Petri Nets

2. Applications

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.

Part 1 – Overview and Foundations, *Resonance*, Vol.4, No.8, p.66–77, 1999.

Petri nets offer a versatile modeling framework for complex, distributed, concurrent systems and have been used in a wide range of modeling applications. In Part 1 of this two-part article, we have seen important features and representational power of the Petri net model. We have also seen how the application of firing rules enables Petri nets to capture the dynamics or behavior of the modeled system. In this part, we will first understand how important system properties are modeled by Petri nets and then look into the applications of Petri net models.

An Example: The Dining Philosophers

We will start by constructing a Petri net model for the celebrated *Dining Philosophers Problem* (DPP) first proposed and studied by Edsger Dijkstra in 1968. The DPP is a popular model of reality encountered in numerous concurrent systems with contention for shared resources. We consider here a simplified version of the DPP. In this problem, there are three philosophers each of whom alternates between a *thinking* phase and an *eating* phase. The thinking times and eating times are random. The philosophers are seated at a large round table on which is placed some food that can only be eaten with the help of two forks. See *Figure* 1. Between each pair of philosophers, a fork is placed. To eat the food, each philosopher requires two forks, namely the one on the left and the one on the right. Thus there are three philosophers and three forks which are shared in the manner described. The problem, of course, is that if all philosophers pick up the fork on their left and then wait for the fork on their right, they will have to wait forever and starve (since no philosopher is willing to part with the fork he already has grabbed). Such a situation is popularly referred to as a

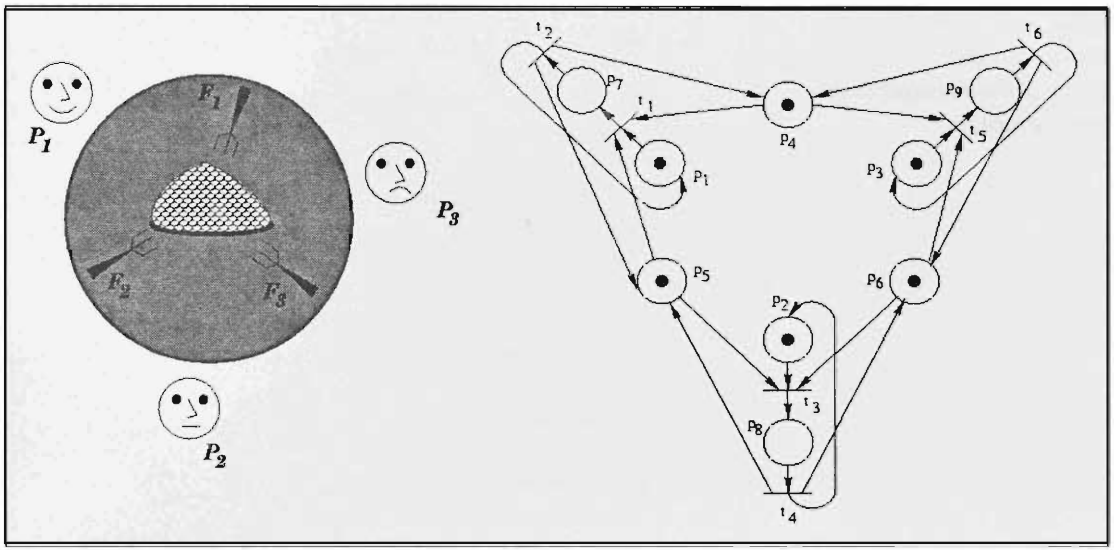

Figure 1(left). Dining philosophers.
Figure 2(right). Petri net model of a dining philosophers problem

deadlock in computer operating systems. In the computer operating systems context, each philosopher represents a *process* and each fork represents a certain *resource* such as CPU, memory, or disk required by the process. In an automated manufacturing context, each philosopher may represent a robotic assembly process that requires two robots for assembling workpieces, with the forks representing the assembly robots.

Let the three philosophers be denoted by P_1, P_2, P_3 and the three forks by F_1, F_2, F_3. P_1 needs forks F_1 (left fork) and F_2 (right fork); P_2 needs F_2 (left fork) and F_3 (right fork); P_3 needs F_3 (left fork) and F_1 (right fork). *Figure* 2 shows the Petri net model for this problem, with 9 places and 6 transitions. The semantics of the places and transitions is provided in *Table* 1.

Note that the initial marking of this model (see *Figure* 2) is such that there is one token in the places $p_1,$ $, p_6$ and no tokens in p_7, p_8, p_9. This means that all the philosophers are in the thinking state and all the forks are available for use.

