transition
 5:    if the same pre–set has not been chosen earlier then
 6:       if none of the selected places are in conflict or precedence relationship then
 7:          for all places in the pre–set of the chosen transition do
 8:             Find a copy in the occurrence net
 9:             if copy found then
10:                Mark it with a token
11:                Make a copy of the transition in the occurrence net
12:                Connect the transition with its pre–set
13:                Make a copy of the the post–set of the transition and
                   connect them with the transition
14:             end if
15:          end for
16:       end if
17:    end if
18: until Ad Infinitum
```

Algorithm 4: Petri net unfolding process [37]

The research described in this document takes the route of determining sufficient conditions for reachability in a general Petri net. Toward this end, an acyclic transformation technique is developed that converts a given Petri net into an acyclic Petri net with known necessary and sufficient conditions for reachability. This dissertation research also explores some reachability related problems, namely, the sub–marking reachability problem and deadlock avoidance in light of the new research results related to reachability. Finally, a novel application of the Petri net reachability problem to the order due date determination problem in a discrete part manufacturing system is explored.

The various research objectives can be stated as follows.

Reachability in general Petri nets: The objective is to develop a reachability analysis technique for a general Petri net. The approach is similar in spirit to the unfolding technique, but the resulting net falls into the class of Petri nets for which necessary and sufficient conditions for reachability are known.

Problems related to reachability: The objective is to analyze problems that are related to the reachability problem in light of the new research results related to reachability in general Petri nets. Specifically, the sub–marking reachability problem and the deadlock avoidance problem are explored.

Application of reachability: The objective is to explore novel applications of the reachability problem in a manufacturing environment. The order due date determination problem in a discrete part manufacturing environment is formulated as a reachability problem.

The research conducted to address these three objectives are presented in the following chapters.

Chapter 3

An Acyclic Transformation Technique

Summary

A new approach to the Petri net reachability problem is presented in this chapter. For an acyclic Petri net, it has been shown that the existence of a non–negative integer solution to the system state equation is necessary and sufficient for reachability. An acyclic transformation is presented in this chapter that expands any given general Petri net into an acyclic net. This makes a general Petri net amenable for analysis using the reachability criteria for acyclic nets. The practical issues involved in executing this transformation are also discussed.

3.1 Introduction

The main idea behind the acyclic transformation is to expand a Petri net into multiple stages, with the places in the 1^{st} stage forming the input for the 2^{nd} stage and so on. This converts any arbitrary Petri net into an acyclic Petri net, thus enabling the applicability of known sufficient conditions for reachability in acyclic nets to any arbitrary Petri net. By this process, if the net expansion has been performed up to $\mathcal{N}$ stages, then all the states that could be reached by firing a maximum of $\mathcal{N}$ transitions are readily available. In fact, as the discussions in the following sections would reveal, this procedure also gives a straightforward method

for finding the transition firing sequences contained in $\mathcal{N}$–stage expansion. This technique could be used for any arbitrary Petri net. However, the major issue that needs to be addressed is in bounding the number of stages $\mathcal{N}$ that is sufficient for deciding a specific reachability problem, which is the topic of the following chapter. We restrict the presentation in this chapter to the exposition of the acyclic transformation procedure.

3.1.1 Net Expansion

The net expansion procedure is at the heart of the acyclic transformation approach. The basic idea is to expand the Petri net into multiple stages, which is accomplished by replicating the places with one set of places forming the input places for the transitions, and the other forming the set of output places. This ensures that there are no directed circuits in the net. In addition, fictitious transitions are added between "equivalent" places to transfer any additional tokens that might be left over. This would make them available for the next stage of the net. These additional transitions are called "*transporters,*" and are graphically represented by two parallel bars. At any given stage, one or more or all of the conflict free transitions could be fired.

The following example explains the net expansion process. The prefix of 1 (i.e., ${}_1X$) is used to denote a one–stage expansion.

Example 3.1.1 *Consider the Petri net shown in Figure 3.1(a). This net has four places and three transitions. For a one–stage expansion of the net, the four places are replicated. For a transition, say* t_1 *in the original net, the pre–set is* ${}^\bullet t_1 = \{p_1, p_3\}$*, and the post–set is* $t_1^\bullet = \{p_2\}$*. In the expanded net, the transition* t_1 *represented by* ${}_1t_{11}$ *has the following pre and post–sets:* ${}^\bullet{}_1t_{11} = \{{}_1p_{11}, {}_1p_{13}\}$ *and* ${}_1t_{11}^\bullet = \{{}_1p_{22}\}$*, where* ${}_1p_{11}$*,* ${}_1p_{13}$*, and* ${}_1p_{22}$ *can be regarded as equivalent to* p_1*,* p_3*, and* p_2*, respectively, in*

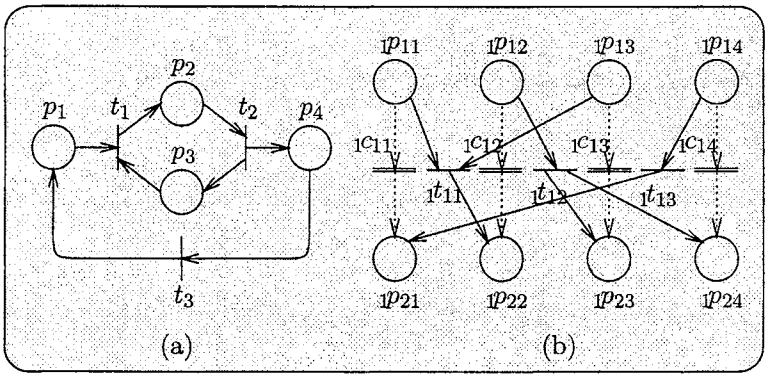

Figure 3.1: One–stage net expansion example

the original net. This accomplishes the net expansion. Next, to transport any left over tokens from the current stage to the next stage, the four transporters $_1c_{1i}, 1 \leq i \leq 4$ *are introduced. A complete one–stage expansion is shown in Figure 3.1(b).*

The following definition formalizes this idea for a one–stage expansion.

Definition 3.1.1 *Given a Petri net PN, its one–stage expanded net is a 7-tuple,* $_1PN = (_1P, _1T, _1C, _1F, _1G, _1W, _1\mathcal{M}_0)$*, where*

1. $_1P = \{_1p_{11}, _1p_{12}, \cdots, _1p_{1m}, _1p_{21}, _1p_{22}, \ldots, _1p_{2m}\}$ *is a set of* $2m$ *places*
2. $_1T = \{_1t_{11}, _1t_{12}, \cdots, _1t_{1n}\}$ *is a set of* n *transitions*
3. $_1C = \{_1c_{11}, _1c_{12}, \cdots, _1c_{1m}\}$ *is a set of* m *transporters*
4. $_1F \subseteq (_1P \times _1T) \cup (_1T \times _1P)$ *is a set of directed arcs connecting the places and transitions*
5. $_1G = \{_1P \times _1C\} \cup \{_1C \times _1P\}$*, is a set of directed arcs connecting the places and transporters, which move the unconsumed tokens from the current stage to the next stage*

6. ${}_1W : {}_1F \rightarrow \{1, 2, 3, \cdots\}$ *is a weight function*[1] *which assigns a positive integer to all the directed arcs in* ${}_1F$ *such that,*

   (a) $w({}_1p_{1i}, {}_1t_{1j}) = w(p_i, t_j), \forall w(p_i, t_j) > 0$
   (b) $w({}_1t_{1j}, {}_1p_{2i}) = w(t_j, p_i), \forall w(t_j, p_i) > 0$

   and

7. ${}_1\mathcal{M}_0 : {}_1P \rightarrow \{0, 1, 2, \ldots\}$ *is the initial marking such that,*

$$ {}_1\mathcal{M}_0(i) \;=\; \begin{cases} \mathcal{M}_0(i) & 1 \leq i \leq m \\ 0 & (m+1) \leq i \leq 2m \end{cases} $$

Within the expanded net, the execution is very similar to that of any Petri net. The transition enabling and firing rules (Definitions 1.1.2 & 1.1.3) still hold good. From the definition, it can be observed that the expansion task is to separate the input arcs (places to transition) from the output arcs (transition to places). The incidence matrix of the one–stage expanded net can now be defined in terms of the original Petri net.

Given a Petri net *PN*, the incidence matrix of the one–stage expanded net of $2m \times (n+m)$ ((input places, output places) $\times$ (regular transitions, transporters from input places to the output places)) dimension is ${}_1\mathcal{A} = [{}_1a_{ij}]$, and the typical entry of the incidence matrix, ${}_1a_{ij}$, is given by,

$$ {}_1a_{ij} = \begin{cases} a_{ij} & \forall (i \leq m), (j \leq n) \text{ if } a_{ij} < 0 \\ a_{(i-m)j} & \forall (m+1 \leq i \leq 2m), (j \leq n) \text{ if } a_{ij} > 0 \\ -1 & \forall (i \leq m), (n+1 \leq j \leq n+m) \text{ if } i = j \\ 1 & \forall (m+1 \leq i \leq 2m), (n+1 \leq j \leq n+m) \\ & \qquad\qquad \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.1} $$

[1] the default arc weight is 1, and is often not indicated in graphical representations

The incidence matrix of the expanded net can also be represented in terms of the original Petri net as,

$$ {}_1\mathcal{A} = [{}_1a_{ij}^-] - [{}_1a_{ij}^+] = \begin{pmatrix} \mathcal{A}^- & -I \\ \mathcal{A}^+ & I \end{pmatrix} \tag{3.2} $$

This expansion procedure can be easily extended to multiple stages. An $\mathcal{N}$–stage expansion of the net utilizes the output places of the previous stage as the input places for the current stage. It can also be thought of as placing multiple one–stage expansions one after the other, which is best illustrated by the structure of the incidence matrix. The required number of stages of expansion is determined by the specific reachability problem. The discussion on this is deferred until Chapter 4.

Definition 3.1.2 *Given a Petri net PN, the incidence matrix of its $\mathcal{N}$–stage expanded net is,*

$$ {}_{\mathcal{N}}\mathcal{A} = \begin{pmatrix} \mathcal{A}^- & 0 & 0 & \cdots & 0 & 0 & -I & 0 & \cdots & 0 & 0 \\ \mathcal{A}^+ & \mathcal{A}^- & 0 & \cdots & 0 & 0 & I & -I & \cdots & 0 & 0 \\ 0 & \mathcal{A}^+ & \mathcal{A}^- & \cdots & 0 & 0 & 0 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \mathcal{A}^+ & \mathcal{A}^- & 0 & 0 & \cdots & I & -I \\ 0 & 0 & 0 & \cdots & 0 & \mathcal{A}^+ & 0 & 0 & \cdots & 0 & I \end{pmatrix} \tag{3.3} $$

The dimension of ${}_{\mathcal{N}}A$ is $(\mathcal{N}+1)m \times (\mathcal{N}n + \mathcal{N}m)$.

Having introduced the notion of net expansion, the utility of it in the reachability analysis of a Petri net is discussed next. The following two issues have to be addressed before using the expanded net for the reachability analysis of a given net.

Relationship between the reachability sets: The problem is to establish the relationship between the set of all reachable states in

the given Petri net, and in its $\mathcal{N}$–stage expanded net, and vice versa.

Integrity of the transition firing sequences: The second issue is to establish the equivalence or relationship between the transition firing sequences in the given Petri net, and in its $\mathcal{N}$–stage expanded net.

3.1.2 Relationship between the Reachability Sets

The net expansion scheme can be effectively used for reachability analysis only if it can be shown that for every reachable marking in the given net, there exists an equivalent marking in the expanded net.

Consider a Petri net PN. Let $\mathcal{M}_1$ be the system state after firing an enabled transition t_j. In its one–stage expanded net equivalent ${}_1PN$, the transition t_j would be represented by ${}_1t_{1j}$. Note that if $t_j \in \Gamma(\mathcal{M}_0)$, then ${}_1t_{1j} \in \Gamma({}_1\mathcal{M}_0)$, in other words if transition t_j is enabled in marking $\mathcal{M}_0$, then the equivalent transition ${}_1t_{1j}$ in the one stage expanded net is also enabled in the equivalent marking ${}_1\mathcal{M}_0$. The resulting marking by firing the enabled transition ${}_1t_{1j}$ in ${}_1PN$ can be represented as,

$$\begin{aligned} {}_1\mathcal{M}_1 &= {}_1\mathcal{M}_0 + {}_1\mathcal{A}_1 u \\ &= \begin{pmatrix} \mathcal{M}_0 \\ \mathcal{Z}(m) \end{pmatrix} + \begin{pmatrix} \mathcal{A}^- & -I \\ \mathcal{A}^+ & I \end{pmatrix} \begin{pmatrix} u_t \\ u_c \end{pmatrix} \\ &= \begin{pmatrix} \mathcal{M}_0 + \mathcal{A}^- u_t - I u_c \\ \mathcal{Z}(m) + \mathcal{A}^+ u_t + I u_c \end{pmatrix} \end{aligned} \tag{3.4}$$

where, u_t is the firing count vector for the transitions, u_c is the firing count vector for the transporters, and $\mathcal{Z}(m)$ is a vector of zeros of length m.

The relationship between the resulting marking $\mathcal{M}_1$ of the original Petri net *PN* and that of its expanded equivalent is discussed next.

$$\begin{aligned} \mathcal{M}_1 &= \mathcal{M}_0 + \mathcal{A}u \\ &= \mathcal{M}_0 + (\mathcal{A}^- + \mathcal{A}^+)u \\ &= (\mathcal{M}_0 + \mathcal{A}^- u) + (\mathcal{Z}(m) + \mathcal{A}^+ u) \end{aligned} \tag{3.5}$$

Equation 3.5 shows that every legal marking of *PN* that could be reached as a result of firing a single transition can be decomposed into two components namely, $(\mathcal{M}_0 + \mathcal{A}^- u)$ and $(\mathcal{Z}(m) + \mathcal{A}^+ u)$. These two components constitute the markings of the input and output places of the transitions in the expanded net after firing the equivalent transition as given by Equation 3.4, assuming that none of the transporters needs to be fired (i.e., $u_c = 0$). The firing count vector for the transporters can be determined by the number of tokens left in the input places after the transitions have been fired, and it can be determined by,

$$u_c = \mathcal{M}_0 + \mathcal{A}^- u_t \tag{3.6}$$

Now the resultant marking of the expanded net can be determined as,

$$\begin{aligned} {}_1\mathcal{M}_1 &= \begin{pmatrix} \mathcal{M}_0 + \mathcal{A}^- u_t - I(\mathcal{M}_0 + \mathcal{A}^- u_t) \\ \mathcal{Z}(m) + \mathcal{A}^+ u_t + I(\mathcal{M}_0 + \mathcal{A}^- u_t) \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ \mathcal{M}_0 + (\mathcal{A}^- + \mathcal{A}^+)u_t \end{pmatrix} \end{aligned} \tag{3.7}$$

where the marking of the output places of the transitions are the same as the markings of the places of the original net in marking $\mathcal{M}_1$. However, not all the markings of the original net have an equivalent marking in the one–stage expanded net. This following example would help illustrate these ideas.

