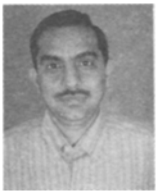
Petri Nets

1. Overview and Foundations

Y Narahari



Y Narahari is currently an Associate Professor of Computer Science and Automation at the Indian Institute of Science, Bangalore. His research interests are broadly in the areas of stochastic modeling and scheduling methodologies for future factories; and object oriented modeling.

Petri nets offer a versatile modeling framework for complex, distributed, concurrent systems and have been used in a wide range of modeling applications. The first part of this two-part article provides an overview of Petri nets and presents important conceptual underpinnings of Petri net modeling.

Introduction

Modeling is a central part of all activities that lead up to the design, implementation, and deployment of systems. We build models so that we can understand better the system we are developing. Models enable to communicate the desired structure and behavior of our system and provide a basis for designing high-performance systems.

Petri nets constitute a versatile modeling tool applicable to many systems. They are graphical in nature and are backed up by a sound mathematical theory. They can be used by practitioners and theoreticians alike and their applications range over a wide variety of disciplines. They have been primarily used to describe and study *discrete event dynamical systems*. A discrete event dynamical system is a system in which the evolution of the activities in time depends on the occurrence of *discrete* events. Examples of such systems include computer systems, automated manufacturing systems, communication networks, air traffic networks, power systems, office automation systems, business processes, etc. In a computer system, for example, typical discrete events include: arrival of a new job into the system; finishing of execution of a program; commencement of an I/O operation; or a disk crash. When such an event occurs, the state of the system might change; some old events may get disabled; and some new events may get enabled. In order to capture

Box 1. History

Petri nets have an interesting history and have come quite a long way from the time Carl Adam Petri proposed them in his doctoral work in the early 1960s. His doctoral dissertation was submitted to the faculty of Mathematics and Physics at the Technical University of Darmstadt, West Germany in 1962. Petri currently works in an institution called GMD in Bonn, Germany. The primary motivation behind Petri's work was to model concurrency and asynchronism in distributed systems through a formalism more powerful than finite state automata. Petri's pathbreaking work came to the attention of A W Holt in the mid-1960s. Holt led the Information System Theory project of Applied Data Research in the United States. This project brought out a series of influential reports on Petri net theory in the mid and late 1960s. From 1970 to 1975, the Computation Structures Group at the Massachusetts Institute of Technology became a leading centre for Petri net research and from then on, Petri nets became an active research area in several universities, particularly in Europe. Starting from 1980, a series of annual conferences have been held, initially called as the European Workshops on Applications and Theory of Petri Nets and currently called as the International Workshops on Applications and Theory of Petri Nets. These workshops are meant exclusively for Petri net related papers.

Research and development in the area of Petri nets can be categorized into several streams. The research in the 1960s and 1970s was mostly on Petri net theory with less emphasis on applications. The theory focused on issues such as Petri net languages; characterization of Petri nets as a model of computation; and qualitative analysis of systems. During the 1980s, a large number of Petri net based packages were developed for use in modeling and analysis of concurrent systems. Significant applications of Petri net theory were explored in the 1980s, primarily in the areas of computer operating systems, distributed computer systems, computer networks, and automated manufacturing systems. Timed Petri nets or stochastic Petri nets became prominent in these applications and Petri nets emerged as a major tool for quantitative performance analysis of systems. 1990s have seen the emergence of user-friendly modeling and analysis tools based on Petri nets for both qualitative and quantitative analyses and even non-specialists are able to effectively use these powerful tools. However, to this day, several researchers, including Carl Adam Petri, continue to advance the theory of Petri nets in many interesting directions.

the structure and dynamics of such a system, Petri nets offer a natural and effective modeling methodology.

In Part 1 of this article, we will first briefly trace the history of Petri net modeling *Box* 1. We will then understand the notation and meaning of Petri net models, using an illustrative example (that of a simple manufacturing plant with two machine centres). Following this, we will understand how Petri net models can capture the behavior or dynamics of a modeled system. Next, we will look into the representational power of Petri nets while modeling systems.

