Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturing Systems Using Petri Net Models

N. VISWANADHAM, SENIOR MEMBER, IEEE, Y. NARAHARI, AND TIMOTHY L. JOHNSON, MEMBER, IEEE

Abstract—Deadlocks constitute an important issue to he addressed in the design and operation of flexible manufacturing systems (FMS's). In this paper, we show that prevention and avoidance of FMS deadlocks can he implemented using Petri net models. For deadlock prevention, we use the reachability graph of a Petri net model of the given FMS, whereas for deadlock avoidance, we propose a Petri net-based on-line controller. We discuss the modeling of the General Electric FMS at Erie, PA. For such real-world systems, deadlock prevention using the reachability graph is not feasible. We develop a generic, Petri net-based on-line controller for implementing deadlock avoidance in such real-world FMS's.

***Key* Words—Flexible manufacturing system (FMS), General Electric FMS, deadlock prevention, deadlock avoidance, Petri Net models.**

I. Introduction

IN THIS paper, we investigate the use of Petri net (PN) models in the prevention and avoidance of deadlocks in flexible manufacturing systems (FMS's). We first show that PN's constitute an effective modeling framework for real-world FMS's by taking the example of the General Electric FMS (GE FMS) at Erie, PA. We then show that PN models can be used in the prevention and avoidance of deadlocks. Deadlock prevention refers to static resource allocation policies for eliminating deadlocks, whereas deadlock avoidance refers to dynamic resource allocation policies. For deadlock prevention, we use the reachability graph of the PN model to arrive at static resource allocation policies. For deadlock avoidance, we propose a PN-based on-line monitoring and control system. We illustrate deadlock prevention for a simple manufacturing system comprising a machine and an automated guided vehicle (AGV) and observe that prevention can be implemented effectively only for reasonably small systems. Deadlock avoidance is the preferred technique for real-world FMS's such as the GE FMS.

A. Deadlocks in Automated Manufacturing Systems

Automated manufacturing systems, including FMS's, belong to the class of discrete event dynamical systems (DEDS) that are gaining in prominence in the recent literature [1]. In a typical FMS, raw parts of various types enter the system at discrete points of time and are processed concurrently, sharing a limited number of resources such as numerically controlled (NC) machines, robots, material handling system (MHS), fixtures, and buffers. In such resource-sharing systems, deadlocks [2]–[4] constitute a major issue to be addressed at the design and operation phases. A deadlock is a highly undesirable situation in which each of a set of two or more jobs keeps waiting indefinitely for the other jobs in the set to release resources. The occurrence of a deadlock can cripple the entire system and renders automated operation impossible. In addition, a deadlock, occurring in a subsystem of the given system, can propagate to other parts of the system, eventually completely stalling all activities in the entire system. Deadlocks usually arise as the final state of a complex sequence of operations on jobs flowing concurrently through the system and are thus generally difficult to predict. In an improperly designed **FMS,** the only remedy for deadlock may be manual clearing of buffers or machines and restart of the system from an initial condition that is known to produce deadlock-free operation under normal production conditions. Both the lost production and the labor cost in resetting the system in this way can be avoided by proper design and careful operation.

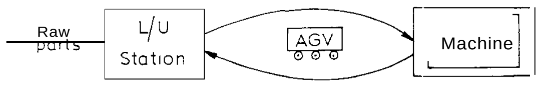


Fig. 1. Simple manufacturing system comprising an **AGV** and an NC machine.

To visualize a simple example of a deadlock in a manufacturing system, consider the system depicted in Fig. 1. There is a load/unload (L/U) station at which raw parts are always available. An AGV carries a raw part from the L/U station to an NC machine, which carries out some operations on the raw part. The finished part is carried by the AGV to the L/U station, where it is unloaded. It is assumed that the AGV can only carry one part at a time, and the NC machine can only process one part at a time. In addition, the AGV takes a certain amount of time to carry a part from L/U to machine or from machine to L/U. However, if it is not carrying a part, it can travel very quickly between the L/U and AGV. Imagine the following sequence of events, starting with an initial state in which the AGV and the machine are free, and raw parts are available: 1) The AGV carries a raw part, say part 1, and loads it onto the NC machine, which starts processing part 1; 2) the AGV returns to the L/U station and

Manuscript received November 11, 1988; revised June 8, 1990.

N. Viswanadham and Y. Narahari are with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India.

T. L. Johnson is with the Control Technology Branch, General Electric R&D Center, Schenectady, NY 12301.

IEEE Log Number 9038621.

carries another raw part, say part 2, to the machine but waits for the machine, which is still processing part 1. Thus, the AGV gets blocked waiting for the machine; 3) the machine finishes the operations on part 1 and starts waiting for the AGV to carry the finished part 1 to the L/U station. At this juncture, the machine gets blocked waiting for the AGV. If the machine and the AGV can only accommodate one part at a time and there is no additional buffer space, the two resources here are then involved in a deadlock since each keeps waiting for the other indefinitely. Even if some buffer space is provided for raw parts and finished parts in the above system, a deadlock can still occur because the AGV can fill the entire buffer with raw parts during the processing of part 1 by the machine.

In the recent literature, several efforts have focused on the problem of deadlocks in automated manufacturing systems [5]-[9]. One of the major traditional applications of PN's [10], [11] has been in the deadlock analysis of concurrent systems. In manufacturing systems, studies on deadlocks, using PN-based models are presented in [5]-[8] and [12]-[14]. These studies essentially prove the existence (absence) of deadlocks using the invariants of the PN model. In this paper, we address the important issues of prevention and avoidance of deadlocks in automated manufacturing systems using PN-based techniques. The terms prevention and avoidance have been used in the Computer Science literature on deadlocks [2]-[4] to mean static and dynamic policies, respectively, for eliminating deadlocks. It is known that deadlock prevention policies that are usually implemented in the design stage lead to inefficient resource utilization. Deadlock avoidance policies that can be enforced during the operation of a system lead to better resource utilization and throughput.

B. Outline of the Paper

Section II is devoted to a systematic introduction to the notation of PN's. The definitions presented are based on those in [5], [10], [11], [15]-[17]. In Section III, we demonstrate the use of PN's in the modeling of a real-world FMS (namely, the GE FMS at Erie, PA) and present a generic deadlock situation in the GE FMS. Sections IV and V are devoted to deadlock prevention and deadlock avoidance, respectively. In Section IV, we show that scheduling rules for ensuring deadlock prevention in a given FMS can be devised by carrying out an exhaustive path analysis of the reachability graph of the PN model of the FMS. However, the reachability graph for the PN models of real-world FMS's, such as the GE FMS, can contain tens of thousands of states and arcs, and even off-line analysis may be intractable. This provides the motivation for employing deadlock avoidance, which can be implemented without generating the reachability graph. In Section V, we first show for a simple example that deadlock avoidance can be guaranteed by looking ahead into the evolution of the system by a certain number of steps. The process of looking ahead into the system evolution can be done in a natural way using a PN model of the system. We then propose an on-line monitoring and control system that could avoid most deadlocks for any given FMS. With a finite look ahead, deadlocks may not be totally avoided, but the probability of occurrence of deadlock will diminish appreciably with increasing value for look ahead. The proposed on-line controller can be used effectively for real-world FMS's such as the GE FMS.

