A Simple Deadlock Avoidance Algorithm in Flexible Manufacturing Systems

Paul E. Deering, PhD
Department of Industrial Technology
Ohio University
deering@ohio.edu

Abstract

As Flexible Manufacturing Systems (FMS) become more flexible and complex the subject of deadlock avoidance becomes essential. This paper presents a simple yet effective algorithm that can be implemented at two levels of complexity to avoid deadlock in a FMS. This paper discusses the differences between primary deadlock and impending deadlock; it models a FMS using digraphs to calculate slack, knot, order and space to avoid deadlock. Several examples are provided demonstrating the method.

Introduction

Allowing a manufacturing system to enter only live states (deadlock-free states) and avoid any dead states (deadlocked states) can save both loss of production and labor costs as well as provide better resource utilization. Moving the wrong part in a live FMS can cause deadlock that can both cripple the entire manufacturing system and stall production. The only recourse is to manually resolve the deadlock and reset the FMS to a known state that is live. To prevent manual deadlock resolution in a FMS, a deadlock avoidance algorithm that can determine which parts to move must be incorporated into the controller of the FMS.

There are two types of deadlock that can occur in a manufacturing system. The most basic type is primary deadlock; this situation occurs when each part on a circuit requests the next resource in its process plan. This situation is illustrated in Figure 1.

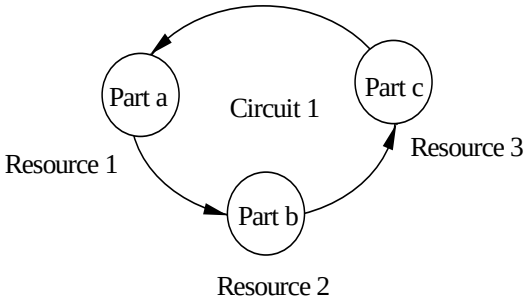

Figure 1: Example of primary deadlock

Assume that part *a* in resource 1 has to go to resource 2, part *b* in resource 2 has to go to resource 3, and part *c* in resource 3 has to go to resource 1 before each part is completed. If resource 1, resource 2 and resource 3 can only hold one part at a time, no parts can move without intervention. Circuit 1 in Figure 1 is said to be in primary deadlock. A more complex and difficult-to-detect type of deadlock is called impending deadlock; this occurs when parts can move through the system but will terminate in primary deadlock after a finite number of moves. Consider the system shown in Figure 2 and assume that each resource can only hold one part. Assume that part *a* is occupying resource 1 and that part *a* first requires resource 2, then resource 3. Likewise, assume part *b* is occupying resource 3 and the next resource required by part *b* is resource 2 followed by resource 3. Although part *a* can move to resource 2, the system will terminate in primary deadlock on circuit 2. Part *b* can also move to resource 2, but primary deadlock will result on circuit 1.

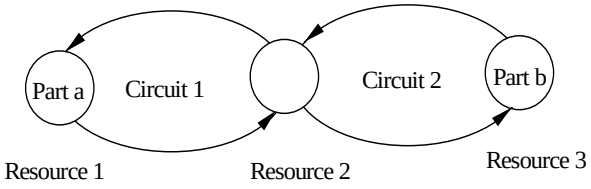

Figure 2: Example of impending deadlock

The two main approaches to solving the deadlock situation in manufacturing systems include: detection and resolution and avoidance. Deadlock detection and resolution methods [3, 4, 10, 13, and 14] allow deadlocks to occur. The deadlock is resolved by implementing a deadlock recovery procedure which moves parts to buffers and reset the system to a live state. Deadlock avoidance methods [1, 2, 5-9, 11, 12, 15, 16-18] avoid deadlock by controlling the mix of parts in the system at any given time. A part can be moved or introduced into the system only if the move does not cause deadlock. If a move is found to cause deadlock, then the move is not allowed to occur thus avoiding the deadlock state.

This paper presents the results of [16] and demonstrates a simple algorithm which can be implemented at two levels of complexity to avoid both primary and impending deadlock. This paper is organized as follows: the first section discusses previous research on deadlock in a FMS; the next section defines a mathematical model of a manufacturing systems; circuit parameters slack, knot, order and space is then defined; the next section proves sufficient conditions for a deadlock-free system; several examples demonstrating the method is then provided; next the deadlock avoidance algorithm is presented; and finally concluding remarks about the method and future research.