In Part 2, we will first illustrate Petri net modeling through a representative example, that of the *dining philosophers* problem. We will then understand how important system properties are captured by the dynamics of a Petri net model of the system. Following this, we investigate different ways in which Petri nets models can be used in system modeling.

Classical Petri Nets

Petri nets are bipartite graphs and provide an elegant and mathematically rigorous modeling framework for discrete event dynamical systems. In this section an overview of Petri nets is presented with the aid of several definitions and an illustrative example. In the following, N and R denote respectively the set of non-negative integers and the set of real numbers.

We start with some elementary definitions in classical Petri nets and illustrate the definitions with some examples.

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

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

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

$$P \cup T \neq \phi, \ \ P \cap T \neq \phi$$

$IN : (PXT) \rightarrow N$ is an *input function* that defines directed arcs from places to transitions, and

$OUT : (PXT) \rightarrow N$ is an *output function* that defines directed arcs from transitions to places.

Pictorially, places are represented by circles and transitions by horizontal or vertical bars. If $IN(p_i, t_j) = k$, where $k > 1$ is an integer, a directed arc from place p_i to transition t_j is drawn with label k. If $IN(p_i, t_j) = 1$, we include an unlabeled directed arc. If $IN(p_i, t_j) = 0$, no arc is drawn from p_i to t_j. Some places have black dots or tokens within them. The significance of the tokens will be introduced soon.

Places of Petri nets usually represent conditions or resources in the system while transitions model the activities in the system. In all subsequent definitions, we assume a Petri net (P, T, IN, OUT) as given in the above definition. Also, we assume that the index i takes on the values $1, 2, ...n$, while the index j takes on the values $1, 2, ...m$.

Example 1. Let us consider a simple manufacturing system comprising two machines M_1 and M_2 and processing two different types of parts. Each part type goes through one stage of operation, which can be performed on either M_1 or M_2. On completion of processing, the part is unloaded from the system and a fresh part of the same type is loaded into the system. *Figure* 1 depicts a Petri nets model of this system and *Table* 1 gives the interpretation of the places and transitions in the model. For this Petri net,