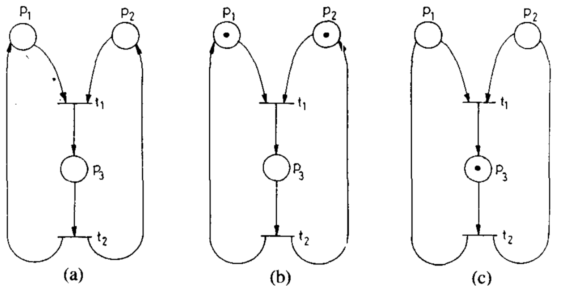


Fig. 2. (a) Petri net model of a single machine system; (b) initial marking M_o of the above model; (c) another marking M_1 of the above model.

II. Petri Nets—An Overview

We now present an overview of PN's [10], [11], [15]-[17] and state the most relevant results. In the following, N denotes the set of nonnegative integers.

Definition 2.1: A Petri net G is a four-tuple (P, T, IN, OUT) where

$$P = \{p_1, p_2, p_3, \cdots, p_n\} \text{ is a set of places}$$

$$T = \{t_1, t_2, t_3, \cdots, t_m\} \text{ is a set of transitions}$$

$$P \cup T \neq \emptyset, P \cap T = \emptyset$$

and where IN: $(P \times T) \rightarrow N$ is an input function that defines directed arcs from places to transitions and where OUT: $(P \times T) \rightarrow N$ is an output function that defines directed arcs from transitions to places.

Pictorially, places are represented by circles and transitions by horizontal bars. If IN $(p_i, t_j) = k$, where $k \geq 1$ is an integer, a directed arc from place p_i to transition t_j is drawn with label k. If IN $(p_i, t_j) = 0$, no arc is drawn from p_i to t_j. Similarly, if OUT $(p_i, t_j) = k$, a directed arc is included from transition t_j to place p_i, with label k if $k > 1$ and without label if $k = 1$. If $k = 0$, no arc is included from t_j to p_i.

Example 1: Let us consider a machine that processes one job at a time. As soon as the processing is over, another job is made available, and the machine starts processing again. Fig. 2 depicts a PN model (PNM) of the above system. The places and the transitions have the following interpretation:

p_1	Machine ready to process (machine "free")
p_2	job waiting for processing
p_3	job undergoing machining (machine "busy")
t_1	machining commences
t_2	machining finishes.

In the above example, places represent various conditions in the system, and transitions represent the starting or finishing of activities. For example, place p_1 models the condition "machine is free". We have assumed that the machine, if it fails, will be repaired and will resume its operation on the job. As such, for the sake of simplicity, failures and repairs have not been explicitly modeled in this PNM.

For the above PNM

$$P = \{p_1, p_2, p_3\}; \quad T = \{t_1, t_2\}; \quad \text{and}$$
$$\text{IN}(p_1, t_1) = \text{IN}(p_2, t_1) = \text{IN}(p_3, t_2) = 1$$
$$\text{IN}(p_1, t_2) = \text{IN}(p_2, t_2) = \text{IN}(p_3, t_1) = 0$$
$$\text{OUT}(p_1, t_2) = \text{OUT}(p_2, t_2) = \text{OUT}(p_3, t_1) = 1$$
$$\text{OUT}(p_1, t_1) = \text{OUT}(p_2, t_1) = \text{OUT}(p_3, t_2) = 0.$$

Definition 2.2: Let 2^P be the powerset of P. We then define functions ZP: $T \to 2^P$ and OP: $T \to 2^P$ as follows:

$$IP(t_j) = \{p_i \in P: \text{IN}(p_i, t_j) \neq 0\} \, \forall t_j \in T$$
$$OP(t_j) = \{p_i \in P: \text{OUT}(p_i, t_j) \neq 0\} \, \forall t_j \in T$$

where $IP(t_j)$ is the set of input places of t_j and $OP(t_j)$ is the set of output places of t_j.

Example 2: For the PN of Fig. 2(a)

$$IP(t_1) = OP(t_2) = \{p_1, p_2\} \quad \text{and}$$
$$OP(t_1) = IP(t_2) = \{p_3\}.$$

Definition 2.3: A marking M of a Petri net G is a function $M: P \to N$. A marked Petri net W is a Petri net G together with a marking defined on it. We denote it by (G, M), and write $W = (G, M)$. We always associate an initial marking M_o with a given PN. M_o will represent the initial state of the system that the PN is modeling.

It can be noted that a marking of a PN with n places is an $(n \times 1)$ vector and associates with each place a certain number of tokens, which are represented by means of dots inside the places.

Example 3: Fig. 2(b) gives a marked PN with marking M_o given by

$$M_o = \begin{bmatrix} M_o(p_1) \\ M_o(p_2) \\ M_o(p_3) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

The marking M of the PNM of Fig. 2(c) is given by

$$M_1 = \begin{bmatrix} M_1(p_1) \\ M_1(p_2) \\ M_1(p_3) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Definition 2.4: A transition t_j of a PN is said to be enabled in a marking M if

$$M(p_i) \geq \text{IN}(p_i, t_j) \, \forall p_i \in \text{IP}(t_j).$$

An enabled transition t_j can fire at any instant of time. When a transition t_j enabled in a marking M fires, a new marking M' is reached according to the equation

$$M'(p_i) = M(p_i) + \text{OUT}(p_i, t_j) - \text{IN}(p_i, t_j) \, \forall p_i \in P.$$

We say marking M' is reachable from M and write $M \xrightarrow{t_j} M'$.

Example 4: In Fig. 2(b), transition t_1 is enabled in marking M_o. When t_1 fires, the marking M_1 is reached. Transition t_2 is enabled in M_1, and when t_2 fires, the new marking is M,. It can be seen that reachability of markings is a transitive relation on the set of all markings. In addition, by convention, we regard that a given marking is reachable from itself in zero steps (that is, by firing no transition).

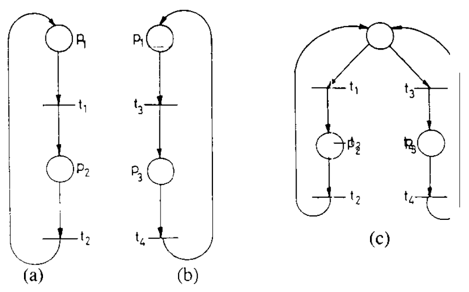


Fig. 3. (a), (b) Two Petri net models; (c) union of the above **two** Petri net models.

Definition 2.5: The set of all markings reachable from an initial marking M_o of a PN is called the reachability set of M_o and is denoted by $R[M_o]$.

