

Deadlock Avoidance in the Operating Systems with Using Coloured Petri Nets

¹Shahram Moharrami and ²Adalat Karimov

¹Department of Computer Engineering, Parsabad Moghan Branch,
Islamic Azad University, Parsabad Moghan, Iran

²Department of Information Economic and Technologies,
Azerbaijan State Economic University, Baku, Azerbaijan

Abstract: Now a days with advances in computer science and increase in processor speed, modeling methods have found extensive applications in industrial fields. Coloured Petri Nets are one of these modeling methods. Coloured Petri Nets are based on graph theory and are applied specifically for concurrent and asynchronous applications. As executable models, they are capable of graphical description of complicated systems. On the other hand, development of hardware and other peripheral computer resources and development of various computer software systems call for efficient and powerful operating systems so that users can use the software and hardware items in an effective manner. The purpose of this research is to study the application of Coloured Petri Nets for modeling resource management in operating systems with the aim of optimal utilization of resources and Deadlock Avoidance in the operating systems.

Key words: Coloured Petri Nets, resources, deadlock avoidance, operating system, place, transition

INTRODUCTION

A model is a simple and understandable representation of the structure and behavior of the system under study which in most cases can be expressed by mathematical formulas. Decision rules can be obtained directly from a pattern (Karimov and Moharrami, 2007, 2010). In fact, using the model, researchers can acquire knowledge and information about the modeled phenomenon without experiencing the costs and risks associated with the real phenomenon.

In the same way, Coloured Petri Nets are used for acquiring information about structure and function of modeled systems. Petri Nets were developed by Mr. Carl Adam Petri in 1962 (Fereydu and Safa, 2008). His research focused on information systems. Numerous groups were formed in Germany and several other countries to conduct research projects on applications of Petri Nets. In this study, after a brief introduction to Petri Nets, researchers will investigate the problem of resource management modeling in the operating system using Petri Nets.

PETRI NETS

Petri Nets are based on graph theory. They are executable and have the capability for graphical

description of complicated systems. The Theory of Petri Nets has grown in two directions (Ruhollah and Nets, 2007):

- Applied: The Applied Theory of Petri Nets
- Theoretical: The Pure Theory of Petri Nets

The objective of the developments in the applied domain is to provide essential tools, techniques and relationships for application of Petri Nets. In fact, strong theory is a prerequisite for more effective applications. Developments in the theoretical direction consider that Petri Nets are useful for real world problems. Accordingly, we take into account both theoretical and applied aspects in this article. First, we define some formulas and then we describe the power and applications of this tool.

Basic definition: In this study, researchers present the official definition of hierarchical Petri Nets. This definition is required for a basic understanding and study of Petri Nets. The formulation is based on the theory of sets. In fact, these nets provide a new class of machines called Petri Net automata.

Structure of Petri Nets: Since, Petri Nets (PN) are a special form of graphs, researchers start with a few basic concepts from the graph theory.

Definition 1: A graph consists of two components, the nodes and the edges and a method for connecting these elements (Peterson, 1981).

In other words, a Graph $G(V, E, Q)$ consists of a non-empty set V which is called the set of nodes of the graph, a set E called edges of the graph and a mapping Q from the set E onto the set of pairs V . If the pair of nodes that are connected by an edge is an ordered pair then the edge is directed and an arrow is placed on the edge to denote this fact. If all edges of a graph are directed, the graph is called a directed graph (a digraph). Two nodes that are connected by an edge of the graph are called adjacent nodes. Nodes need not necessarily be denoted by points; they can be shown using circles, bars, boxes or any other conventional shape. When a graph contains directed parallel edges (edges that connect an identical pair of nodes) that graph is called a multi-graph. A few instances of multi-graphs are shown in Fig. 1. The property of Petri Nets as graphs is that they are bi-partite graphs, i.e., they have two kinds of nodes. To differentiate these two kinds of nodes, different shapes are used to represent them. The first type of nodes is called place nodes and they are shown by a circle or ellipse. The second type of nodes are called transition nodes and are denoted by a solid bar or rectangle. The edges of Petri Nets are called arcs and are always directed. A bi-partite graph has special properties. An edge can only connect two nodes of different types. So, the edges can be arcs form a place to a transition or from a transition to a place. It is not possible for an arc to connect a place to another place or a transition to another transition (Bostan-Korpeoglu and Yazici, 2007). Figure 1 shows two instances sample graphs.

