Performance Analysis of Automated Manufacturing Systems with Blocking and Deadlock

N. Viswanadham
Department of Computer Science and Automation
Indian Institute of Science
Bangalore, India

T.L. Johnson
Control Systems Laboratory
GE Corporate Research & Development
Schenectady, NY 12301

Y. Narahari
Department of Computer Science and Automation
Indian Institute of Science
Bangalore, India

Abstract

The flow of multiple concurrent jobs in an automated manufacturing system (AMS), all competing for a finite set of resources, often leads to a deadlock situation. In this paper, we develop Petri net and Markov chain models for manufacturing systems with blocking and deadlock. We compute the probability that the system in an absorbing (deadlocked) state, the mean time to deadlock, and the throughput before deadlock, etc., which can be used to compare performance of various prevention and avoidance strategies. A systematic method for designing locks and interlocks for deadlock avoidance using the reachability graph of the Petri net model is presented.

Deadlocks and Deadlock Prevention in Automated Manufacturing Systems

Deadlock [1] represent a highly undesirable phenomenon of resource sharing in concurrent systems. A deadlock is a situation where each of a set of two or more jobs keeps waiting indefinitely for the other jobs in the set to release resources. In the context of flexible manufacturing systems (FMS's), jobs are associated with the completion of the sequence of steps required to manufacture a part or batch of parts. Resources refer to machines, buffers, fixtures, etc., several of which may be required at each stage of production. Deadlocks may arise as the final state of a complex sequence of operations on concurrent jobs passing through a system, and are thus generally difficult to predict. Deadlocks ultimately result, however, in zero throughput. In an improperly designed AMS, the only remedy for deadlock may be manual clearing of buffers or machines, and restart of the system from an initial condition which is known to produce deadlock-free operation under nominal production conditions. Both the lost production and the labor cost in resetting the system in this way can be avoided by proper design.

Necessary conditions for deadlock to occur are known: **(1)** ***Mutual exclusion*** -- a resource cannot be used simultaneously by two or more parts; **(2)** ***No preemption*** -- jobs hold onto resources until they finish with them; **(3)** ***Hold and wait*** -- jobs hold onto resources while waiting for additional resources to become available; and **(4)** ***Circular wait*** -- there exists a set of parts $(P_1, P_2, ..., P_n)$ such that P_1 is waiting for a resource held by P_2, and so on, with P_{n-1} waiting for a resource held by P_n, while finally P_n is waiting for release of a resource by P_1. (Circular wait is known as gridlock in New York City.) The first three conditions appear to be "common sense", while the fourth is often overlooked due to the large number of possible cycles which may occur in such systems; thus, deadlocks have actually been reported in existing FMS **[2]** under certain conditions.

The principle of deadlock prevention is easy: falsify one of the above conditions! Existing algorithms for deadlock prevention, however, often lead to conservative solutions which reduce throughput during normal operation in order to avoid deadlock under abnormal conditions that are seldom experienced.

A Simple Example of Deadlock and Deadlock Avoidance

To visualize an example of a deadlock in a manufacturing system, consider a transfer line comprising two machines, m_1 and m_2, and an automated guided vehicle **(AGV),** shown in Figure **1.** Assume that a given part undergoes manufacturing operations in the following sequence:

(i) A fresh part, say p, enters the system when m_1 is free;

(ii) after the machining by m_1 is complete, the AGV picks up p and delivers it to m_2 if m_2 is free, otherwise the part waits for the AGV;

(iii) when m_2 finishes processing p, the AGV unloads the part from the system.

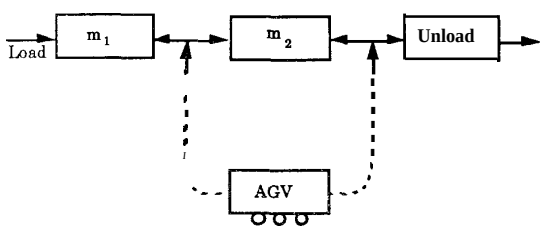

Figure 1: A simple transfer line with two machines and an AGV

0-8186-1966-X/90/0000/0064/$01.00

Now, imagine the following sequence of events, starting with an initial state in which m_1 and m_2 are both free and the AGV is available:

(i) A fresh part, say pi, enters the system, gets processed by m_1 and is delivered at m_2 by the AGV. m_2 starts processing $\mathbf{p_1}$.

(ii) A second part, say p_2, enters the system, gets processed by m_1, is put on the AGV and transported to $\mathbf{m_2}$, but the part waits on theAGV since $\mathbf{m_2}$ is busy.