Example 5: It can be seen from Figs. 2(a) and (b) that

$$R[M_o] = R[M_1] = \{M_o, M_1\}.$$

Definition 2.6: Let $G_1 = (P_1, T_1, \text{IN}_1, \text{OUT}_1)$ and $G_2 = (P_2, T_2, \text{IN}_2, \text{OUT}_2)$ be two PN's such that there exists no pair $(p, t) \in (P_1 \cap P_2) \times (T_1 \cap T_2)$ satisfying either

$$\text{IN}_1(p, t) \neq 0 \text{ and } \text{IN}_2(p, t) \neq 0$$

or

$$\text{OUT}_1(p, t) \neq 0 \text{ and } \text{OUT}_2(p, t) \neq 0.$$

We define the union of G_1 and G_2 as the Petri net G = (P, T, IN, OUT), where $P = P_1 \cup P_2$; $T = T_1 \cup T_2$; IN = $\text{IN}_1 \cup \text{IN}_2$, and OUT = $\text{OUT}_1 \cup \text{OUT}_2$. The union of any finite number of PN's nets is also defined likewise.

Example 6: The PN of Fig. 3(c) is the union of the Petri nets in Figs. 3(a) and (b).

Definition 2.7: Given a marked net (G, M_o), a reachable marking $M \in R[M_o]$ is called a deadlocked marking (or a deadlock) if no transition is enabled in M. A marked net (G, M_o) in which no reachable marking is deadlocked is said to be deadlock free.

We now introduce the notation of generalized stochastic PN's [16] (GSPN's), which are a special class of timed PN's.

Definition 2.8: A GSPN is a six-tuple (P, T, IN, OUT, M_o, F) where a) (P, T, IN, OUT, M_o) is a marked PN, b) T is partitioned into two sets T_I of immediate transitions and T_T of timed transitions, c) F is a function with domain $R[M_o] \times T_T$, which associates to each $t \in T_T$ in each $M \in R[M_o]$ a continuous random variable that indicates the firing time of t in M, and d) each $t \in T_I$ has zero firing time in all reachable markings.

In the graphical representation of GSPN's, a horizontal line represents an immediate transition, and a rectangular bar represents a timed transition. GSPN markings are classified into two types: vanishing markings (those in which at least one immediate transition is enabled) and tangible markings

TABLE I
DETAILS OF FIXTURE TYPES IN THE GE FMS

Part type	Stage of operation	Number of fixtures available
509	OP10	4
509	OP20	3
509	OP30	1
640	OP10	3
640	OP10	2
640	OP30	1

(those in which only timed transitions are enabled). In vanishing markings, as a rule, only an immediate transition is selected to fire even if timed transitions are enabled.

Example 7: Consider the PN of Fig. 2(b). Let t_1 be an immediate transition denoting the starting of a machine operation and t_2 be a timed transition denoting the actual machining operation. If we associate to transition t_2 a random variable equal to the processing time, this then becomes a GSPN model. M_o will then be a vanishing marking, and M_1 will be a tangible marking.

III. MODELING OF GE FMS

In this section, we develop a PN model for the General Electric FMS at Erie, PA, and exhibit typical deadlocks in the GE FMS.

A. Architecture of GE FMS

The GE FMS is designed to manufacture locomotive parts of two types called type 509 and type 640. Parts of type 509 undergo 17 operations in a sequence, and parts of type 640 undergo 18 operations in a sequence. The operations of each part type are divided into three different stages called $OP10$, $OP20$, and $OP30$.

There are 12 machines $M1$, $M2, \cdots, M12$, which are organized as seven different workstations $S1$, $S2, \cdots, S7$. Of these, $M1$ and $M3$ are special vertical milling machines; M4, $M9$, and $M10$ are large horizontal milling machines; $M5$ is a small horizontal milling machine; M7, M14, and M15 are medium horizontal milling machines; M13 and M17 are fixturing machines; M12 is the load/unload machine. Each workstation has two input buffers and one output buffer. There is no central storage in this FMS.

For each part type, different fixture types are required for the stages of operation $OP10$, $OP20$, and $OP30$. Thus, there are six types of fixtures. The number of fixtures of each type available in the GE FMS is given in Table I.

A part of a given type is loaded into the system and fixtured onto a fixture meant for its $OP10$. The part goes through several operations, and after finishing the stage $OP10$, it is defixtured and then fixtured onto a fixture meant for its $OP20$. After undergoing $OP20$, the part is again defixtured and then fixtured onto a fixture meant for its $OP30$. At the end of $OP30$, the part is defixtured and finally unloaded from the system.

Fig. 4 shows the routing table for the GE FMS. This table gives details of all operations on both part types. In addition to the machines involved in the particular operation, the routing table gives the processing times in minutes. These processing times are not the actual processing times but are those available from a simulation of the GE FMS.

There is an automated transporter that carries one part at a time from any source workstation to any destination workstation. The transportation times are insignificant compared with the processing times.

B. Petri Net Model of the GE FMS

In the PNM of the GE FMS, a place represents one of the following: available machines in a workstation, busy machines in a workstation, blocked machines in a workstation, parts waiting in an input buffer, parts waiting in an output buffer, and fixtures of a particular type. A transition in the PNM of the GE FMS represents one of the following six epochs of events: 1) commencement of loading operation, 2) commencement of processing (end of wait in input buffer), 3) end of blocking of a machine (commencement of wait in output buffer), 4) end of processing (beginning of blocking phase), 5) end of wait in output buffer (beginning of wait in input buffer of next machine), and 6) commencement of fixture changeover operation.

In the GE FMS, each part of type 509 goes through 17 operations, whereas each part of type 640 goes through 18 operations (see Fig. 4). Thus, the overall operation of the GE FMS involves 35 different types of operations. In terms of PN representation, this means that the overall PNM of the GE FMS is the union (see **Definition 2.6**) of the PNM's of the 35 individual operations [5], [13]. Therefore, to construct a PNM for the GE FMS, we first construct a PNM for each of the 35 operations and coalesce these PNM's using the paradigm of union of PN's. The detailed PNM's are available in [18].

C. Deadlock Situations in the GE FMS

It is reasonable to expect a complex system such as the GE FMS to have several deadlocks. Here, we give an example of a deadlock in the GE FMS. Consider a state of the GE FMS in which the configuration of the workstations $S1$, $S2$, and $S3$ is as shown in Fig. 5. The figure shows the two input buffers of each station on the left, the machines in the workstation at the center, and the output buffer of each station on the right. The input and output buffers and all machines except M5 carry a workpiece. The state of a workpiece is described by Jpi, where $p = 1, 2$ and $i = 0, 1, 2, \cdots, 17$. Jpi refers to a job of type 509 or 640 (depending on whether $p = 1$ or $p = 2$) undergoing operation i. Looking at the routing table of the GE FMS (Fig. 4), we can see that the parts in the output buffers of $S2$ and S3 are waiting for a slot in the input buffer of $S1$, whereas the part in the output buffer of $S1$ is waiting for a slot in the input buffer of either S2 or S3. However, the input buffers of $S1$, S2, and S3 are full, and the machines $M1$, M3, M4, M9, and $M10$ are blocked after finishing the processing of workpieces. A situation of this type leads to indefinite waiting, which is never resolved and represents a deadlocked state.

Such a state of the system is reachable from the initial state of the GE FMS, as can be seen from the following sequence of events occurring in three phases.