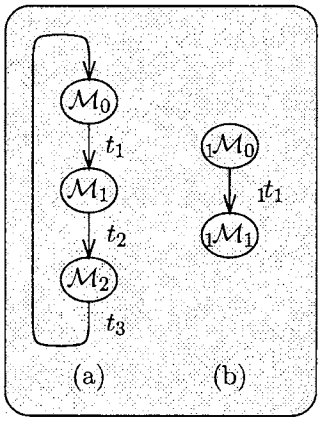

Figure 3.2: Reachability trees for the one–stage net expansion example

Example 3.1.2 *Consider the Petri net shown in Figure 3.1(a) and its one–stage expanded equivalent in Figure 3.1(b). Given an initial marking of* $\mathcal{M}_0 = \{1,0,1,0\}$ *for the original net, the equivalent initial marking for the one–stage expanded net is* ${}_1\mathcal{M}_0 = \{1,0,1,0,0,0,0,0\}$ *(see definition 3.1.1). For these initial markings, the corresponding reachability trees are shown in Figure 3.2. It can be verified from the marking legends of the reachability tree that the marking* ${}_1\mathcal{M}_1$ *of the one–stage expanded net is equivalent to the marking* $\mathcal{M}_1$ *of the original Petri net. However, there exists no equivalent marking in the one–stage expanded net for the other marking* $\mathcal{M}_2$ *of the original net.*

Marking Legend

Original Petri net		**One–stage expanded net**	
$\mathcal{M}_0$	$\{1,0,1,0\}$	${}_1\mathcal{M}_0$	$\{1,0,1,0,0,0,0,0\}$
$\mathcal{M}_1$	$\{0,1,0,0\}$	${}_1\mathcal{M}_1$	$\{0,0,0,0,0,1,0,0\}$
$\mathcal{M}_2$	$\{0,0,1,1\}$		

Table 3.1: Marking legend for the reachability trees in Figure 3.2

The markings of the original net that have no equivalent markings in

the one–stage expanded nets are the ones that can be reached by firing two or more one transitions. A multi–stage expansion would capture these relationships. For an $\mathcal{N}$–stage expanded net, the state equation can be rewritten as,

$$
\begin{aligned}
{}_{\mathcal{N}}\mathcal{M}_d &= \mathcal{M}_0 + {}_{\mathcal{N}}\mathcal{A}u \\
&= \mathcal{M}_0 + \begin{pmatrix}
\mathcal{A}^- & 0 & 0 & \cdots & 0 & -I & 0 & \cdots & 0 & 0 \\
\mathcal{A}^+ & \mathcal{A}^- & 0 & \cdots & 0 & I & -I & \cdots & 0 & 0 \\
0 & \mathcal{A}^+ & \mathcal{A}^- & \cdots & 0 & 0 & I & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & \mathcal{A}^- & 0 & 0 & \cdots & I & -I \\
0 & 0 & 0 & \cdots & \mathcal{A}^+ & 0 & 0 & \cdots & 0 & I
\end{pmatrix} \\
&\quad \begin{pmatrix} u_t(1) & u_t(2) & \cdots & u_t(\mathcal{N}) & u_c(1) & \cdots & u_c(\mathcal{N}-1) & u_c(\mathcal{N}) \end{pmatrix}^T \\
&= \begin{pmatrix}
\mathcal{M}_0 + \mathcal{A}^- u_t(1) - u_c(1) \\
\mathcal{Z}(m) + \mathcal{A}^+ u_t(1) + \mathcal{A}^- u_t(2) + u_c(1) - u_c(2) \\
\mathcal{Z}(m) + \mathcal{A}^+ u_t(2) + \mathcal{A}^- u_t(3) + u_c(2) - u_c(3) \\
\vdots \\
\mathcal{Z}(m) + \mathcal{A}^+ u_t(\mathcal{N}-1) + \mathcal{A}^- u_t(\mathcal{N}) + \\
\qquad u_c(\mathcal{N}-1) - u_c(\mathcal{N}) \\
\mathcal{Z}(m) + \mathcal{A}^+ u_t(\mathcal{N}) + u_c(\mathcal{N})
\end{pmatrix}
\end{aligned}
\tag{3.8}
$$

where the components of the firing count vector $u_t(i),\ 1 \leq i \leq \mathcal{N}$ and $u_c(i),\ 1 \leq i \leq \mathcal{N}$ are the transition firing counts of transitions and transporters at each stage, respectively. Each row of the above decomposition represents the markings of places at each stage. The state equation of

the original net can be decomposed as,

$$
\begin{aligned}
\mathcal{M}_d &= \mathcal{M}_0 + \mathcal{A}u \\
&= \mathcal{M}_0 + (\mathcal{A}^- + \mathcal{A}^+)(u_1 + u_2 + \cdots + u_{\mathcal{N}}) \\
&= \mathcal{M}_0 + \mathcal{A}^- u_1 \\
&\quad + \mathcal{A}^+ u_1 + \mathcal{A}^- u_2 \\
&\quad \ddots \\
&\quad + \mathcal{A}^+ u_{(\mathcal{N}-1)} + \mathcal{A}^- u_{\mathcal{N}} \\
&\quad + \mathcal{A}^+ u_{\mathcal{N}} \\
&= (\mathcal{M}_0 + \mathcal{A}^- u_1) + (\mathcal{Z}(m) + \mathcal{A}^+ u_1 + \mathcal{A}^- u_2) + \\
&\quad (\mathcal{Z}(m) + \mathcal{A}^+ u_2 + \mathcal{A}^- u_3) + \cdots + (\mathcal{Z}(m) + \mathcal{A}^+ u_{(\mathcal{N}-1)} + \\
&\quad \mathcal{A}^- u_{\mathcal{N}}) + (\mathcal{Z}(m) + \mathcal{A}^+ u_{\mathcal{N}}) \\
&= {}_{\mathcal{N}}\mathcal{M}_d(1:m) + {}_{\mathcal{N}}\mathcal{M}_d(m+1:2m) + \cdots + \\
&\quad {}_{\mathcal{N}}\mathcal{M}_d(\mathcal{N}m+1:\mathcal{N}m+m)
\end{aligned}
\tag{3.9}
$$

where $X(x : y)$ refers to the elements of vector X from position x to position y.

The above expansion of the state equations shows the relationship between the reachability sets of a net and its generic expansion. The firing count vector for the transporters for any stage i can be expressed in terms for the firing counts of the transitions as,

$$u_c(i) = \mathcal{M}_{i-1} + \mathcal{A}^- u_t(i) \tag{3.10}$$

Substituting the firing counts of the transporters in the state equation

of the expanded net (Equation 3.8),

$$
{}_{\mathcal{N}}\mathcal{M}_d = \begin{pmatrix}
\mathcal{M}_0 + \mathcal{A}^- u_t(1) - (\mathcal{M}_0 + \mathcal{A}^- u_t(1)) \\
\mathcal{A}^+ u_t(1) + \mathcal{A}^- u_t(2) + (\mathcal{M}_0 + \mathcal{A}^- u_t(1)) - (\mathcal{M}_1 + \mathcal{A}^- u_t(2)) \\
\mathcal{A}^+ u_t(2) + \mathcal{A}^- u_t(3) + (\mathcal{M}_1 + \mathcal{A}^- u_t(2)) - (\mathcal{M}_2 + \mathcal{A}^- u_t(3)) \\
\vdots \\
\mathcal{A}^+ u_t(\mathcal{N}-1) + \mathcal{A}^- u_t(\mathcal{N}) + \big(\mathcal{M}_{(\mathcal{N}-2)} + \mathcal{A}^- u_t(\mathcal{N}-1)\big) - \\
\big(\mathcal{M}_{(\mathcal{N}-1)} + \mathcal{A}^- u_t(\mathcal{N})\big) \\
\mathcal{A}^+ u_t(\mathcal{N}) + \big(\mathcal{M}_{(\mathcal{N}-1)} + \mathcal{A}^- u_t(\mathcal{N})\big)
\end{pmatrix}
$$

$$
= \begin{pmatrix}
0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \mathcal{M}_0 + A(u_t(1) + u_t(2) + \cdots + u_t(\mathcal{N}))
\end{pmatrix} \tag{3.11}
$$

To summarize, in the original net every marking that is reachable from the initial marking by firing a sequence of $\mathcal{N}$ transitions has an unique equivalent marking in the $\mathcal{N}$–stage expanded net. Also, firing sequences of the original Petri net are preserved in the expanded Petri net. In fact, at each stage, by firing only one transition, the solution to the state equation of the expanded Petri net delivers the transition firing sequence also. At each stage, multiple transitions that are not in conflict could be fired. Hence, the following Theorem 3.1.1.

Theorem 3.1.1 *Consider a Petri net PN and its $\mathcal{N}$–stage expanded net ${}_{\mathcal{N}}PN$. For every reachable marking in the original net,*

1. *there exists an $\mathcal{N}$ such that there is a unique equivalent marking in the $\mathcal{N}$–stage Petri net, i.e.,*

$$\exists \mathcal{N} : \forall \mathcal{M} \in \mathcal{R}(\mathcal{M}_0), \exists_{\mathcal{N}}\mathcal{M} \in {}_{\mathcal{N}}\mathcal{R}({}_{\mathcal{N}}\mathcal{M}_0)$$

2. *the transition firing sequences that connect the equivalent markings in the original net and the expanded net are equivalent, i.e.,*

$$\Big(\sigma \in \mathcal{L}(\mathcal{M}_0) : \mathcal{M}_0[\sigma \succ \mathcal{M}_d\Big) \equiv \Big({}_{\mathcal{N}}\sigma \in {}_{\mathcal{N}}\mathcal{L}({}_{\mathcal{N}}\mathcal{M}_0) : {}_{\mathcal{N}}\mathcal{M}_0[{}_{\mathcal{N}}\sigma \succ {}_{\mathcal{N}}\mathcal{M}_d\Big)$$

Proof: Refer to the above discussions.

The above theorem establishes the relationship between a given Petri net, and its expanded version using the acyclic transformation.

3.2 Sufficient Condition for Reachability

The net expansion scheme developed in the previous sections converts a general Petri net into an acyclic Petri net. Also, the previous sections shown that all markings that could be reached in the original Petri net by firing transitions at most $\mathcal{N}$ times have equivalent markings in the $\mathcal{N}$–stage expanded Petri net (Theorem 3.1.1). Hence, in a general Petri net, a marking $\mathcal{M}_d$ is reachable if there exists a positive integer $\mathcal{N}$ such that the equivalent marking ${}_{\mathcal{N}}\mathcal{M}_d$ is reachable in the expanded acyclic Petri net ${}_{\mathcal{N}}PN$. Thus, the sufficient condition for reachability in a general

Petri net can be formalized as follows.

Theorem 3.2.1 *In a general Petri net PN, a marking* $\mathcal{M}_d \in \mathcal{R}(\mathcal{M}_0)$, *iff there exists a positive integer* $\mathcal{N}$ *such that the state equation of the* $\mathcal{N}$*-stage expanded Petri net* ${}_{\mathcal{N}}\mathcal{M}_d = {}_{\mathcal{N}}\mathcal{M}_0 + {}_{\mathcal{N}}\mathcal{A}u$ *has a non-negative integer solution.*

Proof: Refer to the above discussions.

If the required number of stages for a reachability problem can be determined exactly or at least bounded, then the above Theorem 3.2.1 can be used to determine reachability in a general Petri net. This is the subject of the discussions in the next chapter.

Chapter 4

Reachability Analysis Using Net Expansion

Summary

The acyclic transformation expands any given Petri net into an acyclic net with a specified number of stages. In order to utilize the expansion procedure for reachability analysis, the number of stages for expansion needs to be determined. In this chapter, issues related to the determinination of the number of stages are discussed, and ways of reducing the complexity in computing the same are explored.

4.1 Introduction

The acyclic transformation in conjunction with Theorem 2.2.2 can be used for the reachability analysis of a general Petri net if the number of stages of net expansion could be bounded. In certain situations, the problem context could provide an upper bound for net expansion. For example, in Petri net models used for machine scheduling, the number of jobs would dictate the required number of stages. In the absence of such a direct insight from the problem context, the model is expected to provide that insight. However, gaining such an insight on bounding the net expansion is not easy. In the following sections, the issues involved

in determining the number of net expansion stages are discussed. Also, ways of reducing the computational complexity are developed.

4.2 Reachability Analysis Using Net Expansion

In the previous chapter, the relationship between the reachable markings of the original Petri net and its expansion was discussed. However, one parameter that still needs to be determined is the number of stages of net expansion. Knowledge of the required number of stages of net expansion, or an upper bound on the same is essential to transform the reachability problem of any given Petri net into a reachability problem for the expanded Petri net. The reachability problem (Definition 2.1.1) can be restated in terms of the expanded net as,

Definition 4.2.1 *Given a Petri net PN and a marking $\mathcal{M}_d$, the reachability problem determines if for some positive integer $\mathcal{N}$, ${}_{\mathcal{N}}\mathcal{M}_d \in \mathcal{R}({}_{\mathcal{N}}\mathcal{M}_0)$, where ${}_{\mathcal{N}}\mathcal{M}_0$ and ${}_{\mathcal{N}}\mathcal{M}_d$ are the equivalent initial and final markings for the expanded net ${}_{\mathcal{N}}PN$.*

Definition 4.2.1 redefines the reachability problem of any given Petri net into an equivalent problem for an acyclic net. Theorem 2.2.2 provides the sufficient condition for reachability in an acyclic net. Simply restated, the existence of a non–negative integer solution for the state equation is a necessary and sufficient condition for reachability in an acyclic net. The acyclic transformation presented earlier has reformulated the reachability problem of a general Petri net such that the results available for the acyclic net could be used for deciding the reachability. The one open question is the determination of the number of stages of net expansion.