Previous Research

Many researchers use Petri Nets [1, 2, 5, 10-12, 14] as a formalism to describe deadlock in a manufacturing system. Banaszak and Krogh [1] proposed a deadlock avoidance algorithm (DAA) that developed a restriction policy based upon production route information to guarantee that no circular wait situations will occur. Their DAA is sufficient for avoiding deadlocks but is not an optimal solution. Viswanadham, Narahari and Johnson [10] developed a deadlock avoidance algorithm which employed a look-ahead policy. This algorithm did not detect all deadlocked states and suggested using a recovery mechanism in case of system deadlock. Zhou and DiCesare [11] and Zhou [12] generalized the sequential mutual exclusions (SME) and parallel mutual exclusions (PME) concepts and derived the sufficient conditions for a Petri net (PN) containing such structures to be bounded, live and reversible. In general, PN solutions are suitable for manufacturing systems which contain few resources but become very complicated for larger systems.

Another formalism to describe the manufacturing system is to use Graphs [3, 4, 6-9, 13-19]. In this approach, the vertices represent resources and the edges represent part flows between resources. Wysk, Joshi, and Yang [13] were the first to develop a specialized directed graphical structure called a Wait Relation Graph (WRG) to model a manufacturing system. In [13] they developed a string manipulation procedure which yields a set of control actions to detect and recover from primary deadlock. Cho, Kumaran and Wysk [3] used system status graphs to develop the concept of simple and non-simple bounded circuits with empty and non-empty shared resources to detect part flow deadlock and impending part flow deadlock. This method introduced the concept of a bounded circuit to detect deadlock. The method detected deadlock based on characteristics of this bounded circuit. The methods in [13] and [3] could only handle single capacity resources. Fanti, Maione and Turchiano [4] used a graph called Working Procedure Digraph and developed a simple graph-theoretic method for deadlock detection and recovery in systems with multiple capacity resources. This algorithm did not prevent deadlock from occurring either and suggested a suitable recovery strategy.

Judd and Faiz [6] expanded upon the original formulation proposed by [13] and the first to define slack, order and space to avoid deadlock. This method provided sufficient conditions for deadlock by satisfying a set of linear inequalities. Lipset, Deering and Judd [8] extended [6] to precisely quantify both necessary and sufficient conditions for deadlock to exist. In this research, they redefined the order of a knot, defined a special state called an evaluation state and the concept of order reduction. The approach was to put the system into an evaluation state and then compute the order. Deering [16] improved [8] by further refining the order of a knot, evaluation state and eliminated the need for order reduction. Zhang, Judd and Deering [15] developed a Deadlock Avoidance Algorithm (Daa) based on [8] and [16] that avoided deadlock and executed in polynomial time. Zhang, Judd and Deering [18] expanded upon [15] and [16] to quantify the sufficient conditions for a system state to be live and derived the liveness necessary and sufficient conditions for an evaluation state. Zhang and Judd [19] extended [18] to allow choice in process flow or flexible part routing.

Modeling a Manufacturing System

A Flexible Manufacturing System consists of a set *R* of finite resources, e.g. robots, buffers, and machines, which produce a finite set *P* of products. Each resource $r \in R$ has a capacity of cap(*r*) units that can perform the required operations. The capacity function can be extended to a set of resources, that is

$$\text{cap}(R_1) = \sum_{\forall r \in R_1} \text{cap}(r), \quad \text{for any } R_1 \subseteq R. \tag{1}$$

For each product $p \in P$, the process plan $\text{plan}(p) = r_1 r_2 \ldots r_m$ defines the sequence of resources that are required to produce *p*. Resource r_m is the terminal resource for product *p*. It is assumed that all process plans are fixed, finite and sequential. A part is an instance of a product that flows through the system. At any given time, a manufacturing system is working on a set Q of parts. The function $\text{class}(q)$ returns the product *p* to which part *q* belongs.

A manufacturing system can be represented by a Wait Relation Graph (WRG) $G = (V, A)$. Each vertex represents a resource; that is, *V*=*R*. A directed arc is drawn from vertex r_1 to vertex r_2, if r_2 immediately follows r_1 in at least one process plan. Each arc will be labeled with the part(s) that will flow through it. A subgraph $G_1 = (R_1, A_1) \subset G$ of a WRG consists of a subset of the resources and arcs of *G* so that all the arcs in A_1 connect resources in R_1. The union (intersection), denoted by $G_1 \cup G_2$ $(G_1 \cap G_2)$, of two subgraphs is the union (intersection) of the component resource and arc sets. A path $P = (R_p, A_p)$ is a subgraph whose resources and arcs can be ordered in the list $r_1 a_1 r_2 a_2 \ldots a_{n-1} r_n$ where each arc in the list connects the resources on either side. When specifying a path, writing the arcs is redundant. Therefore, only the resources will be enumerated when a path is defined. A simple path is a path with no repeated elements in the ordered list. A closed path is a path with the same first and last element. A simple circuit is a closed path with no repeated elements in the ordered list except the first and last element.