(iii) A third part, p_3, enters thesystem and gets processed by m_1. Meanwhile, m_2 finishes processing $\mathbf{p_1}$.

At this stage, m1 is blocked because the AGV is not available; the AGV will not become available unless m_2 is free; but m_2 cannot be free until the AGV is available. This is a deadlock, and the whole system comes to a standstill as shown by the event sequence diagram in Figure **2.**

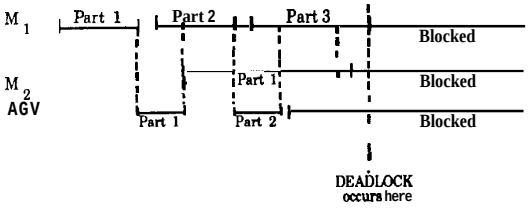

Figure 2: An Event Sequence Diagram showing a sequence leading to a deadlock

A state diagram for this example *can* be derived. Let the state of the system be denoted by (M_1, $\mathbf{M_2}$, AGV), where $\mathbf{M_1} \in$ {B,I,X} is the state set of m1 and $M_2 \in$ B,I,X} is the state set of m2. Here B represents "busy", I represents "idle", and X represents "blocked". The state set of the AGV is AGV = {B,I,X, T,U} where T represents "transport" of **a** part from m1 to m_2 and U represents "unloading" of a part from m_2 to the unloading station. From the transition diagram in Figure 3, it is seen that blocking corresponds to the existence of an absorbing (or trap) state in the diagram, labelled (X,X,X).

One approach to deadlock avoidance is resource scheduling. By examining the transitions labelled 1 and 2 in Figure 3, it is seen that a control rule which keeps m_1 idle when the AGV is transporting (state T), and keeps the AGV idle (state I) when M_2 is busy would prevent these transitions from occurring, and thus prevent the system from entering the deadlocked state. This solution avoids deadlock by reducing throughput under certain conditions.

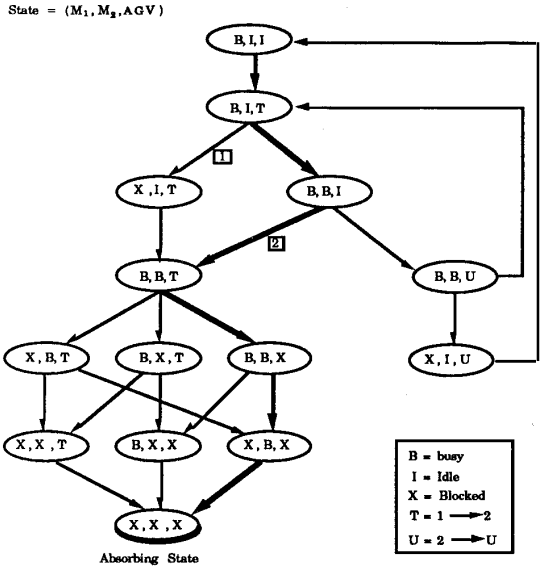

Figure 3: State Diagram

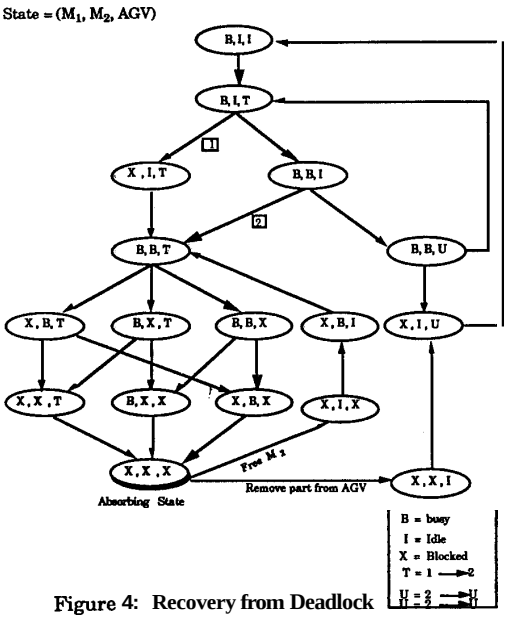

Figure 4: Recovery from Deadlock

Another solution is to use two AGV's, removing the resource contention; this solution would preserve throughput.

An augmentation of the state diagram, Figure **4,** illustrates recovery from deadlock. Here, the state (X,I,X) is added by the possibility of freeing $\mathbf{m_2}$, while the state (X,X,I) is made possible by removing **a** part from the AGV. Either of these possibilities could be realized by adding a buffer resource.