$$P = \{p_1, p_2, ...p_8\}; \;\; T = \{t_1, t_2, ...t_8\}$$

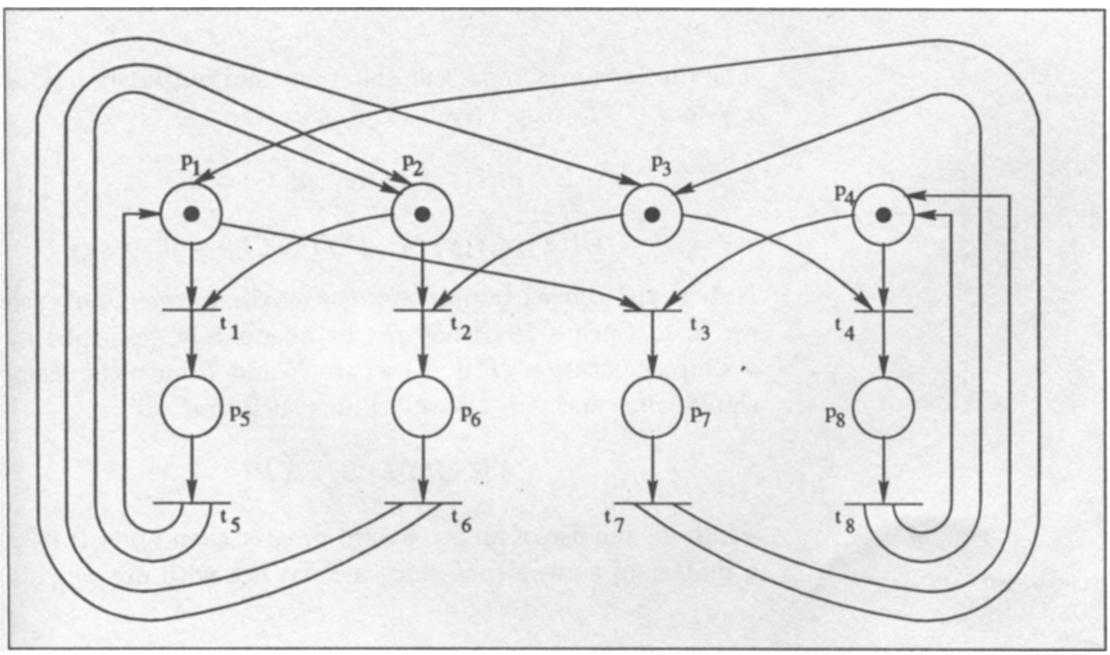


Figure 1. Petri net model of a simple manufacturing system.

Table 1. Places and transitions in the Petri net model of a manufacturing plant.

Places

p_1 : Raw parts of type 1
p_2 : Machine M_1 available
p_3 : Raw parts of type 2
p_4 : Machine M_2 available
p_5 : M_1 processing a part of type 1
p_6 : M_1 processing a part of type 2
p_7 : M_2 processing a part of type 1
p_8 : M_2 processing a part of type 2

Transitions

t_1 : M_1 starts processing a part of type 1
t_2 : M_1 starts processing a part of type 2
t_3 : M_2 starts processing a part of type 1
t_4 : M_2 starts processing a part of type 2
t_5 : M_1 finishes processing a part of type 1
t_6 : M_1 finishes processing a part of type 2
t_7 : M_2 finishes processing a part of type 1
t_8 : M_2 finishes processing a part of type 2

The directed arcs represent the input and output functions IN and OUT, respectively. For example,

$$IN(p_1, t_1) = 1; \;\; IN(p_6, t_2) = 0.$$

$$OUT(p_5, t_1) = 1; \;\; OUT(p_6, t_6) = 0.$$

Note in the above example that the maximum weight of each arc is 1. Such a Petri net can be adequately described by a simpler notation (P, T, A) where P and T have the usual significance and A is the set of arcs such that

$$A \subseteq (PXT) \cup (TXP)$$

Indeed, the use of an arc weight greater than unity is only a matter of convenience since a Petri net with arc weights greater than unity can always be represented by another Petri net having maximum arc weight unity.

Example 2. For the Petri nets of example 1, the set A is given by

$$\{(p_1,t_1),(p_2,t_1),(p_2,t_2),(p_3,t_2),(p_1,t_3),(p_4,t_3),(p_3,t_4),(p_4,t_4),$$
$$(p_5,t_5),(p_6,t_6),(p_7,t_7),(p_8,t_8),(t_1,p_5),(t_2,p_6),(t_3,p_7),(t_4,p_8),$$
$$(t_5,p_1),(t_5,p_2),(t_6,p_2),(t_6,p_3),(t_7,p_1),(t_7,p_4),(t_7,p_3),(t_7,p_4)\}.$$

Definition: Given a transition t_j, the set of *input places* of t_j, denoted by $IP(t_j)$ and the set of *output places* of t_j, denoted by $OP(t_j)$, are defined by

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

Definition: Given a place p_i, the set $IT(p_i)$ of *input transitions* of p_i and the set $OT(p_i)$ of *output transitions* of p_i are defined by

$$IT(p_i) = \{t_j \in P : OUT(p_i,t_j) \neq 0\}$$
$$OT(p_i) = \{t_j \in P : IN(p_i,t_j) \neq 0\}$$

Example 3. For the Petri net of *Figure* 1, we have

$$IP(t_1) = OP(t_5) = \{p_1,p_2\};\ \ IP(t_5) = OP(t_1) = \{p_5\}$$

The other sets of input places and output places can be obtained similarly. Also,

$$IT(p_1) = \{t_5,t_7\};\ \ OT(p_1) = \{t_1,t_3\}$$

The other sets of input transitions and output transitions can be obtained similarly.