In this initial marking, the transitions t_1, t_2, t_3 are enabled. However, only one of them may fire since each of these transi-

Table 1. Places and transitions in the Petri net model of the dining philosophers problem.

Places

p_1 : Philosopher P_1 in thinking state
p_2 : Philosopher P_2 in thinking state
p_3 : Philosopher P in thinking state
p_4 : Fork F_1 available for use
p_5 : Fork F_2 available for use
p_6 : Fork F_3 available for use
p_7 : Philosopher P_1 in dining state
p_8 : Philosopher P_2 in dining state
p_9 : Philosopher P_3 in dining state

Transitions

t_1 : Philosopher P_1 starts dining
t_2 : Philosopher P_1 finishes dining
t_3 : Philosopher P_2 starts dining
t_4 : Philosopher P_2 finishes dining
t_5 : Philosopher P_3 starts dining
t_6 : Philosopher P_3 finishes dining

tions conflicts with the other two transitions. In this model, the features that are modeled are: sequential execution; conflict; and synchronization. It can be seen that this Petri net has four reachable markings including the initial marking.

The DPP originally described by Dijkstra and popularly discussed in operating systems textbooks has five philosophers and five forks. One can easily develop a Petri net model for such a DPP. The resulting model will have 15 places and 10 transitions. In addition to sequential execution, conflict, and synchronization, the model will have concurrency and confusion.

Properties of Petri Nets

We now define some important desirable properties that Petri net models of physical systems should exhibit.

Definition: A place p_i of a marked Petri net is said to be *k-bounded* ($k > 0$, integer) in marking M_0 if

$$M(p_i) \leq k \quad \forall M \in R[M_0].$$

If $k = 1$, p_i is said to be *safe*. If all places of a Petri net are k-bounded (safe) in a marking M_0, the Petri net itself is said to be bounded (safe) in M_0.

Boundedness refers to a finite requirement of resources and also models absence of overflows in buffers. It can be shown that a bounded Petri net has a finite reachability set. This is an important requirement for carrying out performance analysis using Petri net models. If places represent conditions, then safety implies that the conditions are constrained to have only two boolean values: truth and falsity. In such a case, if the place in question in unsafe, it will point to incorrect modeling or incorrect logic in the system.

Boundedness refers to a finite requirement of resources and also models absence of overflows in buffers.

Definition: A marked Petri net is said to be *conservative* if

$$\sum_{i=1}^{n} M(P_i) = \text{constant} \quad \forall M \in R[M_0].$$

Conservativeness implies that the sum of tokens will remain constant in all reachable markings. Such a Petri net will represent a system with a fixed number of resources or jobs.

Definition: A marked Petri net with initial marking M_0 is said to be *proper* or *reversible* if

$$M_0 \in R[M_0] \quad \forall M \in R[M_0].$$

In a proper Petri net, the initial marking is reachable in one or more steps from every reachable marking. Properness implies reinitializability of the system and assumes significance in ensuring recovery from failure states. Properness is often referred to as reversibility and reinitializability.

Definition: A transition t_j of a marked Petri net is said to be *live* under a marking M_0 if, for all markings $M_0 \in R[M_0]$, there exists a sequence of transition firings which results in a marking that enables t_j. A Petri net is said to be live if all its transitions are live.

Properness implies reinitializability of the system and assumes significance in ensuring recovery from failure states.

Liveness of a Petri net implies absence of deadlocked states. In Petri net terms, a deadlocked state is a reachable marking

A Petri net is said to be live if all its transitions are live. Liveness of a Petri net implies absence of deadlocked states.

There are many other important properties of Petri nets such as *fairness* which we are not considering here. Also, there is extensive literature on the existence of necessary and sufficient conditions for boundedness, properness, and liveness of Petri nets.

Petri Net Applications

Having presented the definitions and details of Petri net modeling, we now examine different ways in which they can be used and the various disciplines in which they have been applied. First of all, Petri nets can be used to provide a

faithful representation of the structure and behavior of a system. They are particularly attractive for capturing features such as concurrency, non-determinism, asynchronous operation, synchronization, and flow of control. Once a Petri net model of a system is created, it can be put to use in a variety of ways.