For a given reachability problem, if the firing count vector[1] is known, then the number of stages can be bounded by the sum of the firing counts of the transitions, i.e., $\mathcal{N} \leq \sum_{j=1}^{n} u_j$. This is an upper bound because, at any given stage, the conflict–free transitions could be fired simultaneously. Thus, for a given firing count vector the reachability in a general Petri net could be decided without developing/searching the entire reachability tree. Theorem 3.1.1 in conjunction with Theorem 2.2.2 ensures that if there exists no non–negative integer solution to the state equation of the $\mathcal{N}$–stage expanded acyclic net, then the marking is not reachable with the given firing count vector or any firing count vector that adds to $\mathcal{N}$. This is true because the firing count vector is used only for bounding the number of stages of net expansion.

Example 4.2.1 *Consider the Petri net shown in Figure 3.1(a). The state equation of this net has a solution of the form* $(\alpha, \alpha, \alpha - 1)$ *for both* $(\mathcal{M}_d, \mathcal{M}_0) = (\{0,0,0,1\}, \{1,0,0,0\})$ *and* $(\mathcal{M}_d, \mathcal{M}_0) = (\{0,0,1,1\}, \{1,0,1,0\})$. *For all integer values of* $\alpha \geq 1$, *the transition firing count vector is non–negative and integer, satisfying the necessary condition for reachability stated in Theorem 2.2.1. However, reachability can be ascertained only by searching the reachability tree.*

A two–stage expansion of the net would show that the state equation is inconsistent[2] *for* $(\mathcal{M}_d, \mathcal{M}_0) = (\{0,0,0,1\}, \{1,0,0,0\})$. *In the case of* $(\mathcal{M}_d, \mathcal{M}_0) = (\{0,0,1,1\}, \{1,0,1,0\})$, *the solution to the state equation consists of firing the transition* t_1 *once in the first stage and transition* t_2 *once in the second stage, and none of the transporters need to be fired.*

However, in some situations the transition firing count vector cannot be determined exactly by solving the state equation (Equation 2.2). The structure of the state equation introduces certain issues related to uniqueness of the solution that need to be addressed while using it for

[1] a non–negative integer vector that is as long as the number of transitions, with each entry representing the number of times a transition needs to be fired

[2] there exists no non–negative firing count vector that satisfies the state equation

determining the number of stages of net expansion. In the following sections of this chapter, the use of the solution to the state equation as a means of bounding the net expansion is discussed. Finally, the algorithmic structure for reachability analysis using net expansion is documented.

4.3 Spurious Solutions and Reachability

The system state equation is a linear transformation of $u \in \mathcal{I}^n$ into $\Delta\mathcal{M} \in \mathcal{I}^m$. The range of this transformation is the subspace generated by $\Delta\mathcal{M}$, and is the subspace of $\mathcal{I}^m$ spanned by the columns of $\mathcal{A}$. Thus, the dimension of the range is the maximum number of linearly independent columns of $\mathcal{A}$, also called the rank of the incidence matrix $\mathcal{A}$.

In Section 2.2.2, the necessary condition for reachability based on the rank of the incidence matrix was discussed which can be restated as $\mathbb{R}[\mathcal{A}] \equiv \mathbb{R}[\mathcal{A} \mid \Delta\mathcal{M}]$. This condition checks for the existence of a solution to the system state equation. However, depending on the size of the incidence matrix and its rank, the solution to the system state equation can be either unique or infinite in number. The system state equation can be classified into one of the six categories shown in Table 4.2.

When $\mathbb{R}(\mathcal{A}) = m = n$ (§1) and $\mathbb{R}(\mathcal{A}) = n < m$ (§3), the system state equation has a unique solution. In all the other cases, the system state equation has a infinite number of numerical solutions [51]. If the state equation has an unique solution, then there is no ambiguity in the number of stages for net expansion. However, if the state equation does

Exactly determined system	§1. $\mathbb{R}(\mathcal{A}) = m = n$
	§2. $\mathbb{R}(\mathcal{A}) < m = n$
Over determined system	§3. $\mathbb{R}(\mathcal{A}) = n < m$
	§4. $\mathbb{R}(\mathcal{A}) < n < m$
Under determined system	§5. $\mathbb{R}(\mathcal{A}) = m < n$
	§6. $\mathbb{R}(\mathcal{A}) < m < n$

Table 4.2: State equation classification

not have a unique solution, then the number of stages for net expansion cannot be determined definitely just by solving the state equation. This ambiguity is because the solution to the state equation would have one or more free variables. Different numerical values for the free variables would result in different firing count vectors, and hence, different values for the required number of stages. The following example illustrates these issues.

Example 4.3.1 *Consider the Petri net shown in Figure 4.1, with an initial marking of* $\mathcal{M}_0 = \{1, 0, 0, 0, 0, 0, 0\}$. *The problem is to determine if the desired final marking of* $\mathcal{M}_d = \{0, 0, 0, 0, 0, 1, 1\}$ *is reachable from the given initial marking. If the acyclic transformation is to be used for deciding reachability, then the number of stages of expansion has to be determined first. The state equation for this net can be classified as a rank deficient exactly determined system (§2). The incidence matrix has a rank of five, two less than the maximum possible. Hence, the general solution to the state equation* $u = \{\alpha_1, \alpha_2, \alpha_1, \alpha_1, \alpha_2, \alpha_2, \alpha_1 + \alpha_2 - 1\}$ *has two free variables. Different numerical values would give different firing count vectors, and hence different net expansion stages.*

The issue to be addressed is the determination of appropriate numerical values for the free variables. Then the reachability problem could be decided using the acyclic transformation with the number of stages equal to the sum of the transition firing counts determined using those numerical values for the free variables. The next few sections contain ways of

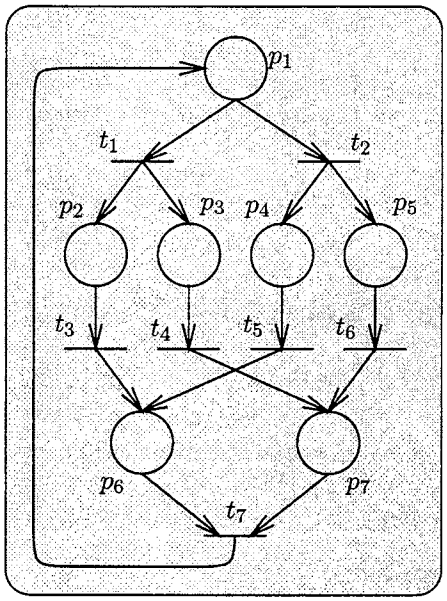

Figure 4.1: Petri net example to demonstrate spurious solutions to state equation

reducing the number of free variables and consequently, the complexity in determining the appropriate number of net expansions. In order to address this issue, a more detailed examination of the source of degeneracy in the state equation is carried out. This is followed by the development of some potential ways to reduce the degeneracy under certain situations.

4.3.1 Source of system degeneracy

The linear dependence among the vectors of the incidence matrix reduces its rank such that it is less than the maximum permissible rank, resulting in the introduction of free variables in the solution to the state

equation. This increases the complexity of searching the state space. The discussions in the following sections are directed towards eliminating these linear dependencies, if possible, and analyzing the impact of such a modification.

The linear dependency among the places (transitions) is called place (transition) invariant in Petri net terminology. As previously mentioned in Section 1.2.2, the weighted sum of all the tokens in a place invariant set remains the same in all possible markings, while firing the transitions in the transition invariant set returns the net to the original marking. The presence of invariant sets are crucial for the correctness of a model (system) due to its influence on system properties such as boundedness and reversibility. Ironically, with regard to the solution of the system state equation, the presence of invariant sets is detrimental. A formal definition of place and transition invariant sets is presented in Definitions 4.3.1 and 4.3.2.

Definition 4.3.1 *An $m \times 1$* ***non–negative*** *integer vector $x \in \mathfrak{N}^m$ is called a place invariant or P–invariant if,*

$$\mathcal{A}^T x = 0 \tag{4.1}$$

Definition 4.3.2 *An $n \times 1$* ***non–negative*** *integer vector $y \in \mathfrak{N}^n$ is called a transition invariant or T–invariant if,*

$$\mathcal{A} y = 0 \tag{4.2}$$

A P–invariant set is denoted by π, and the set of all place invariant sets of a Petri net is denoted by Π.

It is often mentioned in literature that there are $(m - \mathbb{R}(\mathcal{A}))$ minimal P–invariant sets in a Petri net *PN* with m places [11]. However, investigations into the reasons for rank deficiency revealed a discrepancy in this

statement. The reason for this discrepancy is due to the non–negativity requirement on the place invariant set as specified in Definition 4.3.1. The following example elucidates the issues involved.

Example 4.3.2 *Consider the Petri net shown in Figure 4.2. This net has 8 places and 8 transitions. In this net there are no P–invariant sets, i.e.,*

$$\nexists\, x : \mathcal{A}^T x \;=\; 0,\; x_i \geq 0,\; i = \{1, 2, \ldots, m\}$$

But the rank of the incidence matrix for the Petri net is $\mathbb{R}(\mathcal{A}) = 7$. *For* $x^T = [0, -1, 0, 1, 1, 0, 1, 0]$, $\mathcal{A}^T x = 0$, *and hence the* $|\Pi| \neq m - \mathbb{R}(\mathcal{A})$. *Thus the number of P–invariant sets of a Petri net is not always the difference between the number of places and the rank of the incidence matrix.*

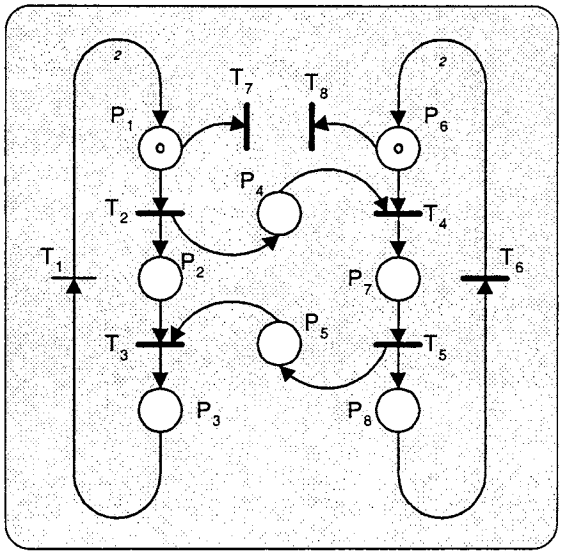

Figure 4.2: An illustration for place invariant analysis

As the example illustrates, the reason for the discrepancy in the relationship connecting the number of P–invariants and the rank of the

incidence matrix is the non–negativity restriction in the definition of the P–invariant sets (Definition 4.3.1). Hence, the concept of *extended place and transition invariant sets* is introduced [47].

Definition 4.3.3 *An $m \times 1$ integer vector $x \in \mathfrak{I}^m$ is called the extended place invariant or ExP–invariant if,*

$$\mathcal{A}^T x = 0 \tag{4.3}$$

Definition 4.3.4 *An $n \times 1$ integer vector $y \in \mathfrak{I}^n$ is called the extended transition invariant or ExT–invariant if,*

$$\mathcal{A} y = 0 \tag{4.4}$$

The extended place and transition invariant sets are nothing but the null spaces along the row and column spaces, respectively. These null spaces introduce degeneracy, and hence free variables in the solution to the state equation. When $\mathbb{R}(\mathcal{A}) = m < n$, the solution to system state equation would have free variables, in spite of having a full rank incidence matrix. However, when the rank is less than the maximum permissible (§2, §4, & §6) due to the nullity along the row or column space, the null space introduces additional free variables. If the null space can be eliminated, then there will be no introduction of free variables in the solution to the state equation. If the null space could be eliminated or partially reduced, the impact of it on the system dynamics needs to be understood. In the next section, methods for breaking the nullity are explored for the purposes of solving the state equation.

4.4 Invariant Free Petri Net

The null space in the sub–space spanned by the rows of the incidence matrix (ExP–invariant) has the special property that the weighted sum

of the tokens remains a constant in all the reachable markings of the Petri net. If this token balance could be broken or destabilized, then the place invariant would become void implying that the null space would reduce along one of the dimensions. The token balance could be broken by increasing the arc weights of one of the arcs incident into one of the places in the place invariant set. However, this would leave additional tokens in the place, and thus altering the state space of the system. This necessitates that these additional tokens be removed by some means. Adding sink transitions to the destabilized places is an option. These transitions that are used to remove the additional tokens are termed as *stabilizers*. Though this approach would reduce the null space, the effective number of free variables would remain unchanged due to the addition of the sink transition. Hence, in order for the approach of destabilizing the place invariant set to be effective, the null space should reduce faster than the introduction of any additional dimension. The following example would help illustrate these issues.

Example 4.4.1 *Consider the following Petri net with 7 places and 6 transitions defined as follows:*

Places: $P = \{p_1, p_2, p_3, p_4, q_1, q_2, q_3\}$

Transitions: $T = \{t_1, t_2, t_3, r_1, r_2, r_3\}$

Directed arcs: $F \subseteq (P \times T) \cup (T \times P)$, *expressed in terms of the pre and post sets of the places are,*

Place pre-sets: ${}^\bullet p_1 = \{t_3\}$, ${}^\bullet p_2 = \{t_1\}$, ${}^\bullet p_3 = \{t_2\}$, ${}^\bullet p_4 = \{t_2\}$, ${}^\bullet q_1 = \{t_1, t_2, r_2\}$, ${}^\bullet q_2 = \{r_1\}$, ${}^\bullet q_3 = \{r_3\}$

Place post-sets: $p_1^\bullet = \{t_1\}$, $p_2^\bullet = \{t_2\}$, $p_3^\bullet = \{t_3\}$, $p_4^\bullet = \{t_1\}$, $q_1^\bullet = \{r_1\}$, $q_2^\bullet = \{t_1, t_2, r_3\}$, $q_3^\bullet = \{r_2\}$

Arc weights: $w(p,t) = w(t,p) = 1, \forall (p \times t) \wedge (t \times p) \in F$

The ExP–invariant sets are,

$$\begin{aligned} \pi_1 &= \{1,0,1,-1,0,0,0\} \\ \pi_2 &= \{0,1,0,1,0,0,0\} \\ \pi_3 &= \{0,0,0,0,1,1,1\} \end{aligned}$$

The place invariant set π_1 could be destabilized by changing $w(t_3,p_1)=2$. This would increase the rank of the incidence matrix by one. However, if a sink transition on place p_1 is introduced to remove any additional tokens, the maximum permissible rank changes from 6 to 7. Hence, the number of free variables in the state equation would remain the same.