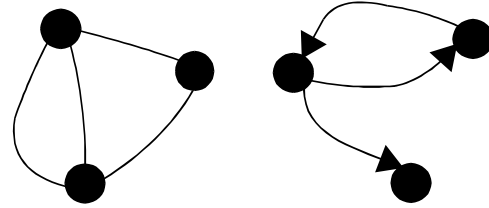


Fig. 1: Two sample graphs

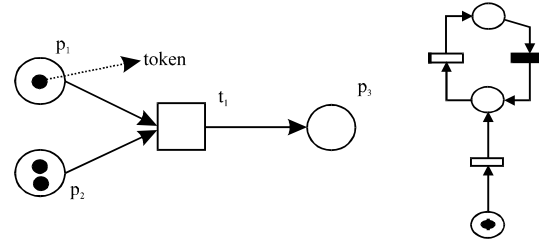


Fig. 2: Two sample Petri Nets

Definition 2: A Petri Net is composed of 4 components (Fereydun and Safa, 2008; Motameni, 2006):

- $P = \{p_1, p_2, \dots, p_k\}, k > 0$ is a finite set of places
- $T = \{t_1, t_2, \dots, t_l\}$ is a finite set of transitions (with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$)
- $I: P \times T \rightarrow \mathbb{N}$ is an input function that specifies weights of arcs directed from places to transitions
- $O: P \times T \rightarrow \mathbb{N}$ is an output function that specifies weights of arcs directed from transitions to places

I/O functions are the connecting bridge of Transitions (T) and Places (P) and connect T, P pairs to each other. The input function determines the set of input places $I(t_i)$ for each transition t_i while the output function determines the set of output places $O(t_i)$ for each transition t_i (Jensen, 1997). The structure of a Petri Net is defined with places, transitions and input and output functions. Figure 2 shows two instances of Petri Nets.

Petri Net notations: Notation m marks the placement of a series of tokens on the places of the Petri Net. Each token has an initial concept of T, P in the Petri Net. After the tokens are defined and put in Places (P) of Petri Net, the number and state of tokens can change during execution of a Petri Net. In other words, tokens are used to define the execution of the Petri Net. $M = (C, m)$ is a Petri Net, where $C = (P, T, I, O)$ and m is a notation. This is often written as (Peterson, 1981):

$$M = (P, T, I, O, m)$$

Vector m shows the number of tokens for each place p_i . m_i is the number of token at place P_i , i.e., $M(P_i) = m_i$.

Execution rules of Petri Nets: Petri Nets are executed by firing transitions (Murata, 1989). A net is executed by means of separate indexed tokens. Tokens are put in places and control the execution of net transitions. A transition T is executed by removing tokens from input places and creating new separate tokens in output places. A transition fires only if it is permissible or active and it is permissible if each of input places has at least the number of tokens equal to input arcs of that transition (which are arrows from place P to transition T).

The number of tokens needed for input arcs to activate a transition T are called activating tokens. Transition T_i in a signed Petri Net $C = (P, T, I, O)$ with sign m is activated when the number of provided tokens p_i is greater than or equal to the minimum number of incoming arcs to T_i (Fereydun and Safa, 2008; Jensen *et al.*, 1996).

Example: In Fig. 3, t_1 is fired when there are at least one token in p_1 and t_3 is fired when there are one token in p_3 . When a transition is fired, it pushes out all activator tokens and sends them to output places.

Colored Petri Nets and their creation: Colored Petri Nets are a more general form of ordinary Petri Nets. These nets make it possible to perform a more accurate and more detailed modeling of complicated asynchronous processes. The tokens can be different so, that a new attribute, known as color is added to the tokens. The arcs can include mathematical expressions which are made by a combination of the set of colors and their related variables (Bostan-Korpeoglu and Yazici, 2007).