PART TYPE 509
OP10

0	1	2	3	4	5
P1	P2	P6	P3	P6	P9
20	30	112	168	22	36

PART TYPE 509
OP 20

6	7	8	9	10	11
P2	P1	P6	P4	P9	P1
20	20	78	93	42	20

PART TYPE509
OP3 0

12	13	14	15	16
P2	P8	P6	P2	P1
15	69	13	15	20

PART TYPE 640
OP10

0	1	2	3	4	5	6
P1	P2	P3	P6	P3	P6	P9
20	30	101	70	59	17	26

PART TYPE 640
O P20

7	8	9	10	11	12
P2	P1	P6	P4	P9	P1
20	20	64	101	52	20

PART TYPE 640
O P30

13	14	15	16	17
P2	P7	P5	P2	P1
15	56	16	15	20

MACHINES INVOLVED-

P1 -	M12
P2 -	M13 or M17
P3 -	M1 or M3
P4 -	M5
P5 -	M9 or M10
P6 -	M4 or M9 or M10
P7 -	M7
P8 -	M14 or M15
P9 -	M7 or M14 or M15

Fig. 4. Routing table of the GE FMS. In each table, the first row gives operation numbers, the second row gives the machines for the operations, and the third row gives the corresponding processing times.

Phase I: A part of type 509 and a part of type 640 are admitted into the system. They finish $OP10$.

Phase 2: Three parts of type 509 and two parts of type 640 are allowed to enter the system and complete $OP10$. Meanwhile, the two parts of phase 1 complete $OP10$.

Phase 3: Four parts of type 509 and three parts of type 640 are admitted into the system. Now, there are 14 jobs in the system, and all 14 fixtures are utilized. These 14 jobs eventually distribute themselves among stations $S1$, S2, and $S3$ in the manner shown in Fig. 5.

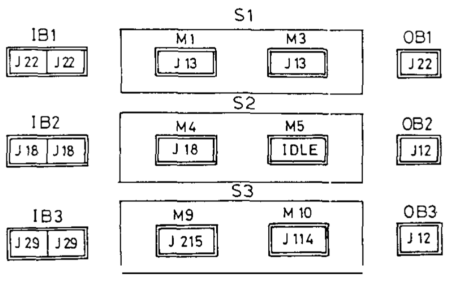


Fig. 5. Deadlock situation in GE FMS.

Using the invariants [5] of the PNM of the GE FMS, it can be shown formally that the above state corresponds to a deadlocked marking [18]. Invariants can often be used to prove the absence of deadlocks as well [5], [6], [8], [11], [12]. The invariants can be computed efficiently in the above case by invoking Theorem 1 of [5], which facilities the computation of the invariants of the union of a finite number of PN's in terms of the invariants of the individual nets.

IV. Deadlock Prevention

An FMS can be considered to be a concurrent system with several processes and resources. Processes correspond to parts inside the system, whereas resources in an FMS are the machines, input buffers, output buffers, conveyors, fixtures, etc. Parts inside an FMS compete for these shared resources. In the Computer Science literature [2]-[4], four conditions have been identified as necessary conditions for the occurrence of deadlock. These include the following:

1) ***Mutual exclusion:*** A resource cannot be used by two or more processes simultaneously,

2) No ***preemption:*** When a resource is being used, it is not released unless the process using it finishes with it.

3) ***Hold and wait:*** There must exist a process that is holding at least one resource and is waiting to acquire additional resources that are currently being held by other processes.

4) ***Circular wait:*** There must exist a set $\{p_1, p_2, \cdots, p_n\}$ of waiting processes such that p_1 is waiting for a resource that is held by p_2, p_2 is waiting for a resource that is held by $p_3, \cdots, p_{n-1}$ is waiting for a resource that is held by p_n, and p_n is waiting for a resource that is held by p_1.

Deadlock prevention consists of falsifying one or more of these necessary conditions using static resource allocation policies so that deadlocks are completely eliminated. We now show how the reachability graph of a PNM of a given FMS can be used to arrive at resource-allocation policies that enforce deadlock prevention. As an example, we consider the single-machine, single-AGV system of Fig. 1. Fig. 6 shows a PNM of this system and a description of the places and transitions is given in Table II. This is a GSPN model (Definition 2.8) where we distinguish between immediate transitions and timed transitions. Immediate transitions fire as soon as they are enabled and represent logical changes in states. Timed transitions fire a certain time after being enabled. We assume that these times are continuous random variables. We designate t_1, t_2, t_3, t_5, t_6, and t_8 as immedi-

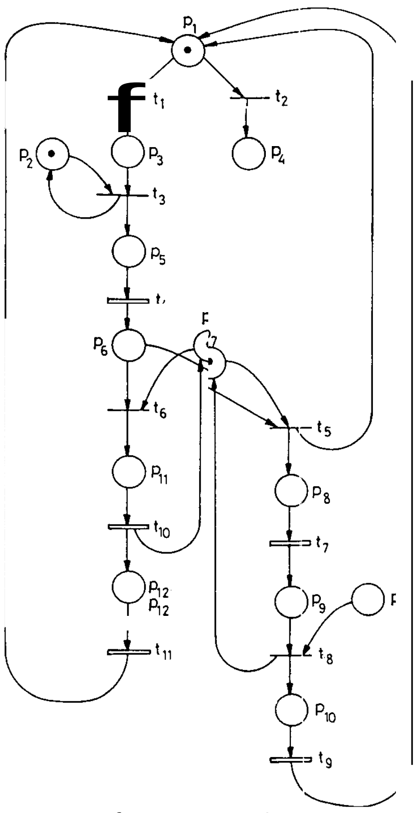


Fig. 6. GSPN model of the simple manufacturing system comprising an AGV and an NC machine.

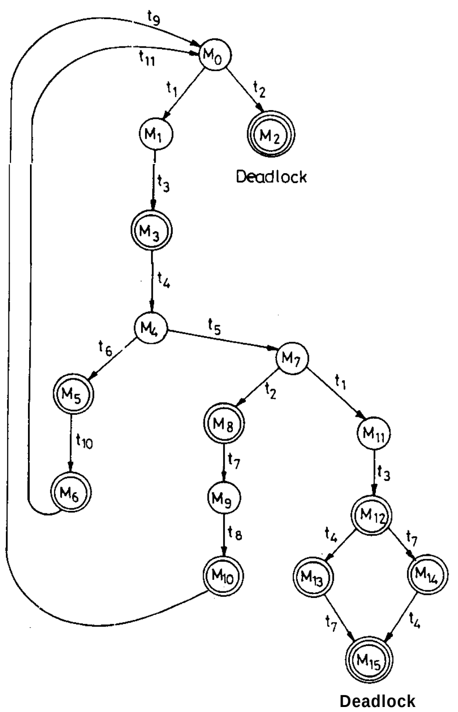


Fig. 7. Reachability graph of the GSPN model of Fig. 6. Single circles are vanishing markings, double circles are tangible markings, and triple circles are deadlocks.