As the example illustrates, independent introduction of sink transitions to stabilize the places in the ExP–invariant that was destabilized (by changing the arc weights leading into the places) does not help in reducing the number of free variables in the solution to the state equation. The introduction of additional dimensions, and hence the number of free variables could be slowed if the accumulated tokens in multiple destabilized places could be removed simultaneously. Hence, the objective should be to have a common stabilizer for multiple destabilized places. Two possible ways of achieving this objective are,

Utilizing the base transitions: In this approach, instead of introducing additional sink transitions to remove additional tokens, the possibility of using the transitions that are part of the actual Petri net themselves is explored. The column space is maintained a constant while reducing the null space along the row space. However, note that this introduces additional conditions to be satisfied for the transitions affected to be enabled.

Multi purpose sink transitions: In this approach, multiple destabilized places are to be stabilized by a single sink transition, which

ensures that the drop in the row null space is faster than the increase in the column space.

Both these procedures depend on the knowledge of the firing count relationship among the transitions. It is not the knowledge of exact firing count vector that is essential, rather relationship in terms of the free variables. Because these procedures rely on the firing count vectors, they are very specific to a reachability problem. The conditions to be satisfied for each of the above two approaches to be effective are discussed next.

4.4.1 Utilizing Base Transitions

As the name suggests, the transitions that are already part of the Petri net are evaluated for use as transitions that remove the additional tokens accumulated. If a transition has to qualify for this purpose, then the firing count of that transition has to be equal to or a multiple of the firing count of the transition that destabilizes the place. Suppose in a Petri net place p_i is destabilized by increasing the weight of the arc incident from transition t_j to place p_i, for a transition $t_{j\prime}$ to qualify as a stabilizer for place p_i, the firing count of transition $t_{j\prime}$ should be equal to or a multiple of transition t_j. If this condition is satisfied, then any additional tokens left behind by transition t_j could be removed by transition $t_{j\prime}$. Additionally for the transition $t_{j\prime}$ to be acceptable, transitions t_j and $t_{j\prime}$ should not be part of the same ExT–invariant. If this condition is violated, then the destabilization of the place does not help in increasing the rank of the incidence matrix.

Note that by using transition $t_{j\prime}$ as the stabilizer for the destabilized

place p_i, the enabling condition for the transition $t_{j'}$ has been changed. This additional condition on the transition will be satisfied if the transition $t_{j'}$ is fired only after the transition t_j has fired. The characterization of these transitions is one of the problems that needs to be investigated further.

The following example should help illustrate this technique.

Example 4.4.2 *Consider the Petri net defined in Example 4.4.1. For this Petri net, consider the reachability problem which seeks to determine if* $\mathcal{M}_d = \{0,0,10,1,0,0,1\}$ *is reachable from* $\mathcal{M}_0 = \{10,0,0,1,0,0,1\}$. *The maximum permissible rank is 6 for this Petri net, and it has a rank of 4. Since there are 6 transitions, and the incidence matrix has a rank of 4, the solution to the state equation will have 2 free variables. The general solution to the state equation for the set of markings specified above is* $u = \{\alpha, \alpha, \alpha-10, \beta, \beta, 2\alpha+\beta\}$. *This Petri net has 3 ExP–invariant sets, given by* $\Pi = \{a_1, a_2, a_1, a_2 - a_1, a_3, a_3, a_3\}$; *and 2 ExT–invariant sets, given by* $\Gamma = \{b_1, b_1, b_1, 2b_1 + b_2, b_2, b_2\}$. *Place* p_2 *could be destabilized by setting* $w(t_1, p_2) = 2$. *This would mean that every time transition* t_1 *is executed, two tokens would be added to place* p_2 *instead of one. This additional token has to be removed in order to maintain the system dynamics. The candidate transition for acting as stabilizer for place* p_2 *would be transition* t_2 *since both* t_1 *and* t_2 *have the same firing counts. However, it is not valid because, both the transitions* t_1 *and* t_2 *are part of the same ExT–invariant set.*

4.4.2 Multi–purpose Sink Transitions

In this technique, additional sink transitions are added to the Petri net to act as stabilizers. While adding these additional sink transitions, they are linked to multiple destabilized places that are part of different ExP–invariant sets. This requires that the destabilized places have exactly the same number of additional tokens or even multiples of additional tokens. In order to reduce the number of free variables in the solution to state

equation, a newly added sink transition should act as a stabilizer for at least two destabilized places.

Note that the destabilization of places and the addition of stabilizers changes the set of all reachable markings in the Petri net. However, all reachable markings of the original Petri net will be part of the reachability set of the modified Petri net. This can be verified by imposing the condition that the additional tokens that destabilize the places would be moved from those places only through the stabilizers. Hence, all the markings in the original Petri net have equivalent markings in the modified net.

The following example should help illustrate this technique.

Example 4.4.3 *Continuing the Example 4.4.2, places p_2 and p_3 could be destabilized by setting $w(t_1, p_2) = 2$ and $w(t_2, p_3) = 2$. Places p_2 and p_3 are part of different ExP-invariant sets. Note the choice of transitions that destabilize the places. In the solution to the state equation, both have the same firing count, and hence would leave the same number of additional tokens. These places could now be stabilized by a common stabilizer.*

4.4.2.1 Summary

Both methods provide ways to reduce the number of free variables in the solution to the state equation. Though it is desirable to remove all the free variables from the solution, it may not be possible. However, the reduction in the free variables count does reduce the uncertainty in the solution. For a specific reachability problem analysis, the free variables are to be assigned various numerical values iteratively to determine the number of stages of net expansion. The following section formalizes these

ideas into an algorithm for reachability analysis.

4.5 Reachability Analysis Using Net Expansion Revisited

At this stage, approaches discussed in this chapter and the previous chapter are combined to determine reachability in a general Petri net, and presented in an algorithmic form (Algorithm 5). In the algorithm, the solution to the state equation of the original Petri net and the invariant–free version of the same are used to determine the number of stages of net expansion by acyclic transformation. Once the number of stages is determined, the expanded version of the net is developed and checked for reachability using the necessary and sufficient condition for reachability in acyclic Petri nets.

Since the required number of stages for conclusively determining reachability may not be known or bounded due to the existence of free variables in the solution to state equation, Algorithm 5 may terminate without a conclusion. Hence, further research is required to bound the required number of stages for a reachability problem.

Require: In a Petri net determine if $\mathcal{M}_d \in \mathcal{R}(\mathcal{M}_0)$

1: Find the general solution to the state equation $\mathcal{M}_d - \mathcal{M}_0 = \mathcal{A}u$
2: Reduce the number of free variables in the solution by converting it into invariant–free Petri net
3: **if** The invariant–free net has a unique solution to state equation **then**
4: Add the elements of the unique solution to determine the required number of stages of net expansion
5: Using acyclic transformation build the expanded acyclic Petri net
6: **if** state equation of the acyclic expanded Petri net has non–negative integer solution **then**
7: marking $\mathcal{M}_d$ is reachable from $\mathcal{M}_0$
8: **else**
9: marking $\mathcal{M}_d$ is not reachable from $\mathcal{M}_0$
10: **end if**
11: **else**
12: **repeat**
12: Set free variables in the general solution to state equation to increasing non-negative integer values and determine a candidate number of stages for net expansion
13: **if** state equation of the acyclic expanded Petri net has non–negative integer solution **then**
14: marking $\mathcal{M}_d$ is reachable from $\mathcal{M}_0$
15: **else**
16: marking $\mathcal{M}_d$ is not reachable from $\mathcal{M}_0$
17: **end if**
18: **until** Marking is determined to be reachable or time allocated has expired
19: **end if**

Algorithm 5: Reachability Using Acyclic Transformation

Chapter 5

Some Related Problems

Summary

In this chapter, two problems that are closely related to the reachability problem are studied. First, in the submarking reachability problem, the desired final marking is specified for only a subset of the places. The relationship between the submarking reachability problem and the regular reachability problem is established. Next, the implications of the new reachability results in tackling the deadlock avoidance problem are discussed.

5.1 Submarking Reachability

Whereas, in the reachability problem studied thus far, a complete specification of the final marking is available, in the submarking reachability problem the final marking is specified for only a subset of the places. There has been published work related to submarking reachability of marked graphs starting with the article by Kumagai et al. [30]. For a marked graph, Kumagai et al. provide an approach for constructing a marking with the specified submarking that may or may not be reachable from the given initial marking. This work was later expanded to study the relationship of the sub–marking reachability problem with network programming problems [7]. The results presented in this section build

on these ideas and extend the theory of submarking reachability to a general Petri net.

Definition 5.1.1 *The Submarking Reachability Problem. Given a Petri net PN, and marking for a subset of the places $\mathcal{M}_d(p^c) : p^c \in P^c \subset P$, the submarking reachability problem determines if there exists a marking $\mathcal{M} \in \mathcal{R}(\mathcal{M}_0)$ with the specified token distribution $\mathcal{M}_d(p^c)$ for the places $p^c \in P^c$.*

The places for which a final token distribution is specified are called *controlled places* (P^c), while the rest are called *free places* (P^f). Similarly, the transitions that are incident on any of the controlled places are called *controlled transitions* (T^c), while the rest are called *free transitions* (T^f). These notions are formally defined below [7].

Definition 5.1.2 *Controlled and Free Places. A place p^c is said to be* controlled *in a given final marking $\mathcal{M}_d$ if $\mathcal{M}_d(p^c)$ is defined. A place p^f is said to be* free *in a given final marking $\mathcal{M}_d$ if $\mathcal{M}_d(p^c)$ is not defined.*

Definition 5.1.3 *Controlled and Free Transitions. A transition t^c is said to be* controlled *in a given final marking $\mathcal{M}_d$ if at least one of the places incident on the transition is in the set of controlled places, i.e., if $\{{}^\bullet t^c \cup t^{c\bullet}\} \cap P^c \neq \{\emptyset\}$, then t^c is said to be controlled. If $\{{}^\bullet t^c \cup t^{c\bullet}\} \cap P^c = \{\emptyset\}$, then t^c is said to be a free transition.*

Using these basic categorization of places and transitions, the relationship between the sub–marking and full-marking reachability problems is established.

5.1.1 SubM Petri Net

Let there be m^c controlled places ($m^c =\mid P^c \mid$) and n^c controlled transitions ($n^c =\mid T^c \mid$), and m^f free places ($m^f =\mid P^f \mid$) and n^f free

transitions ($n^f = \mid T^f \mid$). Note that $m = m^c + m^f$ and $n = n^c + n^f$. This classification of places and transitions partitions the incidence matrix as follows:

$$\mathcal{A} = \begin{pmatrix} \mathcal{A}^{cc} & \mathbf{0} \\ \mathcal{A}^{fc} & \mathcal{A}^{ff} \end{pmatrix} \begin{matrix} \text{controlled} \\ \text{free} \end{matrix} \tag{5.1}$$

where $\mathcal{A}^{cc}$ is an $m^c \times n^c$ matrix corresponding to the relationship between places and transitions that are controlled, while $\mathcal{A}^{fc}$ and $\mathcal{A}^{ff}$ are $m^f \times n^c$ and $m^f \times n^f$ dimensioned matrices corresponding to the relationship of free places as influenced by controlled and free transitions, respectively. With these partitions of the incidence matrix, the state equation can be rewritten as

$$\begin{pmatrix} \mathcal{M}_d^c \\ \mathcal{M}_d^f \end{pmatrix} = \begin{pmatrix} \mathcal{M}_0^c \\ \mathcal{M}_0^f \end{pmatrix} + \begin{pmatrix} \mathcal{A}^{cc} & \mathbf{0} \\ \mathcal{A}^{fc} & \mathcal{A}^{ff} \end{pmatrix} \begin{pmatrix} u^c \\ u^f \end{pmatrix} \tag{5.2}$$

where u^c and u^f are the firing counts of controlled and free transitions, respectively. The above state equation can be rearranged as follows:

$$\begin{aligned} \begin{pmatrix} \mathcal{M}_d^c \\ \mathbf{0} \end{pmatrix} &= \begin{pmatrix} \mathcal{M}_0^c \\ \mathcal{M}_0^f \end{pmatrix} + \begin{pmatrix} \mathcal{A}^{cc} & \mathbf{0} \\ \mathcal{A}^{fc} & \mathcal{A}^{ff} \end{pmatrix} \begin{pmatrix} u^c \\ u^f \end{pmatrix} - \begin{pmatrix} \mathbf{0} \\ \mathcal{M}_d^f \end{pmatrix} \\ &= \begin{pmatrix} \mathcal{M}_0^c \\ \mathcal{M}_0^f \end{pmatrix} + \begin{pmatrix} \mathbf{0} & \mathcal{A}^{cc} & \mathbf{0} \\ -I^f & \mathcal{A}^{fc} & \mathcal{A}^{ff} \end{pmatrix} \begin{pmatrix} \mathcal{M}_d^f \\ u^c \\ u^f \end{pmatrix} \end{aligned} \tag{5.3}$$

where I^f is an identity matrix of dimension $m^f \times m^f$.

By observing the rearranged state equation (Equation 5.3), it can be seen that the new incidence matrix corresponds to a Petri net with m places and $n + m^f$ transitions, where the additional m^f transitions are sink transitions on the free places of the original Petri net. This net is called as the *SubM Petri net*, and is formally defined as follows.

Definition 5.1.4 *SubM Petri Net. Given a Petri net PN and desired token distribution in a subset of the places of the Petri net, the incidence matrix of its* **SubM Petri Net** *sPN of dimension $m \times (n + m^f)$ is,*

$$\mathcal{A}^s = \begin{pmatrix} \mathbf{0} & \mathcal{A}^{cc} & \mathbf{0} \\ -I^f & \mathcal{A}^{fc} & \mathcal{A}^{ff} \end{pmatrix} \tag{5.4}$$

The reachability set of the SubM Petri net is denoted by $\mathcal{R}^s(\mathcal{M}_0)$. Next, the submarking reachability criteria are defined in terms of its SubM Petri net.