The function $\text{n}(q)$ returns a positive integer that represents the position in $\text{plan}(\text{class}(q))$ of the operation that is currently processing *q*. When a new part q is added to the system, then $\text{n}(q) = 1$. As the part is moved from resource to resource according to its plan, $\text{n}(q)$ is incremented until it reaches the end of its plan and exits the system. The state *n* of a manufacturing system is a vector containing the current n(*q*) for all $q \in Q$. A state *n* of a manufacturing system is live if a sequence of part movements exist that will empty the system. A state *n* of a manufacturing system is dead, or deadlocked, if it is not *live*.

Given a manufacturing system $G = (R, A)$, let $a \in A$ and $r \in R$. Then the function $\text{tail}(a)$ returns the resource at the tail of the given arc; the function $\text{head}(a)$ returns the resource at the head of the arc. A unit of the resource $r = \text{tail}(a)$ is said to be committed to arc *a* if it is processing a part *q* whose next resource in its process plan is $\text{head}(a)$. It is important to note that the number of resource units committed to the outgoing arcs of *r* can be less than the number of busy units. This happens when some of the busy units are being used for terminal operations. A resource unit is free if it is not committed to an arc; by this definition a busy unit which is not committed is still termed free. A resource is free if any of its units are free. A resource is empty if it contains no parts. The commitment function $\text{com}(a, n)$ returns the number of resource units that are committed to arc *a* when the system is in state *n*. The commitment function is extended to a set of arcs as follows:

$$\text{com}(A_1, n) = \sum_{\forall a \in A_1} \text{com}(a, n), \quad \text{for any } A_1 \subseteq A. \tag{2}$$

A part is enabled if either the next resource in its process plan contains at least one resource unit that is not busy or the part is in the last step of its process plan. Suppose that the system is in state n_0; there exists an arc *a* such that resource $r_2 = \text{head}(a)$ is free and the part in the resource $r_1 = \text{tail}(a)$ is committed to *a*. Then, when r_1 finishes its operation, this part can be moved to resource r_2. This process is called propagation. The symbol n_k is used to denote the state of the system after the k^{th} propagation. A part *q* in WRG G can be shifted to resource *r* if it can be propagated to *r* without propagating any other part in *G*. A part *q* in WRG G is said to have a free exit if it can be shifted its terminal resource r_m in *G*.

Slack, Knot, Order and Space

This section will summarize the major concepts and results from [6, 8, 13, 16]. This section defines the concept of slack, knot, order and space.

The slack is the number of free resources units available for parts to flow on a subgraph.

Definition 1: The slack of any subgraph $G_1 = (R_1, A_1) \subseteq G$ is given by

$$\text{slack}(G_1, n) = \text{cap}(R_1) - \text{com}(A_1, n). \tag{3}$$

A closed path *c* in a WRG *G* is in primary deadlock in state *n* if $\text{slack}(c, n) = 0$.

An interesting phenomenon happens when two simple circuits are joined by a single capacity resource as opposed to a multiple capacity resource. Consider the two manufacturing system depicted in Figure 3 and Figure 4. Here, two simple circuits are joined by resource r_0, and all parts are committed to their outgoing arcs. In Figure 3 and

Figure 4, the labels indicating which parts flows through each arc have been left off for simplicity.

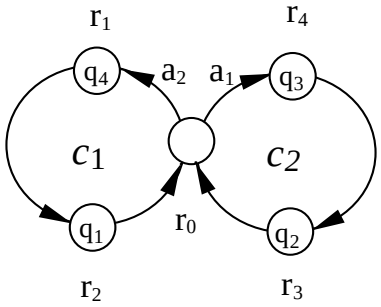

Figure 3: Two simple circuits intersecting at a single capacity resource

Assume that all resources in both systems are of a capacity of one except where $\text{cap}(r_0) = 2$ and part q_5 is committed to arc a_1 (see Figure 4). Further assume in both systems that if q_1 is moved to resource r_0, it will be committed to arc a_1 and if part q_2 is moved to resource r_0, it will be committed to arc a_2.