Definition: Let T_1 be a subset of T. The transitions of T_1 are said to be *conflicting* if

$$\bigcap_{t\in T_1} IP(t) \neq \phi$$

and *concurrent* if

$$IP(t_j) \cap IP(t_k) = \phi \, \forall t_j, t_k \in T_1.$$

Example 4. In the Petri net of *Figure* 1, the sets of transitions that are conflicting are $t_1, t_3; t_1, t_2; t_2, t_4$; and t_3, t_4. Some of the concurrent sets of transitions are $t_1, t_4; t_2, t_3; t_5, t_8$; and t_1, t_8. Petri nets capture concurrency of activities through concurrent transitions and non-deterministic activities through conflicting transitions. Further, they can also model co-existence of concurrent and non-deterministic activities. Elegant representation of such features is an important facet of Petri net modeling.

Definition: A *marking* M of a Petri net is a function $M : P \to N$. A *marked* Petri net is a Petri net with an associated marking.

A marking of a Petri net with n places is an $(n \times 1)$ vector, which associates with each place a certain number of tokens represented by black dots, and represents a state of the Petri net. We always associate an initial marking M_0 with a given Petri net model. In the rest of the article, we use the words *state* and *marking* interchangeably. Also, unless otherwise specified, a Petri net henceforth will refer to a marked Petri net.

Example 5. In *Figure* 1, the marking of the Petri nets is given by

$$M_0 = \begin{bmatrix} M_0(p_1) \\ \vdots \\ M_0(p_8) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

This corresponds to a state of the system when both machines are available for use and one fresh part of each type is waiting to be processed.

Dynamics of Petri Nets

Petri nets model both the structure and behavior of systems. Structural modeling is accomplished through the graphical structure and the input-output relationships among the places and transitions. Behavioral modeling is achieved through execution of *firing rules* (also called as the *token game*) that capture the dynamics of the modeled system over time. First we introduce the important notion of reachability set for a Petri net.

Definition: A transition t_j of a Petri net is said to be *enabled* in a marking M if

$$M(p_i) \geq IN(p_i, t_j) \quad \forall \; p_i \in IP(t_j)$$

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

$$M'(p_i) = M(p_i) + OUT(p_i, t_j) - IN(p_i, t_j) \quad \forall \; p_i \in P \quad (1)$$

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

We consider that every marking is trivially reachable from itself by firing no transition. Also, if some marking M_j is reachable from M_i and M_k is reachable from M_j, then it is easy to see that M_k is reachable from M_i. Thus reachability of markings is a *reflexive* and *transitive* relation on the set of markings.

Definition: The transitive closure of the reachability relation, which comprises all markings reachable from the initial marking M_0 by firing zero, one, or more transitions, is called the *reachability set* of a Petri net with initial marking M_0. It is denoted by $R[M_0]$.

Definition: For a marked Petri net with initial marking M_0, the *reachability graph* is a directed graph (V, E) where $V = R[M_0]$ is the set of vertices, and E, the set of directed arcs, is given by: $(M_1, M_2) \in E$ if

(i) $M_1, M_2 \in R[M_0]$ and (ii) either there exists a transition $t \in T$ such that $M_1 \xrightarrow{t} M_2$ or there exists a set, $T_1 \subseteq T$, such that T_1 is a set of concurrent transitions by firing all of which M_1 reaches M_2.