TABLE II
DESCRIPTION OF THE GSPN MODEL OF FIG. 6

Places:

1	: AGV available
2	: Raw parts available
3	: AGV available to carry a raw part
4	: AGV available to carry a finished part
5	: AGV carrying a raw part to the NC machine
6	: AGV, with raw part, waiting for the NC machine
7	: NC machine available
8	: NC machine processing a part: AGV released
9	: NC machine waiting for AGV, after finishing processing
10	: AGV unloading the finished part
11	: NC machine processing a part: AGV not released
12	: AGV. not released during processing by Machine. unloading a finished part

Immediate Transitions:

1	: AGV assigned to raw part
2	: AGV assigned to finished part
3	: AGV starts transporting a raw part
5	: AGV released after finding machine free
6	: AGV not released after finding machine free
8	: AGV Starts unloading a finished part

Timed Transitions:

4	: AGV carrying a raw part to the NC machine
7	: Machine processing a part: AGV released
9	: AGV carrying a finished part to L/U station
10	: Machine processing a part: AGV not released
11	: AGV, not released during processing by machine, carrying a finished part to L/U Station

ate transitions and t_4, t,, t_9, t_{10}, and t_{11} as timed transitions. In the above **PNM,** there are two sets of conflicting immediate transitions: $\{t,, t_2\}$ and $\{t,, t_6\}$. The set $\{t,, t_2\}$ models the assignment of AGV to a raw part or a finished part. The set $\{t,, t_6\}$ models whether or not the AGV is released after carrying a part from the L/U station to the machine and finding the machine **free.** t, represents the release of the AGV, whereas t_6 models the holding of the

TABLE III
DESCRIPTION OF THE REACHABLE MARKINGS OF THE GSPN MODEL OF FIG. 6

Marking	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12
M_0	1	1	0	0	0	0	1	0	0	0	0	0
M_1	0	1	1	0	0	0	1	0	0	0	0	0
M_2	0	1	0	1	0	0	1	0	0	0	0	0
M_3	0	1	0	0	1	0	1	0	0	0	0	0
M_4	0	1	0	0	0	1	1	0	0	0	0	0
M_5	0	1	0	0	0	0	0	0	0	0	1	0
M_6	0	1	0	0	0	0	1	0	0	0	0	1
M_7	1	1	0	0	0	0	0	1	0	0	0	0
M_8	0	1	0	0	0	0	0	1	0	0	0	0
M_9	0	1	0	1	0	0	0	0	1	0	0	0
M_{10}	0	1	0	0	0	0	1	0	0	1	0	0
M_{11}	0	1	1	0	0	0	0	1	0	0	0	0
M_{12}	0	1	0	0	1	0	0	1	0	0	0	0
M_{13}	0	1	0	0	0	1	0	1	0	0	0	0
M_{14}	0	1	0	0	1	0	0	0	1	0	0	0
M_{15}	0	1	0	0	0	1	0	0	1	0	0	0

AGV until the machine finishes processing and the AGV unloads the finished part.

Fig. 7 depicts the reachability graph of the above **PNM.** There are **16** markings: M_0, $M_1, \cdots, M_{15}$. The description of these markings is given in Table III. We distinguish the markings into three classes: vanishing markings (those in which at least one immediate transition is enabled), tangible markings (those in which only timed transitions are enabled) [16], and deadlocks in which none of the transitions is enabled. Vanishing markings model the states in which the system stays for zero time, and they only indicate logical changes of state. Tangible markings are those in which the system will sojourn for nonzero time due to the progress of one or more timed activities in the system. Deadlocks are absorbing states in which the system will have to stay forever. In Fig. 7, vanishing markings are shown as single circles, tangible markings as double circles, and deadlocks as triple circles. The labels on the arcs indicate the transitions to be fired. From the graph, the following can be inferred:

1) The deadlock M_2 can be prevented by firing $t,$ in preference to $t,$ in the marking M_o, that is, the deadlock M_2 can be prevented by assigning the AGV to only a raw part when no finished part is waiting.

2) The deadlock M_{15} can be prevented by firing t_6 in preference to t_5 in the marking M_4. This means that we do not release the AGV after the AGV transports a raw part to the machine, and the machine takes up the raw part for processing. In this case, we hold the AGV until the machine finishes processing and the AGV unloads the finished part.

3) The deadlock M_{15} can also be prevented by firing t_2 in preference to t_1 in marking M_7, that is, by assigning the AGV to a finished part when a finished part is waiting.

As is shown above, an exhaustive path analysis of the reachability graph can lead to a set of resource allocation policies that prevent the occurrence of deadlocks. It is enough to do such an analysis just once in order to devise deadlock-prevention policies. Such a method has earlier been used in the context of safety critical systems by Leveson and Stolzy [19].

V. Deadlock Avoidance

Deadlock prevention is accomplished by static policies and is known to result in poor resource utilization [3], [4]. In addition, the reachability analysis technique to arrive at deadlock prevention policies can become infeasible if the state space is very large, as in the case of a real-life FMS such as the GE FMS. Deadlock avoidance is the preferred alternative in such cases. In deadlock avoidance, we attempt to falsify one or more of the necessary conditions in a dynamic way by keeping track of the current state and the possible future conditions. The idea is to let the necessary conditions prevail as long as they do not cause a deadlock but falsify them as soon as a deadlock becomes a possibility in the immediate future. As a result, deadlock avoidance leads to better resource utilization.

In this section, we present an on-line monitoring and control system, based on PN's, for implementing deadlock avoidance. This system will avoid most of the deadlocks and for deadlocks that are not predicted by this scheme, recovery mechanisms have to be used. We first present some definitions.

Definition 5.1: The look ahead of a deadlock-avoidance policy is the number of steps of future evolution of the system computed before making a resource-allocation decision.

Definition 5.2: Given a PNM (P, T, IN, OUT, M_o), a marking $M \in R[M_o]$ is said to be blocked if there exists a $t \in T$ such that a) t has two or more input places, b) there exists a $p \in IP(t)$ such that $M(p) \geq IN(p, t)$, c) t is disabled in M.

Note: The motivation for the above definition is to capture markings in which processes are blocked waiting for resources. Blocking can be represented by a partially enabled transition having two or more input places. A blocked marking is to be distinguished from a deadlocked marking in which all transitions are disabled **(Definition 2.7).** Blocking is a necessary but not a sufficient condition for the occurrence of a deadlock. A blocked marking is often a good portent of a deadlock.

Definition 5.3: A marking M of a PNM is designated safe if it is neither blocked nor deadlocked.

Note: The term "safe" here is inspired by the Operating Systems literature [2]–[4] and is not to be confused with the safeness property of PN's in classical PN literature.

Notation: A marking M can only be of three types: safe, blocked, and deadlocked. We use the labels S, B, and D, respectively, to designate a marking.

Definition 5.4: Given a PNM (P, T, IN, OUT, M_o) and a marking $M \in R[M_o]$, the future set of markings reachable from M in i steps $i \geq 0$ is denoted and defined by $L_i(M) = \{(a, M', t)$, where M' is reachable from M in exactly i steps by firing the transition sequence and is of type t where t may be S, B or $D\}$.

