24

Petri Nets for Dynamic Event-Driven System Modeling

Jiacun Wang
Monmouth University

24.1 Introduction **24**-1
24.2 Petri Net Definition **24**-1
24.3 Transition Firing **24**-3
24.4 Modeling Power **24**-4
24.5 Petri Net Properties **24**-5
Reachability • Safeness • Liveness
24.6 Analysis of Petri Nets **24**-7
Reachability Analysis • Incidence Matrix and State Equation • Invariant Analysis • Simulation
24.7 Colored Petri Nets **24**-10
24.8 Timed Petri Nets **24**-12
Deterministic Timed Petri Nets • Stochastic Timed Petri Nets
24.9 Concluding Remark **24**-16

24.1 Introduction

Petri nets were introduced in 1962 by Dr. Carl Adam Petri (Petri, 1962). Petri nets are a powerful modeling formalism in computer science, system engineering, and many other disciplines. Petri nets combine a well-defined mathematical theory with a graphical representation of the dynamic behavior of systems. The theoretical aspect of Petri nets allows precise modeling and analysis of system behavior, while the graphical representation of Petri nets enables visualization of the modeled system state changes. This combination is the main reason for the great success of Petri nets. Consequently, Petri nets have been used to model various kinds of dynamic event-driven systems such as computer networks (Ajmone Marsan et al., 1986), communication systems (Merlin and Farber, 1976; Wang, 2006), manufacturing plants (Venkatesh et al., 1994; Zhou and DiCesare, 1989; Desrochers and Ai-Jaar, 1995), command and control systems (Andreadakis and Levis, 1988), real-time computing systems (Mandrioli and Morzenti, 1996; Tsai et al., 1995), logistic networks (van Landeghem and Bobeanu, 2002), and workflows (van der Aalst and van Hee, 2000; Lin et al., 2002) to mention only a few important examples. This wide spectrum of applications is accompanied by wide spectrum different aspects, which have been considered in the research on Petri nets.

AQ1

24.2 Petri Net Definition

A Petri net is a particular kind of bipartite directed graphs populated by four types of objects. These objects are *places*, *transitions*, *directed arcs*, and *tokens*. Directed arcs connect places to transitions or transitions to

0-8493-XXXX-X/02$0.00+$1.50

places. In its simplest form, a Petri net can be represented by a transition together with an input place and an output place. This elementary net may be used to represent various aspects of the modeled systems. For example, a transition and its input place and output place can be used to represent a data processing event, its input data and output data, respectively, in a data processing system. To study the dynamic behavior of a Petri net modeled system in terms of its states and state changes, each place may contain zero or a positive number of tokens. Tokens are a primitive concept for Petri nets in addition to places and transitions. The presence or absence of a token in a place can indicate whether a condition associated with this place is true or false, for instance.

Denote by N the set of nonnegative integers. A Petri net is formally defined as a five-tuple $N = (P, T, I, O, M_0)$, where

(1) $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places.
(2) $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions. $P \cup T \neq \oslash$ and $P \cap T = \oslash$.
(3) I: $T \times P \to N$ is an *input matrix* that specifies directed arcs from places to transitions; its entry $I(t_i, p_j)$ represents the number of arcs connecting place p_j to transition t_i.
(4) O: $T \times P \to N$ is an *output matrix* that specifies directed arcs from transitions to places; its entry $O(t_i, p_j)$ represents the number of arcs connecting transition t_i to place p_j.
(5) M_0: $P \to N$ is the *initial marking*.

A *marking* in a Petri net is an assignment of tokens to the places of a Petri net. Tokens reside in the places of a Petri net. The number and position of tokens may change during the execution of a Petri net. The tokens are used to define the execution of a Petri net.

Most theoretical work on Petri nets is based on the formal definition of Petri nets. However, a graphical representation of a Petri net is much more useful for illustrating the concepts of Petri net theory. A Petri net graph is a Petri net depicted as a bipartite directed multigraph. Corresponding to the definition of Petri nets, a Petri net graph has two types of nodes: a *circle* that represents a place, and a *bar* or *box* that represents a transition. Directed arcs (arrows) connect places and transitions, with some arcs directed from places to transitions and other arcs directed from transitions to places. An arc directed from a place p_j to a transition t_i defines p_j to be an *input place* of t_i, denoted by $I(t_i, p_j) = 1$. An arc directed from a transition t_i to a place p_j defines p_j to be an *output place* of t_i, denoted by $O(t_i, p_j) = 1$. If $I(t_i, p_j) = k$ (or $O(t_i, p_j) = k$), then there exist k directed (parallel) arcs connecting place p_j to transition t_i (or connecting transition t_i to place p_j). Usually, in the graphical representation, parallel arcs connecting a place (transition) to a transition (place) are represented by a single directed arc labeled with its multiplicity, or weight k. A circle contains a *dot* represents a place contains a token (Peterson, 1981).

AQ2

Example 1

A simple Petri net.

Figure 24.1 shows a simple Petri net. In this Petri net, we have

$P = \{p_1, p_2, p_3, p_4\}$;
$T = \{t_1, t_2, t_3\}$;
$I(t_1, p_1) = 2, I(t_1, p_i) = 0$ for $i = 2, 3, 4$;

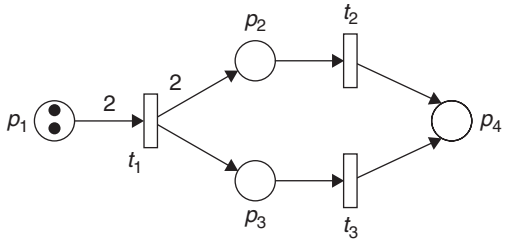

FIGURE 24.1 A simple Petri net.

$I(t_2, p_2) = 1, I(t_2, p_i) = 0$ for $i = 1, 3, 4$;
$I(t_3, p_3) = 1, I(t_3, p_i) = 0$ for $i = 1, 2, 4$;
$O(t_1, p_2) = 2, O(t_1, p_3) = 1, O(t_1, p_i) = 0$ for $i = 1, 4$;
$O(t_2, p_4) = 1, O(t_2, p_i) = 0$ for $i = 1, 2, 3$;
$O(t_3, p_4) = 1, O(t_3, p_i) = 0$ for $i = 1, 2, 3$;
$M_0 = (2\ 0\ 0\ 0)$.