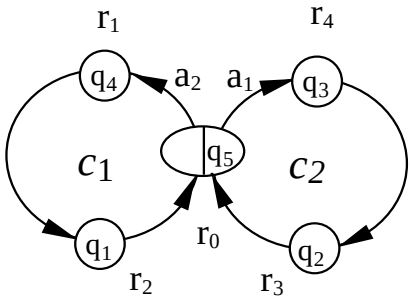

Figure 4: Two simple circuits intersecting at a multiple capacity resource

Even though resource r_0 is free in both manufacturing systems, Figure 3 shows a dead system and Figure 4 a live one. In Figure 3, if either part q_1 or q_2 is moved into r_0, then primary deadlock will result on circuit c_2 or c_1, respectively. In Figure 4, if part q_1 were moved into r_0, then primary deadlock will result on circuit c_2. However, moving part q_2 will not cause deadlock since this move will allow the parts to propagate along circuit c_2. This observation motivates the following definitions.

Definition 2: Let c_1 and c_2 be any two closed paths in a WRG of a manufacturing system. If $c_1 \cap c_2$ consists of exactly one resource with a capacity of one, then this resource is called a *knot* with respect to $c_1 \cup c_2$.

Clearly, r_0 in Figure 3 is a knot with respect to $c_1 \cup c_2$ since cap(r_0) = 1 and $c_1 \cap c_2 = r_0$. Resource r_0 in Figure 4 is not a knot with since cap(r_0) = 2. The next two definitions are needed to define the order of a knot.

Definition 3: Let c_1 and c_2 be two closed paths in a WRG *G*. Path c_1 is *connected* to c_2 if $c_1 \cap c_2 \neq 0$ and a part currently exists in the system that must propagate from c_1 to c_2 without leaving $c_1 \cup c_2$.

Definition 4: Given two closed paths c_1 and c_2, then c_1 and c_2 are *cross connected* if c_1 is connected to c_2 and c_2 is connected to c_1.

Definition 5: Let the closed path *c* in state *n* consist of two closed paths, c_1 and c_2, such that $c = c_1 \cup c_2$ and $c_1 \cap c_2 = k$ where *k* is a knot. The order of knot *k* with respect to the closed path c in state *n* is defined as

$$\text{order}(k,c,n) = \begin{Bmatrix} 1, \text{if } c_1 \text{ and } c_2 \text{ are cross connected.} \\ 0, \text{otherwise.} \end{Bmatrix} \tag{4}$$

The order of any simple circuit is zero.

The order of r_0 in Figure 3 is one since c_1 and c_2 are cross connected since $\text{order}(r_0, c_1 \cup c_2, n) = 1$.

Definition 6: Let *c* be a closed path in a WRG *G* in state *n* that contains *m* knots. Then, the order of *c* is given by

$$\text{order}(c,n) = \sum_{i=1}^{m} \text{order}(k_i, c, n). \tag{5}$$

Definition 7: Let *c* be a closed path in a WRG *G* of a manufacturing system in state *n*. The free space on a closed path *c* is the difference between the slack and the order

$$\text{space}(c,n) = \text{slack}(c,n) - \text{order}(c,n) \quad \forall c \in C_G \tag{6}$$

where C_G is the set of all closed paths in *G*.

The following theorem proves that if all closed paths of a WRG *G* have space greater than zero, *G* is *live*.

Theorem 1: Let C_G be the set of all closed paths in a non-empty WRG *G* in state *n*. If

$$\text{space}(c,n) > 0 \quad \forall c \in C_G, \tag{7}$$

then *G* is live.

Proof. See [16].

Examples

This section consists of three examples demonstrating the method. The third example will show a condition where a live system is evaluated to be dead.

Example 1: Let the WRG *G* in Figure 5 be in state *n*. The manufacturing system is composed of six resources, r_1, r_2, r_3, r_4, r_5 and r_6, all with unit capacity. The system contains seven closed paths. Let C_G represent the set of closed paths in *G*, i.e.

$$C_G = \{c_1, c_2, c_3, c_1 \cup c_2, c_1 \cup c_3, c_2 \cup c_3, c_1 \cup c_2 \cup c_3\}.$$

Suppose that the system manufactures three products, p_1, p_2 and p_3, specified by the following process plans: $\text{plan}(p_1) = r_1 r_6 r_2 r_3$, $\text{plan}(p_2) = r_3 r_4 r_6 r_5$, and $\text{plan}(p_3) = r_5 r_6 r_1$. Assume that parts *a*, *b*, and *c* belong to product classes p_1, p_2 and p_3 respectively. Suppose that the system is in state $n = [\text{n}(a), \text{n}(b_1), \text{n}(b_2), \text{n}(c)] = [1,2,1,1]$.