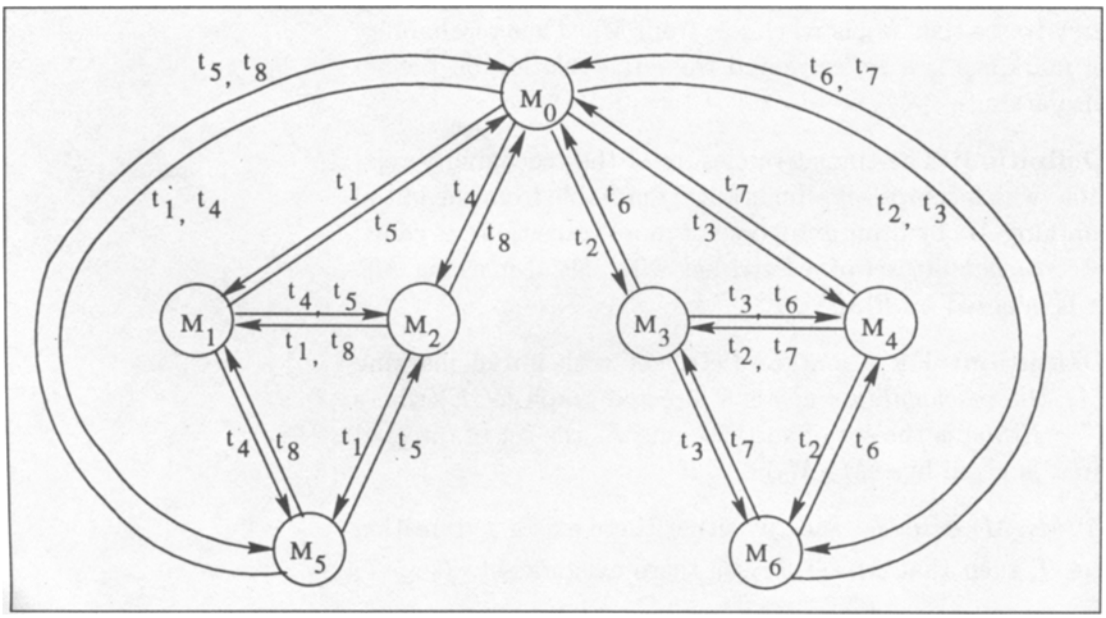
In the reachability graph, the nodes are labeled by the markings they represent and the directed arcs are labeled by the transition or the set of concurrent transitions whose firing takes the source node to the destination node.

Example 6. In the marked Petri net of *Figure* 1, the transitions t_1, t_2, t_3, and t_4 are all enabled. When t_1 fires, the new marking reached is M_1 where $M_1 = (00111000)^T$. Thus, $M_0 \xrightarrow{t_1} M_1$. Also, $M_0 \xrightarrow{t_4} M_2$, $M_0 \xrightarrow{t_2} M_3$ and $M_0 \xrightarrow{t_3} M_4$ where $M_2 = (11000001)^T$, $M_3 = (10010100)^T$, and $M_4 = (01100010)^T$. It can be shown that $R[M_0] = \{M_0, M_1, M_2, M_3, M_4, M_5, M_6\}$ where the details of the markings are given in *Table* 2. *Figure* 2 gives the reachability graph of this Petri net.

Representational Power

The typical structural and behavioral characteristics exhibited by the activities in a complex system, such as concurrency, decision making, synchronization, and priorities, can be modeled elegantly by Petri nets. We have already seen

Figure 2. Reachability graph of the marked Petri net of Figure 1.



Marking	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
M_0	1	1	1	1	0	0	0	0
M_1	0	0	1	1	1	0	0	0
M_2	1	1	0	0	0	0	0	1
M_3	1	0	0	1	0	1	0	0
M_4	0	1	1	0	0	0	1	0
M_5	0	0	0	0	1	0	0	1
M_6	0	0	0	0	0	1	1	0

Table 2. Reachable markings of the Petri net of Figure 1.

in Example 4 how concurrency and conflicts are represented. Here we identify Petri net constructs for representing characteristics of various features. *Figure* 3 depicts these constructs.