Petri Net Models and Deadlock Detection

By evaluating the reachability graph **of** a Petri net model of an AMS, sequences **of** events which lead to deadlocks can be identified **[3], [4].** These sequences of events can be detected, and can be prevented from completing, by using resource scheduling as in the example above. This process is illustrated by continuing with the analysis of the example in Figure **1.**

A Generalized Stochastic Petri Net (GSPN) for this two-machine system is shown in Figure **5.** The places and transitions **of** the the model are defined in Table **1.**

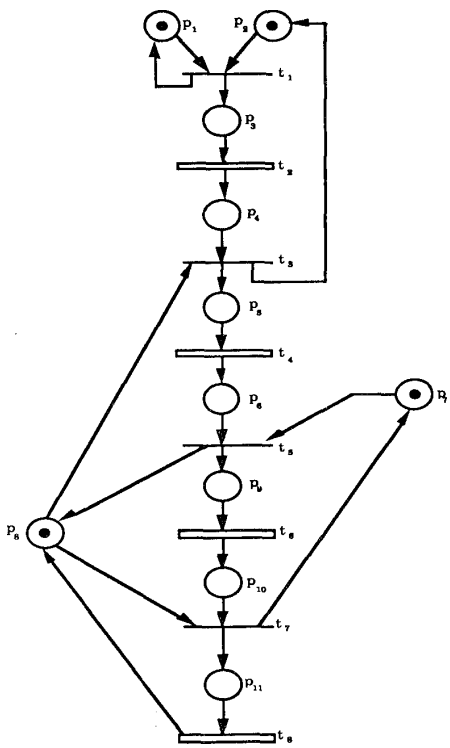

Figure *5:* GSPN Model For The Transfer Line of Figure **1**

Table **1:** Definition **of** Places and Transitions corresponding to Figure **5**

Places:		
p_1	:	**At least one raw part waiting for processing by M_1**
p_2	:	**M_1 idle**
p_3	:	**M_1 processing a part**
p_4	:	**M_1 waiting for AGV, after finishing processing**
p_5	:	**AGV transporting a part, from M_1 to M_2**
p_6	:	**AGV waiting for M_2, after bringing a part to M_2**
p_7	:	**M_2 idle**
p_8	:	**AGV idle**
p_9	:	**M_2 processing a part**
p_{10}	:	**M_2 waiting for AGV, after finishing processing**
p_{11}	:	**AGV transporting a part from M_2 to unload station**
Transitions		
t_1	:	**M_1 starts processing a part**
t_2	:	**Processing of a part by M_1**
t_3	:	**AGV starts transporting a part from from M_1 to M_2**
t_4	:	**Travelling of AGV from M_1 to M_2**
t_5	:	**M_2 starts processing a part**
t_6	:	**Processing of a part by M_2**
t_7	:	**AGV starts transporting a part from M_2 to unload station**
t_8	:	**Travelling of AGV from M_2 to unload station**
t_1, t_3, t_5, t_7	:	Immediate transitions
t_2, t_3, t_5, t_7	:	**Exponential transitions**

The reachability graph for this GSPN model can be constructed by standard means. It has 8 vanishing markings and **14** tangible markings. The marking $\{p_1 p_4 p_6 p_{10}\}$, for instance, corresponds to a deadlock since none of the transitions of the GSPN model is enabled in the marking. This corresponds to the absorbing state **of** the state machine model **of** Figure **2.** By doing an extensive path analysis **of** the reachability graph, all transition firing sequences which lead to each deadlock marking can be enumerated. Though complex, this path analysis can be automatically computed for Petri Net graphs of modest size. The firing sequences leading to deadlock can be detected and avoided by scheduling using operating rules, as described in the example **of** the previous section. This method can be used to

design a real-time controller for deadlock avoidance having the architecture shown in Figure 6.

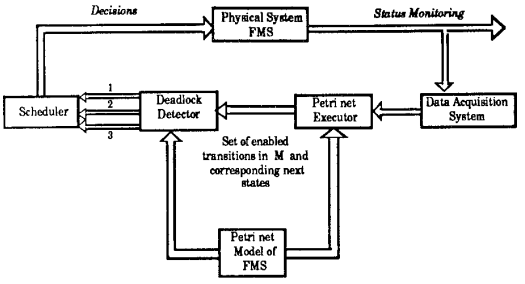

Figure 6: Real-Time Controller for Deadlock Avoidance

This type of deadlock avoidance can be implemented as an extension of an existing monitoring an control system. For instance, if a data acquisition system measures machine states, the corresponding Petri net marking may be deduced via the "Petri Net Executor". Markings which can lead to firing sequences resulting in deadlock can be detected by identifying sets of enabled transitions and corresponding next states. This is the basis for deadlock detection. The scheduler disables certain transitions by enforcing a scheduling (event sequencing) discipline when these markings are encountered, maneuvering the system to avoid the deadlock. Such a discipline is readily implemented, for instance, in a Programmable Logic Controller (PLC).