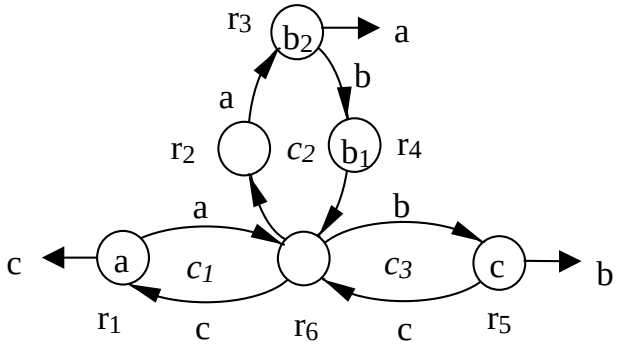

Figure 5: Manufacturing system for example 1

Table 1 on the next page shows the capacity, commitment, slack, order and space computations in state *n*.

Table 1: Circuit parameters for example 1

Subgraph	Capacity	Commitment	Slack	Order	Space
c_1	2	1	1	0	1
c_2	4	2	2	0	2
c_3	2	1	1	0	1
$c_1 \cup c_2$	5	3	2	0	2
$c_1 \cup c_3$	3	2	1	0	1
$c_2 \cup c_3$	5	3	2	0	2
$c_1 \cup c_2 \cup c_3$	6	4	2	1	1

Clearly, the space of all closed paths is greater than zero. The manufacturing system is live according to Theorem 1. Table 2 shows one possible sequence of moves to empty the system.

Table 2: Part movements to empty system in example 1

Part Movement	Resulting State after move
a to r_6	2 2 1 1
a to r_2	3 2 1 1
c free exit	3 2 1 -
b_1 free exit	3 - 1 -
b_2 free exit	3 - - -
a free exit	- - - -

Example 2: Let the WRG G in Figure 6 be in state n_0. The process plans for parts a, b and c are as presented in Table 3. Let $c_1 = r_1 r_4 r_2 r_1$, $c_2 = r_2 r_3 r_4 r_2$, and $c_3 = r_4 r_5 r_6 r_4$. Assume that the state of the system is $n_0 = [n(a_1), n(a_2), n(b), n(c)] = [2, 1, 2, 1]$.

Table 3: Process plans for example 2

Part	Process Plan
a	$r_2 r_1 r_4 r_2$
b	$r_2 r_3 r_4 r_5 r_6$
c	$r_6 r_4 r_2$

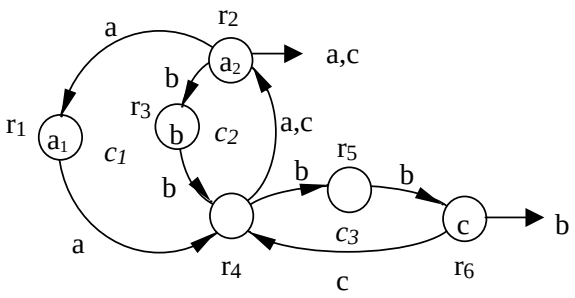

Figure 6: Manufacturing system for example 2

Table 4: Circuit parameters for example 2

Subgraph	Capacity	Commitment	Slack	Order	Space
c_1	3	2	1	0	1
c_2	3	1	2	0	2
c_3	3	1	2	0	2
$c_1 \cup c_2$	4	3	1	0	1
$c_1 \cup c_3$	5	3	2	0	2
$c_2 \cup c_3$	5	2	3	1	2
$c_1 \cup c_2 \cup c_3$	6	4	2	1	1

Clearly, the space of all closed paths is greater than zero. The manufacturing system is live according to Theorem 1. Table 5 shows one possible sequence to empty the system.

