The 32nd Annual Symposium of the Washington Operations Research - Management Science Council
November 5, 1997, Washington, DC

Frontiers in OR/MS Track: Session I

PETRI NETS – TUTORIAL AND APPLICATIONS

Jeffrey W. Herrmann
Department of Mechanical Engineering and
Institute for Systems Research
University of Maryland
College Park, MD 20742
Email:jwh2@isr.umd.edu

Edward Lin
Institute for Systems Research
University of Maryland
College Park, MD 20742
Email: lin@isr.umd.edu

ABSTRACT

Petri nets are graphical and mathematical tools for studying discrete event systems. Petri nets have been under extensive development since Petri defined the language in 1962. Various extensions of Petri nets have been developed to study concurrent, asynchronous, distributed, parallel, deterministic, and stochastic system behavior. This paper provides a quick tutorial to learn the basic terminology, concepts, and applications of Petri nets. We introduce the fundamentals of Petri nets and black-and-white Petri nets. We show how to use Petri nets to model and analyze discrete event systems. We also describe some extensions of Petri nets and their applications. We provide a list of references for further reading.

1. Introduction

Very few mathematical tools are available to study the dynamics of discrete event systems. Petri nets are commonly used to model a variety of discrete event systems, such as communication protocols and networks [1, 2], manufacturing, production and scheduling systems [5, 12, 15, 18, 19], sequence controllers [4, 11], and the design, specification, simulation, and validation of software systems [9, 16].

Petri nets are graphical and mathematical tools. Petri nets provide a uniform environment for modeling and formal analysis. The same model supports the construction of discrete event simulators and controllers as well as the formal verification of such behavioral properties as precedence relations of events, concurrent operations, appropriate synchronization, freedom from deadlock, and mutual exclusion of shared resources.

This paper provides a quick tutorial to learn the basic terminology, concepts, and applications of Petri nets. The remainder of this paper is structured as follows: Section 2 introduces the fundamentals of Petri nets and black-and-white Petri nets. Section 3 describes properties of Petri nets. Section 4 presents analysis methods. Section 5

describes some extensions of Petri nets. Section 6 discusses the issues and provide a list of references for further reading.

2. Fundamentals

A Petri net is a bipartite directed graph that contains places (represented by circles), transitions (represented by boxes or bars), and directed arcs connecting places to transitions and transitions to places. The dynamic nature of the modeled system in a Petri net is represented by the movement of tokens. A token, represented as a small dot, is placed in a circle to indicate that the state is active. The current location and distribution of tokens in a Petri net is called its marking. The marking of the Petri net defines the state of the system. A Petri net containing tokens is called a marked Petri net.

A black-and-white Petri net can be formally defined as a four-tuple:
$PN = <P, T, I, O>$
where:
P is a finite non-empty set of places. $P = \{p_1, p_2, ..., p_n\}$
T is a finite non-empty set of transitions. $T = \{t_1, t_2, ..., t_s\}$
I is an input function. I: $(T \times P) \rightarrow \{0,1\}$, $I(t, p) = 1$ if there exists an arc from p to t. If so, p is an input place for t;
O is an output function. O: $(T \times P) \rightarrow \{0,1\}$, $O(t, p) = 1$ if there exists an arc from t to p. If so, p is an output place for t;

M: $P \rightarrow N$ is a marking that assigns a non-negative integer to each place. Such a number represents the number of tokens in the place. $<P, T, I, O, M>$ represents a marked Petri net.

The following rules are used to govern the flow of tokens in a Petri net:

1. Enabling Rule:
   A transition, t, is enabled if each input place p of t contains at least one token.
2. Firing Rule:
   Firing an enabled transition t removes one token from each input place of t and adds one token to each output place of t.

Using this notation, Petri nets can model discrete event systems and can capture important system characteristics. These include actions that must occur in a sequence, actions that can occur concurrently, actions that compete for resources, actions that must be synchronized, actions that occur in cycles, and actions that cannot occur simultaneously.

3. Petri Nets Properties

The following are some fundamental questions that one can ask about Petri nets. In general, these questions ask whether the Petri net model exhibits some desirable property or not. In turn, this property describes system behavior.

3.1 Reachability

Can the modeled system reach a specific state as a result of a functional behavior? In other words, does there exist a transition firing sequence that will lead to the desired state (marking) from the initial state (marking)? If so, the target state is a reachable state.