24.3 Transition Firing

The execution of a Petri net is controlled by the number and distribution of tokens in the Petri net. By changing distribution of tokens in places, which may reflect the occurrence of events or execution of operations, for instance, one can study the dynamic behavior of the modeled system. A Petri net is executed by *firing* transitions. We now introduce the enabling rule and firing rule of a transition, which govern the flows of tokens:

(1) *Enabling Rule*: A transition t is said to be *enabled* if each input place p of t contains at least the number of tokens equal to the weight of the directed arc connecting p to t, i.e., $M(p) \geq I(t, p)$ for all p in P. If $I(t, p) = 0$, then t and p are not connected, so we do not care about the marking of p when considering the firing of t.
(2) *Firing Rule*: Only enabled transitions can fire. The firing of an enabled transition t removes from each input place p the number of tokens equal to $I(t, p)$, and deposits in each output place p the number of tokens equal to $O(t, p)$.

Mathematically, firing t at M yields a new marking

$$M'(p) = M(p) - I(t, p) + O(t, p) \quad \text{for all } p \text{ in } P$$

Note that since only enabled transitions can fire, the number of tokens in each place always remains nonnegative when a transition is fired. Firing a transition can never try to remove a token that is not there.

A transition without any input place is called a *source transition*, and one without any output place is called a *sink transition*. Note that a source transition is unconditionally enabled, and that the firing of a sink transition consumes tokens, but does not produce tokens.

A pair of a place p and a transition t is called a *self-loop*, if p is both an input place and an output place of t. A Petri net is said to be *pure* if it has no self-loops.

Example 2

Transition firing.

Consider the simple Petri net shown in Figure 24.1. Under the initial marking, $M_0 = (2\ 0\ 0\ 0)$, only t_1 is enabled. Firing of t_1 results in a new marking, say M_1. It follows from the firing rule that

$$M_1 = (0\ 2\ 1\ 0)$$

The new token distribution of this Petri net is shown in Figure 24.2. Again, in marking M_1, both transitions of t_2 and t_3 are enabled. If t_2 fires, the new marking, say M_2, is

$$M_2 = (0\ 1\ 1\ 1)$$

If t_3 fires, the new marking, say M_3, is

$$M_3 = (0\ 2\ 0\ 1)$$

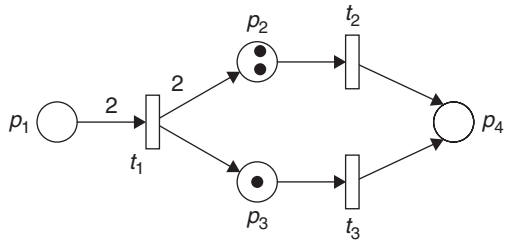

FIGURE 24.2 Firing of transition t_1.

24.4 Modeling Power

The typical characteristics exhibited by the activities in a dynamic event-driven system, such as concurrency, decision making, synchronization, and priorities, can be modeled effectively by Petri nets.

1. *Sequential Execution*. In Figure 24.3(a), transition t_2 can fire only after the firing of t_1. This imposes the precedence constraint "t_2 after t_1." Such precedence constraints are typical of the execution of the parts in a dynamic system. Also, this Petri net construct models the causal relationship among activities.
2. *Conflict*. Transitions t_1 and t_2 are in conflict in Figure 24.3(b). Both are enabled but the firing of any transition leads to the disabling of the other transition. Such a situation will arise, for example, when a machine has to choose among part types or a part has to choose among several machines. The resulting conflict may be resolved in a purely nondeterministic way or in a probabilistic way, by assigning appropriate probabilities to the conflicting transitions.
3. *Concurrency*. In Figure 24.3(c), the transitions t_1 and t_2 are concurrent. Concurrency is an important attribute of system interactions.
4. *Synchronization*. It is quite normal in a dynamic system that an event requires multiple resources. The resulting synchronization of resources can be captured by transitions of the type shown in Figure 24.3(d). Here, t_1 is enabled only when each of p_1 and p_2 receives a token. The arrival of a token into each of the two places could be the result of a possibly complex sequence of operations elsewhere in the rest of the Petri net model. Essentially, transition t_1 models the joining operation.
5. *Mutually exclusive*. Two processes are mutually exclusive if they cannot be performed at the same time due to constraints on the usage of shared resources. Figure 24.3(e) shows this structure. For example, a robot may be shared by two machines for loading and unloading. Two such structures are parallel mutual exclusion and sequential mutual exclusion.
6. *Priorities*. The classical Petri nets discussed so far have no mechanism to represent priorities. Such a modeling power can be achieved by introducing an *inhibitor arc*. The inhibitor arc connects an input place to a transition, and is pictorially represented by an arc terminated with a small circle. The presence of an inhibitor arc connecting an input place to a transition changes the transition-enabling conditions. In the presence of the inhibitor arc, a transition is regarded as enabled if each input place, connected to the transition by a normal arc (an arc terminated with an arrow), contains at least the number of tokens equal to the weight of the arc, and no tokens are present on each input place connected to the transition by the inhibitor arc. The transition firing rule is the same for normally connected places. The firing, however, does not change the marking in the inhibitor arc connected places. A Petri net with an inhibitor arc is shown in Figure 24.3(f). t_1 is enabled if p_1 contains a token, while t_2 is enabled if p_2 contains a token and p_1 has no token. This gives priority to t_1 over t_2: in a marking in which both p_1 and p_2 have a token, t_2 would not be able to fire until t_1 is fired.
7. *Resource constraint.* Petri nets are well suited to model and analyze systems that are constrained by resources. For instance, Figure 24.4 depicts the Petri net model of a queue with two servers. The transition *a* models the arrival of clients, *b* and *c* indicate the start and end of the service, whereas

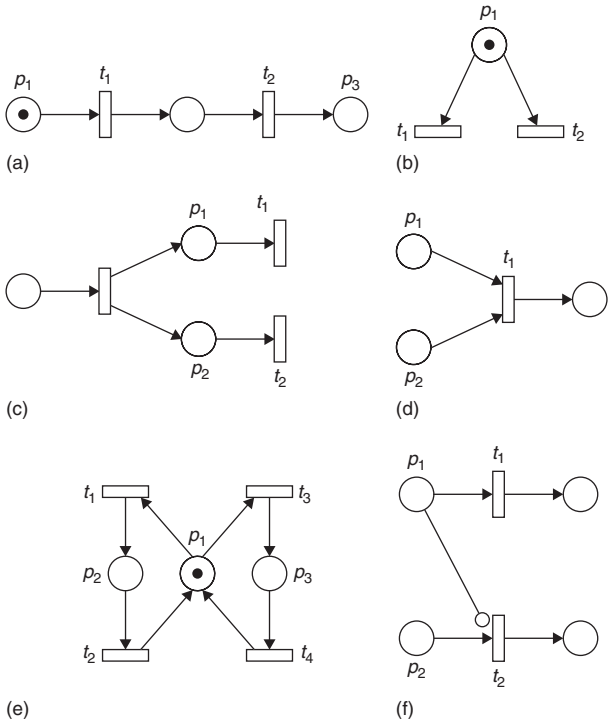

FIGURE 24.3 Petri net primitives to represent system features. (a) Sequential, (b) conflict, (c) concurrent, (d) synchronization, (e) mutual exclusive, and (f) priority.