Table 5: Part movements to empty system in example 2

Part Movement	Resulting State after move
a_1 to r_4	3 1 2 1
a_2 to r_1	3 2 2 1
a_1 free exit	- 2 2 1
a_2 free exit	- - 2 1
b to r_4	- - 3 1
b to r_5	- - 4 1
c free exit	- - 4 -
b free exit	- - - -

Example 3: Theorem 1 can conclude that a system is live if the space of all closed paths is greater than zero. If the space is zero the system may be live or dead. Consider the following two cases.

Case 1: Suppose that the system in Figure 7 has the process plans depicted in Table 6.

Table 6: Process plans for example 3

Part	Process Plan
a	$r_1 r_2 r_3 r_4 r_6$
b	$r_5 r_3 r_4 r_2 r_1$
c	$r_6 r_4 r_2 r_3 r_5$

Assume that the system is in state $n = [a, b, c] = [1,1,1]$.

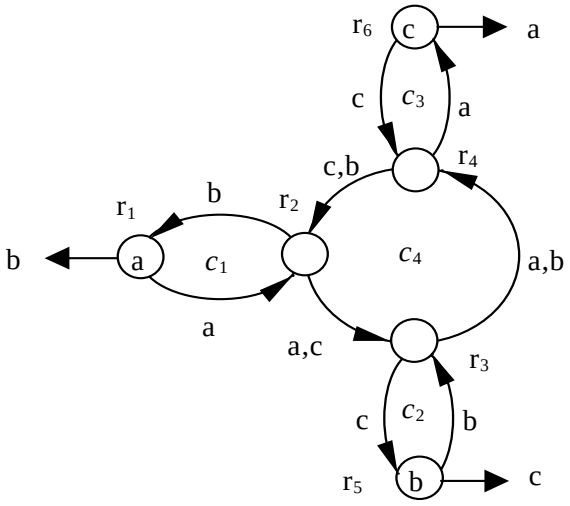

Figure 7: Manufacturing system for example 3, Case 1

The space of all closed paths in Figure 7 is greater than zero except for the closed path which contains the entire system, $\text{space}(c_1 \cup c_2 \cup c_3 \cup c_4, n_0) = 0$. Even though the space is zero the system is live. Now consider a WRG with the same structure except for a different system state and part routings.

Case 2: Suppose the system in Figure 8 has the process plans as depicted in Table 7.

Table 7: Process plans for example 3

Part	Process Plan
a	$r_1 r_2 r_3 r_5$
b	$r_5 r_3 r_4 r_6$
c	$r_6 r_4 r_2 r_1$

Assume that the system is in state $n = [n(a), n(b), n(c)] = [1,1,1]$.

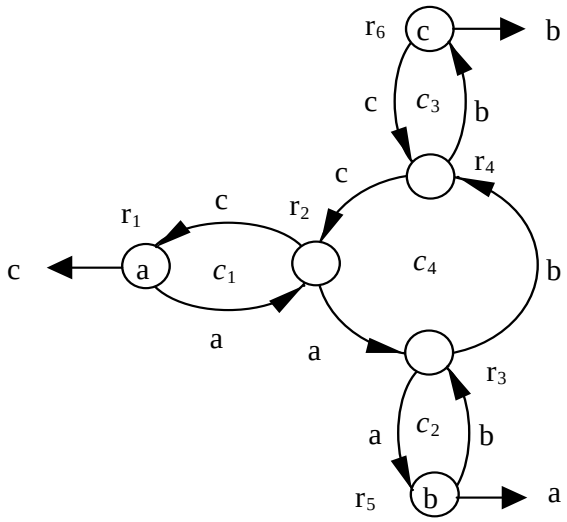

Figure 8: Manufacturing system for example 3, Case 2

As in case 1, the space of all closed paths in Figure 8 is greater than zero, except for the closed path that contains the entire system, $\text{space}(c_1 \cup c_2 \cup c_3 \cup c_4, n_0) = 0$. In this case, the system is dead. The space condition cannot distinguish between the two cases.

The Deadlock Avoidance Algorithm

An algorithm that implements the methods presented herein can be applied to any process control systems to avoid deadlock. The algorithm insures propagating an enabled part would not transition a live system to a dead state. The algorithm can be implemented in two different levels. A first level implementation, which is less restrictive, would be to define the order of all knots to as one. A second level of implementation would compute the order of each knot per Definition 5. Implementing the algorithm at this second level would allow more live states but would add more complexity. A flowchart of these two implementations is depicted in Figure 9.

