Sufficient Conditions for a Flexible Manufacturing System to be Deadlocked

Paul E. Deering, PhD
Department of Engineering Technology and Management
Ohio University
deering@ohio.edu

Abstract

In recent years, researchers have been interested in scheduling algorithms to avoid deadlock in Flexible Manufacturing Systems (FMS). FMS are discrete event systems characterized by the availability of resources to produce a set of products. Raw parts, which belong to various product types, enter the system at discrete times and are processed concurrently while sharing a limited number of resources. In such systems, a situation may occur in which parts become permanently block. This is called deadlock. This paper presents the sufficient conditions for deadlock to exist in a FMS; it models a FMS using digraphs to calculate slack, knot, order and space; it identifies three types of circuits that are fundamental in determining if a FMS is in deadlock.

Introduction

Moving the wrong part in a manufacturing system could place the live (deadlock-free) system into a deadlocked state or dead state. The only recourse would be to manually resolve the deadlock and reset the FMS to a live state. Clearly, avoiding deadlock altogether would lead to increased production and decreased labor costs. To prevent manual deadlock resolution a Deadlock Avoidance Algorithm (DAA) was developed in [1]. The DAA did not allow the system to enter any dead states and proved sufficient conditions for the system to be live. The DAA introduced the idea of space. If *space > 0* of all closed paths in the manufacturing then deadlock would be avoided. The only problem was that some live states were detected dead states. See figure 1. The DAA in [1] only proved sufficient conditions for a system to be live.

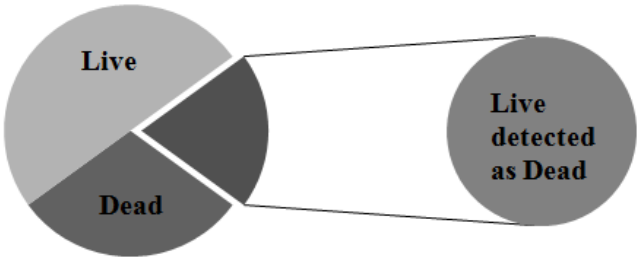

Figure 1: Live States detected as Dead

This paper will prove sufficient condition for the manufacturing system to be dead and is the partial results of reference [2]. 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 parameter slack, knot order and space is then defined; the next section introduces three types of circuit uses to proves sufficient conditions for a manufacturing systems to be dead.

Related Research

Many researchers use Petri nets [3-9] as a formalism to describe deadlock in a manufacturing system. Banaszak and Krogh [3] proposed a deadlock avoidance algorithm (DAA), which developed a restriction policy based on production route information to guarantee that no circular wait situations would occur. Their DAA is sufficient for avoiding deadlocks but is not an optimal solution. Viswanadham, Narahari, and Johnson [6] developed a deadlock avoidance algorithm that employed a look-ahead policy. This algorithm did not detect all deadlocked states, and the authors suggested using a recovery mechanism in case of system deadlock. Zhou and DiCesare [7] and Zhou [8] 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 that contain few resources but become very complicated for larger systems.

Another formalism to describe the manufacturing system is to use graphs [10-20]. In this approach, the vertices represent resources and the edges represent part flows between resources. Wysk, Joshi, and Yang [16] were the first to develop a specialized directed graphical structure called a wait relation graph (WRG) to model a manufacturing system. In reference [16], they developed a string manipulation procedure that yields a set of control actions to detect and recover from primary deadlock. Cho, Kumaran, and Wysk [10] 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 references [16] and [10] could only handle single capacity resources. Fanti, Maione, and Turchiano [11] 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, but it suggested a suitable recovery strategy.

Judd and Faiz [12] expanded on the original formulation proposed by reference [16] 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 [14] extended reference [12] to precisely quantify 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 defined the concept of order reduction. The approach was to put the system into an evaluation state and then compute the order. Deering [2] and [1] improved reference [14] by further refining the order of a knot and evaluation state, as well as eliminating the need for order reduction. Zhang, Judd, and Deering [17] developed a deadlock avoidance algorithm (DAA) based on references [14] and [2], which avoided deadlock and was executed in polynomial time. Zhang, Judd, and Deering [19] expanded upon references [17] and [2] 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 [20] extended reference [19] to allow choice in process flow or flexible part routing.

Modeling a Manufacturing System

An FMS consists of a set, *R,* of finite resources, such as 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 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 an 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 elements.