Colored Petri Nets are created according to class charts. The first condition is that relationships are converted into intermediate classes. The second condition is that each class possesses properties or actions. Usually, a hierarchical model is used. A transition is considered for each main class.

Actions of main classes are replaced by equivalent transitions. Intermediate classes are converted into places of the model. Properties of each of these intermediate classes are transformed into the color set for each of the places. The following example shows the function of the colored Petri Net. Proposition and function t_1 (Fig. 4):

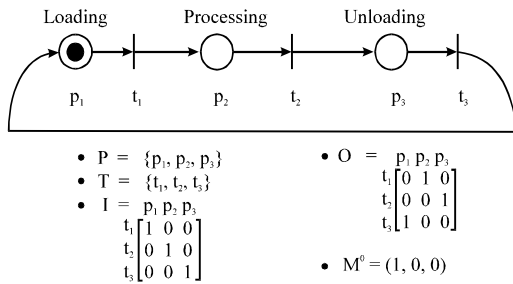


Fig. 3: Places, transitions and transition conditions

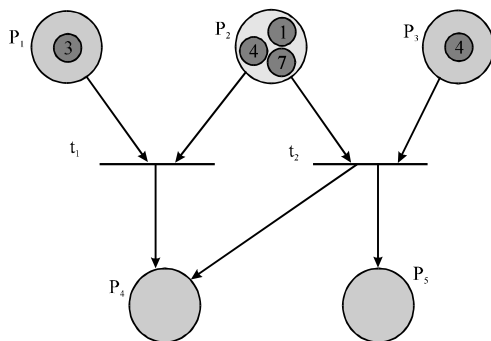


Fig. 4: Transition in colored Petri Net. Numbering into small black circles show colour of token's

$$P_2 > P_1$$

$$P_4 = P_2 + P_1$$

Proposition and function t_2 :

$$P_3 > P_2$$

$$P_4 = P_3 + P_2$$

$$P_5 = P_2 + P_3$$

OPERATING SYSTEM, RULES AND METHODS OF MODELING

The operating system is the main program that acts as an interface between the user and computer hardware and is usually the first program that is loaded into the memory after booting up the computer (Abraham, 1999). The main part of the operating system that performs its important tasks is called the kernel. The kernel is always running on the computer. The operating system has two main duties:

- It simplifies working with the computer. In other words, the user is not forced to deal with hardware issues or to call hardware system operations like writing to the disk directly
- The second task of the operating system is resource management. Resource management enables optimal and cost-effective utilization of system resources (processors, memories, disks, etc.)

The operating system works as a resource manager and allocates resources to programs according to their requirements. Evidently, the number of requested resources should not be more than the total number of available resources. If each process in a set waits for an event that can occur only by another process in that set, a deadlock state ensues (Abraham, 1999). The goal in modeling resource management is to detect and repair deadlock conditions in the system. Therefore, researchers must model requirements in a way that shows their role in the occurrence of deadlock. In fact, the effects and results of requests on the system are very important. On the other hand, modeling requests is not separate from modeling available resources. Therefore, the role of requirements in available resources should be elucidated clearly. If a program abuses a requirement, it can accentuate its role in creating deadlock. As a matter of fact, the requirements lead to enhancement of available resources. Therefore, in the modeling, the available resources should be indicated in addition to requirements (Rasul, 2005).

THE PROPOSED MODEL

Due to the presence of color in colored Petri Nets which enables us to easily detect the characteristics and number of a requirement as well as the characteristics and number of a specific resource and noting the availability of powerful modeling and analytical tools, we decided to use colored Petri Nets for modeling. The proposed model consists of two essential parts. The first part includes characteristics of the requirements for the processes and the second part includes the characteristics of available resources. In this modeling, a separate Petri Net is used for each process, consisting of a transition and two places. One place includes the tokens denoting the characteristics of resource requirements of the process (including their number and type). The other place includes tokens denoting characterizes of available resources (their type and number). Figure 5 shows a general schematic representation of a basic Petri Net.