Qualitative Analysis: The Petri net model can be subjected to qualitative analysis to check system properties such as safety, boundedness, liveness, absence of deadlocks, fairness, etc. To accomplish this, there are a variety of rigorous analysis methodologies such as:

Reachability Analysis: Here, the reachability graph of the Petri net model under a designated initial marking is obtained through graph theoretic techniques. It may be noted that boundedness is a necessary and sufficient condition for the reachability set to be finite. However, even in a bounded Petri net, the number of reachable markings could be of an exponential order in the number of places and transitions. For this reason, reachability based methods are often intractable. In such situations, the following methods may be used for analysis.

Analysis through Invariants: By defining an incidence matrix for the Petri net and using linear algebraic techniques, one can compute *place invariants* and *transition invariants* for a given Petri net model. The invariants can often be used in efficiently evaluating properties such as bounded-

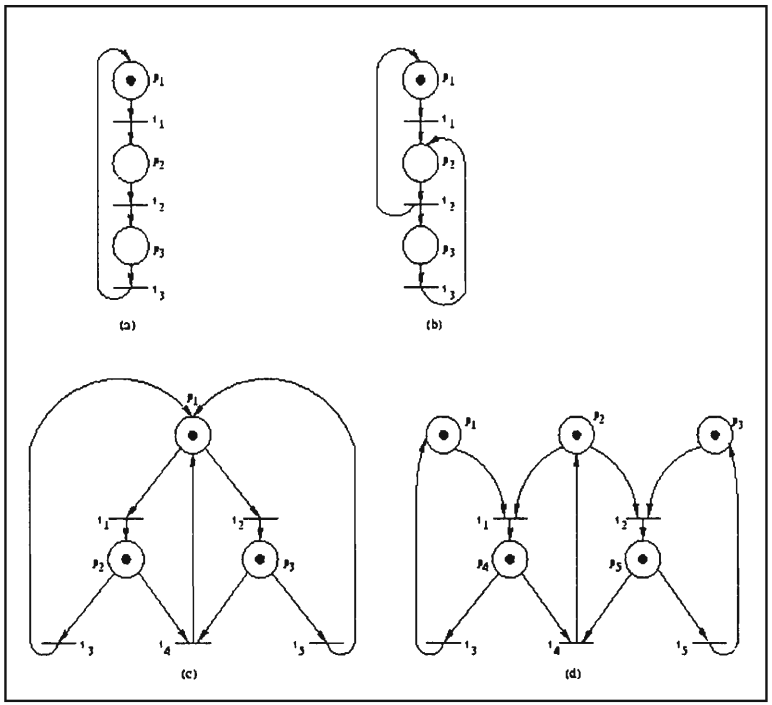

Figure 3. Petri net model of a dining philosophers problem with three philosophers and three forks.

in which none of the transitions is enabled. Liveness is a very desirable property of Petri net models.

Example 1. It can be seen that the Petri net of *Figure* 1 is bounded in each of the markings $M_0, M_1, ...M_6$, shown in *Table* 2(see previous part of this article). In fact, it is safe in all these markings. Also, the Petri net is proper and live in each of these markings. However, the Petri net is not conservative.

Example 2. The Petri net model of the dining philosophers problem (*Figure* 2) can be seen to be bounded and safe in each reachable marking. Also, the Petri net is proper and live in each of the markings. However, the Petri net is not conservative.

Example 3. *Figure* 2 shows four Petri nets satisfying different sets of these properties. The Petri net in *Figure* 3(a) is safe, bounded, conservative, proper, and live. The Petri net in *Figure* 3(b) is not safe, not bounded, not conservative, not proper, but live. In *Figure* 3(c), the model is safe, bounded, conservative, proper, but not live. Finally, the Petri net in *Figure* 3(d) is safe, bounded, not conservative, not proper, and not live.

Petri nets are particularly attractive for capturing features such as concurrency, non-determinism, asynchronous operation, synchronization and flow of control.

Petri nets have been used to investigate the presence or absence of deadlocks in distributed systems; to verify and validate complex communication protocols.

ness, conservativeness, and liveness.

Reduction Techniques: There is extensive literature on reducing the size of a Petri net model, without affecting underlying properties such as boundedness and liveness. Reduction techniques rely on replacing an entire subnet by a single place or a single transition.

Petri nets have been used to investigate the presence or absence of deadlocks in distributed systems; to verify and validate complex communication protocols; etc. All the techniques described above have been employed in these applications.

Quantitative Analysis: By representing timing of various activities in the system, a Stochastic Petri Net (SPN) model can be created. Such a model can be used in quantitative performance analysis by transforming the SPN model into a Markov chain or a more general stochastic process. Performance measures such as response time, throughput rate, resource utilization, and reliability can be computed by analyzing the SPN models either through Markov chain techniques or queueing theory techniques. For example, SPNs have been used in the performance evaluation of flexible manufacturing systems and multiprocessor systems.