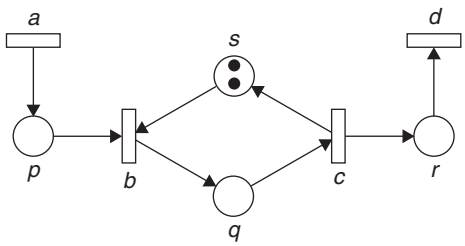

FIGURE 24.4 A queue with two servers.

d models the departure. The place *p* indicates clients that are waiting to be served, *q* models clients that are being served. A client being served occupies a resource, the *server*. Place *s* indicates free resources; initially there are two resources (servers) available. To fire *b*, a waiting client in place *p* and a free server in place *s* have to be available. When the first client arrives, a server becomes busy. When the second client arrives before the first one has finished, the second server becomes busy as well. When the third arrives before the other two have finished, he has to wait until one of the two servers becomes available. The modeling of situations like the one sketched, where behaviors are concurrent up to a certain degree, can be done very naturally by means of Petri nets.

24.5 Petri Net Properties

As a mathematical tool, Petri nets possess a number of properties. These properties, when interpreted in the context of the modeled system, allow system designer to identify the presence or absence of the

application domain-specific functional properties of the system under design. Two types of properties can be distinguished, behavioral and structural. The behavioral properties are those which depend on the initial state or marking of a Petri net. In contrast, the structural properties do not depend on the initial marking of a Petri net. They depend on the topology, or net structure, of a Petri net. Here we provide an overview of some of the most important, from the practical point of view, behavioral properties: reachability, safeness, and liveness.

24.5.1 Reachability

An important issue in designing event-driven systems is whether a system can reach a specific state, or exhibit a particular functional behavior. In general, the question is whether the system modeled with a Petri net exhibits all desirable properties as specified in the requirement specification, and no undesirable ones.

To find out whether the modeled system can reach a specific state as a result of a required functional behavior, it is necessary to find such a transition firing sequence that would transform its Petri net model from the initial marking M_0 to the desired marking M_j, where M_j represents the specific state, and the firing sequence represents the required functional behavior. In general, a marking M_j is said to be *reachable* from a marking M_i if there exists a sequence of transition firings that transforms M_i to M_j. A marking M_j is said to be *immediately reachable* from M_i if firing an enabled transition in M_i results in M_j. The set of all markings reachable from marking M is denoted by $R(M)$. We will explain how to get $R(M)$ later.

24.5.2 Safeness

In a Petri net, places are often used to represent information storage areas in communication and computer systems, product and tool storage areas in manufacturing systems, etc. It is important to be able to determine whether proposed control strategies prevent from the overflows of these storage areas. The Petri net property, which helps to identify the existence of overflows in the modeled system, is the concept of *boundedness*.

A place p is said to be *k-bounded* if the number of tokens in p is always less than or equal to k (k is a nonnegative integer number) for every marking M reachable from the initial marking M_0, i.e., $M \in R(M_0)$. It is *safe* if it is 1-bounded.

A Petri net $N = (P, T, I, O, M_0)$ is k-bounded (safe) if each place in P is k-bounded (safe). It is *unbounded* if k is infinitely large. For example, the Petri net of Figure 24.1 is 2-bounded, but the net of Figure 24.4 is unbounded.

24.5.3 Liveness

The concept of liveness is closely related to the *deadlock* situation, which has been situated extensively in the context of computer operating systems.

A Petri net modeling a deadlock-free system must be *live*. This implies that for any reachable marking M, any transition in the net can eventually be fired by progressing through some firing sequence. This requirement, however, might be too strict to represent some real systems or scenarios that exhibit deadlock-free behavior. For instance, the initialization of a system can be modeled by a transition (or a set of transitions) that fires a finite number of times. After initialization, the system may exhibit a deadlock-free behavior, although the Petri net representing this system is no longer live as specified above. For this reason, different levels of liveness are defined. Denote by $L(M_0)$ the set of all possible firing sequences starting from M_0. A transition t in a Petri net is said to be

(1) $L0$-live (or dead) if there is no firing sequence in $L(M_0)$ in which t can fire.
(2) $L1$-live (potentially firable) if t can be fired at least once in some firing sequence in $L(M_0)$.
(3) $L2$-live if t can be fired at least k times in some firing sequence in $L(M_0)$ given any positive integer k.
(4) $L3$-live if t can be fired infinitely often in some firing sequence in $L(M_0)$.
(5) $L4$-live (or live) if t is $L1$-live (potentially firable) in every marking in $R(M_0)$.

For example, all the three transitions in the net of Figure 24.1 are $L1$-live because t_1 and t_3 can only fire once each while transition $t_2$can fire twice. However, all transitions in the net of Figure 24.4 are $L4$-live, because they are all $L1$-live in every reachable marking.

24.6 Analysis of Petri Nets

We have introduced the modeling power of Petri nets in the previous sections. However, modeling by itself is of little use. It is necessary to *analyze* the modeled system. This analysis will hopefully lead to important insights into the behavior of the modeled system.

There are four common approaches to Petri net analysis: (1) reachability analysis, (2) the matrix-equation approach, (3) invariant analysis, and (4) simulation. The first approach involves the enumeration of all reachable markings, but it suffers from the state-space explosion issue. The matrix-equations technique is powerful but in many cases it is applicable only to special subclasses of Petri nets or special situations. The invariant analysis determines sets of places or transitions with special features, as token conservation or cyclical behavior. For complex Petri net models, discrete-event simulation is an option to check the system properties.

24.6.1 Reachability Analysis

Reachability analysis is conducted through the construction of reachability tree if the net is bounded. Given a Petri net N, from its initial marking M_0, we can obtain as many "new" markings as the number of the enabled transitions. From each new marking, we can again reach more markings. Repeating the procedure over and over results in a tree representation of the markings. Nodes represent markings generated from $M_0$and its successors, and each arc represents a transition firing, which transforms one marking to another.