However, each basic Petri Net related to a process has a transition in addition to the two places. The inputs to this transition come from the above-mentioned places and consist of resource requirements of the process and the available resources. The outputs are related to available resources where allocated resources are added to available resources after termination of the process. Each transition has a conditional expression for activation so that the transition takes effect if the condition is met.

The operators in these conditional expressions consist of the usual logical operators and their operands are the number of input tokens to the transition. In fact, the transition shows the possibility of the execution of the process. The transition reviews the possibility of the process according to available resources and if the process is possible, the transition is activated and the process is executed. The allocated resources return to available resources after termination of the process. This

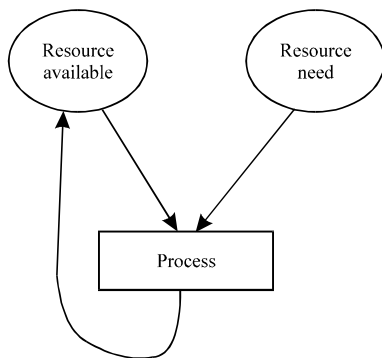


Fig. 5: General sketch of a basic Petri Net

transition prevents the occurrence of a deadlock in the system. This transition avoidance of the occurrence a deadlock in the system.

MODELING DEADLOCK CONCEPTS

An important concept associated with resource allocation is deadlock. Within Petri Net Theory an important question is does the net deadlock? A Petri Net is deadlocked if no transitions are enabled. The first subsection presents two simple net models that provide a means to introduce the concept of deadlock. In the second subsection a graphical model of the bankers algorithm and deadlock detection algorithm is discussed. This model is a completely different model from the first two and exhibits a very different use of a Petri Net Model for the same general subject area.

Introducing deadlocks with Petri Net: The following model is used to introduce the concept of deadlock. In Fig. 6 is a simple Petri Net Model of a resource allocation scheme with only two instances of a single kind and two processes that require both resources before completing. The place R represents the single resource and each token represents an instance of the resource. The remaining places represent the thread of control of the two processes. The transitions t_1 and t_2 , respectively represent the events of process a and process b requesting an instance of the resource. The transitions t_3 and t_4 represent a second request from each. Finally, transitions t_5 and t_6 Model the releasing of both resource instances. This net can deadlock if the following firing sequence occurs: t_1, t_2 . At this point no transitions are enabled. Although, resource allocation graphs are useful, the Petri Net Model adds the dynamic aspects explicitly, i.e., the possible sequences of events leading up to a deadlock. One may begin to examine what firing sequences cause deadlocks.

Once the first model is understood and the concept of deadlock is introduced, the subject of deadlock

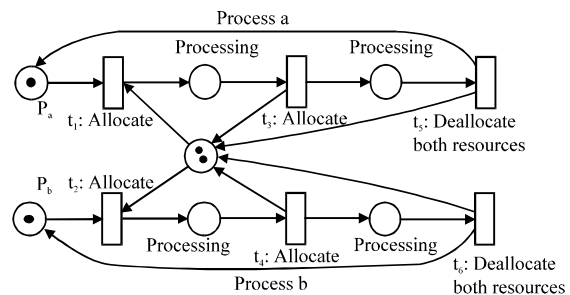


Fig. 6: Resource Allocation Petri Net Model 1

prevention and the related costs can be discussed. To illustrate the prevention protocol that calls for processes to allocate all their resources before execution begins one could use the following Petri Net in Fig. 7. One can see that if t_1 fires then t_2 and t_4 cannot be enabled until t_3 has been fired. This models the fact that process a has all the resources and process b cannot make any progress. Following the firing of t_3 , both t_1 and t_2 are enabled.