Design and Synthesis: Petri net models can be composed in a variety of ways, preserving important system properties. This enables Petri net models to be used to synthesize complex systems from simpler building blocks in a hierarchical way. For example, detailed models of complex manufacturing plants have been built from simple building blocks using bottom-up methods.

Formal Specification Tool: The Petri net or SPN model of a system can become the basis of formal system specification. The model can be used in system design, system implementation, and system simulation. For example, Petri nets have been used in describing the semantics of programming languages; for specifying the requirements of communication protocols; etc.

Box 1. Applications of Petri Nets

Numerous disciplines in engineering, science, and humanities have used Petri nets in one of the ways mentioned above. Application areas include: computer operating systems, multiprogrammed computer systems, parallel and distributed computer systems, distributed databases, computer communication networks, software engineering, flexible manufacturing systems, business processes in manufacturing and service organizations, industrial process control systems, fault-tolerant computer systems, programmable logic controller and VLSI arrays, office automation systems, workflow management systems, formal languages, legal systems and jurisprudence, project management, complex command and control systems, and air traffic networks.

To give an idea of the wide range of real-world problems in which Petri nets have proved to be of great value in decision-making, we provide a sample list of such applications:

- Design of fail-safe traffic signaling systems
- Design of fault-tolerant multiprocessor systems used in avionics systems and in missile control applications
- Detection, prevention, and avoidance of deadlocks in highly automated manufacturing plants
- Performance evaluation of complex manufacturing architectures, leading to the design of optimal manufacturing strategies
- Modeling and verification of the X.25 protocol used in computer networks
- Requirements specification of software for real-time systems
- Description of semantics of programming languages
- Design and analysis of high-performance cache coherence protocols for shared memory multiprocessors
- Design of conflict-free business processes in workflow management systems
- Building programmable logic controllers for industrial process control
- As a simulation model in commercial discrete event simulation packages

Current Status

The basic Petri net formalism may be extended in a variety of ways. Indeed there are numerous extended versions of Petri nets and SPNs. The extensions have been proposed either to impart more modeling power or to enable domain-specific features to be more conveniently modeled. Extensions of classical Petri nets include: colored Petri nets, predicate-transition nets, predicate-action nets, high level Petri nets, fuzzy Petri nets, etc. Extensions of stochastic Petri nets include: generalized stochastic Petri nets, extended stochastic Petri nets, generalized timed Petri nets, stochastic activity networks, etc. Current use of Petri nets

Box 2. Petri Net Analysis Software

At the Department of Computer Science and Automation, Indian Institute of Science, Bangalore, we have built software for analyzing and simulating stochastic Petri nets and we have used it in performance analysis of automated manufacturing plants, business processes such as supply chain networks and product design processes, and fault-tolerant multiprocessor systems. Interested readers are urged to contact the author for further details.

is mostly in the form of these extended formalisms.

Numerous computer aided software packages are now available to assist in Petri net modeling and analysis of systems. This shows the theoretical maturity of the area. The area of Petri nets has grown to a degree where familiarity with technical aspects of the Petri net formalism is not needed any longer to effectively use some of the available packages. Some of these packages are commercial products and others are available in the public domain. The packages are being effectively used in many of the areas and disciplines mentioned above.

All this does not mean that research in Petri nets has come to a standstill. High quality research in Petri net theory continues to engage numerous researchers, particularly in Europe. Prominent topics of research include: connections between Petri nets and other models of computation; integrated modeling formalisms combining Petri nets with other promising formalisms for concurrent systems modeling; Petri nets as a tool for specification, analysis, and synthesis of complex discrete event and real-time systems; and Petri nets as a specification tool in software engineering. Concurrently, the list of Petri net applications continues to grow at an impressive pace, thus making knowledge of this versatile formalism an important core-competence for system modelers and system designers.

Suggested Reading

[1] J L Peterson, Petri nets, *ACM Computing Surveys*, Vol.9, No.4, 223–252, 1977.

[2] T Agerwala, Putting Petri nets to work, *IEEE Computer*, Vol.12, No.12, 85–94, 1979.

Address for Correspondence
Y Narahari
Department of Computer Science and Automation
Indian Institute of Science
Bangalore560 012, India.
Email: hari@csa.iisc.ernet.in