Note: Given a marking M of type t, we have

$$L_0(M) = \{(\epsilon, M, t)\}$$

where ϵ is the null transition sequence. In addition, for $i \geq 0$, $j \geq 0$, if $M' \in L_i(M)$, then the elements of $L_j(M')$ will be contained in $L_{i+j}(M)$. Hence, if $L_{i+j}(M)$ is known, $L_j(M')$ can be obtained. We first motivate PN-based deadlock avoidance using an example and then discuss the on-line controller.

A. Example to Illustrate Deadlock Avoidance

Here, we consider again the single-machine–single-AGV system depicted in Fig. 1. A PNM of this system is shown in Fig. 6, and the reachability graph is shown in Fig. 7. We discuss this example for look ahead 1. Therefore, we look at the $L_1(\cdot)$ function only. Let us say we start in the initial marking M_o. This is a vanishing marking in which two conflicting immediate transitions $t,$ and $t,$ are enabled. When we fire $t,$, we obtain marking M_1, which is a safe state. When we fire $t,$, we obtain marking M_2, which is a deadlock. Thus, we have

$$L_1(M_o) = \{(t_1, M_1, S), (t_2, M_2, D)\}.$$

To avoid the deadlock, we have to fire t_1 in preference to t_2, that is, we should assign the resource AGV to a raw part. In this case, we have predicted a deadlock with a look ahead of 1. After firing t_1, the system reaches the state M,. M_1 is a vanishing marking in which only one immediate transition is enabled. We have

$$L_1(M_1) = \{(t_3, M_3, S)\}$$

Therefore, we can fire $t,$, which means that the AGV starts transporting the raw part. M_3 is a tangible marking that represents the transport of a raw part by the AGV from the L/U station to the machine. As soon as the AGV finishes, the PN marking can be updated to M_4. State M_4 is a vanishing marking in which two conflicting immediate transitions t_5 and t_6 are enabled. We see that

$$L_1(M_4) = \{(t_6, M_5, S), (t_5, M_7, S)\}.$$

Since both of the next states are safe, we can choose any transition to fire. Let us choose *t,,* that is, we release the AGV after the AGV reaches the machine and finds the machine available. Thus, the current marking is M_7, which is, again, a vanishing marking with two conflicting transitions *t,* and *t,*. We find

$$L_1(M_7) = \{(t_2, M_8, S), (t_1, M_{11}, S)\}.$$

The choice here is between assigning the released AGV to a raw part or a finished part. Let us say we fire *t,,* that is, we assign the AGV to the next raw part (which is already available). We reach the marking M_{11}, which is a vanishing marking with *t,* as the only enabled transition. We have

$$L_1(M_{11}) = \{(t_3, M_{12}, S)\}.$$

The firing of t_3 means that the AGV starts transporting a fresh raw part. The marking M_{12} is a tangible marking in which the machine and the AGV are both busy. Depending on whichever finishes faster, we will reach M_{13} or M_{14}. If we assume that the AGV transport time and the machine processing time are independent continuous random variables, then the AGV and the machine cannot finish simultaneously. We have

$$L_1(M_{12}) = \{(t_4, M_{13}, B), (t_7, M_{14}, B)\}.$$

Since t_4 and t_7 are activities in the physical system, we do not have any control over their progress. However, whether t_4 or t_7 fires first, we end up in a blocked state. In M_{13}, the AGV is blocked while waiting for the machine (t_5 and t_6 are disabled), whereas in M_{14}, the machine is blocked while waiting for the AGV (t_8 is disabled). Let us say that the AGV finishes first and that we reach the marking M_{13}. Now

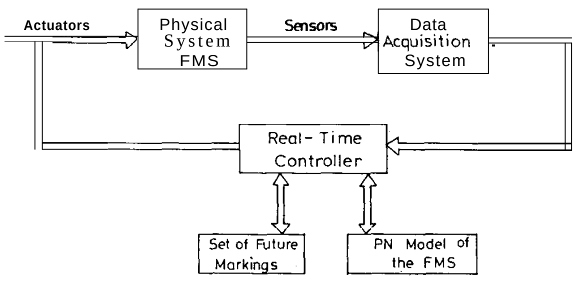
$$L_1(M_{13}) = \{(t_7, M_{15}, D)\}.$$

M_{13} is a tangible marking, and eventually, t_7 fires, resulting in the deadlocked state M_{15}, in which both the AGV and the machine are blocked. Thus, using a look ahead of 1, we are able to avoid only one deadlock (M_2). This will be the case with look aheads of 2 and 3 as well. It can be shown that a look ahead of 4 will avoid both the deadlocks. We can make the following observations:

1) Greater look ahead implies greater probability of avoiding deadlocks. However, there can be systems where only infinite look ahead will guarantee total deadlock avoidance. For this reason, deadlock avoidance may have to be supplemented by deadlock recovery.

2) In the case of look ahead equal to 1, the deadlock M_{15} is predicted in M_{13} or M_{14}. In the case of look ahead of 2, the deadlock is predicted in M_{12} (two steps earlier), and if look ahead equals 3, the deadlock is predicted in M_{11} itself. Therefore, the cost of deadlock recovery becomes less with increasing look ahead.

3) The PN framework is suitable for implementing deadlock avoidance. Vanishing markings with conflicting transitions naturally model resource-allocation decisions; tangible markings model the progress of timed activities, which are not controllable once started. The evolution of the system can be easily determined by computing the future markings using the *L* function.



B. On-line Controller for Deadlock Avoidance

We now present an on-line controller for deadlock avoidance in any FMS using PN's. The controller is basically an on-line monitoring system. Fig. 8 shows the components of the proposed controller. These components are described below.

Physical System: This block corresponds to the actual FMS in operation.

Data Acquisition System: This unit is responsible for gathering, using various sensors, status information of all resources in the FMS. The output of this unit can be used to determine the current marking of the PNM of the FMS.

Petri Net Model: This corresponds to a data structure that efficiently stores the PNM of the FMS. The construction of this model can be carried out easily using the paradigm of union of PN's, as is detailed in Section III. This data structure also includes a field for the current marking, which is updated constantly by the real-time controller.

Set of Future Markings: This is another data structure that efficiently stores the sets $L_1(M)$, $L_2(M), \cdots, L_n(M)$, where n is the look ahead employed and M is a current marking. These sets are crucially used by the real-time controller to select the immediate transitions to fire. When the marking of the PNM changes, the L sets for the new marking can be computed easily from those of the current marking.

Real-time Controller (RTC): The inputs to this unit are the look ahead to be employed and the sensor output data for the current state of the FMS. The controller has access to the two data structures, namely, PNM and the set of future markings. This unit mainly performs three functions in each iteration.

1) Determination of the current marking of the PNM.
2) Classification of the current marking into deadlock, tangible marking, or vanishing marking.
3) Looking ahead into the system evolution and initiation of appropriate actions.