The function $\mathrm{n}(q)$ returns a positive integer that represents the position in plan[class(q)] of the operation that is currently processing q. When a new part q is added to the system, then $\mathrm{n}(q)=1$. As the part is moved from resource to resource according to its plan, $\mathrm{n}(q)$ 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 $\mathrm{tail}(a)$ returns the resource at the tail of the given arc; the function $\mathrm{head}(a)$ returns the resource at the head of the arc. A unit of the resource $r=\mathrm{tail}(a)$ is said to be committed to arc a if it is processing a part q whose next resource in its process plan is $\mathrm{head}(a)$. 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 that 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 $\mathrm{com}(a,n)$ 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:

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

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 $r_2=\mathrm{head}(a)$ is free and the part in the resource $r_1=\mathrm{tail}(a)$ 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 shift its terminal resource r_m in G.

Slack, Knot, Order, and Space

This section will summarize the major concepts and results from [2, 12, 14, 16]. This section defines the concept of slack, knot, order, and space.

The slack is the number of free resource 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 $\text{slack}(c, n) = 0$.

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 $c_1 \cup c_2$.

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.

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:

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

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 reference [2].

Sufficient Conditions for a System to be Dead

The previous section proved sufficient conditions for a manufacturing system to be live; that is, if the space of all closed paths in a manufacturing system is greater than zero, then the system is live. This section will prove sufficient conditions for a manufacturing system to be dead. Unfortunately, this cannot be proven in the general case, since there is insufficient information in the WRG to determine these conditions. However, when the system is in a special system state called an evaluation state, it can be shown that a manufacturing system is dead if one of the closed paths equals zero. The following example will demonstrate this more clearly.

Example 1: Let the WRG *G* in Figure 2 be in state n. Suppose that the process plans for parts *a*, *b* and *c* appear as presented in Table 1. Assume that the state of the system is $n = [n(a), n(b), n(b)] = [1, 1, 1]$. Table 2 depicts the order and space computations for this system.

Table 1 Process plans for example 1

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

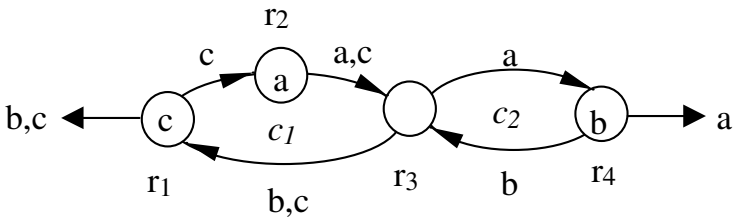

Figure 2 Manufacturing system for example 1

Table 2 Order and space computations for example 1

Circuit	Order	Space
c_1	0	1
c_2	0	1
$c_1 \cup c_2$	1	0

Since the space of the union between c_1 and c_2 is zero, the method previously presented in reference [1] cannot conclude whether the system is live or dead. This revised method will show that the order in which parts flow through order-one knots is required to describe sufficient conditions for a dead system. For example, knowing that parts *a* and *b* must pass through r_3 before any other parts can leave c_1 and c_2 and that the space of the union between c_1 and c_2 is zero, will allow researchers to know that the system in Figure 1 to be dead.

This section will contain three parts: the first section shows necessary and sufficient conditions which render basic closed path as dead; the next two sections show sufficient conditions for deadlock of chained and complex closed paths.

Basic Closed Paths

Definition 8: A basic closed path c is a closed path in a WRG *G* in state *n* such that $\text{order}(c,n) = 0$.

Theorem 1: Given a basic closed path $c = (R,A)$ in state *n*. If $\text{space}(c,n) = 0$ then *c* is dead.

Proof: See reference [2].

Theorem 1 allows us to conclude that space greater than zero of a basic closed path is necessary and sufficient for the system to be live. The next section addresses a particular closed path that contains order-one knots.

Chained Closed Paths

This section defines a chained closed path and introduces a special state called an evaluation state. A series of definitions, some lemmas and a theorem will prove that if a chained closed path is in an evaluation state and its space is equal to zero, then the chained closed path is dead.

Definition 9: A chained closed path c is a closed path containing one or more order-one knots with respect to *c*, such that *c* can be decomposed into a set of basic closed paths which intersect at only the order-one knots.