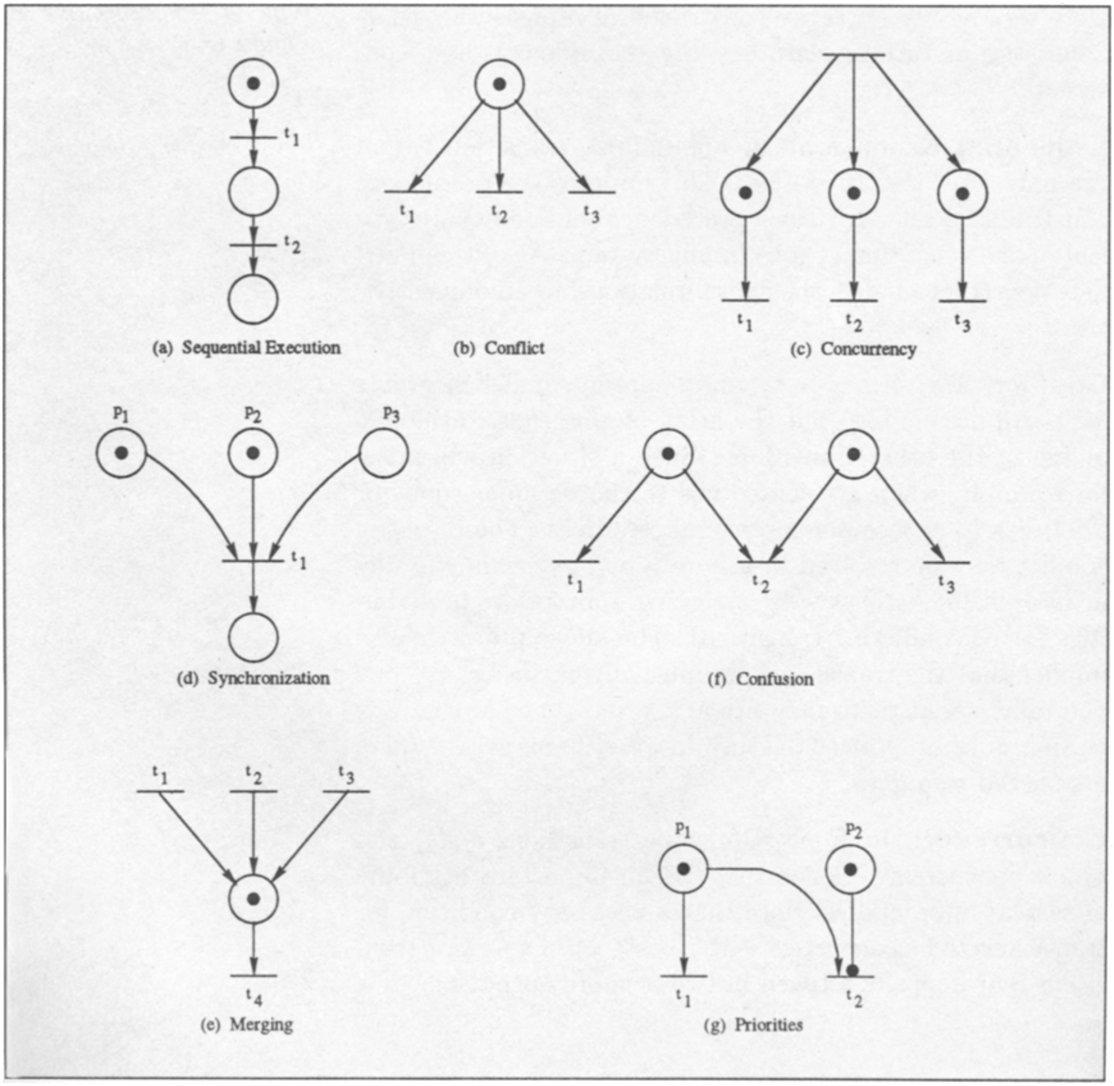
Sequential Execution: In *Figure* 3(a), transition t_2 can fire only after the firing of t_1. This imposes the precedence constraint 't_2 after t_1'. Such precedence constraints are typical of the execution of jobs in any system. Also, this Petri nets construct models the causal relationship among activities.

Conflict: Transitions t_1, t_2, and t_3 are in conflict in *Figure* 3(b). All are enabled but the firing of any leads to the disabling of the other transitions. Such a situation will arise, for example, when a resource has to choose among jobs or a job has to choose among several resources. The resulting conflict may be resolved in a purely *non-deterministic* way or in a *probabilistic* way, by assigning appropriate probabilities to the conflicting transitions. The above primitive also implies that the transitions are mutually exclusive, i.e. one and only one of them may fire at a given time. Mutual exclusion is an important feature in all systems where there are shared resources.