The above tree representation, however, will grow infinitely large if the net is unbounded. To keep the tree finite, we introduce a special symbol ω, which can be thought of as "infinity." It has the properties that for each integer n, $\omega > n$, $\omega + n = \omega$, and $\omega \geq \omega$. Generally, we do not know if a Petri net is bounded or not before we perform the reachability analysis. However, we can construct a *coverability tree* if the net is unbounded or a *reachability tree* if the net is bounded according to the following general algorithm:

1. *Label the initial marking M_0 as the root and tag it "new."*
2. *For every new marking M:*
   - 2.1 *If M is identical to a marking already appeared in the tree, then tag M "old" and go to another new marking.*
   - 2.2 *If no transitions are enabled at M, tag M "dead-end" and go to another new marking.*
   - 2.3 *While there exist enabled transitions at M, do the following for each enabled transition t at M:*
     - 2.3.1 *Obtain the marking M' that results from firing t at M.*
     - 2.3.2 *On the path from the root to M if there exists a marking M'' such that $M'(p) \geq M''(p)$ for each place p and $M' \neq M''$, i.e., M'' is coverable, then replace $M'(p)$ by ω for each p such that $M'(p) > M''(p)$.*
     - 2.3.3 *Introduce M' as a node, draw an arc with label t from M to M', and tag M' "new."*

If ω appears in a marking, then the net is unbounded and the tree is a coverability tree; otherwise, the net is bounded and the tree is a reachability tree. Merging the same nodes in a coverability tree (reachability tree) results in a coverability graph (*reachability graph*).

Example 3

Reachability analysis.

Consider the Petri net shown in Figure 24.1. All reachable markings are $M_0 = (2, 0, 0, 0)$, $M_1 = (0, 2, 1, 0)$, $M_2 = (0, 1, 1, 1)$, $M_3 = (0, 2, 0, 1)$, $M_4 = (0, 0, 1, 2)$, $M_5 = (0, 1, 0, 2)$, and $M_6 = (0, 0, 0, 3)$. The reachability tree of this Petri net is shown in Figure 24.5(a), and the reachability graph is shown in Figure 24.5(b).

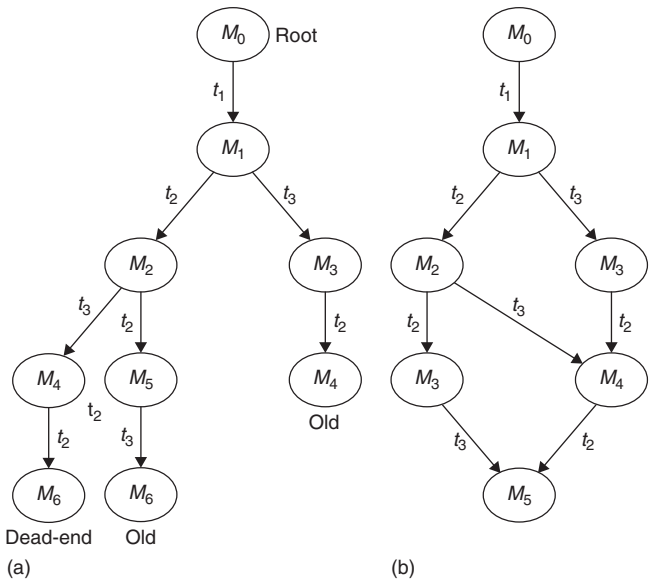

FIGURE 24.5 (a) Reachability tree. (b) Reachability graph.

24.6.2 Incidence Matrix and State Equation

For a Petri net with n transitions and m places, the incidence matrix $A = [a_{ij}]$ is an $n \times m$ matrix of integers and its entry is given by

$$a_{ij} = O(t_i, p_j) - I(t_i, p_j)$$

In writing matrix equations, we write a marking M_k as an $m \times 1$ column vector. The jth entry of M_k denotes the number of tokens in place j immediately after the kth firing in some firing sequence. The kth firing or *control vector* u_k is an $n \times 1$ column vector of $n-1$ zeroes and one nonzero entry, a 1 in the ith position indicating that transition i fires at the kth firing. Since the ith row of the incidence matrix A denotes the change of the marking as the result of firing transition i, we can write the following state equation for a Petri net:

$$M_k = M_{k-1} + A^T u_k, \quad k = 1, 2, \ldots$$

Suppose that a destination marking M_d is reachable from M_0 through a firing sequence $\{u_1, u_2 \ldots, u_d\}$. Writing the state equation for $k = 1, 2, \ldots, d$ and summing them, we obtain

$$M_d = M_0 + A^T \sum_{k=1}^{d} u_k$$

This state equation specifies a necessary condition for marking M_d being reachable from M_0, which is that there is a nonnegative solution of the firing vector $\sum_{k=1}^{d} u_k$.

24.6.3 Invariant Analysis

In a Petri net, arcs describe the relationships among places and transitions. They are represented by two matrices I and O. By examining the linear equations based on the execution rule and the matrices, one can find subsets of places over which the sum of the tokens remains unchanged. One may also find that a transition firing sequence brings the marking back to the same one. The concepts of *S-invariant* and *T-invariant* are introduced to reflect these properties.

Mathematically, an S-invariant is an integer solution y of the homogeneous equation

$$Ay = 0$$

and a T-invariant is an integer solution x of the homogeneous equation

$$A^T x = 0$$

The nonzero entries in an S-invariant represent weights associated with the corresponding places so that the weighted sum of tokens on these places is constant for all markings reachable from an initial marking. These places are said to be covered by an S-invariant. The nonzero entries in a T-invariant represent the firing counts of the corresponding transitions, which belongs to a firing sequence transforming a marking M_0 back to M_0. Although a T-invariant states the transitions comprising the firing sequence transforming a marking M_0 back to M_0, and the number of times these transitions appear in this sequence, it does not specify the order of the transition firings.

Invariant findings may help in the analysis of some Petri net properties. For example, if each place in a net is covered by an S-invariant, then it is bounded. However, this approach is of limited use since invariant analysis does not include all the information of a general Petri net.

The set of places (transitions) corresponding to nonzero entries in an S-invariant $y \geq 0$ (T-invariant $x \geq 0$) is called the *support of an invariant* and is denoted by $\|x\|$ ($\|y\|$). A support is said to be *minimal* if there is no other invariant y_1 such that $y_1(p) \leq y(p)$ for all p. Given a minimal support of an invariant, there is a unique minimal invariant corresponding to the minimal support. We call such an invariant a *minimal-support invariant*. The set of all possible minimal-support invariants can serve as a generator of invariants. That is, any invariant can be written as a linear combination of minimal-support invariants (Memmi and Roucairol, 1980).

Example 4

Figure 24.6 shows a simple manufacturing system with a single machine and a buffer. The capacity of the buffer is 1. A raw part can enter the buffer only when it is empty, otherwise it is rejected. As soon as the part residing in the buffer gets processed, the buffer is released and can accept another coming part. Fault may occur in the machine when it is processing a part. After being repaired, the machine continues to process the uncompleted part. The places and transitions in this Petri net are as follows:

p_1: The buffer available.
p_2: A part in the buffer.
p_3: The machine available.
p_4: The machine processing a part.
p_5: The machine failed.
t_1: A part arrives.
t_2: The machine starts processing a part.
t_3: The machine ends processing a part.

