Journal of the Korean Institute
of Industrial Engineers
Vol.20, No.1, March., 1994.

FMS에서의 Deadlock 탐지와 방지에 관한 연구

임동순

한남대학교 산업공학과

Abstract

Deadlock in flexible manufacturing systems (FMS) which refers to the stop state of job flow in the system can be commonly occurred in operating FMS. This state, mainly due to bad movements of jobs and complicated job routings, should be avoided to maximize the utilization of high-capital resources. In this study, the deadlock generated from the conflict between flow objects competing to occupy space resources in FMS is investigated. Capacity Designated Directed Graph (CDG) is constructed to represent the space resources and flow object routings. From the characteristics of CDG, an algorithm for the detection of the deadlock possibility is proposed. Finally, two deadlock avoidance rules are proposed and implemented in the control of an Automated Guided Vehicle system in an FMS.

1. 서 론

유연 생산 시스템(Flexible Manufacturing System; FMS)은 여러 종류의 작업물이 각자의 routing에 따라 동시에 가공되는 복잡한 시스템이다. 이러한 시스템을 운용하는데 있어서 어느 작업물도 이동될 수 없는 deadlock 현상은 쉽게 일어날 수 있다. Deadlock 이란 단지 기다림만으로 해결될 수 없는 시스템의 정지상태를 의미한다[7]. 예를 들어, 두 기계 MC1 과 MC2로 구성된 FMS가 있다고 하자. 현재 두 기계에는 작업이 끝난 작업물이 다음 작업으로 이동을 원하고 있다. 만약, MC1과 MC2에 있는 작업물들의 다음 작업을 위한 기계가 각각 MC2, MC1 이고, 한 로봇이 작업물들의 이동을 수행하고 있다면, 두 작업물들은 서로 이동될 수 없는 deadlock 상태에 있게된다. 이러한 현상은 효율적인 작업물의 흐름과 투자비용이 큰 설비의 이용 극대화를 위하여 효과적으로 방지되어야 한다.

FMS에서의 deadlock에 관한 연구들이 보고

된바 있다. 그러나, 컴퓨터 시스템에서의 deadlock 문제에 관해 많은 연구가 있었던 반면에, FMS에서의 deadlock은 아직도 많은 연구가 필요한 실정이다[1]. Egbelu 와 Tanchoco[5]는 자동반송차량(Automated Guided Vehicle; AGV) 시스템을 운용하는데 있어서 발생하는 작업장내의 deadlock 현상을 언급하였다. AGV 운행 규칙들을 비교, 평가하기 위한 그들의 연구에서 작업장내의 작업물이 이동될 수 없는 현상을 해결하기 위하여 자동간섭(automatic intervention)과 중앙버퍼(central buffer)가 이용되었다. 즉, deadlock 상태가 발생하면, 그 상태를 유발한 작업물을 중앙버퍼에 이동시킨다. 그리고, 중앙버퍼에 있는 작업물은 deadlock이 해결됐을 경우에만 다음 목적지로 이동될 수 있다. Okogbaa 와 Huang[7]은 중앙버퍼 와 유사한 개념인 공용버퍼(common buffer)를 이용한 job-dispatching 정책을 제안하였다. 이 정책하에서, 한 작업물이 이미 다른 작업물로 가득찬 설비를 요구할 경우 이 작업물은 공용버퍼로 이동된다. 그리고, 공용버퍼에 있는 모든 작업물들이 그들의 원래 목적지로 보내질때 까지 시스템에로의 새로운 작업물 유입은 금지된다. Sabuncouglu 와 Hommertzheim[8]은 시스템내에 많은 수의 작업물이 공장내의 작업물 이동량과 혼잡을 증가시켜, deadlock의 가능성을 증가시킬 수 있음을 보였다. 공장의 deadlock 현상을 줄이기 위하여 그들은 새로운 작업물이 시스템으로 과다하게 유입되는 것을 저지하는 Gate라고 불리는 job release 정책을 사용하였다.