Model for deadlock avoidance and detection algorithms:

Deadlock avoidance and detection are two other major subtopics. Typically, the banker’s algorithm is presented when discussing deadlock avoidance. The banker’s (safety) algorithm checks for the safeness of allocating resources to a process p_i each time a request is made by process p_i . A similar algorithm is used for detecting deadlocks. The deadlock detection algorithm is run periodically. The output of the detection algorithm is either a sequence of processes which indicates how the system of processes may complete and thus indicate that the system is not deadlocked or the indication that the system is deadlocked.

A Petri Net Model for the deadlock detection algorithm is presented next for a system of processes with several instances of each resource type. In the following discussion we use X as an unconventional index variable to emphasize that resources are indexed by letters rather than non-negative integers. Given n processes and m resource types, the algorithm uses an $n \times m$ allocation matrix, Allocation whose entry, Allocation $[i, X] = k$ indicates that k instances of resource X are allocated to process i . Similarly, an $n \times m$ request matrix, request has entries, request $[i, X] = k$ that indicate process i is currently requesting k instances of resource X . An m -vector, available whose entry, available $[X] = k$ indicates k instances of resource X are currently available.

The generic net model is described. There is a process place and transition pair for each of the processes

in the system (each correspondingly subscripted). There is also a resource place for each resource type. Let p_i and p_x denote process and resource places, respectively. The arc set F and labeling function W consist of:

- Arcs (P_i, x_i) each with weight one, for each process i
- Arcs from transitions t_i to resource places p_x with weight label k if Allocation $[i, X] = k$ and $k > 0$
- Arcs from resource places p_x to transitions t_i with weight label k if request $[i, X] = k$ and $k > 0$

In the algorithm, a Boolean n -vector, finish is used to keep track of what processes were examined and can have their requests met. Finish $[i] = \text{true}$ when the i th process has been examined by the algorithm and its request can be met. Initially, Finish $[i] = \text{false}$ for all i where $1 \leq i \leq n$.

The place markings correspond to finish. A process place P_i is marked with one token if $\text{finish}[i] = \text{false}$ and marked with zero tokens otherwise. After a transition t_i fires, a token is removed from process place P_i . This corresponds to setting Finish $[i]$ to true. The meaning of firing a transition is discussed shortly. The resource places p_x are marked with k tokens iff available $[X] = k$. In addition to the data structures discussed above, the algorithm also uses an m -vector called work which is modified during the execution of the algorithm. Work is initialized by the available vector.

At each point in the execution of the algorithm, the value work $[X]$ corresponds to the number of tokens in p_x . When request $[i, X] \leq \text{work}[X]$ for each resource kind X and Finish $[i] = \text{False}$, the transition t_i is correspondingly enabled. This corresponds to the situation where the requests of the i th process can be met. Note that more than one request might be met. Thus, several transitions might be enabled. One is nondeterministically chosen. If process i is chosen then the work vector is updated through vector addition of itself with the i th allocation row. This corresponds to the firing of transition t_i .

Suppose that five processes, $\{p_0, p_1, \dots, p_4\}$ and three resource types, $\{A, B, C\}$ are given. Resources A, B and C have 7, 2 and 6 instances, respectively. Request matrix is request. Suppose that the matrix representing the state of resource allocation at same time T are shown in Table 1.

The following net model in Fig. 8 allows one to graphically depict the deadlock detection algorithm. One begins to arbitrarily (nondeterministically) fire enabled transitions until no transitions are enabled.

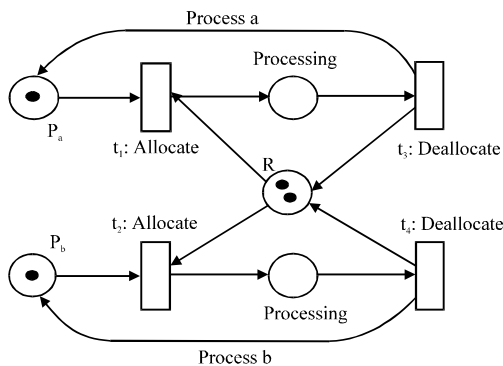


Fig. 7: Resource Allocation Net Model 2

Table 1: Processes, resource types and requests

Process	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
P ₀	0	1	0	0	0	0	0	0	0
P ₁	2	0	0	2	0	2	-	-	-
P ₂	3	0	3	0	0	0	-	-	-
P ₃	2	1	1	1	0	0	-	-	-
P ₄	0	0	2	0	0	2	-	-	-