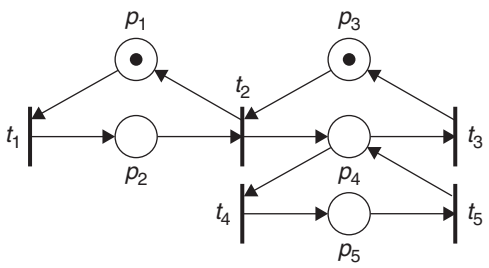

FIGURE 24.6 A simple manufacturing system with a single machine and a buffer.

t_4: The machine fails.
t_5: Repair the machine.

We are going to find the S-invariants of the net and use them to see how this simple manufacturing system works. First, we obtain the incidence matrix directly from the model:

$$A = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

Then, by solving $Ay = 0$ we get two minimal-support S-invariants, $y_1 = (1\,1\,0\,0\,0)^T$ and $y_2 = (0\,0\,1\,1\,1)^T$, where $\|y_1\| = \{p_1, p_2\}$ and $\|y_2\| = \{p_3, p_4, p_5\}$ are corresponding minimal supports. Since the initial marking is $M_0 = (1\,0\,1\,0\,0)^T$, then $M_0^T y_1 = 1$ and this leads to the fact that

$$M(p_1) + M(p_2) = 1$$

It shows that the buffer is either free or busy. Similarly, it results from $M_0^T y_2 = 1$ that

$$M(p_3) + M(p_4) + M(p_5) = 1$$

It shows how the machine spends its time. It is either up and waiting, or up and working, or down.

24.6.4 Simulation

For complex Petri net models, simulation is another way to check the system properties. The idea is simple, that is, using the execution algorithm to run the net. Simulation is an expensive and time-consuming technique. It can show the presence of undesirable properties but cannot prove the correctness of the model in general case. Despite this, Petri net simulation is indeed a convenient and straightforward yet effective approach for engineers to validate the desired properties of a discrete-event system. The algorithm is given as follows:

(1) Initialization: decide the initial marking and the set of all enabled transitions in the marking.
(2) If the number of preset simulation steps or certain stopping criteria is met, stop. Otherwise, if there is no transition enabled, report a deadlock marking and either stop or go to Step 1.
(3) Randomly pick a transition to fire. Remove the same number of tokens from each of its input places as the number of arcs from that place to the transition and deposit the same number of tokens to each of its output places as the number of arcs from the transition to that place.
(4) Remove all disabled transitions from the enabled transition set, and add all newly enabled ones to the enabled transition set. Go to Step 2.

The above algorithm can be modified to simulate extended Petri nets such as timed ones. The advantage of the simulation methods is to allow one to derive the temporal performance for a system under very realistic assumptions. A list of Petri net simulation tools along with feature descriptions can be found in the following *Petri Nets World* website: `http://www.informatik.uni-hamburg.de/TGI/PetriNets/`.

24.7 Colored Petri Nets

In a standard Petri net, tokens are indistinguishable. Because of this, Petri nets have the distinct disadvantage of producing very large and unstructured specifications for the systems being modeled. To tackle this issue, high-level Petri nets were developed to allow compact system representation. Colored

Petri nets (CPNs) (Jensen, 1981) and Predicate/Transition (Pr/T) nets (Genrich and Lautenbach, 1981) are among the most popular high-level Petri nets. We will introduce colored Petri nets in this section.

Introduced by Kurt Jensen in 1981, a CPN has its each token attached with a color, indicating the identity of the token. Moreover, each place and each transition has a set of colors attached. A transition can fire with respect to each of its colors. By firing a transition, tokens are removed from the input places and added to the output places in the same way as that in original Petri nets, except that a functional dependency is specified between the color of the transition firing and the colors of the involved tokens. The color attached to a token may be changed by a transition firing and it often represents a complex data-value. CPNs lead to compact net models by using of the concept of colors. This is illustrated by Example 5.

Example 5

A manufacturing system.

Consider a simple manufacturing system comprising two machines M1 and M2, which process three different types of raw parts. Each type of parts goes through one stage of operation, which can be performed on either M1 or M2. After the completion of processing of a part, the part is unloaded from the system and a fresh part of the same type is loaded into the system. Figure 24.7 shows the (uncolored) Petri net model of the system. The places and transitions in the model are as follows:

$p_1(p_2)$: Machine M1 (M2) available.
$p_3(p_4, p_5)$: A raw part of type 1 (type 2, type 3) available.
$p_6(p_7, p_8)$: M1 processing a raw part of type 1 (type 2, type 3).
$p_9(p_{10}, p_{11})$: M2 processing a raw part of type 1 (type 2, type 3).
$t_1(t_2, t_3)$: M1 begins processing a raw part of type 1 (type 2, type 3).
$t_4(t_5, t_6)$: M2 begins processing a raw part of type 1 (type 2, type 3).
$t_7(t_8, t_9)$: M1 ends processing a raw part of type 1 (type 2, type 3).
$t_{10}(t_{11}, t_{12})$: M2 ends processing a raw part of type 1 (type 2, type 3).

Now let us take a look at the CPN model of this manufacturing system, which is shown in Figure 24.8. As we can see, there are only 3 places and 2 transitions in the CPN model, compared at 11 places and 12 transitions in Figure 24.7. In this CPN model, p_1 means machines are available (corresponding to places p_1 and p_2 in Figure 24.7), p_2 means parts available (corresponding to places p_3-p_5 in Figure 24.7), p_3 means processing in progress (corresponding to places p_6-p_{11} in Figure 24.7), t_1 means processing starts (corresponding to transitions t_1-t_6 in Figure 24.7), and t_2 means processing ends

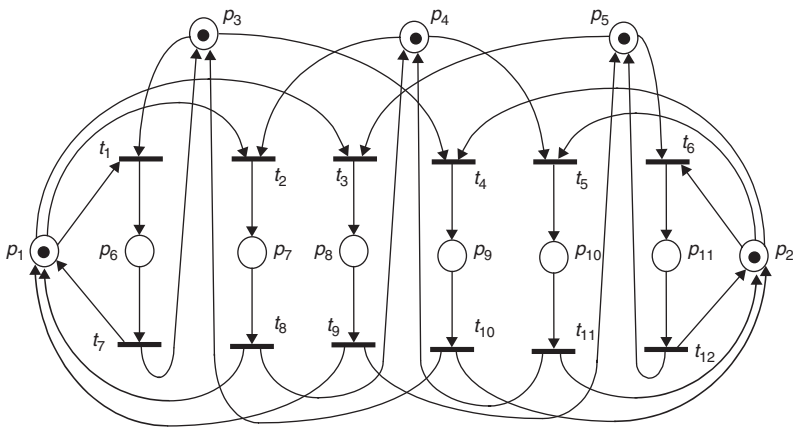

FIGURE 24.7 Petri net model of a simple manufacturing system.