The following is a simple example of a chained closed path:

Example 2: Consider the manufacturing system in Figure 3. Assume that all *a* part types flow to the right from c_1 to c_3, and that all *b* part types flow to the left from c_3 to c_1. In this state, resources r_2 to r_3 are order-one knots. The manufacturing system can be decomposed into three simple circuits, c_1, c_2 and c_3. Let $c = c_1 \cup c_2 \cup c_3$. In this example, *c* is a chained closed path, since *c* can be decomposed into basic closed paths so that each circuit intersects each other at only the order-one knots (i.e. $c_1 \cap c_2 = r_2$ and $c_2 \cap c_3 = r_3$).

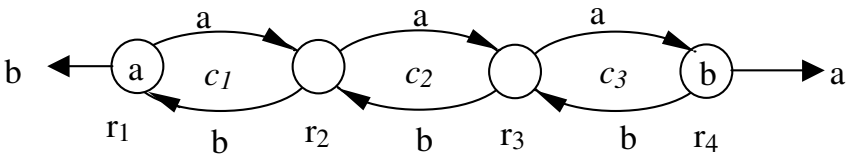

Figure 3 A chained closed path

The following example will help the reader conceptualize the need for an evaluation state:

Example 3. Suppose that a part exists in all the resources shown in Figure 4 except r_4. Each part is committed to the outgoing arc of its resource. Assume that all part a types must flow to circuit c_2 before completion and parts d_1 and d_2 must flow to circuit c_1 before completion. Call this state *n*. This state may, or may not, be dead, depending on the ultimate destination of part *b* in the resource r_7.

Case 1. Suppose part *b* must move to resource r_4 and then to r_5 and exit the system. Clearly, in this case, state *n* is a live state.

Case 2. Suppose part *b* must flow to r_4 and commit to circuit c_1. Then state *n* is a dead state.

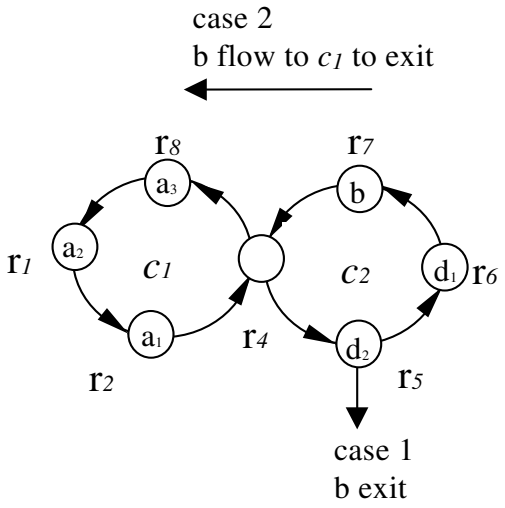

Figure 4 Manufacturing system for example 3

To distinguish and to evaluate these two cases, the dynamics of the part crossing through the knot should be analyzed more closely. Notice that in both cases, all part *a's* must cross knot r_4 before any other part on c_1 can leave c_1. But the part crossing dynamics are different on circuit c_2 in the two cases. Notice that in case 1, part *b* can leave circuit c_2 before part d_1 must cross knot *k*. In other words, a resource may become free on c_2 before part d_1 must cross the knot. In this state, we conclude that state *n* is not in an evaluation state. The method in the reference [2] cannot determine if deadlock exists by computing the space in state *n*. Notice that in Case 2, part *b* must cross knot r_4 before any other part can leave c_2. In this situation, no part can escape c_2 before the crossing must occur. The state of the system in case 2 is considered to be an evaluation state. These ideas motivated the following definitions.

Definition 10: Let c_1 and c_2 be two closed paths in a WRG *G* such that $c_1 \cap c_2 = k$ where *k* is an order-one knot. If a part *q* on c_1 propagates to *k* and commits to an arc on c_2, then *q* is said to *cross* knot *k*.

Definition 11: A basic closed path in a WRG *G* is always in an evaluation state.

Definition 12: An empty chained closed path in state *n* is in an evaluation state.

Definition 13: Let a non-empty chained closed path *c* be in state *n*. A chained closed path *c* can be divided into two closed paths, c_1 and c_2, at any order-one knot *k* such that $c_1 \cap c_2 = k$. Then chained closed path c is in an evaluation state if

1. all order-one knots are empty, and

2. for each order-one knot *k,* two parts, q_1 and q_2 exist, such that

   a. part q_1 must cross from c_1 to c_2 before any other part can leave c_1, and c_2 is in an evaluation state after the move; and

   b. part q_2 must cross from c_2 to c_1 before any other part can leave c_2, and c_1 is in an evaluation state after the move.