Concurrency: In *Figure* 3(c), the transitions t_1, t_2, and t_3 are concurrent. Concurrency is an important attribute of system interactions. Note that a necessary condition for transitions to be concurrent is the existence of a *forking* transition that deposits a token in two or more output places.

Synchronization: Often, jobs in a system wait for resources and resources wait for appropriate jobs to arrive (as in assembly operations). The resulting synchronization of activities can be captured by transitions of the type shown in *Figure* 3(d). Here, t_1 will be enabled only when each of its input places has a minimum appropriate number of tokens. In the present case, p_1 and p_2 have tokens but p_3 does not have a token. Therefore, for t_1 to be enabled, we will have to wait for a token to arrive into p_3. The arrival of a token into this place could be the result of a possibly complex se-

Figure 3. Petri net primitives to represent system features.



quence of operations elsewhere in the rest of the Petri nets model.

Merging: When parts from several streams arrive for service at the same resource, the resulting situation can be depicted as in *Figure* 3(e). Another example is the arrival of several jobs from several sources to a centralized location.

Confusion: Confusion is a situation where concurrency and conflicts co-exist. An example is depicted in *Figure* 3(f). Both t_1 and t_3 are concurrent while t_1 and t_2 are in conflict, and t_2 and t_3 are also in conflict.

Priorities: The classical Petri nets discussed so far have no mechanism to represent priorities. *Inhibitor nets* include special arcs called *inhibitor arcs* to model priorities. A portion of an inhibitor net is shown in *Figure* 3(g). p_2 is called an inhibitor place of t_2. An inhibitor arc from p_2 to t_2 is drawn as shown in *Figure* 3(g). The transition t_2 is enabled only if p_1 has a token and p_2 does not have a token. This enables a higher priority to be given to t_1 over t_2. In *Figure* 3(g), for example, t_1 is enabled but not t_2 because p_2 has a token. It is to be noted that the reachability set of an inhibitor net is a subset of the same net but with the inhibitor arcs removed. Inhibitor arcs enhance the modeling power of the Petri net model and indeed, it has been shown that Petri nets with inhibitor arcs are equivalent in power to the Turing machines. It has been proved that the classical Petri nets, i.e., without inhibitor arcs, are less powerful than Turing machines (see Article-in-a-box, *Resonance*, Vol. 2, No. 7, July 1997)and can only generate a proper subset of context-sensitive languages.

In this article thus far, we have looked into the history of Petri nets; important notation and semantics of Petri net models; and their modeling power. In the second part of the article, we will first present the example of the dining philosophers problem to illustrate Petri net modeling. Then we present features of Petri nets that make them an attractive and versatile modeling tool and provide an overview of their applications.

Suggested Reading

[1] J L Peterson, *Petri net theory and the modeling of systems*, Prentice Hall, Englewood Cliffs, New Jersey, 1981.

[2] N Viswanadham and Y Narahari, *Performance Modeling of Automated Manufacturing Systems*, Prentice Hall, Englewood Cliffs, New Jersey, 1992 (also available from Prentice Hall of India, New Delhi, 1998).

[3] R Johnsonbaugh and T Murata, Petri nets and marked graphs: mathematical models of concurrent computation, *American Mathematical Monthly*, Vol. 89, No. 8, October 1982, 553–566.

[5] T Murata, Petri nets: properties, analysis, and applications, *Proceedings of the IEEE*, Vol. 77, No. 4, April 1989, 541–580.

[6] W Reisig, Petri nets: An introduction, EATCS monographs in Theoretical Computer Science, Springer-Verlag, New York, 1985.

Address for Correspondence
Y Narahari
Department of Computer Science and Automation
Indian Institute of Science
Bangalore 560 012, India.
Email: hari@csa.iisc.ernet.in