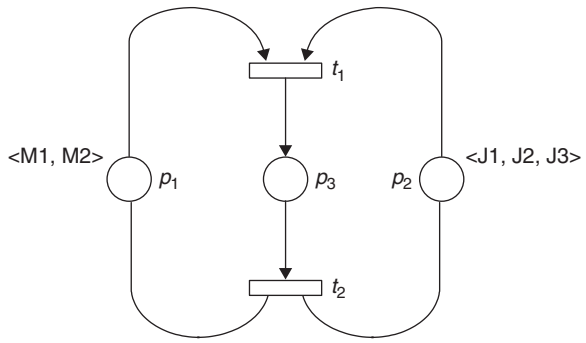

FIGURE 24.8 Colored Petri net model of the manufacturing system.

(corresponding to transitions t_7-t_{12} in Figure 24.7). There are three color sets: *SM*, *SP*, and $SM \times SP$, where $SM = \{M1, M2\}$, $SP = \{J1, J2, J3\}$. The color of each node is as follows:

$C(p_1) = \{M1, M2\}$
$C(p_2) = \{J1, J2, J3\}$
$C(p_3) = SM \times SP$
$C(t_1) = C(t_2) = SM \times SP$

CPN models can be analyzed through reachability analysis. As for ordinary Petri nets, the basic idea behind reachability analysis is to construct a reachability graph. Obviously, such a graph may become very large, even for small CPNs. However, it can be constructed and analyzed totally automatically, and there exist techniques that make it possible to work with condensed occurrence graphs without losing analytic power. These techniques build upon equivalence classes. Another option to the CPN model analysis is simulation. Readers are referred to Jensen (1997) for a detailed description of the concepts, analysis methods, and practical use of colored Petri nets.

24.8 Timed Petri Nets

The need for including timing variables in the models of various types of dynamic systems is apparent since these systems are real time in nature. In the real world, almost every event is time related. When a Petri net contains a time variable, it becomes a *timed Petri net* (Wang, 1998). The definition of a timed Petri net consists of three specifications:

- the topological structure,
- the labeling of the structure, and
- firing rules.

The topological structure of a timed Petri net generally takes the form that is used in a conventional Petri net. The labeling of a timed Petri net consists of assigning numerical values to one or more of the following things:

- transitions,
- places, and
- arcs connecting the places and transitions.

The firing rules are defined differently depending on the way the Petri net is labeled with time variables. The firing rules defined for a timed Petri net control the process of moving the tokens around.

The above variations lead to several different types of timed Petri nets. Among them, deterministic timed Petri nets (DTPNs) (Ramchandani, 1974) and stochastic timed Petri nets (STPNs) (Molloy, 1982;

Bause and Kritzinger, 2002), in which time variables are associated with transitions, are the two most widely used extended Petri nets.

24.8.1 Deterministic Timed Petri Nets

The introduction of deterministic time labels into Petri nets was first attempted by Ramchandani (1974). In his approach, the time labels were placed at each transition, denoting the fact that transitions are often used to represent actions, and actions take time to complete. The obtained extended Petri nets are called *deterministic timed Petri nets* (DTPNs). (Ramamoorthy and Ho, 1980) used such an extended model to analyze system performance. The method is applicable to a restricted class of systems called *decision-free nets.* This class of nets involves neither decisions nor nondeterminism. In structural terms, each place is connected to the input of no more than one transition, and to the output of no more than one transition.

A DTPN is a six-tuple (P, T, I, O, M_0, τ), where (P, T, I, O, M_0) is a Petri net and $\tau\colon T \to R^+$ a function that associates transitions with deterministic time delays.

A transition t_i in a DTPN can fire at time τ if and only if

(1) for any input place p of this transition, there have been the number of tokens equal to the weight of the directed arc connecting p to t_i in the input place continuously for the time interval $[\tau - \tau_i, \tau]$, where τ_i is the associated firing time of transition t_i;
(2) after the transition fires, each of its output places, p, will receive the number of tokens equal to the weight of the directed arc connecting t_i to p at time τ.

An important application of DTPN is to calculate the cycle time of a Petri net model. For a decision-free Petri net where every place has exactly one input arc and one output arc, the minimum cycle time (maximum performance) C is given by

$$C = \max\left\{\frac{T_k}{N_k} : k = 1, 2, \Lambda, q\right\}$$

where $T_k = \sum_{t_i \in L_k} \tau_i$ is the sum of the execution times of the transitions in circuit k; $N_k = \sum_{p_i \in L_k} M(p_i)$ the total number of tokens in the places in circuit k; and q the number of circuits in the net.

Example 6

A communication protocol.

Consider the communication protocol between two processes, one indicated as the sender and the other as the receiver. The sender sends messages to a buffer, while the receiver picks up messages from the buffer. When it gets a message, the receiver sends an acknowledgment (ACK) back to the sender. After receiving the ACK from the receiver, the sender begins processing and sending a new message. Suppose that the sender takes 1 time unit to send a message to the buffer, 1 time unit to receive the ACK, and 3 time units to process a new message. Then, the receiver takes 1 time unit to get the messages from the buffer, 1 time unit to send back an ACK to the buffer, and 4 time units to process a received message. The DTPN model of this protocol is shown in Figure 24.9. The legends of places and transitions and timing properties are as follows:

p_1: The sender ready.
p_2: Message in the buffer.
p_3: The sender waiting for ACK.
p_4: Message received.
p_5: The receiver ready.
p_6: ACK sent.
p_7: ACK in the buffer.
p_8: ACK received.

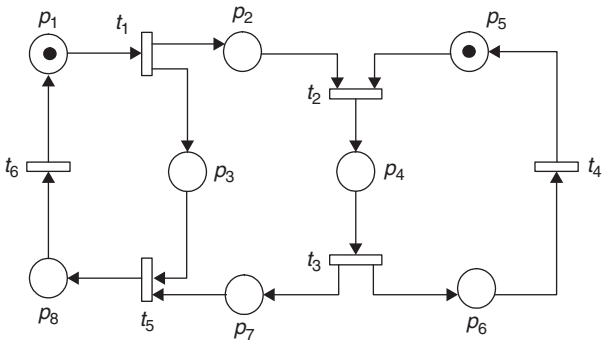

FIGURE 24.9 Petri net model of a simple communication protocol.

t_1: The sender sends a message to the buffer. Time delay: 1 time unit.
t_2: The receiver gets the messages from the buffer. Time delay: 1 time unit.
t_3: The receiver sends back an ACK to the buffer. Time delay: 1 time unit.
t_4: The receiver processes the message. Time delay: 4 time units.
t_5: The sender receives the ACK. Time delay: 1 time unit.
t_6: The sender processes a new message. Time delay: 3 time units.