3.2 Boundedness

Tokens can represent information or parts. Places can represent data storage in information systems, buffers in manufacturing systems, and other storage locations. In general, these places have limited capacity. For example, in a manufacturing system, a buffer may have space to store at most three parts. It will be infeasible to store more than three (3) parts in this buffer at any time. So, we ask, will the system ever overfill the buffer? Equivalently, one could ask if the place is 3-bounded. A Petri net is k-bounded if each and every place is k-bounded. In general, a place p is k-bounded if there exists no reachable marking M: $M(p) > k$.

3.3 Safeness

Safeness is a special case of the boundedness property. A place in a Petri net is safe if the number of tokens in that place cannot exceed one. A Petri net is safe if each and every place is safe. In other words, a Petri net is safe if it is 1-bounded.

3.4 Liveness

Consider a simple manufacturing cell consists of one machine, one robot, and one buffer with size one. Assume that the machine is processing a part and there is a part in the buffer. After the machine finishes its operation, a deadlock occurs since the robot can not unload the part from the machine to the buffer and the machine can not process the part in the buffer. Deadlock has also been the subject of a number of studies in computer science. A deadlock in a Petri net is a marking in which no transition can fire. A net is said to be deadlock-free if at least one transition can always be fired. Liveness is a stronger condition than deadlock-free. A transition t is potentially firable at a given marking M if there exists a transition firing sequence μ leading to a marking M' in which t is enabled. A transition is live if it is potentially firable in all reachable markings. In other words, a transition is live if it never loss the possibility of firing. A net is live if all the transitions are live. Liveness guarantees that any operation that can be performed by the system is reachable, regardless of the sequence of previously performed operations.

4. Analysis Methods

Various methods exist to verify the above properties and to answer other questions about Petri nets. This section will discuss the following three: enumeration, linear algebraic techniques, and simulation.

4.1 Enumeration

Enumeration attempts to identify all possible markings that a Petri net can reach. For a k-bounded Petri net, one can create a reachability tree. For an unbounded Petri net, one can create a coverability tree.

4.1.1 Reachability Tree

The Reachability Problem [13] asks the following: Given a Petri net PN with an initial marking M^0 and a marking M', is $M' \in R(PN, M^0)$? The reachability set of a Petri net PN with initial marking M^0 contains markings which are reachable from M^0. Denote this set by $R(PN, M^0)$,

Reachability analysis approach for bounded system is based on the exhaustive, sequential simulation of all possible marking evolutions.

4.1.2 Coverability Tree

The Coverability Problem [13]: Give a Petri net *PN* with initial marking *M* and a marking *M'*, is there a reachable marking *M"* such that *M" ∈ M'*? Coverability analysis locates patterns in the marking evolutions. The following algorithm will create a coverability tree:

The coverability tree is computed according to the following algorithm:

1. Let the initial marking be the root node of the tree and mark this node with "new"
2. While there exist a node marked "new"
    - select a node *A* marked "new" and let *M* be the marking of node *A*
    - If *M* is identical to another marking in the tree, then mark *A* "old." Go to 2
    - If no transition is enabled in *M*, then mark *A* "terminate." Go to 2
    - Else, for each enabled transition *t*:
        - Compute *M'* which results from firing *t* in *M*. Denote *M'* as node *C*.
        - If, on the path from the root node to *C*, there exists a node *D* with a marking *M"* such that $M'(p) \geq M"(p)$ for every $p \in P$ and $M'(p) > M"(p)$ for at least one place $p \in P$,
          then let $M'(p) = \omega$ for any $p \in P$ such that $M'(p) > M"(p)$.
        - Add *C* to the tree, add an arc joining *A* to *C*, and mark this arc with *t*.
        - If there already exists a node with the same marking *M'*, then mark *C* "old", otherwise mark *C* "new"
    - Mark *A* "old" and go to 2

4.2 Linear Algebraic Technique

A second approach to the representation and analysis of Petri nets is based on linear algebra. In this approach an incidence matrix represents the Petri net, and vectors represent markings. Equations represent the dynamic behavior.

4.2.1 State Equation

Consider a Petri net PN=<*P*, *T*, *I*, *O*>,
Let *I* be a matrix with s rows and n columns. $I_{ij} = I(t_i, p_j)$
Let *O* be a similar matrix, $O_{ij} = O(t_i, p_j)$
Let $A = O - I$ be the incident matrix of *PN*.
Let the initial marking M^0 be a n element column vector. Given a sequence σ of feasible transitions from M^0, we can compute the new marking *M'* as follows: Let μ be a row vector with *s* elements, μ_i represents the number of times t_i fires in σ, then $M' = M^0 + \mu A$.