5.1.2 Submarking Reachability Criteria

In a SubM Petri net, the sink transitions on the free places could be used to remove any tokens that might be accumulated in them. Assuming that in the reachability set of the original Petri net there exists a marking with the desired token distribution on the controlled places, the tokens in the free places if any could be removed with the sink transitions. Except for the sink transitions on the free places the net structure of both the original Petri net and its SubM derivative are identical. Other than removing any tokens in the free places, the sink transitions cannot affect the system dynamics in any way. In fact, if the sink transitions are never executed, the reachability set of the original Petri net and its SubM derivative would be identical. Hence the following Lemma.

Lemma 5.1.1 *Given a Petri net PN, every marking in the reachability set of the Petri net has an equivalent marking in its SubM Petri net's sPN reachability set with the same token distribution, i.e., $\forall \mathcal{M} \in \mathcal{R}(\mathcal{M}_0), \exists \mathcal{M}^s \in \mathcal{R}^s(\mathcal{M}_0) : \forall p \in P, \mathcal{M}^s(p) = \mathcal{M}(p)$*

However, the converse may not be true. In other words every reachable marking of SubM Petri net may not have an equivalent marking in the reachability set of its parent net. For the purposes of submarking reachability, the converse is not relevant.

Utilizing the functionality of the sink transitions, the submarking reachability problem of the original Petri net can be posed as a reachability problem in its SubM Petri net. When the tokens in the free places are removed by the sink transitions leaving the free places empty, the submarking reachability problem of the original Petri net is equivalent to reachability of a marking with zero tokens in the free places and identical token distribution in the controlled places of the SubM Petri net. If a given submarking is reachable, then the associated token distribution of the free places in the original Petri net would be the firing counts of the sink transitions on the free places in the SubM Petri net. This equivalence of the reachability problem is formalized next.

Theorem 5.1.1 *A specified submarking $\mathcal{M}_d$ is reachable in a Petri net PN if in the associated SubM Petri net $_{\mathcal{S}}PN$ the marking $\mathcal{M}_d^{\mathcal{S}} \in \mathcal{R}^{\mathcal{S}}(\mathcal{M}_0)$, where $\forall p^c \in P^c, \mathcal{M}_d(p^c) = \mathcal{M}_d^{\mathcal{S}}(p^c)$ and $\forall p^f \in P^f, \mathcal{M}_d^{\mathcal{S}}(p^f) = 0$.*

Theorem 5.1.1 provides a means for deciding submarking reachability by posing it differently. Discussion in the previous chapters shows that deciding reachability is not straightforward for all situations, and hence, a transformation procedure is introduced to utilize known reachability results. More specifically, much emphasis is laid on a class of Petri nets called the acyclic Petri nets. It is of interest to note here that if the original Petri net is acyclic, introduction of the sink transitions for the

free places does not alter the acyclic nature. The sink transitions do not introduce any loops and an acyclic Petri net retains this property with the addition of the sink transitions.

Property 5.1.1 ***Preservation of the acyclic net structure.*** *If a Petri net PN is acyclic, then any of the associated SubM Petri net is also acyclic.*

The above property of the SubM Petri net, allows the use of known necessary and sufficient condition of reachability for acyclic nets (Theorem 2.2.2).

5.2 Deadlock Avoidance Problem

Deadlock avoidance is concerned with avoiding deadlock states of the system, and initiating automatic recovery procedures when a deadlock cannot be avoided. The deadlock avoidance problem is known to be NP–complete [21]. It has been shown to be recursively equivalent to the reachability problem [9].

Deadlock prevention problem is closely associated with the deadlock avoidance problem. Deadlock prevention is concerned with falsifying one or more of the necessary conditions for deadlock. It is a static policy and it generally results in poor resource utilization. The deadlock avoidance techniques counter the poor resource utilization by using dynamic resource allocation policies.

The deadlock avoidance policies try to determine if by allocating a resource to a requesting proces the system would get into deadlock situations. More often, the deadlock states might be deeply embedded in the state space, and may not be readily apparent if we were to only look

for deadlock states as a result of allocating a resource. An approach to deadlock avoidance is to restrict the number of processes that can simultaneously request a resource [3]. Though this approach guarantees to avoid all deadlocks, it is very conservative, meaning, it could unnecessarily restrict safe resource allocations. Another approach to deadlock avoidance is the n–step look ahead logic proposed by Viswanadham et al. [53]. Theoretically, if the number of look ahead steps are not bounded, then it would be possible to avoid all deadlocks. However, this is practically infeasible. When n is limited, this algorithm cannot guarantee the avoidance of all deadlock states.

This restriction on the number of look–ahead steps can be eliminated if it is possible to perform reachability analysis. The basic idea here is to determine if after allocating the resource to a process, the system can reclaim the resource. If this can be established, then the impact of allocating the resource is none with regards to deadlock. For a Petri net model, this can be formulated as a submarking reachability problem, which as the discussions in the previous section show, in turn can be formulated as a full–marking reachability problem. The acyclic transformation and related techniques provide a promising avenue for reachability analysis in general Petri nets, and hence could have an impact on the deadlock avoidance problem also. A detailed investigation is required for practical application of this approach.

Example 5.2.1 *Consider a simple production system (See Figure 5.1) consisting a load–unload station, a single–machine work station, and an AGV for transporting parts between them [53, 19]. Further, there are no buffers in the system, the raw parts are always available, and the AGV can carry only one part at a time. There are two deadlock states in this system:*

Deadlock state 1: *The AGV with a raw part is waiting to load it on the machine that has a finished part.*

Deadlock state 2: *The AGV has been assigned to carry a finished part from the machine, while the machine which is empty is waiting for a raw part .*

A simplified Petri net model of this system is shown in Figure 5.2, with the explanation for the nodes in Table 5.1 [19]. The reachability graph of this reduced Petri net is given in Figure 5.3. The marking legends are given in Table 5.2. The marking $\mathcal{M}_5$ corresponds to the above defined deadlock state 1, while the marking $\mathcal{M}_7$ corresponds to the above defined deadlock state 2.

From the reachability graph, it can be seen that the system enters a deadlock state if the AGV is assigned to pick up a raw part from the load–unload station, when a part is already being processed by the machine. Also, if the AGV is assigned to pick up a finished part from the machine when the machine is empty, the system enters a deadlock state.

The key resource allocation decision is taken when the place A_3 is marked. The place A_3 is marked in the markings $\mathcal{M}_2$ and $\mathcal{M}_4$. In both these markings, the transitions t_3 or t_4 can be fired. In order to avoid deadlock, when the system is in the marking $\mathcal{M}_2$, transition t_3 needs to be fired, and in marking $\mathcal{M}_4$, the transition t_4 needs to be fired. This information was obtained by analyzing the reachability graph. However, such an analysis may not be always possible.

Assuming that the reachability graph is unavailable, when the system is in marking $\mathcal{M}_2$, the two possible resource allocation decisions corresponding to transitions t_3 and t_4 need to be evaluated. By firing the transition t_3 the system enters marking $\mathcal{M}_3$. From this marking a sub–marking reachability problem is solved in which the final desired marking has the place A_3 marked again. This problem is solved by using the acyclic transformation technique and the SubM Petri net modification. Using these two transformations, it can be verified that there exists a marking in which the resource allocation decision can be made again. However, if the transition t_4 is fired, the place corresponding to resource allocation decision will not be marked again. Hence, in state $\mathcal{M}_2$, the transition t_4 should not be fired.

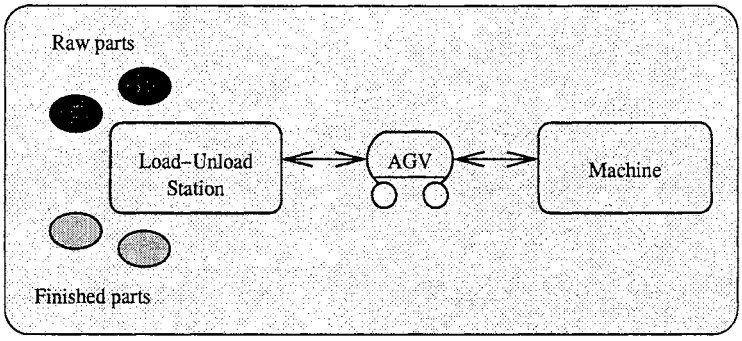

Figure 5.1: Deadlock avoidance: An illustration

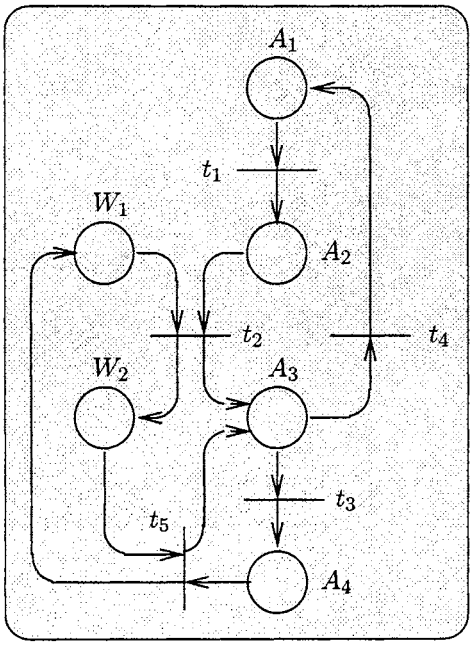

Figure 5.2: Reduced Petri net model for the system in Figure 5.1

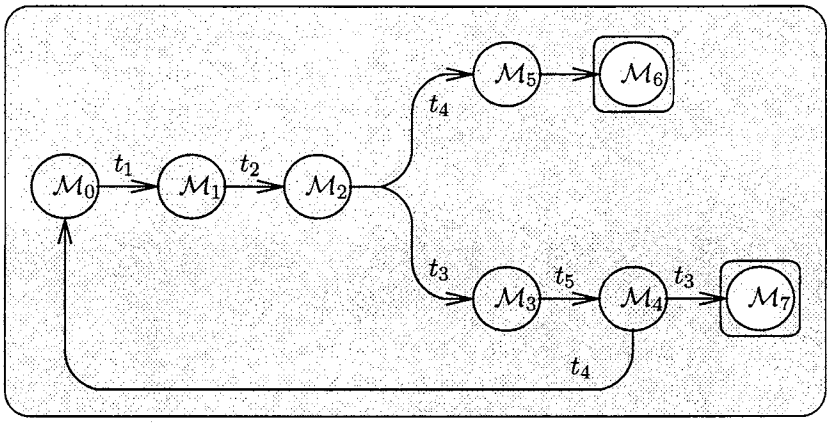

Figure 5.3: Reachability graph for the Petri net model in Figure 5.1

W_1	Machine waiting for AGV to load raw part
W_2	Machine waiting for AGV to unload finished part
A_1	AGV waiting for raw part
A_2	AGV waiting at machine to load raw part
A_3	AGV waiting for resource allocation decision
A_4	AGV waiting to unload finished part from machine
t_1	Load raw part on the AGV and move the AGV to the machine
t_2	Load raw part on the machine from the AGV
t_3	Decision: AGV to wait for finished part at the machine
t_4	Decision: AGV to wait for raw part at the load–unload station
t_5	Unload finished part from the machine on to the AGV

Table 5.1: Explanation for the places and transitions in the Petri net in Figure 5.2

Marking/Place	W_1	W_2	A_1	A_2	A_3	A_4
$\mathcal{M}_0$	1	0	1	0	0	0
$\mathcal{M}_1$	1	0	0	1	0	0
$\mathcal{M}_2$	0	1	0	0	1	0
$\mathcal{M}_3$	0	1	0	0	0	1
$\mathcal{M}_4$	1	0	0	0	1	0
$\mathcal{M}_5$	0	1	1	0	0	0
$\mathcal{M}_6$	0	1	0	1	0	0
$\mathcal{M}_7$	1	0	0	0	0	1

Table 5.2: Marking legend for the reachability graph in Figure 5.3

Chapter 6

An Application of Reachability Analysis in Discrete Manufacturing

Summary

The reachability problem lends itself very well for decision problems involving discrete optimization. This relationship is utilized to formulate the due date quotation problem in a discrete manufacturing environment as a reachability problem. To facilitate this, Petri net constructs for modeling a production system are developed. The advantage of following the Petri net reachability approach for this problem is that the analysis also provides an operational road map for the system.

6.1 Introduction

The increasing availability of system wide inventory information with the implementation of supply chain integration software is enabling manufacturing enterprises to respond quickly with due date and price quotes. These advanced order processing methodologies are being necessitated for online buy and sell systems also. The Web interface makes the process of buying and selling easier for the consumers by enabling them to easily check among the competitors. Hence, decision support systems that quote realistic due dates and prices are needed. There is a growing

interest in real–time decision making based on the current state of the system to deal with custom orders of small sizes. Hence, a real–time decision support system would have to coordinate the lead time quotation decisions with scheduling and inventory management systems. The decisions would have to take into account the uncertainties in the inventory and the production system and the current state of the system. The importance of accounting for the current system status and uncertainties was first noted by Conway [8]. This was much before the advent of technology that makes it possible. The approach developed in this chapter addresses the due date quotation problem that accounts for the current state of the production system.

Typically, due dates are determined by adding a certain multiple of the flow time's standard deviation to the steady state expected flow time [23]. The steady state approach works well for stable environments with large lot sizes. However, this approach falls short in a dynamic environment with increasing product variety and small lot sizes. There are a number of order due date quoting algorithms in the job–shop scheduling literature based on a number of assignment rules and control methods [5, 24, 13]. These typically do not consider the current state of the system. The performance improvements to be gained by considering the system status are well documented in literature [14, 4, 31]. This leads to the classification of order due date assignment policies as those that consider current system status and those that do not consider the current system status [5, 40]. The research described in this chapter, belongs to the category that considers the current system status.

A mathematical programming model is the underlying structure of most due date assignment models in the literature. Recently, models

that dynamically allocate and reallocate resources have been developed. However, in the operational phase, materials are released to the shop floor based on a production plan, with very little or no guidance to achieve operational objectives as decided by the model. To overcome this shortcoming, an integrated approach to the due date assignment problem is proposed. This approach, in addition to supporting due date decisions, provides an operational road map to achieve those objectives.

Traditionally, the application of Petri nets to manufacturing systems has been in the area of supervisory control to prevent the system from entering deadlock states [19, 6, 53], and scheduling in flexible manufacturing systems [32, 49]. In general, reachability analysis forms the basis of such applications. It is shown here that the reachability analysis lends itself well for due date quotation decision support systems also. The current state of the production system is mapped to the initial state of its Petri net model. Any new order is transformed into an appropriate desired state definition, and reachability analysis is used to determine if there exists a sequence of events that makes the order completion possible. In addition to addressing the decision problem, the analysis also provides an operational road map.