FMS의 Petri net 모델은 deadlock의 탐지와 방지에 이용될 수 있다. Banaszak 과 Krogh [1]는 FMS의 Petri net 모델에 근거한 Deadlock Avoidance Algorithm(DAA)을 제안하였다. 그러나, DAA는 동일한 routing을 갖는 한 종류의 작업물들이 설비들을 공유하는데서 발생하는 deadlock의 방지에 기초한다. 비록, 그들이 여러 종류의 작업물이 있는 FMS에서의 DAA 유효성을 보였지만, 여러 종류의 작업물이 한 공용설비를 공유하는데서 발생하는 deadlock을 DAA는 방지할 수 없음을 보일 수 있다. 즉, 설비 1에서 기다리고 있는 작업물 종류 A가 설비 2를 원하고 있고, 설비 2에서 기다리고 있는 작업물 종류 B가 설비 1을 원한다고 할때, DAA 하에서는 deadlock이 발생한다. Viswanadham등[10]은 Petri net의 reachability 분석을 통한 deadlock 예방정책 과 look-ahead에 의한 방지를 제안하였다. 그러나, Petri nets의 모델링 능력과 분석력은 서로 상반된 관계에 있다. 즉, 복잡한 시스템을 표현한 petri net 모델의 reachability 분석은 불가능할 수 도 있다. 또한, 그들이 제안한 look-ahead에 의한 deadlock 방지는 엄밀한 의미로 deadlock을 없애지는 못한다. 그들의 연구에서 언급했듯이 look-ahead로 탐지할 수 없는 deadlock을 처리하기 위하여 별도의 deadlock 회복(recovery) 정책이 필요하다.

이 연구는 FMS에서 deadlock을 유발하는 요소 중 작업물, 공구, 차량등과 같은 시스템내의 흐름객체와 이 흐름객체가 필요로 하는 공간적인 자원(space resource)들을 고려하여, 흐름객체들이 공간적인 자원을 공유하는데서 발생하는 deadlock 현상을 대상으로 한다. FMS의 deadlock에 관한 연구들은 deadlock이 발생하였을 경우의 deadlock 회복 방안에 대

한 것들이 대부분이다. 본연구의 목적은 deadlock의 회복보다는 deadlock의 탐지, 예방을 위한 방안을 제시하는데 있다. 제시된 방안은 흐름객체와 공간적인 자원을 표현한 용량을 갖는 그래프 (Capacitated Designated Directed Graph, CDG)의 특성에 기초한다. 이 연구의 구성은 다음과 같다. 제 2장에서는 컴퓨터 시스템과 FMS에서의 deadlock 필요조건을 비교한다. 제 3장은 이 연구의 기본이 되는 CDG에 대한 특성과 이를 통한 deadlock 필요조건을 설명한다. CDG를 이용하여 deadlock을 탐지, 방지하는 알고리듬과 방안을 각각 4, 5장에서 설명한다. AGV 조정시스템에서 제안된 deadlock 방지 정책이 어떻게 이용될 수 있는지를 6장에서 보인다. 마지막으로 제 7장에서 결론을 맺는다.

2. 컴퓨터 시스템과 FMS에서의 deadlock 조건

컴퓨터 시스템에서의 deadlock 연구는 시스템내 process와 resource의 두가지 요소를 고려하여, 한 process가 여러 resource를 동시에 (또는 연속, 부가적으로) 필요로 한다는 상황을 설정한다. 본 연구에서는 AGV, 로봇등과 같이 컴퓨터에 의해 조정되는 수송설비가 작업물등을 옮길 때의 deadlock 현상을 대상으로 한다. 이를 위해 FMS 내에서의 많은 요소 중 (1) 흐름객체와 (2) 공간적인 자원을 고려한다. 흐름객체는 FMS 내에서 공간적인 이동을 하는 객체를 의미하며, 크게 능동적 객체 와 수동적 객체로 나뉠 수 있다. 수동적 객체는 작업물, 공구, 펠리트등과 같이 수송설비의 도움을 받아야만 움직일 수 있는 객체를 의미하며, 능동적 객체는 수동적 객체를 움직일 수 있는, 컴퓨터로 조정되는 수송설비들을 의미한다. FMS에서의 능동적 객체는 어느 수동적 객체의 이동요청이 있을 때 그 수동적 객체가 원하는 다음 공간적 자원으로 옮기며, 이는 다음 공간 자원에 충분한 공간적 여유가 있을 경우에 가능하다.