The system in Example 2 is in an evaluation state. Resources r_2 and r_3 are order-one knots. For order-one knot r_2, part *a* must cross from c_1 to $c_2 \cup c_3$ before any other part can leave c_1, and part *b* must cross $c_2 \cup c_3$ to c_1 before any other part can leave $c_2 \cup c_3$. For order-one knot r_3, part *a* must cross from $c_1 \cup c_2$ to c_3 before any other part can leave $c_1 \cup c_2$ and part *b* must cross c_3 to $c_1 \cup c_2$ before any other part can leave c_3. After moving either part *a* or part *b*, both $c_2 \cup c_3$ and $c_1 \cup c_2$ are in evaluation states.

The next lemma will show how the parts are committed when a chained closed path is in an evaluation state.

Lemma 1: Given a chained closed path $c = (R, A)$ in a WRG *G* that is in an evaluation state *n*, $space(c, n) = 0$ if, and only if, all order-one knots are empty in *c* and all other resources in *c* are filled and committed to resources on *c*.

Proof: See reference [2].

The next two lemmas are preliminary results that are required to prove the final theorem of this section.

Lemma 2: Given a chained closed path c that is in an evaluation state *n*, if $\text{space}(c, n) = 0$ then a part *q* exists such that when it is moved, it will fill an order-one knot and commit an outgoing arc of that knot on *c*.

Proof: See reference [2]

Lemma 3: Given a non-empty chained closed path c that is in an evaluation state n_0, if $\text{space}(c, n_0) = 0$ then propagating any part will create a chained closed path c_2 such that $c_2 \subset c$ and $\text{space}(c_2, n_1) = 0$.

Proof: See reference [2]

Definition 14: If any subgraph in a WRG *G* is dead, then *G* is dead.

Theorem 2: Given a non-empty chained closed path c that is in an evaluation state n_0, if $\text{space}(c, n_0) = 0$, then c is dead.

Proof: See reference [2]

Complex Closed Paths

Closed paths that are not basic closed paths or chained closed paths are classified as complex closed paths. This section will introduce complex closed paths. It will also be shown, if a complex closed is in an evaluation state and it contains a path with space equal to zero, then this is sufficient for determining if the system is dead.

We will first define a complex closed path and its various components, then follow these definitions with an example.

Definition 15: A complex closed path is a closed path that contains one or more order one knots that is not a chained closed path.

Definition 16: A complex path can be decomposed into two paths, one being a chained closed path and the other is called the auxiliary closed path. The intersection of the auxiliary closed path intersects and the chained closed path must contain one or more order one knots of the chained path.

Definition 17: A bypass path is the portion of the auxiliary path that does not intersect the chained closed path.

Definition 18: The first arc on the bypass path is a bypass arc.

Consider the following example.

Example 4: Suppose that the system in Figure 5 has the following parts and process plans as depicted in Table 3.

Table 3 Process plans for example 4

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

Assume that the system is in state $n = [n(a_1), n(a_2), n(b), n(d_1), n(d_2)] = [3,1,1,3,2]$.

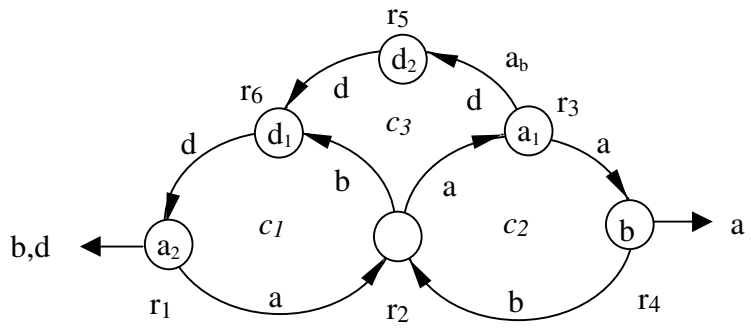

Figure 5 Complex closed path for example 4

The system consists of three simple closed paths: $c_1 = r_1 r_2 r_6 r_1$, $c_2 = r_2 r_3 r_4 r_2$, and $c_3 = r_2 r_3 r_5 r_6 r_1 r_2$. The order$(r_2, c_1 \cup c_2, n) = 1$. Clearly, the manufacturing system in Figure 5 is not a basic closed path. The system cannot be a chained closed path either since $(c_1 \cup c_2) \cap c_3$ is not a knot. According to Definition 15, the system in Figure 5 is a complex closed path. The complex closed path can be decomposed into a chained closed path, (i.e., $c_1 \cup c_2$, and an auxiliary closed (i.e., c_3). Closed path c_3 is an auxiliary closed path since the intersection of c_3 and the chained closed path $c_1 \cup c_2$ contain the order-one knot r_3. The simple path $r_3 r_5 r_6$ is a bypass path that joins c_1 and c_2 together. Arc a_b on resource r_3 is a bypass arc since it leaves r_3 along the auxiliary closed path c_3.