4.2.2 Invariant Analysis:

A positive integer solution *X* of $AX^T = 0$ is called a P-invariant. By analysis such solutions, one can find a set of places that has a constant number of tokens. For example, if the Petri net is the model of a manufacturing system, and if tokens represent parts, then we conclude that the work-in-process (WIP) is constant in the manufacturing system.

The positive integer solution Y of $YA = 0$ is called a T-invariant. If σ is a feasible transition sequence for a marking m^0 and Y is the vector of transition firing, $YA = 0$ implies that the marking is unchanged after σ. From the manufacturing system point of view, this means that if we use line 1 or/and line 2 to manufacture a given number of units of products, then we come back to the same initial state (the same WIP). This result is more precise than the previous one which only claimed that the total number of parts in the system are constant.

4.3 Simulation

Petri nets can be used to simulate the system behavior. In particular, simulation methods are extremely useful when time is associated with the net evolution or when we wish to know the response of the system. However, in general, simulation methods can not prove properties, even if they might help us understand the system. The simulation-based model validation can produce a limited set of states and thus can show that the model has errors but cannot prove that the model has no errors.

5. Some Extensions of Petri Nets

Petri nets were named after Carl A. Petri who created in 1962 a net-like mathematical tool for the study of communication with automata [13]. A variety of extensions have been developed including Place-Transition nets, Individual-Token nets (Colored Petri nets), Time/Timed Petri nets, and stochastic Petri nets [13, 17].

Place-Transition nets extend basic Petri nets by assigning a weight to each arc. A transition is fired if each input place of the transition has the required number of tokens specified by the weight associated with the arc from the place to the transition. Firing the transition removes multiple tokens from each input place and adds multiple tokens to each output place, the number of tokens equals the weight of the arc.

In these Petri nets, tokens are indistinguishable. However, in some applications, it is necessary to use tokens to represent different types of real-world components. Colored Petri nets can indicate different types of tokens [8]. In addition, each arc has an expression that indicates the types of tokens and the number of tokens that can flow between a place and a transition.

In some systems time is a critical system performance measure. For example, Petri nets can be used to derive production cycle time and identify bottleneck workstations in a production system. Several extended Petri net models can represent time [7, 12]. Ramchandani's timed Petri nets are derived from Petri nets by associating a fixed finite firing time with each transition. In his model, a transition takes time to fire and a transition must fire as soon as it is enabled. Merlin's time Petri nets are more general than timed Petri nets. In Merlin's model, a transition is associated with a time interval, a minimum time and a maximum time. A transition is fired at a random time within the interval once it is enabled. When the time associated with a transition is modeled as a random variable with some probability distribution the net is called a stochastic timed Petri net. A stochastic timed Petri net in which the time delay for each transition is assumed to be stochastic and exponentially distributed is called a stochastic Petri net. Petri net models that incorporate probabilistic time functions allow one to obtain production rates, through put, average delays, resource utilization, and reliability measures. Certain stochastic nets can be solved by constructing and solving the underlying (semi)-Markov processes, and for general cases they can be analyzed via Petri net-driven discrete event simulation [10].

Large systems can be modeled with hierarchical nets such as Hierarchical Colored Petri nets, Hierarchical Object Oriented Design (HOOD), and PROTOB (an object oriented language and methodology based on PROT net) [1, 6, 8, 9].

6. Summary

Petri nets are a useful graphical and mathematical tool for modeling discrete event systems. This paper introduced the fundamental Petri net notation and described properties that a Petri net can have; these include reachability, boundedness, safeness, and liveness. Enumeration, linear algebra, and simulation can be used to analyze Petri nets. Petri net extensions can represent different types of tokens, time, randomness, and stochastic systems. Although, many successful Petri net applications have been documented, there exist few inexpensive software tools for developing industrial Petri net applications. Also, there are no systematic Petri net construction approaches -- modeling remaining an art.