컴퓨터 시스템에서는 다음에 설명할 네가지 조건을 모두 만족할 경우 deadlock이 발생한다[3, 8]. 만약, 한조건이라도 만족치 않게 된다면 deadlock의 발생을 방지할 수 있다. 이러한 조건들이 FMS에서도 적용될 수 있는지를 서술한다.

2.1 Mutual exclusion

한 resource는 한 process에서만 사용될 수 있고, 동시에 여러 process에서 사용될 수 없을 경우를 의미한다. 컴퓨터 시스템의 경우, 이러한 mutual exclusion은 피할 수 없는 성질의 것이다[8]. 예를 들어, 한 CPU는 여러 process들을 동시에 처리할 수 없다. 단지 time sharing에 의해 여러 process들을 처리한다. 또한 file은 여러 process에서 내용수정이 없을 경우에만 공유가 가능하다.

FMS에서의 경우 마찬가지로 이러한 mutual exclusion은 피할 수 없다. 예를 들어, 한 turning machine은 한번에 하나의 작업물만 처리할 수 있고, 한 AGV는 동시에 서로 다른 위치에 있는 두개의 물체를 옮길 수 없다.

2.2 Hold and wait

컴퓨터 시스템에서는 한 process가 여러 resource를 동시에(또는 연속, 부가적으로) 필요로 한다는 상황을 설정한다. 이 경우에,

resource는 다른 process에서도 필요로 하는, 여러 process가 사용할 수 있는 성질의 것들이다. 예를 들어, 세개의 tape driver를 가지는 컴퓨터 시스템에서 현재 세개의 process가 있다고 하자. 각 process는 한개 씩의 tape driver를 사용하고 있다. 만약, 각 process들이 또 다른 tape driver들을 추가적으로 필요로 한다면, 세개의 process들은 deadlock 상태에 있게 된다. 각 process들은 다른 process들이 사용하고 있는 tape driver를 release할 때까지 기다릴 수 밖에 없고, 이런 기다림 상태(hold and wait)는 영원히 계속된다.

FMS에서 수동적 객체와 이 객체가 필요로 하는 공간적 자원을 고려하는 경우에도 이러한 hold and wait 현상이 발생할 수 있다. 한 공간을 차지하고 있는 작업물 A는 다음 공정으로의 이동요청을 하고 있고, 작업물의 이동을 담당하고 있는 로봇은 이미 작업물 B를 집어 작업물 A가 차지하고 있는 공간에 놓을려고 한다고 하자. 즉, 작업물 A는 로봇을 필요로 하나 이미 로봇은 작업물 B를 서비스하고 있고, 로봇은 작업물 B를 놓기 위해 공간을 필요로 하나 이미 작업물 A가 그 공간을 차지하고 있다. 결국 두 작업물은 공간적인 자원들을 갖고, 상대 작업물이 그들의 공간적 자원들을 release할 때까지 기다릴 수 밖에 없다. FMS에서 공간적 자원을 고려할 경우의 이러한 hold and wait 현상이 컴퓨터 시스템에서의 경우와 틀린점은, FMS에서는 흐름객체가 다른 공간적 자원을 차지하는 동시에 현재 갖고 있는 공간적 자원을 release한다는 점이다. 이러한 차이로 인하여 FMS에서 공간적 자원을 고려할 경우, hold and wait 현상은 다음에 설명할 circular wait 현상에 포함된다.

2.3 No preemption

어떤 resource를 갖고 있는 process는 그 process를 다 끝낼때에만 resource를 release 하는 경우를 의미한다. 일반적으로 FMS에서는 한 기계가 일을 할 때 그 일을 다 끝내야만 다른 일을 할 수 있다. 그러므로 FMS에서는 preemption을 허락하지 않는것이 보통이다.

2.4 Circular wait

기다림 상태에 있는 process들, 즉 p_0, p_1, p_2,..., p_n이 있다고 하자. p_0는 p_1이 갖고 있는 resource를 필요로 하고, p_1은 p_2가 갖고있는 resource를 필요로 하고, …, p_n은 p_0가 갖고 있는 resource를 필요로 하는 chain 현상이 있다고 하면, 이 상태는 deadlock이 되는 이른바 circular wait 상태가 발생한다. FMS에서 이러한 현상은 쉽게 발견될 수 있다. 즉, hold and wait에서 설명된 예가 이에 해당될 것이다.