The RTC first checks if the previous marking, say P (not to be confused with the notation for the set of places for a

```
Algorithm:  Real-Time Controller
Input:  (1) Petri Net Model of the FMS. (2) n, the look-ahead.
        (3) P, the previous marking. (4) $L_1(P)$, $L_2(P)$,..., $L_n(P)$.
Output: Appropriate Scheduling Decision or Deadlock Recovery
        Action
Local Variables: i (integer); deadlock-flag (boolean);
begin
     if  P  tangible then compute the current marking M after
          reading appropriate sensor outputs
     else compute the current marking M by firing in P the
          transition that was selected to fire;
     Compute E, the set of enabled transitions in M;
     if  E = Ø {M is a deadlock) then initiate deadlock recovery
     else if E contains only timed transitions {M is tangible) then
             initiate monitoring of activities in progress
          else {M vanishing} begin
               i:= 1;
               repeat
                  compute $L_i(M)$ using $L_{i+1}$ (P);
                  if $L_i(M)$ contains only deadlocks then deadlock-
                     flag: = true
                  else i: = i+1
               until (i = n+1 or deadlock-flag);
               if deadlock-flag then initiate appropriate advance
                  deadlock-recovery
               else begin {$L_n(M)$ contains at least one safe state
                          or one blocked state)
                       if L (M) contains at least one safe state then
                          select for firing an immediate
                          transition that leads to one of these
                          safe states
                       else select for firing an immediate transition
                            that leads to one of the blocked states
                    end
          end
end.
```

Fig. 9. Informal algorithm for the real-time controller.

PNM), was tangible or vanishing. If P was tangible, then it obtains the current marking M by reading off appropriate sensor data output values. This is because in a tangible marking, several activities are in progress, and the next marking is decided by the activity that finishes first. The finishing of an activity is indicated by a sensor, which is read off by the data acquisition system. If the previous marking P was vanishing, then the RTC computes the current marking M as the marking obtained by firing in P the transition that was selected to fire in the previous iteration. Having determined M, the RTC updates the PNM to reflect the change in marking.

In the second step, the RTC classifies the current marking M into a deadlock or a tangible marking or a vanishing marking. To this end, the RTC first computes the set E of enabled transitions in M. If E is empty, then M is a deadlock. If E contains only timed transitions, then M is a tangible marking; otherwise, M is a vanishing marking. The actions of the RTC now depend on this classification.

a) If M is a deadlock, the RTC initiates appropriate deadlock recovery actions or informs the operator if necessary.

b) If M is a tangible marking, then one or more activities are in progress. Therefore, we have to monitor these activities to determine the next state of the FMS. The RTC in this case generates signals to activate appropriate sensors to monitor these activities. Note that typical activities include processing by a machine, part transfer by a robot, loading of raw parts, unloading of finished parts, transport of semi-finished parts, etc.

c) If M is a vanishing marking, then at least one immediate transition is enabled and a decision may be required to be made about assigning or releasing some resource. Here, we use the look ahead into the system evolution up to n steps, where n is the look ahead. We select an immediate transition to fire to avoid a deadlock as far as possible. First, the RTC computes $L_1(M)$ by selecting appropriate elements of $L_2(P)$, where P is the previous marking (note that $L_1(M)$

is a subset of $L_2(P)$). If $L_1(M)$ contains only deadlocks, then the **RTC** initiates advance deadlock recovery. Otherwise, it computes $L_2(M)$ using $L_3(P)$. Again, it repeats the steps as in the case of $L_1(M)$. If each $L_i(M)$ contains no deadlocks for $i = 1, 2, \cdots, n - 1$, it computes $L_n(M)$, where n is the look ahead. If $L_n(M)$ contains only deadlocks, the **RTC** initiates advance deadlock recovery. If $L_n(M)$ contains at least one safe state, it will select for firing an immediate transition enabled in M that would lead to a safe state at the end of n steps. If $L_n(M)$ contains no safe states, the **RTC** selects for firing an immediate transition that would lead to a blocked state after n steps. The immediate transition that is finally chosen to fire will depend on the actual system. Depending on the immediate transition selected to fire, appropriate actuators are set.

Fig. 9 gives (in a Pascal-like language) an algorithm that describes the working of the **RTC** in each iteration. It can be seen that such a controller can, in principle, be implemented for real-world FMS's such as the GE FMS.

VI. Conclusion

In this paper, we have demonstrated the use of Petri nets in the modeling of FMS's and in prevention and avoidance of deadlocks in FMS's. We have shown that the paradigm of union of Petri nets can be used in a bottom-up construction of large Petri net models, as in the case of the General Electric FMS. The Petri net model captures all behavioral characteristics of an FMS, including deadlocks. Deadlocks can cause serious performance degradation, and eliminating them is very important for effective automated operation of FMS's. Deadlock handling can take two forms: deadlock prevention in which deadlocks are eliminated by static resource allocation policies and deadlock avoidance in which dynamic policies are employed to avert deadlocks just in time. We have shown the following:

a) Deadlock prevention policies can be devised by conducting an exhaustive path analysis of the reachability graph of a PN model of the given FMS; such an option is feasible for reasonably small systems.

b) Deadlock avoidance can be implemented effectively by an on-line monitoring and control system that employs the PN model to look ahead into the future evolution in order to make a resource-allocation decision; the rare occurrence of deadlocks that cannot be captured by the look ahead that is employed can be handled by suitable deadlock-recovery strategies. Deadlock avoidance is feasible for large real-world FMS's, such as the GE **FMS.**

There are two important issues for future investigation: 1) software implementation of the on-line controller for deadlock avoidance and 2) quantitative analysis in the context of deadlocks.

An effective software implementation of the on-line controller for deadlock avoidance will have to consider the following issues:

1) Suitable data structures for the PN model and the set of future markings
2) classifying a given marking of the PN model into a tangible marking or a vanishing marking and designating it as safe or blocked or deadlocked
3) Efficient computation of future markings and firing sequences for the current marking from those of the previous marking
4) Effective deadlock-recovery strategies.

With respect to a quantitative study of **FMS's** with deadlocks, there is good potential in using the theory of Markov chains with absorbing states [20] to compute the mean time to deadlock and the mean number of parts produced before deadlock. In addition, GSPN models, which have been used in [21] and [22] for performance evaluation of **FMS's,** can be used for comparing the relative effectiveness of different deadlock-prevention algorithms.

References

[1] Y. C. Ho, "Performance evaluation and perturbation analysis of discrete event dynamic systems," *ZEEE Trans. Automat. Contr.*, vol. **AC-32,** no. **7,** pp. **563-472,** July **1987.**

[2] E. G. Coffman, Jr., M. J. Elphick, and A. Shoshani, "System deadlocks," *ACM Comput. Surveys,* vol. **3,** no. **2,** pp. **67-78,** June **1971.**

[3] A. N. Habermann, "System deadlocks," in *Current Trends in Programming Methodology, Vol. III* (K. M. Chandy and R. T. Yeh, Fds.). Englewood Cliffs, NJ: Prentice-Hall, **1977,** pp. **256-297.**

[4] J. L. Peterson and A. Silberschatz, *Operating System Concepts.* Reading, MA: Addison-Wesley, **1985** (2nd ed.).