There are three circuits in the model. The cycle time of each circuit is calculated as follows:

$$\text{circuit } p_1t_1p_3t_5p_8t_6p_1 : C_1 = \frac{T_1}{N_1} = \frac{1+1+3}{1} = 5$$

$$\text{circuit } p_1t_1p_2t_2p_4t_3p_7t_5p_8t_6p_1 : C_2 = \frac{T_2}{N_2} = \frac{1+1+1+1+3}{1} = 7$$

$$\text{circuit } p_5t_2p_4t_3p_6t_4p_5 : C_3 = \frac{T_3}{N_3} = \frac{1+1+4}{1} = 6$$

After enumerating all circuits in the net, we know the minimum cycle time of the protocol between the two processes is 7 time units.

24.8.2 Stochastic Timed Petri Nets

STPNs are Petri nets in which stochastic firing times are associated with transitions. An STPN is essentially a high-level model that generates a stochastic process. STPN-based performance evaluation basically comprises modeling the given system by an STPN and automatically generating the stochastic process that governs the system behavior. This stochastic process is then analyzed using known techniques. STPNs are a graphical model and offer great convenience to a modeler in arriving at a credible, high-level model of a system.

The simplest choice for the individual distributions of transition firing times is negative exponential distribution. Because of the memoryless property of this distribution, the stochastic process associated with the STPN is a continuous-time homogeneous Markov chain (Ethier, 2005) with state space in one-to-one correspondences with marking in $R(M_0)$, the set of all reachable markings. The transition rate matrix of the Markov chain can be easily constructed from the reachability graph given the firing rates of the transitions of the STPN. Exponential timed stochastic Petri nets, often called *stochastic Petri nets* (SPNs), were independently proposed by Natkin (1980) and Molloy (1981), and their capabilities in the performance analysis of real systems have been investigated by many authors.

An SPN is a six-tuple $(P, T, I, O, M_0, \Lambda)$, where (P, T, I, O, M_0) is a Petri net and $\Lambda : T \to R$ a set of firing rates whose entry λ_k is the rate of the exponential individual firing time distribution associated with transition t_k. Natkin and Molloy have shown that the marking process of an SPN is a continuous-time Markov chain. The state space of the Markov chain is the reachable set $R(M_0)$. Suppose there are s markings in $R(M_0)$, and the underlying Markov chain is ergodic, then the steady-state probability distribution $\Pi = (\pi_0, \pi_1, \ldots, \pi_s)$ can be obtained by resolving the following linear system:

$$\Pi Q = 0 \qquad \sum_j \pi_j = 1, \quad \pi_j \geq 0, \quad j = 0, 1, 2, \ldots$$

where Q is a transition rate matrix whose elements outside the main diagonal are the rates of the exponential distributions associated with the transitions from state, while the elements on the main diagonal make the sum of the elements of each row equal to zero. Denote by $E(M_i)$ the set of all enabled transition at marking M_i, and T_{ij} the set of enabled transitions at marking M_i whose firings lead the SPN to another marking M_j. Then Q is determined as follows:

$$q_{ij} = \sum_{t_k \in T_{ij}} \lambda_k \qquad q_i = q_{ii} = -\sum_{t_k \in E(M_i)} \lambda_k$$

The probability of marking M_i changing to M_j is the same as the probability that one of the transitions in the set T_{ij} fires before any of the transitions in the set $T \backslash T_{ij}$. Since the firing times in an SPN are mutually independent exponential random variables, it follows that the required probability has the specific value given by

$$\alpha_{ij} = q_{ij}/q_i$$

In the expression for α_{ij} deduced above, note that the numerator is the sum of the rates of those enabled transitions in M_i, the firing of any of which changes the marking from M_i to M_j; whereas the denominator is the sum of the rates of all the enabled transitions in M_i. Also note that $\alpha_{ij} = 1$ if and only if $T_{ij} = E(M_i)$.

Example 7

A stochastic Petri net.

Figure 24.10 shows a simple SPN model with its reachable markings and its reachable graph. The linear system of steady-state probabilities is

$$\pi_0 + \pi_1 + \pi_2 + \pi_3 + \pi_4 = 1$$

Let $\Lambda = (1\ 1\ 1\ 1)$, then solution to this system is

$$\pi_0 = \pi_4 = 2/7, \qquad \pi_1 = \pi_2 = \pi_3 = 1/7$$

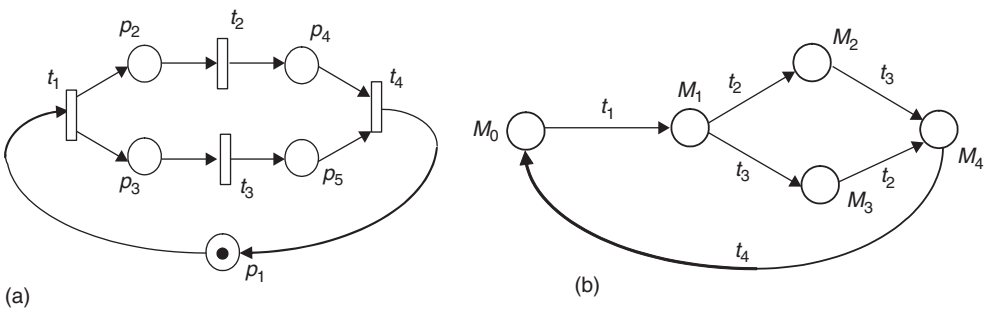

FIGURE 24.10 (a) SPN model. (b) Reachability graph.

The analysis of an SPN model is usually aimed at the computation of more aggregate performance indices than the probabilities of individual markings. Several kinds of aggregate results are easily obtained from the steady-state distribution over reachable markings. In this section, we quote some of the most commonly and easily computed aggregate steady-state performance parameters (Ajmone Marsan, 1990).