결론적으로 FMS에서 흐름객체와 공간적 자원을 고려할 경우, deadlock이 되기 위한 필요 조건은 circular wait 상태이다. Mutual exclusion 과 no preemption은 피할 수 없는 성질의 것이고, hold and wait 현상은 circular wait 현상에 포함되기 때문이다.

3. 용량을 가지는 방향성 그래프 표현

이 연구에서는 FMS 내 한종류의 흐름객체에 대한 공간적 자원을 노드(node), 그리고, 흐름객체의 routing을 아크(arc)로 표현한 그래프(graph)를 통하여 deadlock 현상을 탐지,

예방한다. 어느 시점에서도 한 공간적 자원에 있을 수 있는 흐름객체의 수는 그 자원내의 용량에 의해 제한된다. 예를 들어 한 machining center에 있을 수 있는 작업물의 수는 center 내의 버퍼 크기, 작업물이 가공될 수 있는 기계 수등에 의해 결정된다. 이러한 점을 고려하여, 각 노드들은 정해진 용량을 갖는다. 수식적으로, 이 연구에서 사용되는 그래프는 다음의 형식에 따른다.

용량을 가지는 방향성 그래프(Capacity designated directed graph; CDG)는 $G=(N, A, M)$으로 표현된다. N은 노드 $n_1, n_2, \cdots, n_k$ 들의 집합이고, 노드 n_i는 용량 $C(n_i)$를 갖는다. A는 노드들을 잇는 방향성 아크들의 집합이다. 노드 n_i에서 n_j로 잇는 아크가 존재할 경우 아크 $a(n_i, n_j)$의 값은 1이고, 아크가 없을 경우 0이다. M은 노드의 marking을 의미하며, n_i에 있는 흐름객체의 수를 $M(n_i)$로 표현한다. 노드 n_i에 있는 객체가 다른 노드 n_j로 이동할 수 있는 조건은 (객체흐름 규칙이라 부른다.) 1) $a(n_i, n_j)=1$ 이고, 2) $C(n_j) > M(n_j)$ 이다. 한 객체가 n_i에서 n_j로 이동한 후에 $M(n_i)$값은 하나 감소하고, $M(n_j)$는 하나 증가한다. 이 그래프에서 객체의 이동은 특정한 아크를 요구한다고 가정한다. 즉, 한 노드의 출력아크(output arc)가 둘 이상 있을 때 이 노드에 있는 흐름객체는 특정한 한 아크를 통해 출력노드로 이동되기를 원한다. 그러므로, 위의 객체흐름 규칙에서 $a(n_i, n_j)$가 active한 (n_j에 있는 객체가 요구하는) 아크일 조건을 추가한다.

이 그래프는 다른형태의 그래프인 Marked Directed Graph[4], State Machine[2]과 유사한 형태를 가진다. 표 1은 이들 그래프들의 상이점을 보여준다. 만약, State Machine에서 노드들이 용량을 갖는다면, 본연구에서 사용하는 그래프와 State Machine은 서로 변환 가능하다. 그러므로, 용량을 갖는 State Machine에 대한 연구[6]는 CDG에서도 유효하다.

CDG에서의 deadlock은 어느 노드에 있는 흐름객체가 영원히 움직일 수 없을 때 발생한다. 노드 n_i에 있는 흐름객체가 다른 노드로 영원히 움직일 수 없는 경우는 CDG의 객체흐름 규칙에 의해 다음의 두조건중 하나만 만족하면 된다.

1) 모든 j에 대해서 $a(n_i, n_j)=0$,

2) active 한 아크 $a(n_i, n_j)=1$ 이고, 영원히 $C(n_j)=M(n_j)$.

조건 1)은 CDG가 출력아크를 갖지않는 노드를 포함할 때 발생한다. 출력아크를 갖지 않는 노드의 용량이 무한개일 때 이 노드로의 흐름객체 이동은 항상 허용된다. 앞으로 언급되는 CDG에서 출력아크를 갖지 않는 노드는 무한개의 용량을 갖는다고 가정한다. 이러한 노드를 허용하는 CDG에서 이 노드외의 다른 노드에 흐름객체가 없을 경우 객체흐름은 중지된다. 이러한 경우를 제외 시키기 위하여 어느 시점에도 출력아크를 갖는 노드들중 하나이상에 흐름객체가 존재한다고 가정한다. 조건 2)는 어느 노드에 있는 흐름객체의 수가 영원히 용량과 같을 때 발생하며 다음의 정리를 유도한다.