We next define the evaluation state for a complex closed path.

Definition 19: A complex closed path in a WRG *G* is in an evaluation state if its bypass arcs are *not* committed.

Definition 20: An empty subgraph that is a complex closed path in a WRG *G* in state *n* is in an evaluation state.

Definition 21: A WRG *G* is in an evaluation state if all closed paths in *G* are in an evaluation state.

The system in Example 4 is in an evaluation state. This is because part a_1 in resource r_3 is not committed to the bypass arc a_b. The chained closed path $c_1 \cup c_2$ is in as evaluation state per Definition 13. The space of the chained closed path $c_1 \cup c_2$ is zero. Clearly, the system is dead. The next theorem proves this concept in general.

Theorem 3*:* Given a complex closed path c_p that is in an evaluation state n_0, if any chained closed path $c^* \subset c_p$ has $space(c^*, n_0) = 0$, then c_p is *dead*.

Proof: See reference [2].

Conclusion

Three types of closed paths were identified to prove sufficient conditions for a manufacturing system to be dead. A special state called an evaluation state was introduced. It was showed that if a basic, chained or complex closed path that is in an evaluation state with *space=0* then the system is dead. Unfortunately, determining if a closed path is in an evaluation state is a problem. The problem is there is insufficient information in the WRG to determine if a system is in an evaluation state. This is a topic of future research.

References

[1] Deering, E. P. (2008), "A Simple Deadlock Avoidance Algorithm in Flexible Manufacturing Systems," International Journal of Modern Engineering, vol. 9, no 1, pp. 19-26.

[2] Deering, E. P. (2000), Necessary and Sufficient Conditions for Deadlock in Manufacturing Systems, PhD Dissertation, Ohio University.

[3] 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.

[4] 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, pp. 4119–4124.

[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] 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.

[7] 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.

[8] Zhou, M. (1996), "Generalizing Parallel and Sequential Mutual Exclusions for Petri Net Synthesis of Manufacturing Systems," IEEE Symposium on Emerging Technologies and Factory Automation, vol. 1, pp. 49–55.

[9] Ezpeleta, J., J. Colom, and J. Martinez (1995), "A Petri Net Based Deadlock Prevention Policy for Flexible Manufacturing Systems," IEEE Trans. on Robotics and Automation, vol. 11, no. 2, pp. 173–184.

[10] 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.

[11] 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 Proceedings of the Mediterranean Electrotechnical Conference, vol. 1, pp. 237–241.
[12] 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.
[13] 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.
[14] 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.
[15] 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.
[16] Wysk R., N. Yang, and S. Joshi (1991), "Detection of Deadlocks in Flexible Manufacturing Systems," IEEE Transactions Robotics and Automation, vol. 7, no. 6, pp. 853–858.
[17] Wenle, Z., R. P. Judd, and P. Deering (2003), "Evaluating Order of Circuits for Deadlock Avoidance in a Flexible Manufacturing System," Proceedings of the 2003 American Control Conference, pp. 3679–3683.
[18] Fanti, M.P., B. Maione, S. Mascolo, and B. Turchiano (1995), "Control Polices Conciliating Deadlock Avoidance and Flexibility in FMS Resource Allocation," IEEE Symposium on Emerging Technologies and Factory Automation, vol. 1, pp. 343–351.
[19] Wenle, Z., R. P. Judd, and P. Deering (2004), "Necessary and Sufficient Conditions for Deadlocks In Flexible Manufacturing Systems Based On A Digraph Model," Asian Journal of Controls, vol. 6, no 2, pp. 217–228.
[20] Wenle, Z. and R. P. Judd (2007), "Evaluating Order Of Circuits For Deadlock Avoidance in a Flexible Manufacturing System," Asian Journal of Controls, vol. 9, no. 2, pp. 111–120.

Biography

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