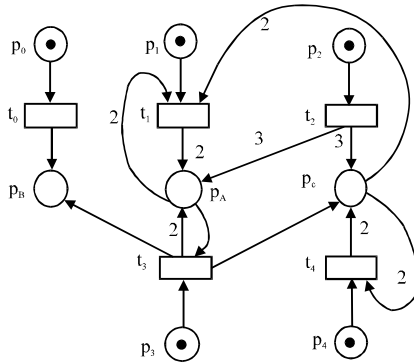


Fig. 8: Deadlock Detection Model

In this model, this does not mean the modelled system is deadlocked. Instead, the deadlock in the modeled system is detected if a token remains in any of the initially marked process places (as opposed to the resource places) and no transition is enabled. If all these process places are unmarked then that firing sequence corresponds to a corresponding process completion sequence. For the above example t_0, t_2, t_3, t_1 and t_4 is one possible firing sequence. This indicates that the above system is not in a deadlocked state. Another example of the many possible firing sequences is: t_2, t_4, t_0, t_3 and t_1 .

The Petri Net Model presented in this subsection is an alternative graphical technique for detecting deadlocks and as shown in this study can be used to model many other concepts in operating systems.

CONCLUSION

Coloured Petri Nets are an appropriate tool for modeling complicated systems and are very useful for studying concurrency and uncertainty. Now a days, modeling and simulation are used extensively in industrial fields. In fact, an industrial system can be studied before its creation and this is very cost-effective from the economic and time point of view. Petri Nets enable us to study the various components of the systems near each other. Using Petri Nets in the operating system research, researchers can investigate stability and concurrency and

prevent the occurrence of deadlock. Using this method which is a new method for modeling resource management, researchers can model complicated processes that require various resources.

REFERENCES

Abraham, S., 1999. Operating System Concepts. 5th Edn., John Wely and Sons, New York, NY, USA.

Bostan-Korpeoglu, B. and A. Yazici, 2007. A fuzzy petri net model for intelligent database. Data Knowledge Eng., 62: 219-247.

Fereydu, S. and Z. Safa, 2008. An *algorithm* for modeling expert systems using fuzzy colored Petri Nets. M.Sc. Thesis, Shahid Beheshti University, Iran.

Jensen, K., 1997. A brief introduction to coloured petri nets. Proceedings of the 3rd International Workshop on Tools and Algorithms for Construction and Analysis of Systems, April 2-4, 1997, Springer Verlag London, UK., pp: 203-208.

Jensen, K., L.M. Kristensen, L. Wells, 1996. Coloured petri nets and cpn tools for modeling. <http://link.springer>.

Karimov, A. and S. Moharrami, 2007. On approaches to automatic classification of object states based on new decision rule. Proc. Int. Conf. Soft Comp. Technol. Econ., 1: 107-116.

Karimov, A. and S. Moharrami, 2010. Automatic classification with neural networks using new decision rule. Int. J. Applied Mathe. Stat., 19: 90-96.

Motameni, H., 2006. Using markov theory for deriving non-functional parameters on transformed petri net from activity diagram. Proceedings of the Software Engineering Conference (Russia), November 16-17, 2006, Moscow, Russi.

Murata, T., 1989. Petri nets: Properties, analysis and applications. Proc. IEEE, 77: 541-580.

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

Rasul, J., 2005. Modeling vulnerabilities of computer networks using colored Petri Nets. Proceedings of the 11th International Computer Conference of Iranian Computer Association, May 19-21, 2005, Istanbul, Turkey.

Ruhollah, B. and P. Nets, 2007. Modeling internet-based remote control systems using Petri Nets. Proceedings of the 1st National Conference on Computer Engineering, March 27, 2008, Islamic Azad University, Lahijan, Iran.