7. Further Readings/Petri Nets Sources

- Petri Nets mailing list: PetriNets@daimi.aau.dk
- World of Petri Nets: http://www.daimi.aau.dk/%7Epetrinet/
- Coloured Petri Nets: http://www.daimi.aau.dk/designCPN/
- Petri Nets Standard: http://www.daimi.aau.dk/%7Epetrinet/standard/ contains the activities in the effort of creating a standard for high-level Petri Nets within the International Organisation for Standardisation (ISO). In standardisation terminology this project is called 7.19.3 (Petri Nets), a sub-project of 7.19 (Diagrams for Software Engineering) and is handled by the standards group called ISO/IEC JTC1/SC7/WG11. Jonathan Billington is the editor of project 7.19.3.
- Jensen: Colored Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Springer-Verlag, Basic Concepts (Vol. 1), ISBN: 3-540-60943-1, Analysis Methods (Vol. 2), ISBN: 3-540-58276-2, and Practical Use (Vol. 3), ISBN: 3-540-62867-3

REFERENCES

1. Baldassari, M. and Bruno, G., “PROTOB: An Object-Oriented Methodology for Developing Discrete Event Dynamic Systems,” *Computer Language* 16, 1, pp. 39-63.
2. Berthomieu, Eernard and Diaz, Michael, “Modeling and Verification of Time Dependent Systems Using Time Petri Nets,” *IEEE Transactions on Software Engineering*, vol. 17, no. 3, March 1991, pp. 259-272.
3. Billington, J., Wheeler, G. R., and Wilbur-Ham, “PROTEAN: A High-level Petri Net Tool for the Specification and Verification of Communication Protocols, “ *IEEE Transactions on Software Engineering*, vol. 14, no. 3, pp. 301-311, 1988.
4. David, Rene and Alla, Hassane, *Petri Nets and Grafcet: Tools for Modeling Discrete Event Systems*, Prentice Hall, 1992.
5. DiCesare, F., Harhalakis, G., Proth, J. M., Silva, M., Vernadat, F. B., *Practice of Petri Nets in Manufacturing*, Chapman & Hall, London, 1993.
6. Drgiovanni, Raffaele, “HOOD Nets,” Advances in Petri Nets, Rozenberg (ed.), Springer-Verlag, pp. 140-160.
7. Freedman, Paul, “Time, Petri Nets, and Robotics,” *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 4, August 1991, pp. 417-433.

8. Jensen, K., “Colored Petri Nets: A High Level Language for System Design and Analysis,” G. Rozenberg (ed.) *Advances in Petri nets 1990*. Lecture Notes in Computer Science, vol. 483, Springer-Verlag, Berlin, pp. 342-416.
9. Lin, Yi-Tzer, *Modeling and Analysis for Message Reachability in Distributed Manufacturing Systems*, Ph.D. dissertation, The Georgia Institute of Technology, Atlanta, GA, 1994.
10. Molloy, M.K., “Performance analysis using stochastic Petri nets,” *IEEE Transactions on Computing*, vol. 3, no. 9, pp. 913-917, 1982.
11. Murata, T., Komoda, N., Matsumoto, K., and Haruna, K. “A Petri net based controller for flexible and maintainable sequence control and its applications in factory automation,” *IEEE Transactions on Industrial Electronics*, vol. 33, no. 1, pp. 1-8, 1986.
12. Murata, T., “Petri nets: properties, analysis and applications,” *Proceedings of the IEEE*, Vol. 77, No. 4, April 1989.
13. Peterson, J. L., *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
14. Plunnecke, H. and Reisig, W., Bibliograph of Petri Nets 1990, in *Advances on Petri Nets 1991*, Lecture Notes in Computer Science, G. Rozenberg, Ed. Berlin: Springer-Verlag, 1991, vol. 524, pp. 317-572.
15. Proth, J. M., Xie, Xiaolan, *Petri Nets: A Tool for Design and Management of Manufacturing Systems*, John Wiley & Sons, Chichester, New York, 1996.
16. Reisig, W., “Petri nets in software engineering,” in *Advances in Petri Nets 1986*, Part II, Lecture Notes in Computer Science, W. Brauer, W. Reisig, and G. Rozenberg (eds.), Springer-Verlag, Berlin, vol. 255, pp. 63-98.
17. Reisig, W. (1985) *Petri Nets: An Introduction*, Springer-Verlag, New York.
18. Silva, M., and Valette, R., “Petri Nets and Flexible Manufacturing,” *Advances in Petri Nets 1989* (G. Rozenberg, ed.) Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1990, pp. 375-417
19. Zhou, MengChu and DiCesare, Frank, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*, Kluwer Academic Publishers, 1994.