[5] Y. Narahari and N. Viswanadham, "A Petri net approach to modelling and analysis of flexible manufacturing systems," *Annals Oper. Res.,* vol. **3,** pp. **449-472, 1985.**

[6] H. Alla, P. Ladet, J. Martinez, and M. Silva, "Modelling and validation of complex systems by Petri nets: Application to FMS," in *Lecture Notes in Computer Science, Vol. 188.* New York: Springer-Verlag, **1985,** pp. **15-32.**

[7] M. Kamath and N. Viswanadham, "Applications of Petri net based models in the modelling and analysis of flexible manufacturing systems," in *Proc. 1986 IEEE Conf. Robotics Automat.,* Apr. **1986,** pp. **312-316.**

[8] J. Martinez, H. Alla, and M. Silva, "Petri nets for specification of FMS's," in *Modelling and Design of FMS* (A. Kusiak (Ed.)). New York: Elsevier, **1986,** pp. **389-406.**

[9] E. S. Acree and M. L. Smith, "Simulation of a flexible manufacturing system—Application of computer operating system techniques,"in *Proc. 18th ZEEE Simulation Symp.*, Mar. **1985,** pp. **205-216.**

[10] J. L. Peterson, *Petri net Theory and the Modelling of Systems.* Englewood Cliffs, NJ: Prentice-Hall, **1981.**

[11] W. Reisig, "Petri nets: An introduction," in *EATCS Monographs on Theoretical Computer Science.* Berlin: Springer-Verlag, **1985.**

[12] N. Viswanadham and Y. Narahari, "Coloured Petri net models for automated manufacturing systems," in *Proc. 1987 IEEE Znt. Conf. Robotics Automat.,* Mar.-Apr. **1987,** pp. **1985-1990.**

[13] Y. Narahari, "Petri net-based techniques for modelling, analysis, and performance evaluation," Doctoral dissertation, Dept. Comput. Sci. Automat., Indian Inst. Sci., Bangalore, India, July **1987.**

[14] C. L. Beck and B. H. Krogh, "Models for simulation and discrete control of manufacturing systems," in *Proc. Int. Conf. Robotics Automat.,* Apr. **1986,** pp. **305-310.**

[15] **T.** Murata, "Modelling and Analysis of Concurrent Systems," in *Handbook of Software Engineering* (C. R. Vick and C. V. Ramamoorty Eds.). New York: Van Nostrand Reinhold, **1984,** pp. **39-63.**

[16] M. A. Marsan, G. Balbo, and G. Conte, "A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems," *ACM Trans. Computer Systems,* vol. **2,** no. **2,** pp. **93-122,** May **1984.**

[17] J. B. Dugan, K. S. Trivedi, R. M. Geist, and V. F. Nicola, "Extended Stochastic Petri Nets: Applications and analysis," in *Proc. Performance '84* (Paris, France), Dec. **1984,** pp. **507-519.**

[18] N. Viswanadham, Y. Narahari, and T. L. Johnson, "Petri net-based investigations on the General Electric flexible manufacturing system,"

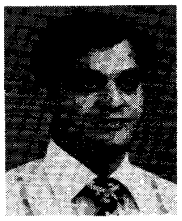
Tech. Rep., Dept. Comput. Sci. Automat., Indian Inst. Sci., Bangalore, May 1987.

[19] N. G. Leveson, and J. L. Stolzy, "Safety analysis using Petri nets," *IEEE Trans. Software Eng.*, vol. SE-13, no. 3, pp. 386–397, Mar. 1987.

[20] K. S. Trivedi, *Probability and Statistics with Reliability, Queueing, and Computer Science Applications.* Englewood Cliffs, NJ: Prentice-Hall, 1982.

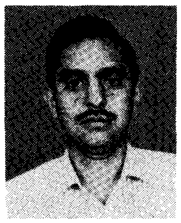
[21] G. Balbo, G. Chiola, Franceschinis, and G. M. Roet, "Generalized stochastic Petri nets for the performance evaluation of FMS," in *Proc. IEEE Int. Conf. Robotics Automat.* (Raleigh, NC), Mar.-Apr. 1987, pp. 1013–1018.

[22] N. Viswanadham and Y. Narahari, "Stochastic Petri nets for performance evaluation of automated manufacturing systems," *Inform. Decision Technol.*, vol. 14, pp. 125–142, 1988.



N. Viswanadham (SM'86) received the Ph.D. degree in 1970 from the Indian Institute of Science (IISc), Bangalore, India. Since August 1987, he has been on the faculty of IISc, where currently, he is a Professor and chairperson of the Department of Computer Science and Automation. He has held several visiting appointments at the University of New Brunswick, the University of Waterloo, and the General Electric Corporate Research and Development Center. He was a GE Research Fellow during 1989. Since 1981, his research investigations have been in the areas of automated manufacturing systems and fault-tolerant control system design. His current research interests are in the areas of fault-tolerant control system design, large-scale dynamic systems, flexible manufacturing systems, and distributed computing systems. He is the author of more than 55 referred journal publications and 60 conference papers. He is a joint author of a book entitled *Reliability in Computer and Control Systems* (North-Holland, 1987). He is currently an Associate Editor at large for the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, Associate Editor of the Journals: *Control Theory and Advanced Technology*, (MITA Press, Japan); *Information and Decision Technologies* (North-Holland, Amsterdam); *Intelligent and Robotics Systems* (Kluwer Academic); and *Sadhana* (Indian Academy of Sciences).

Dr. Viswanadham is a Fellow of Indian National Science Academy and the Indian Academy of Sciences and Indian National Academy of Engineering.



Y. Narahari received the M.E. degree in computer science in 1984 and the Ph.D. degree in 1987 from the Department of Computer Science and Automation, Indian Institute of Science (IISc), Bangalore. His Doctoral Dissertation was on Petri net-based performance analysis of flexible manufacturing systems.

He is currently an Assistant Professor in the Department of Computer Science and Automation at IISc. His current research is focused on stochastic modeling of automated manufacturing systems and on performance modeling of distributed computing systems. He has several research publications in these areas.

Timothy L. Johnson (S'69–M'72) received the S.B., S.M., and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, in 1968, 1969, and 1972, respectively.

He is currently Manager of the Control Systems and Architecture Program at General Electric Corporate Research and Development, Schenectady, N.Y. He was a Senior Scientist with the Automated Systems Department of BBN Laboratories, Inc., from 1980 to 1984 and served as Assistant and Associate Professor of Electrical Engineering and Computer Science at MIT from 1972 to 1980. He has held visiting positions with the Department of Neurology, Boston University, Brown University, Imperial College (London), IRIA (Paris), LAAS (Toulouse), and he is currently an Adjunct Professor of Electrical Engineering at Rensselaer Polytechnic Institute, Troy, N.Y.

He was the recipient of the Donald P. Eckman Award in 1974 and was Edgerton Assistant Professor at MIT from 1973 to 1975. He served as an elected member of the IEEE Control Systems Society Board of Governors from 1983–1989 and as Associate Editor at Large of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL from 1986–1989.