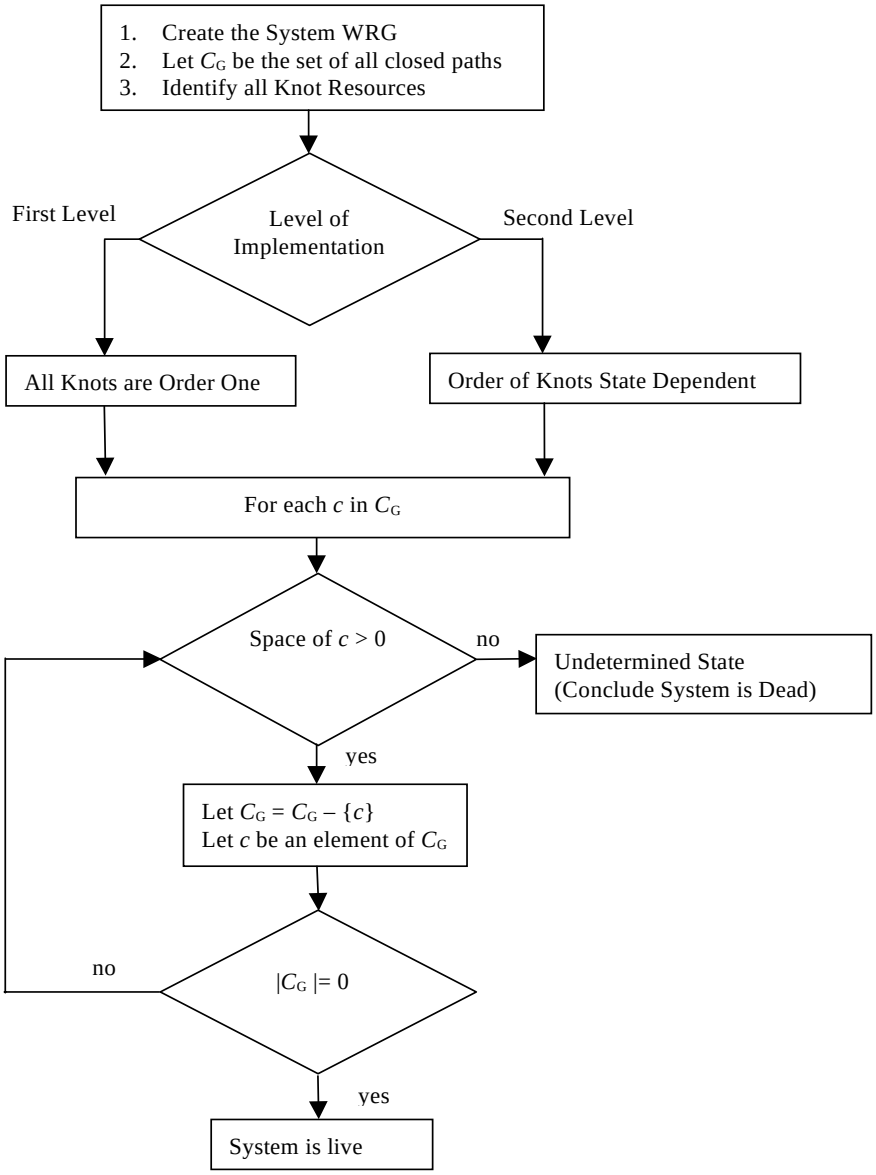

Figure 9: First and second level implementation flowchart

Conclusion

A deadlock avoidance algorithm was developed that avoids both primary and impending deadlock in a Flexible Manufacturing Systems. The concepts of slack, knot, order and space was derived from circuit observations and interactions using Wait Relation Graphs. The algorithm insures deadlock is avoided by not allowing a live system to enter dead states by satisfying a set of linear inequalities, *space* > 0 for all closed paths.

The algorithm does detect all dead states. The algorithm does not detect all live states as shown in Example 3. A special state called the evaluation state presented in [16] and [18] is necessary to determine the liveness of these indistinguishable states. This will be addressed in future publications.

References

[1] Banaszak, Z. and B. Krogh. 1990, “Deadlock Avoidance in Flexible Manufacturing Systems with Concurrently Competing Process Flows.” IEEE Trans. on Robotics and Auto. , vol. 6, no. 6, pp. 724-733.

[2] Barkaoui, K. and I.B. Abdallah 1995, "A Deadlock Method for a Class of FMS," Proceedings of the 1995 IEEE Int. Conf. On Systems, Man and Cybernetics, 1995, pp. 4119-4124.