The rest of the chapter is organized as follows. A description of the Petri net constructs suitable for a due date quoting model is presented in Section 6.2. The use of these constructs for due date analysis is discussed in Section 6.3. The modeling approach and the research issues ahead are discussed in Section 6.3.

6.2 Due Date Quoting Petri Net Model

A Petri net modeling approach is described in this section, which can be used to determine what orders to accept and to recommended order completion times to facilitate due date negotiations. The models developed are for a given plant configuration, the set of currently accepted orders and their quoted due dates, and penalty for violating those quoted due dates. The model balances the resource requirements for the currently accepted orders with that of the new orders and forms the basis of an approach to determine the optimal way of processing those orders. The approach presented here does not consider the future demand to come. However, it is to be noted that consideration of future demand would make the resulting solution more robust. This is due to the evaluation of the trade–offs not only with the currently accepted orders, but also with demand to come. This restriction has been placed since the thrust of this chapter is to explore a new comprehensive approach to the due date quoting system.

For the purposes of this analysis, it is assumed that the order processing time at a resource is known. It has been widely acknowledged that for decisions over a relatively short time horizon, the processes though stochastic by nature can be deemed to be deterministic [20].

6.2.1 Notation and Problem Description

The due-date problem is analyzed here in a myopic setting. The model develops and delivers a production plan based on the current system status. The granularity of the production plan is directly a function of the granularity of the model. Let $\mathbb{O}$ and $\hat{\mathbb{O}}$ represent the set of newly arrived

orders and the previously accepted orders in the system, respectively. The set of resources in the production system is given by $\mathbb{R}$. Let the processing time of the k^{th} processing step of order i be specified by the parameter μ_{ik}^{p}. The mapping function $M_i(k) \in \mathbb{R}$ specifies the resource to be used in the k^{th} processing stage of order i; $N(j) \subseteq \{\mathbb{O} \cup \hat{\mathbb{O}}\}$ specifies the set of all orders to be processed by resource j; and $R(i) \subseteq \mathbb{R}$ specifies the set of all resources required for processing order i. Let $m =\mid \{\mathbb{O} \cup \hat{\mathbb{O}}\} \mid$, $n_j =\mid N(j) \mid$, $o =\mid \mathbb{R} \mid$, and $s_i =\mid R(i) \mid$.

Each order $i \in \hat{\mathbb{O}}$ has a committed due date of T_i. The penalty of not satisfying the committed due date is given by the unit cost function P_i (penalty for each time the order i is delayed beyond the promised due date). The revenue gain from an order $i \in \{\mathbb{O} \cup \hat{\mathbb{O}}\}$ is given by f_i.

6.2.2 Modeling Constructs

In this section, two modeling constructs that can be used to build the model of a system for the purpose of due date quotation analysis are introduced. The first construct models a resource, while the second construct is used to track the performance of the system. These constructs use the notion of a *stage*. Whenever a resource completes processing an order, the resource is said to have passed from one stage on to the next. Alternatively, a resource processes only one order at any of its "stages". Hence, in the model, a resource has as many stages as the number of orders that need processing at that resource. Using this notion of a stage, the two Petri net modeling constructs are defined next.

6.2.2.1 Resource Construct

The Petri net model of a resource has as many stages as there are orders that need processing at that resource. At each stage, the construct is so designed that only one order could be processed. However, at each stage, any of the orders ready to be processed by the resource can be selected for processing. This gives rise to five types of places and three types of transitions:

Places:

- Places to indicate that an order is available to be processed by a resource at each of its stages.
- Places to indicate the availability of the resource to process an order.
- Places to indicate that an order is being processed by the resource at each of the stages.
- Places to indicate that an order has arrived at the resource, but is yet to complete its required processing.
- Places to indicate that an order has completed its required processing at the resource.

Transitions:

- Transitions that indicate the start of processing of an order by a resource in a stage. If an order to be processed along with the resource is available, this transition would be enabled, and the firing of this transition would indicate the start of order processing by the resource.

- Transitions that indicate the completion of processing of an order by a resource in a stage. When the place corresponding to order processing by the resource is populated, this transition would be enabled, and the firing of this transition would indicate the completion of the order processing.
- Transitions that move an unprocessed order from one stage to another. This transition is essential because, at any given stage of a resource, only one order can be processed. It might be better to process order A ahead of B when both are competing for the resource, and these transitions would be used to move B to the next stage of the resource.

The combination of these places and transitions gives rise to the following sub–constructs, which are then assembled to form the resource construct.

Start Processing Sub–construct: In this sub–construct (see Figure 6.1), a waiting order and available resource enables the transition that indicates the start of order processing by the resource. By executing the transition, the order and resource are matched, and the place indicating the progress of order processing is populated.

End Processing Sub–construct: Here, when an order is being processed by a resource, the transition corresponding to the completion of order processing is enabled (see Figure 6.2). When fired, the places corresponding to processed orders and availability of the resource are populated.

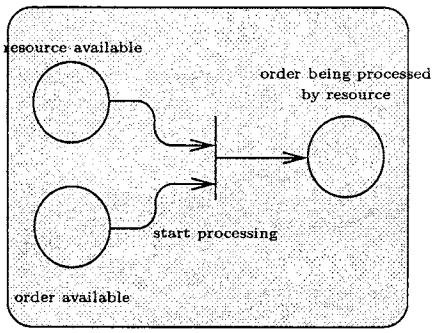

Figure 6.1: Start processing sub–construct

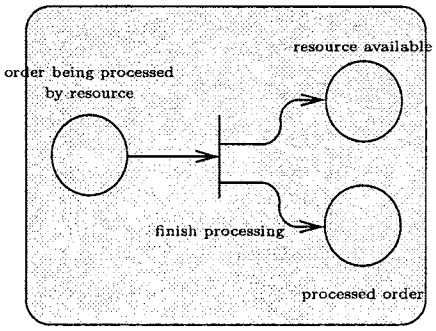

Figure 6.2: End processing sub–construct

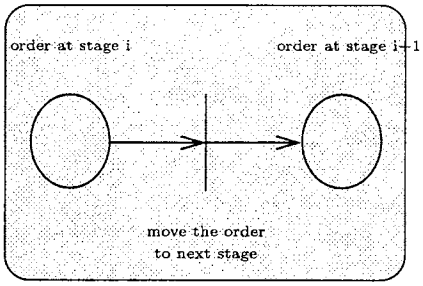

Figure 6.3: Order movement sub–construct

Order Movement Sub–construct: This sub–construct contains transitions that would be used to move the processed and unprocessed orders through to the next stage (see Figure 6.3). This makes an unprocessed order available for processing by the resource at one of its following stages.

Definition 6.2.1 *The Petri net construct of a resource* $j \in \mathbb{R}$ *is a 3–tuple,* $RPN_j = (P_j, T_j, F_j)$*, where*

1. $P_j = \{P1_j, P2_j, P3_j, P4'_j, P4_j\}$ *is a collection of place sets, where*

   (a) $P1_j = \{p1_{11,j}, p1_{21,j}, \ldots, p1_{m1,j}, p1_{12,j}, \ldots, p1_{mn_j,j}, p1_{1(n_j+1),j}, \ldots, p1_{m(n_j+1),j}\}$ *is a set of* $m(n_j + 1)$ *places, with* $p1_{il,j}$ *indicating order* i*'s availability to be processed by resource* j *at resources's* l*th stage, and the* $(n_j + 1)^{th}$ *stage corresponds to a finished order at resource* j

   (b) $P2_j = \{p2_{1,j}, p2_{2,j}, \ldots, p2_{n_j,j}\}$ *is a set of* n_j *places, with* $p2_{l,j}$ *indicating the availability of resource* j *at stage* l

   (c) $P3_j = \{p3_{11,j}, p3_{21,j}, \ldots, p3_{m1,j}, p3_{12,j}, \ldots, p3_{m(n_j-1),j}, p3_{1n_j,j}, p3_{2n_j,j}, \ldots, p3_{mn_j,j}\}$ *is a set of* mn_j *places, with* $p3_{il,j}$ *indicating order* i *being processed by resource* j *at its* l*th stage*

(d) $P4'_j = \{p4'_{1,j}, p4'_{2,j}, \cdots, p4'_{m,j}\}$ *is a set of* m *places, with* $p4'_{i,j}$ *indicating the presence of unprocessed order* i *in resource* j

(e) $P4_j = \{p4_{1,j}, p4_{2,j}, \cdots, p4_{m,j}\}$ *is a set of* m *places, with* $p4_{i,j}$ *indicating the completion of processing of order* i *in resource* j

2. $T_j = \{T1_j, T2_j, T3_j\}$ *is a collection of transition sets, where*

(a) $T1_j = \{t1_{11,j}, t1_{21,j}, \ldots, t1_{m1,j}, t1_{12,j}, \ldots, t1_{m(n_j-1),j}, t1_{1n_j,j}, t1_{2n_j,j}, \ldots, t1_{mn_j,j}\}$ *is a set of* mn_j *transitions, with* $t1_{il,j}$ *indicating the start of processing of order* i *by resource* j *at its* l*th stage*

(b) $T2_j = \{t2_{11,j}, t2_{21,j}, \ldots, t2_{m1,j}, t2_{12,j}, \ldots, t2_{m(n_j-1),j}, t2_{1n_j,j}, t2_{2n_j,j}, \ldots, t2_{mn_j,j}\}$ *is a set of* mn_j *transitions, with* $t2_{il,j}$ *indicating the completion of processing of order* i *by resource* j *at its* l*th stage*

(c) $T3_j = \{t3_{11,j}, t3_{21,j}, \ldots, t3_{m1,j}, t3_{12,j}, \cdots, t3_{mn_j,j}\}$ *is a set of* mn_j *transitions, with* $t3_{il,j}$ *indicating the movement of order* i *in stage* l *of resource* j *to stage* $(l+1)$ *of resource* j

3. $F_j \subseteq (P_j \times T_j) \cup (T_j \times P_j)$ *is a set of directed arcs connecting the places and transitions, expressed in terms of the pre and post sets of the places* $p \in P$ *are as follows.*

(a) $p1^{\bullet}_{il,j} = \{t1_{il,j}, t3_{il,j}\}$; $p1^{\bullet}_{in_j,j} = \{t1_{in_j,j}\}$; $p2^{\bullet}_{l,j} = T1_j$; $p3^{\bullet}_{il,j} = \{t2_{il,j}\}$;

(b) ${}^{\bullet}p1_{il,j} = t3_{i(l-1),j}$; ${}^{\bullet}p1_{i1,j} = \{t2_{i1,j'}, t2_{i2,j'}, \cdots, t2_{in'_j,j'}\}$, *where* $j' = M_i(k-1), n_{j'} =\mid N(j') \mid$ *and* $j = M_i(k)$; ${}^{\bullet}p2_{1,j} = \{\emptyset\}$; ${}^{\bullet}p2_{l,j} = \{t2_{1(l-1),j}, t2_{2(l-1),j}, \cdots, t2_{m(l-1),j}\}$; ${}^{\bullet}p3_{il,j} = \{t1_{il,j}\}$; ${}^{\bullet}p4_{i,j} = \{t2_{i1,j}, t2_{i2,j}, \cdots, t2_{in_j,j}\}$; $p4'^{\bullet}_{i,j} = \{t1_{i1,j}, t1_{i2,j}, \cdots, t1_{in_j,j}\}$

6.2.2.2 Performance Tracking Construct

The resource construct is designed to capture the operational states of a resource. However, the decisions related to the operational plan need

to be taken by considering the system performance implications of the sequence. For this purpose, the performance tracking construct that considers the relative trade-offs between the different orders to be processed is developed.

System performance is determined by tracking the time spent by the orders in the system. In the resource construct, the place $p4_{i,j} \in P4_j$ would be marked if the order i has finished its processing in resource j. If an order is waiting to be processed by the resource j, then one of the places in $p1_{il,j} \in P1_j$ and the place $p4'_{i,j} \in P4'_j$ will be marked. This observation is used in accounting for the order waiting times in a resource.

Places:

- Places to track the time spent by an order at a resource.
- Places to track the processing time of an order in a resource as waiting time for the other orders in line to be processed by the resource.

Transitions:

- Transitions that evaluate the tardiness of an order by applying the cost function that,
  - If the order has been already accepted, it compares the committed completion time with its new estimated completion time under the new plan being developed.

In Petri net terminology, this construct is defined as follows.