Performance Analysis of an AMS with Deadlocks

A continuous time Markov chain can be obtained by eliminating all vanishing markings from the reachability graph, as outlined in [5]. In the case of deadlock, the Markov chain will have absorbing states. The theory of such Markov chains [6] can be used to evaluate performance statistics of systems known to contain deadlocks. Such statistics are useful in predicting how often deadlocks may occur under normal operation. From the preceding discussion, the inherent complexity of the deadlock prevention problem should be clear, and if the expected time to encounter an absorbing state is sufficiently long, the trouble of manually clearing it may be worthwhile, particularly in low volume or single-shift production operations.

For FMS or AMS with deadlocks, the states in which the system is deadlocked correspond to absorbing states of the derived Markov chain. The probability that the system enters an absorbing state approaches 1 as time increases. The exact dynamics may be obtained by solving the Kolmogorov equation

$$\dot{\Pi} = \Pi Q; \ \Pi(0) = \Pi_o \qquad (1)$$

with initial state corresponding to the expected distribution of initial markings (e.g., obtained from a steady-state flow analysis of the desired operation of the system). Let $\pi_j(t)$ be the probability that the system is in the deadlocked state **j** at time t, and let s be the time to deadlock. Then the mean time to deadlock is E(s), where

$$E(s) = \int_0^\infty R_s(t)dt \qquad (2)$$

where the "deadlock rate" R(t) is

$$R_s(t) = Pr(s > t) = 1 - \pi_j(t) \qquad (3)$$

If E(s) is large, the system will not get into deadlock for a long time. If this time is long compared to the time between equipment maintenance actions, new parts entering the system, or shift changes, then manual clearing may be feasible.

Conclusions

Deadlocks in AMS's may be encountered due to particular sequences of parts entering the system, such that the pattern of scarce resource use creates a circular wait condition wherein each part is waiting for one or more resource to be freed by earlier parts, but where a part which has recently entered the system reserves a resource which is needed again for a later operation on an earlier part. The sequences of events which trigger deadlocks are in general **quite** difficult to predict, requiring a path analysis of the reachability tree of a Petri net model of the AMS -- a problem of combinatoric complexity which is only feasible to solve in practice for systems of modest size. When deadlock is known to be a possiblity, a number of remedies are available: adding production resources or buffers, changing the parts routing rules, changing the parts flow or floor layout, or preventing deadlocks by prior detection and on-line resource scheduling. Using a Markov chain model derived from a Petri net model of an AMS, it is possible to choose among these options by predicting the mean time to deadlock, number of parts processed prior to deadlock, and other measures of the seriousness of the problem. Although very high costs are associated with clearing of deadlocks in an AMS, even these costs may be tolerable if deadlock-producing sequences of events are extremely rare or if production rates are low during certain periods.

References

[1] Coffman, Edward, G., Elphick, M.J., and Shoshani, A., "System Deadlocks", ACM Computing Surveys, Vol. 3, No. 2, pp 67-78, June 1971.

[2] Viswanadham, N., Johnson, T.L., "Fault Detection and Diagnosis of Automated Manufacturing Systems", Proceedings of IEEE 27th Conference on Decision and Control, Vol. 3 of 3, pp. 2301-2306, December 1988.

[3] Suri, R., "Quantitative Techniques for Robotic System Analysis", in: Handbood of Industrial Robotics, **S.Y.**Nof (Ed.), John Wiley & Sons, New York, pp 605-638,1985.

[4] Alla, H., Ladet, P., Martinez, P., and Silva, M., "Modelling and Validation of Complex Systems by Coloured Petri Nets: Application to FMS", in Advances in Petri Nets -- 1984, Lecture Notes in Computer Science, Vol. 188, Springer-Verlag, Berlin, West Germany, pp 15-31, 1985.

[5] Marsan, M.A., Balbo, G. and Conte, G., 'Performance Models of Multiprocessor Systems", The MIT Press, Cambridge, Massachusetts, 1986.

[6] Trivedi, K.S., "Probability and Statistics with Reliability, Queueing and Computer Science Applications", prentice-Hall Inc., Englewood Cliffs, N.J., 1982.