- The probability of an event defined through place markings (e.g., no token in a subset of places, or at least one token in a place while another one is empty), can be computed by adding the probabilities of all markings in which the condition corresponding to the event definition holds. Thus, for example, the steady-state probability of the event A defined through a condition that holds for the markings $M_i \in H$ is obtained as

$$P\{A\} = \sum_{M_i \in H} \pi_i$$

- The average number of tokens in a place can be obtained by computing the individual probabilities as those of the event "place p_i contains k tokens."
- The frequency of firing a transition, i.e., the average number of times the transition fires in unit time, can be computed as the weighted sum of the transition firing rate:

$$f_i = \sum_{i:t_j \in E(M_i)} \lambda_j(M_i)\pi_i$$

  where f_j is the frequency of firing t_j and $\lambda_j(M_i)$ the firing rate of t_j at M_i.
- The average delay of a token in traversing a subnet in steady-state conditions can be computed using Little's formula

$$E(T) = \frac{E(N)}{E(\gamma)}$$

  where $E(T)$ is the average delay, $E(N)$ the average number of tokens in the process of traversing the subnet, and $E(\gamma)$ the average input (or output) rate of tokens into (or out of) the subnet. This procedure can be applied whenever the interesting tokens can be identified inside the subnet (which can also comprise other tokens defining its internal condition, but these must be distinguishable from those whose delay is studied) so that their average number can be computed, and a relation can be established between input and output tokens (e.g., one output token for each input token).

24.9 Concluding Remark

Petri nets have been proven to be a powerful modeling tool for various types of dynamic event-driven systems. Since Petri nets were introduced in 1962, numerous research papers have been published. The research has been conducted in many branches, with each branch exploring a promising application aspect of this formalism. Given the rich research results from the Petri net society, it is hard to cover all of them in such a short book chapter. Therefore, this chapter only aims at briefly introducing the most basic concepts, theory and applications of ordinary Petri nets, and a few of the most popular extended Petri nets.

References

Ajmone Marsan, M. 1990. Stochastic Petri nets: An elementary introduction. *Advances in Petri Nets, Lecture Notes in Computer Science* 424.

Ajmone Marsan, M., M.G. Balbo, and G. Conte. 1986. *Performance Models of Multiprocessor Systems*. Cambridge, MA: MIT Press.

Andreadakis, S.K. and A.H. Levis. 1988. Synthesis of distributed command and control for the outer air battle. *Proceedings of the 1988 Symposium on C^2 Research*, SAIC, McLean, VA.

Bause, F. and P. Kritzinger. 2002. *Stochastic Petri Nets—An Introduction to the Theory*. Germany: Vieweg Verlag.

Desrochers, A. and R. Ai-Jaar. 1995. *Applications of Petri Nets in Manufacturing Systems: Modeling, Control and Performance Analysis*. IEEE Press. AQ3

Ethier, S. N. 2005. *Markov Processes*. New York: Wiley.

Genrich, J.H. and K. Lautenbach. 1981. System modeling with high-level Petri nets. *Theoretical Computer Science* 13: 109–136.

Haas, P. 2002. *Stochastic Petri Nets: Modelling, Stability, Simulation*. New York: Springer. AQ4

Jensen, K. 1981. Colored Petri nets and the invariant-method. *Theoretical Computer Science* 14: 317–336.

Jensen, K. 1997. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use* (3 volumes). London: Springer.

Lin, C., L. Tian, and Y. Wei. 2002. Performance equivalent analysis of workflow systems. *Journal of Software* 13(8): 1472–1480.

Lindemann, C. 1998. *Performance Modelling with Deterministic and Stochastic Petri Nets*. New York: Wiley. AQ4

Mandrioli, D., A. Morzenti, M. Pezze, P. San Pietro, and S. Silva. 1996. A Petri net and logic approach to the specification and verification of real time systems. In: *Formal Methods for Real Time Computing* (C. Heitmeyer and D. Mandrioli, eds.). New York: Wiley. AQ4

Memmi, G. and G. Roucairol. 1980. Linear algebra in net theory. *Lecture Notes in Computer Science*, 84: 213–223.

Merlin, P. and D. Farber. 1976. Recoverability of communication protocols—Implication of a theoretical study. *IEEE Transactions on Communications* 1036–1043.

Molloy, M. 1981. *On the Integration of Delay and Throughput Measures in Distributed Processing Models*. Ph.D. Thesis, UCLA, Los Angeles, CA.

Molloy, M. 1982. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers* 31(9): 913–917.

Murata, T. 1989. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE* 77(4): 541–580. AQ4

Natkin, S. 1980. *Les Reseaux de Petri Stochastiques et Leur Application a l'evaluation des Systems Informatiques*. These de Docteur Ingegneur, Cnam, Paris, France.

Peterson, J.L. 1981. *Petri Net Theory and the Modeling of Systems*. New Jersey: Prentice-Hall.

Petri, C.A. 1962. *Kommunikation mit Automaten*. English Translation, 1966: *Communication with Automata*, Technical Report RADC-TR-65-377, Rome Air Dev. Center, New York.

Ramchandani, C. 1974. *Analysis of Asynchronous Concurrent Systems by Petri Nets*. Project MAC, TR-120, MIT, Cambridge, MA.

Ramamoorthy, C. and G. Ho. 1980. Performance evaluation of asynchronous concurrent systems using Petri nets. *IEEE Transaction on Software Engineering* 6(5): 440–449.

Tsai, J., S. Yang, and Y. Chang. 1995. Timing constraint Petri nets and their application to schedulability analysis of real-time system specifications. *IEEE Transactions on Software Engineering* 21(1): 32–49.

van der Aalst, W. and K. van Hee. 2000. *Workflow Management: Models, Methods, and Systems*. Cambridge, MA: MIT Press.

van Landeghem, R. and C.-V. Bobeanu. 2002. Formal modeling of supply chain: An incremental approach using Petri nets. *14th European Simulations Symposium and Exhibition*, Dresden, Germany.

Wang, J. 1998. *Timed Petri Nets, Theory and Application.* Boston, MA: Kluwer.

Wang, J. 2006. Charging information collection modeling and analysis of GPRS networks. *IEEE Transactions on Systems, Man and Cybernetics*, *Part C* 36(6). AQ5

Venkatesh, K., M.C. Zhou, and R. Caudill. 1994. Comparing ladder logic diagrams and Petri nets for sequence controller design through a discrete manufacturing system. *IEEE Transaction on Industrial Electronics* 41(6): 611–619.

Zhou, M.C. and F. DiCesare. 1989. Adaptive design of Petri net controllers for error recovery in automated manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics* 19(5): 963–973.

QUERY FORM

CRC Press

Handbook of Dynamic System Modeling

JOURNAL TITLE:	**C5653**
ARTICLE NO:	**Ch024**

Queries and/or remarks

Query No.	Details Required	Author's Response
AQ1	Mandrioli and Morzenti (1996) is not listed in the reference list.	
AQ2	The sentence "A circle contains ... (Peterson, 1981)" is not clear. Please clarify.	
AQ3	Please provide the location of the publisher for Desrochers and Ai-Jaar (1995).	
AQ4	Haas (2002), Lindemann (1998), Mandrioli et al. (1996) and Murata (1989) are not cited in the text.	
AQ5	Please provide the page range for Wang (2006).	