Definition 6.2.2 *The performance tracking construct of all the orders is a 4-tuple, $PPN = (P, T, F_p, W_p)$, where*

1. $P = \{P5, P6, P7, P8, P9\}$ *is a collection of places where*

   (a) $P5 = \{p5_1, p5_2, \cdots, p5_m, p5^*\}$ *is a set of $m+1$ places, where $p5_i$ keeps track of the time spent by order i before its completion and $p5^*$ is a place that tracks the overall system performance*

   (b) $P6 = \{P6_1, P6_2, \cdots, P6_n\}$ *is a collection of sets of places, where $P6_j = \{p6_{12,j}, p6_{13,j}, \cdots, p6_{1n_j,j}\}$ is a set of n_j places, where the place $p6_{ii',j}, i \neq i'$ in conjunction with the place $p4'_{i,j}$ is used to determine the contribution of the processing time of order i' towards the waiting time of order i*

   (c) $P7 = \{p7_1, p7_2, \cdots, p7_m\}$ *is a set of m places, where $p7_i$ is used to represent the current promised due date of order i*

   (d) $P8 = \{p8_1, p8_2, \cdots, p8_m\}$ *is a set of m places used for determining the cost of any delays for order i*

   (e) $P9 = \{p9_1, p9_2, \cdots, p9_m\}$ *is a set of m places, where $p9_i$ is used to represent the revenue from order i*

2. $T = \{T5, T6, T7, T8, T9, T10\}$ *is a collection of transitions where*

   (a) $T5 = \{T5_1, T5_2, \cdots, T5_n\}$ *is a colletion of sets of transitions, where $T5_j = \{t5_{12,j}, t5_{13,j}, \cdots, t5_{1n_1,j}, t5_{21,j}, t5_{23,j}, \cdots, t5_{n_j(n_j-1),j}\}$ is a set of $n_j(n_j-1)$ transitions where $t5_{ii',j}$ used to evaluate the waiting time contribution of order i' for order i at resource j*

   (b) $T6 = \{t6_1, t6_2, \cdots, t6_m\}$ *is a set of m transitions which is used to determine the difference between the current order due date and the promised order due date, if any*

   (c) $T7 = \{t7_1, t7_2, \cdots, t7_m\}$ *is a set of m transitions which will be used, when for order i, the promised order due date is later than the current due date of the order*

   (d) $T8 = \{t8_1, t8_2, \cdots, t8_m\}$ *is a set of m transitions which will be used, when for order i, the current due date is later than the promised due date*

(e) $T9 = \{t9_1, t9_2, \cdots, t9_m\}$ *is a set of* m *transitions which will be used for determining the cost for any delays for order* i

(f) $T10 = \{t10_1, t10_2, \cdots, t10_m\}$ *is a set of* m *transitions which will be used for determining the net revenue from order* i

3. $F_p \subseteq (P5 \times T) \cup (T \times P5)$ *is a set of directed arcs connecting the places and transitions expressed in terms of the pre and post sets of the places* $p \in P5$ *as*

(a) $p4'i,j^{\bullet} = \{T5_{r_{s_i}}, t6_i\}$; $p7_i^{\bullet} = \{t6_i, t7_i\}$; $p5_i^{\bullet} = \{t6_i, t8_i\}$; $p8_i^{\bullet} = \{t9_i\}$;$p9_i^{\bullet} = \{t9_i, t10_i\}$

(b) ${}^{\bullet}p6_{ii',j} = \{t2_{i'1,j}, t2_{i'2,j}, \cdots, t2_{i'n_j,j}\}$; ${}^{\bullet}p4'i,j = \{T5_{r_{s_i}}, t6_i\}$ ${}^{\bullet}p5_i = \{T5_{r_1}, T5_{r_2}, \cdots, T5_{r_{s_i}}\}$; ${}^{\bullet}p8_i = \{t8_i\}$; $p5^* = \{T10\}$

where $r_k = M_i(k)$

4. $W_p : F_p \rightarrow \{1, 2, 3, \ldots\}$ *is a weight function that assigns a positive integer to all directed arcs in* F_p*, and is defined as follows*

$$\begin{aligned} W_p(t5_{ii',j}, p5_i) &= \mu^p_{i'k}, \text{where } M_{i'}(k) = j \\ W_p(t8_i, p8_i) &= P_i \end{aligned}$$

Note that without any loss of generality, μ^p_{ik} and P_i can be assumed to be integer valued.

6.2.3 Construct Properties

The resource and performance tracking constructs have some important structural properties that are exploited for "optimizing" due date decisions in the next section.

Property 6.2.1 *The resource and performance tracking constructs are* acyclic.

This is a very important property that the constructs possess, because acyclic Petri nets are one among the classes of nets for which both necessary and sufficient conditions for reachability are known. As discussions in an earlier chapter (Chapter 2) indicate, for an acyclic Petri net,

the existence of a non–negative firing count vector for the state equation is sufficient for reachability. Hence, the acyclic nature of the constructs enable the use of known reachability results.

Property 6.2.2 *The set of places* $(P1_j(i) \cup P3_j(i))$ *where* $PX_j(i) \subseteq PX_j : \forall px_{i'l,j} \in PX_j(i), i = i'$, *forms a place invariant set.*

The implication of this property is that, whenever an order enters the confines of a resource, its presence will be indicated at either one of the waiting areas of a stage $P1_j(i)$ or processing areas $P2_j(i)$. If one of these places is marked, then it indicates the fact that the order has already arrived at the resource. When the order is processed, the places in the set $P4_j$ would be marked. Hence, when places in both the sets have been marked, it indicates that the order has already arrived at the resource and has been processed by it.

6.2.4 Modeling Illustration

Next, the process of modeling of a due date quotation problem using the Petri net constructs defined above is illustrated.

Example 6.2.1 *Consider a production system with two orders* $\hat{\mathbb{O}} = \{O_1\}$ *and* $\mathbb{O} = \{O_2\}$. *The orders* O_1 *require processing by resources* R_1 *and* R_2 *in that order, while* O_2 *needs to be processed by resource* R_2 *and* R_1 *in that order. The resource construct for resource* R_1 *is as shown in Figure 6.4. The construct has two stages because both the orders require processing by resource* R_1. *A portion of the performance tracking construct is as shown in Figure 6.5. Specifically, the portion depicted in this figure represents the performance tracking mechanism for order* O_1.

From the graphical model of resource construct of resource R_1 *in Figure 6.4 it can be seen that at each stage there is a set of places to hold orders as input for that stage* ($p1_{il,1}$). *Also, there is a set of places to hold the orders after being processed by the resource. If the order is processed by the resource at its* l^{th} *stage, then the transitions* $t1_{il,1}$ *and* $t2_{il,1}$ *are executed. If an order is not processed by the resource at stage* l, *then the*

order can be moved to the next stage using transition $t3_{il,1}$. The places $p4_{i,1}$ are used to count the number of times an order is processed by the resource. This is essential to ensure that a processing step of an order is executed only once. The utility of this would be more apparent in the next section while discussing the optimization using reachability analysis.

In the performance tracking constructs depicted in Figures 6.5, 6.6, and 6.7, the place $p5_1$ is used to keep track of the time spent by order O_1 in the system before completion. The places $p4'_{i,j}$ is used to determine if an order has reached resource after passing through the previous processing step. The place $p4_i, j$ tracks an order that has completed its processing requirements in resource j. The first two parts of the performance tracking constructs shown in Figures 6.5, and 6.6 together track the total waiting time for an order in a resource. When the order has been processed completely, the place $p5_1$ would represent the total time spent by the order in the system. Using this information, the performance tracking construct in Figure 6.7 computes the overall system performance using the revenue from the order and penalty based on the promised due date, if any.

6.3 Problem Formulation

The resource construct and the performance tracking construct defined in Section 6.2.2 provide the capability to model the process as well as the performance trade off among the different orders in the system. For a given collection of orders, the different resource orders are constructed and are merged such that for a given order, the output from the last stage of k^{th} processing step becomes the input to the first stage of the $k + 1^{th}$ processing step of the order. With the Petri net model thus constructed, the due date assignment problem can be translated into a reachability problem such that all the orders are completed with the maximum number of accumulated tokens in the system–wide performance tracking place $p5^*$. In the mathematical programming world this

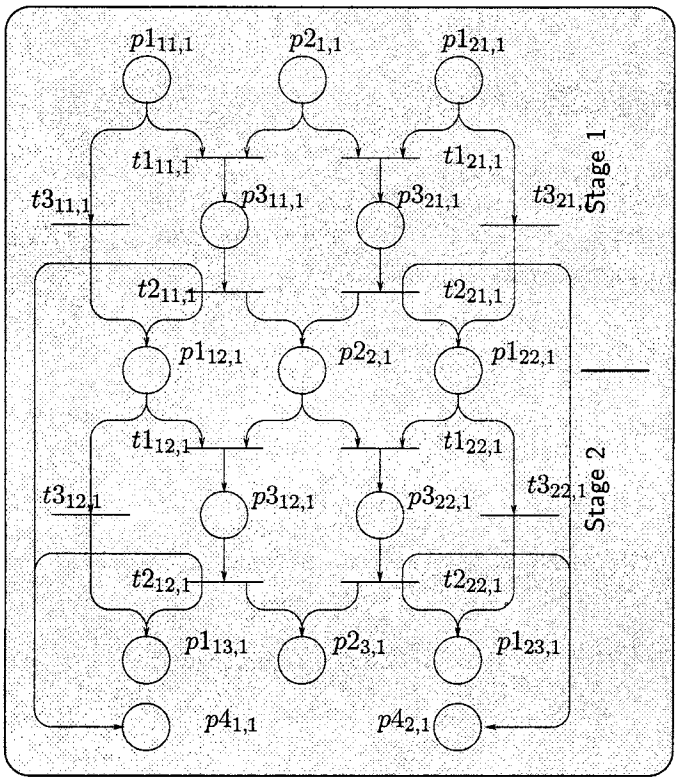

Figure 6.4: Example – Resource construct

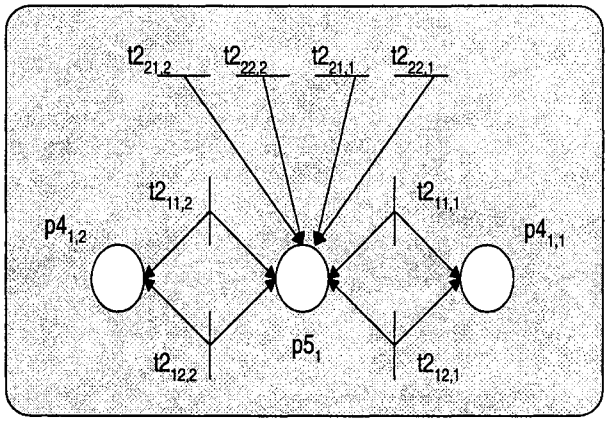

Figure 6.5: Example – Performance tracking construct 1

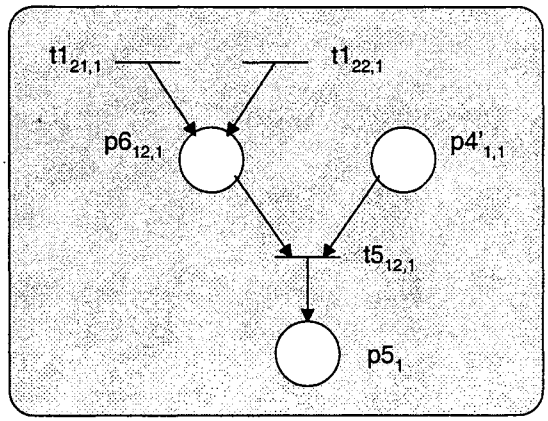

Figure 6.6: Example – Performance tracking construct 2

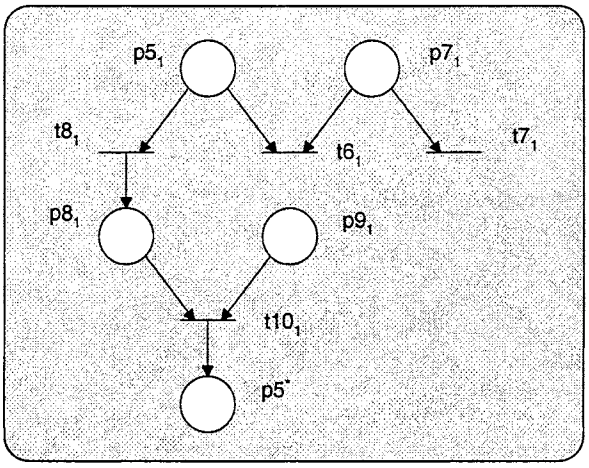

Figure 6.7: Example – Performance tracking construct 3

is equivalent to the maximization of the objective function. Next, the due date quotation problem is formulated in terms of the reachability terminology.

6.3.1 Due Date assignment as a reachability problem

The current state of the system is represented by marking the appropriate places of the Petri net model. From this given state, the problem is to determine the sequence of processing steps that would change the system state to one in which all the orders have been processed with maximum number of tokens in place $p5^*$. The due date quotation problem formulated as a reachability problem is defined as follows.

Definition 6.3.1 *Given a Petri net model of a due date assignment problem defined by a collection of resource constructs and performance tracking constructs, determine a transition firing sequence* $\sigma \in \mathcal{L}(\mathcal{M}_0)$ *that maximizes* $\mathcal{M}_d(p5^*)$ *such that* $\mathcal{M}_0[\sigma \succ \mathcal{M}_d$ *where,*

1. **Initial Marking:**
   $\mathcal{M}_0(p1_{i1,j}) = 1$, $\forall i \in \{\mathbb{O} \cup \hat{\mathbb{O}}\}$, *where* $j = M_i(1)$;
   $\mathcal{M}_0(p2_{1,j}) = 1$, $\forall j \in \mathbb{R}$;
   All other places are empty.

2. **Final Marking:**
   $\mathcal{M}_d(p1_{in_j,j}) = 1$, $\forall i \in \{\mathbb{O} \cup \hat{\mathbb{O}}\}$, *where* $j = M_i(s_i)$;
   $\mathcal{M}_d(p5^*) = M^*$, *where* M^* *is the total cost of the processing plan specified by* σ
   $\mathcal{M}_d(p4_{i,j}) = 1$, $\mathcal{M}_d(p7_1) = T_i$, $\mathcal{M}_d(p9_i) = f_i$ *All other places are empty.*

Note that in the case of partially completed orders that are currently in the system, the processing times are adjusted for the remaining processing times.

A suitable process plan for a given set of orders has to be determined iteratively so that the net revenue of executing the process plan given by the marking $\mathcal{M}_d(p5^*)$ is maximized. The marking in place $p5^*$ is indicative of the net revenue from processing the orders (revenue minus cost associated with the processing plan, which in turn is a function of the order completion times). Hence, a process plan that leads to the maximum number of markings in the place $p5^*$ is desirable. The iterative process of identifying the best process plan would be aided if the number of tokens in the place $p5^*$ could be bounded. If none of the orders incurs any cost in the process plan, then the number of tokens in $p5^*$ would be sum of all the revenues from processing the orders. Hence the iterative process could be set up to begin with $\mathcal{M}_d(p5^*) = \sum_i f_i$, and decrease

$\mathcal{M}_d(p5^*)$ by one for each iteration if a valid firing sequence is not found. The reader is referred to the discussions in the earlier chapters that deal with the actual mechanics of reachability analysis (see Chapters 3 & 4).

Chapter 7

Summary and Research Contributions

Summary

This chapter summarizes the major research findings and the research problems that were brought to the forefront by this dissertation research. The areas that need to be explored in future research activities are also discussed.

7.1 Summary

The main research goal of this dissertation is to explore approaches for deciding reachability in general Petri nets. A formal introduction of the reachability problem is provided in Chapter 2. Also, Chapter 2 reviews the various approaches for deciding reachability that are available in published literature. In Chapter 3, a new acyclic transformation technique is developed that enables the application of the known sufficient condition for reachability in acyclic Petri nets to general Petri nets under certain conditions. Specifically, the application of the sufficient condition of reachability in acyclic Petri nets to any general Petri net with known firing count vector is discussed. Further, the issues involved in using this approach for Petri nets with unknown firing count vectors are presented. Chapter 4 discusses the challenges in obtaining a firing count

vector for a general Petri net reachability problem. More specifically, the presence of the free variables in the solution to state equation is the source of uncertainty in determining the firing count vector. Chapter 4 also explores ways to reduce the number of free variables in the solution to the state equation.