[3] Cho, H., T.K. Kumaran, and R. Wysk 1995, "Graph-Theoretic Deadlock Detection and Resolution for Flexible Manufacturing Systems," IEEE Trans. on Robotics and Auto., vol. 11, no. 3, pp. 550-527.

[4] Fanti, M., G. Maione and B. Turchiano 1996, "Deadlock detection and recovery in flexible production systems with multiple capacity resources," Industrial Applications in Power Systems Computer Science and Telecommunications Procedings of the Mediterranean Electrotechnical Conference, vol. 1, pp. 237-241.

[5] Hsieh, F. and S. Chang 1994, "Dispatching-driven deadlock avoidance controller synthesis for flexible manufacturing systems," IEEE Trans. Robotics and Auto., vol. 10, no. 2, pp. 196-209.

[6] Judd, R. P. and T. Faiz 1995, "Deadlock Detection and Avoidance for a Class of Manufacturing Systems," Proceedings of the 1995 American Control Conference, pp. 3637-3641.

[7] Judd, R. P., P. Deering, and R. Lipset 1997, "Deadlock Detection in Simulation of Manufacturing Systems," Proceedings of the 1997 Summer Computer Simulation Conference, pp. 317-322.

[8] Lipset, R., P. Deering, and R. P. Judd 1997, "Necessary and Sufficient Conditions for Deadlock in Manufacturing Systems," Proceedings of the 1997 American Control Conference, vol. 2, pp. 1022-1026.

[9] Lipset, R., P. Deering, and R. P. Judd 1998, "A Stack-Based Algorithm for Deadlock Avoidance in Flexible Manufacturing Systems," Proceedings of the 1998 American Control Conference.

[10] Viswanadham, N., Y. Narahari, and T. Johnson. 1990. “Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturing Systems Using Petri Net Models,” IEEE Trans. on Robotics and Auto., vol. 6, no. 6, pp. 713-723.

[11] Zhou, M. and F. DiCesare 1992, “Parallel and Sequential Mutual Exclusion for Petri Net Modeling of Manufacturing Systems with Shared Resources.” IEEE Trans. on Robotics and Auto., vol. 7, no. 4, pp. 550-527.

[12] Zhou, M. 1996, "Generalizing parallel and sequential mutual exclusions for Petri net synthesis of manufacturing systems," IEEE Symposium on Emerging Technologies & Factory Automation, vol. 1, 1996, pp 49-55.

[13] Wysk R., Yang, N. and Joshi, S., "Detection of Deadlocks in Flexible Manufacturing Systems," IEEE Transactions Robotics and Automation, Vol. 7, No. 6, pp. 853-858, 1991
[14] Ezpeleta, J., Colom, J., Martinez, J., "A Petri Net Based Deadlock Prevention policy for Flexible Manufacturing Systems," IEEE Trans. on Robotics and Automation, Vol. 11, No. 2, pp. 173-184, April 1995.
[15] Wenle Zhang, Robert P. Judd and Paul Deering, "Evaluating Order Of Circuits For Deadlock Avoidance In A Flexible Manufacturing System", Proceedings of the 2003 American Control Conference, pp. 3679-3683, June 2003, Denver.
[16] Deering, E Paul "Necessary and Sufficient Conditions for Deadlock in Manufacturing Systems", PhD Dissertation, 2000, Ohio University
[17] Fanti, M.P., Maione, B., Mascolo S., and Turchiano, B., "Control Polices Conciliating Deadlock Avoidance and Flexibility in FMS Resource Allocation", IEEE Symposium on Emerging Technologies & Factory Automation., Vol. 1, IEEE, Piscataway, NJ, USA., pp. 343-351, 1995.
[18] Wenle Zhang, Robert P. Judd and Paul Deering,"Necessary and Sufficient Conditions for Deadlocks In Flexible Manufacturing Systems Based On A Digraph Model." 2004 Asian Journal of Controls, Vol 6, No 2, pp. 217-228
[19] Wenle Zhang and Robert P. Judd,"Evaluating Order Of Circuits For Deadlock Avoidance In A Flexible Manufacturing System." 2007 Asian Journal of Controls, Vol 9, No. 2, pp. 111-120

Biography

PAUL DEERING is currently Assistant Professor in the Industrial Technology Department in the Russ College of Engineering and Technology at Ohio University. He has worked in the area of Information Technology for 20+ years and has taught many engineering and computer science courses for the Russ College.