정리 1 : CDG $G=(N, A, M)$이 cycle 을 이루고, G 내의 흐름객체 수 $\sum_{n_i \in N} M(n_i)$와 용량의 합 $\sum_{n_i \in N} C(n_i)$이 같으면 deadlock이 발생한다.

표 1. CDG, Marked Directed Graph, State Machine의 비교

	CDG	Marked Directed Graph	State Machine
구성	노드, 아크 marking	노드(transition), palce 아크, marking	노드(place), trasition 아크, marking
특징	노드는 한 active한 출력아크를 갖는다.	place는 한 입력아크와 한 출력아크를 갖는다.	transition은 한 입력아크와 한 출력아크를 갖는다.
용량	유한의 용량을 갖는 노드	무한의 용량을 갖는 노드	무한의 용량을 갖는 노드

증명 : G가 cycle을 이루다고 하자. 즉, G의 노드들 n_1, n_2, … ,n_p가 cycle을 이룬다. cycle 내의 각 노드 n_i에 있는 흐름객체 수 M(n_i)가 C(n_i)와 같다고 하자. 객체흐름 규칙 2)에 의해 n_1에 있는 흐름객체는 n_2로 움직일 수 없고, n_2에 있는 객체는 n_3로 움직일 수 없고, …, n_p에 있는 객체는 n_1으로 움직일 수 없다. 즉, circular wait 현상으로 어느 객체도 영원히 움직일 수 없다.

정리 2 : CDG G=(N, A, M)의 subgraph G′=(N′, A′, M′)이 cycle 을 이루고, G′ 내의 객체 수 $\sum_{n_i \in N'} M(n_i)$와 용량의 합 $\sum_{n_i \in N'} C(n_i)$이 같으면, deadlock 발생 가능성이 존재한다.

증명 : G′이 cycle을 이루고, A′이 N′의 모든 출력아크를 포함한다면, 정리 1에 의해 deadlock 이 발생한다. 만약, A′이 N′의 출력아크중 일부분을 포함한다면, deadlock 이 발생하지 않을 수 있다. 즉, N′ 내의 노드 n_i의 active한 아크가 A′에 포함되지 않고, active한 아크의 출력노드 n_j에 대해 C(n_j) 〉 M(n_j) 이면 객체흐름은 가능하다. 그러나, active한 아크가 A′에 포함됐다면, 정리 1에 의해 deadlock이 발생한다.

정리 2에 의해 다음의 두 cycle 축소 규칙(cycle reduction rule)을 유도한다.

Cycle reduction rule 1 : G=(N, A, M)에서 circular wait 상태의 가능성을 허용치 않는다면, cycle을 이루는 subgraph G′=(N′, A′, M′)은 용량 $\sum_{n_j \in N'} C(n_j)$-1인 macro 노드로 축소될 수 있다.

Cycle reduction rule 2 : G=(N, A, M)에서 circular wait 상태의 가능성을 허용치 않고, 두 subgraph G′=(N′, A′, M′), G″=(N″, A″, M″)이 cycle 을 이루고, 일부 노드들을 공유하고 있다면, 이 두 cycle은 용량 $(\sum_{n_i \in N'} C(n_i)\text{-}1)+(\sum_{n_j \in N''} C(n_j)\text{-}1)-(\sum_{n_k \in N' \cap N''} C(n_k)$인 macro 노드로 축소될 수 있다.