Chapter 5 presents some of the extensions of the reachability results to related problems. First, the sub–marking reachability problem is discussed. The notion of a SubM Petri net for the purpose of sub-marking reachability analysis is introduced. The relationship of the sub–marking reachability problem to a full–marking reachability problems in its SubM Petri net is established. The preservation of the acyclic nature of a net in its SubM format is shown. Chapter 5 also includes discussions on the impact of the reachability results on deadlock avoidance problems.

Finally, an application of the reachability analysis in the discrete part manufacturing environment is presented in Chapter 6. Petri net modeling constructs for the due–date quotation problem are developed, and these lay the foundations for a new approach to this important problem. It is especially valuable because, the reachability analysis based approach provides an operational road map for the production system in addition to recommending due dates for the orders.

7.2 Research Contributions

The major contribution of this research is the development of a new acyclic transformation technique using net expansion that enables the use of the sufficient condition for reachability in acyclic Petri nets in the context of a general Petri net under certain conditions. This acyclic

transformation approach converts any general Petri net into an acyclic Petri net of a given number of stages. The net result is that for a general Petri net reachability problem, if the number of events that changes the current state to the desired state is known, the sufficienct condition for the reachability problem can be stated in terms of a related acyclic Petri net. In addition, the acyclic transformation based net expansion technique has the added advantage of directly delivering the transition firing sequence.

Another significant contribution of this research is in showing the relationship between the sub–marking reachability problem and a full marking reachability problem. Specifically, a SubM Petri net is defined that can be used to formulate the sub–marking reachability problem of the original Petri net as a reachability problem of the SubM Petri net. Further, the SubM Petri net retains the acyclic property of the original Petri net.

The various research contributions are summarized below:

- The known sufficient condition for reachability for acyclic Petri nets (a sub–class) is made applicable to all Petri nets under certain conditions with the introduction of an acyclic transformation approach.

- Introduced some experimental approaches to reduce the uncertainty in determining the transition firing count vectors for a reachability problem.

- Established a relationship between the sub–marking reachability problem and a full–marking reachability problem. In fact, it is established that the sub–marking reachability problem of a Petri net

could be reformulated directly as full marking reachability problem of its SubM Petri net.

- Modeling constructs are developed for formulating the due–date quotation problem in a discrete part manufacturing environment as a Petri net reachability problem. This lays the foundation for a new, comprehensive approach to the due date quotation problem.

7.3 Future Research Directions

Practical application of the acyclic transformation approach depends on our ability to exactly determine or bound the required number of stages of net expansion. The work on invariant–free Petri nets in Chapter 4 is a step in that direction. Specifically, two approaches for reducing the null space of the incidence matrix are presented. This has a direct implication on the number of free variables in the solution to the state equation, and hence, on bounding the required number of stages of net expansion. The application of these two approaches, namely (1) use of base transitions as stabilizers, and (2) multi–purpose sink transitions as stabilizers need to be explored further. This research characterized their use based on the general solution to the state equation. Another approach could be categorize their usage for specific classes of Petri nets.

On the application of reachability analysis, a novel application to the due date quotation problem in a manufacturing system is explored. In addition to recommending order due dates, this approach has the added advantage of directly providing the operational road map for the system. The power of Petri nets provides the capability to model different business situations as well. An example of such a business situation is

when order A needs to be given priority over order B. Research has been initiated in this direction by defining Petri net modeling constructs for this problem.

The directions for future research can be summarized thus:

- Exploration of invariant free Petri nets for bounding the required number of stages of expansion for a reachability problem, and any new approaches intended for this purpose.
- Testing the Petri net modeling constructs developed herein for the order due date quotation problem in practical situations.
  - The modeling constructs assume that the order processing times are deterministic. Modifications to the constructs to handle stochastic processing times would be a valuable future research effort.
  - Computational studies using these modeling constructs would increase their potential applicability in factory decision support systems.

Bibliography

[1] J. L. BAER AND C. A. ELLIS, *Model design and evaluation of a compiler for a parallel processing environment*, IEEE Transactions on Software Engineering, SE–3 (1977), pp. 394–405.

[2] H. G. BAKER, *Rabin's proof of the undecidability of the reachability set inclusion problem of vector addition systems*, tech. report, MIT Project MAC, CSGM 79, Cambridge, MA, 1973.

[3] Z. A. BANASZAK AND B. H. KROGH, *Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows*, IEEE Transactions on Robotics and Automation, 6 (1990), pp. 724–734.

[4] J. W. M. BERTRAND, *The effect of workload dependent due dates on job shop performance*, Management Science, 29 (1983), pp. 799–816.

[5] T. C. E. CHENG AND M. C. GUPTA, *Survey of scheduling research involving due date determination decisions*, European Journal of Operational Research, 38 (1989), pp. 156–166.

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

[7] M. A. COMEAU AND K. THULASIRAMAN, *Structure of the submarking reachability problem and network programming*, IEEE Transactions on Circuits and Systems, 35 (1988), pp. 89–100.

[8] R. W. CONWAY, *Priority dispatching and job lateness in a job shop*, Journal of Industrial Engineering, 16 (1965), pp. 228–237.

[9] R. DAVID AND H. ALLA, *Petri nets for modeling dynamic systems–A survey*, Automatica, 30 (1994), pp. 175–202.

[10] J. DESEL AND J. ESPARZA, *Reachability in cyclic extended free choice systems*, Theoretical Computer Science, 114 (1993), pp. 93–118.

[11] A. A. DESROCHERS AND R. Y. AL-JAAR, *Applications of Petri Nets in Manufacturing Systems*, IEEE Control System Society: NY., 1994.

[12] M. DIAZ, *Modeling and analysis of communication and co–operation protocols using Petri net based models*, Computer Networks, 6 (1982), pp. 419–441.

[13] I. DUENYAS AND W. J. HOPP, *Quoting customer lead times*, Management Science, 41 (1995), pp. 43–57.

[14] S. E. EILON AND I. G. CHOWDHURY, *Due dates in job shop scheduling*, International Journal of Production Research, 14 (1976), pp. 223–237.

[15] J. ENGELFRIET, *Branching processes of Petri nets*, Acta Informatica, 28 (1991), pp. 575–591.

[16] J. ESPARZA, *Reachability in live and safe free choice Petri nets is NP–complete*, Theoretical Computer Science, 198 (1998), pp. 211–224.

[17] J. ESPARZA, S. RÖMER, AND W. VOGLER, *An improvement of McMillans's unfolding algorithm*, in Proceedings of TACAS'96, LNCS 1055, Springer Verlag, 1996, pp. 87–106.

[18] J. ESPARZA AND C. SCHRÖTER, *Unfoldings based algorithms for the reachability problem*, Fundamenta Informaticae, 46 (2001), pp. 1–17.

[19] R. A. FERNANDES, *Deadlock avoidance in automated manufacturing systems*, master's thesis, School of Industrial Engineering and Management, Oklahoma State University, December 1995.

[20] S. GERSHWIN, *Manufacturing Systems Engineering*, Prentice Hall, 1994.

[21] M. E. GOLD, *Deadlock prediction: Easy and difficult cases*, SIAM Journal of Computing, 7 (1978), pp. 320–336.

[22] M. HACK, *Decidability questions for Petri nets*, Tech. Report 161, MIT, LCS, Cambridge, MA, 1976.

[23] W. J. HOPP AND M. L. SPEARMAN, *Factory Physics: Foundations of Manufacturing Management*, Irwin: Burr Ridge, IL, 1996.

[24] W. J. HOPP AND M. L. STURGIS, *Quoting manufacturing due dates subject to a service level constraint*, IIE Transactions, 32 (2000), pp. 771–784.

[25] A. ICHIKAWA AND K. HIRAISHI, *A class of Petri nets for which necessary and sufficient condition for reachability is obtainable*, Transactions of the Society of Instruments and Control Engineers (In Japanese), 24 (1988).

[26] K. JENSEN, *Coloured Petri nets and the invariant method*, Theoretical Computer Science, 14 (1981), pp. 317–336.

[27] M. KAMATH AND N. VISWANADHAM, *Applications of Petri net based models in the modeling and analysis of flexible manufacturing systems*, in Modeling and Control of Automated Manufacturing Systems, A. A. Desrochers, ed., IEEE Computer Society Press, 1990, pp. 262–267.

[28] R. M. KARP AND R. E. MILLER, *Parallel program schemata*, Journal of Computational Systems Sciences, 3 (1969), pp. 147–195.

[29] S. R. KOSARAJU, *Decidability of reachability in vector addition systems*, in Proceedings of the 14th Annual ACM Symposium on the Theory of Computing, San Francisco, CA, 1982, pp. 267–281.

[30] S. KUMAGAI, S. KODAMA, AND M. KITAGAWA, *Submarking reachability of marked graphs*, IEEE Transactions on Circuits and Systems, 31 (1984), pp. 159–164.

[31] S. R. LAWRENCE, *Estimating flow times and setting due dates in complex production systems*, IIE Transactions, 27 (1995), pp. 657–668.

[32] D. Y. LEE AND F. DICESARE, *Scheduling flexible manufacturing systems using Petri nets and heuristic search*, IEEE Transactions on Robotics and Automation, 10 (1994), pp. 123–132.

[33] R. J. LIPTON, *The reachability problem requires exponential space*, Research Report 62, Department of Computer Science, Yale University, New Haven: CT, 1976.

[34] K. F. MAN, K. S. TANG, AND S. KWONG, *Genetic algorithms: Concepts and applications*, IEEE Transactions on Industrial Electronics, 43 (1996), pp. 519–534.

[35] D. MANDRIOLI, *A note on Petri net languages*, Information and Control, 34 (1977), pp. 169–171.

[36] E. W. MAYR, *An algorithm for the general Petri net reachability problem*, SIAM Journal of Computing, 13 (1984), pp. 441–460.

[37] K. L. MCMILLAN, *A technique of state space search based on unfolding*, Formal Methods in System Design, 6 (1995), pp. 45–65.

[38] S. MELZER AND S. RÖMER, *Deadlock checking using net unfoldings*, in Proceedings of CAV'97, LNCS 1254, Springer Verlag, 1997, pp. 352–363.

[39] T. MIYAMOTO AND S. KUMAGAI, *A graph theoretic approach to reachability problem with Petri net unfoldings*, IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences, E79–A (1996), pp. 1809–1816.

[40] S. A. MOSES, *Due date assignment using feedback control with reinforcement learning*, IIE Transactions, 31 (1999), pp. 989–999.

[41] T. MURATA, *Petri nets: Properties, analysis and applications*, Proceedings of the IEEE, 77 (1989), pp. 541–580.

[42] T. MURATA AND D. ZHANG, *A predicate–transition net model for parallel interpretation of logic programs*, IEEE Transactions on Software Engineering, 14 (1988), pp. 481–497.

[43] N. NIELSEN, G. PLOTKIN, AND G. WINSKEL, *Petri nets, event structures and domains*, Theoretical Computer Science, 23 (1980), pp. 85–108.

[44] N. NILSSON, *Principles of Artificial Intelligence*, Palo Alto, CA: Tioga, 1980.

[45] J. PEARL, *Intelligent Search Strategies for Computer Problem Solving*, Reading, MA: Addison–Wesley, 1984.

[46] J. L. PETERSON, *Petri nets*, Computing Surveys, 9 (1977), pp. 223–252.

[47] P. RAMACHANDRAN AND M. KAMATH, *On the place invariant sets and the rank of incidence matrix of ordinary Petri nets*, in Proceedings of IEEE International Conference on Systems, Man and Cybernetics, IEEE, 1998, pp. 160–165.

[48] C. RAMCHANDANI, *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets*, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1973.

[49] T.-H. SUN, C.-W. CHENG, AND L.-C. FU, *A Petri net based approach to modeling and scheduling for an FMS and a case study*, IEEE Transactions on Industrial Electronics, 41 (1994), pp. 593–601.

[50] K. TAKAHASHI, M. YAMAMURA, AND S. KOBAYASHI, *A GA approach to solving reachability problems for Petri nets*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E79 (1996), pp. 1774–1780.

[51] L. N. TREFETHEN AND D. BAU, *Numerical Linear Algebra*, SIAM, 1997.

[52] N. VISWANADHAM AND Y. NARAHARI, *Performance Modeling of Automated Manufacturing Systems*, Prentice Hall of India, 1994.

[53] N. VISWANADHAM, Y. NARAHARI, AND T. L. JOHNSON, *Deadlock prevention and deadlock avoidance in flexible manufacturing systems using Petri net models*, IEEE Transactions on Robotics and Automation, 6 (1990), pp. 713–723.

[54] K. VOSS, *Using predicate/transition nets to model and analyze distributed database systems*, IEEE Transactions on Software Engineering, SE–6 (1980), pp. 539–544.

[55] S. S. YAU AND M. U. CAGLAYAN, *Distributed software system design representation using modified Petri nets*, IEEE Transactions on Software Engineering, SE–9 (1983), pp. 733–745.

VITA 2

Parthasarathy Ramachandran

Candidate for the Degree of

Doctor of Philosophy

THESIS: An Acyclic Transformation Technique for the Reachability Analysis of Petri Nets

MAJOR FIELD: Industrial Engineering & Management

BIOGRAPHICAL:

Personal Data : Born in Madras, India, May 11, 1972, son of Smt. R. Gomathy and Sri. A. Ramachandran

Education : Graduated from Madras Christian College Higher Secondary School, Madras, India in May 1989; received the Bachelor of Engineering degree in Mechanical Engineering from The University of Madras, India in May 1994; received the Master of Science degree in Mechanical Engineering from Oklahoma State University in December 1995; completed requirements for the Doctor of Philosophy degree at Oklahoma State University in December 2002.

Experience : Graduate Research Assistant, Advanced Control Laboratory, Oklahoma State University, January 1995 – December 1995; Graduate Research Associate, Center for Computer Integrated Manufacturing, Oklahoma State University, September 1996 – June 2000; Instructor, School of Industrial Engineering and Management, Oklahoma State University, Spring 1999, Fall 1999, and Spring 2000; Research Scientist, PROS Revenue Management, Houston, TX, July 2000 – present.

Professional Membership : INFORMS, IEEE, and SIAM