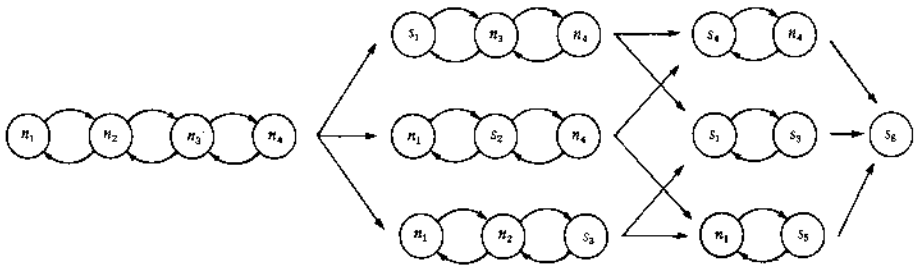
Cycle reduction rule 1은 capacity designated state machine(CSM)에서의 cycle reduction [6]과 유사하다. 다만, cycle을 이루는 subgraph 내의 deadlock 가능성을 허용치 않으므로 cycle 내의 총 객체수는 $\sum_{n_i \in N'} C(n_i)$-1을 초과하지 않는다. 그림 1은 cycle을 이루는 graph의 cycle reduction rule 1에 의한 축소를 보여준다. 모든 노드는 용량 1을 갖는다고 하

자. Circular wait 에 의한 deadlock을 허용치 않으므로 cycle을 이루는 두개의 노드가 축소된 macro 노드 s_1, s_2, s_3 의 용량은 각각 1이다. s_4 와 s_5는 macro 노드와 한 노드가 cycle을 이루어서 축소된 macro 노드로 용량은 각각 1이다. 결국 4개의 노드가 합쳐진 macro 노드 s_6도 용량이 1이 된다. 이는 circular wait 상태의 가능성을 허용치 않는한 어느 시점에도 이 4개의 노드에 있을 수 있는 흐름객체의 총 수는 1을 넘지 않아야 함을 의미한다. 즉, 두개 이상의 흐름객체가 이 그래프에 존재하면 deadlock의 가능성이 있다. 예를 들어 노드 n_1, n_4에 각각 한 흐름객체가 있다고 하자. n_1에 있는 객체의 다음 routing은 n_2, n_3, n_4 이고, n_4에 있는 객체의 routing은 n_3, n_2, n_1 이라면, n_1의 객체는 n_2로 옮길 수 있다. 그러나, 더이상 객체의 움직임은 불가능하다. n_2의 객체는 n_3로 움직일 수 없고, n_4의 객체는 n_3로 움직일 수 없다. 움직일 경우 circular wait 상태에 의한 deadlock이 발생하기 때문이다.

cycle reduction rule 2는 두개의 cycle이 합쳐져 새로운 한 macro 노드로 축소될때에 적용된다. 그림 2는 두개의 cycle s_1과 s_2가 결합되어 구성된 그래프에서의 축소를 보여준다. 각 노드의 용량이 1이라 가정하면, 각 cycle이 macro 노드로 축소될때 용량은 각각 2와 4이다. 이 두 cycle은 두개의 노드(n_3, n_4)를 공유하고 있으므로 cycle들이 합친 macro 노드 s_3의 용량은 4이다. 이 그래프에서 5개이상의 흐름객체가 존재할때 deadlock의 가능성은 자명하다. 그러나, 4개 이하의 흐름객체가 있고, s_1에 2개 이하의 흐름객체가 있을경우 deadlock의 가능성은 없다.

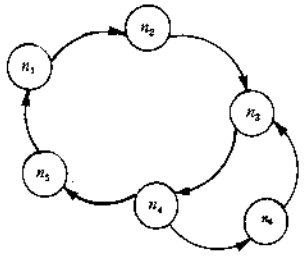
4. deadlock 탐지 와 deadlock 가능성 탐지

어느 시점에서 시스템이 deadlock 상태에 있는지의 조사는 중요한 의미를 지닌다. 뿐만 아니라 deadlock의 가능성을 조사 하는것도 중요하다. deadlock 가능성의 조사는 deadlock을 미리 방지하는데 유용하기 때문이다. 시스템의 상태를 CDG로 표현할때 어떻게 아크를 표현하느냐에 따라 deadlock의 조사 또는 deadlock 가능성의 조사가 결정된다. 시스템의 현상태가 deadlock 인지를 조사하기 위한 그래프는 흐름객체를 갖고 있는 공간적 자원을 노드로 나타내고, 노드의 출력아크는 현재 그 노드에 포함된 흐름객체의 다음 공간적자원(노드)으로 잇는다. 여기서의 흐름객체란 같은 공간적 자원을 공유하는 객체를 의미한다. 예를 들어, 작업물과 차량은 같은 종류의 공간적 자원을 공유하지 않는다. 작업물을 저장하는 버퍼에 차량을 저장할 수 없기 때문이다. 즉, CDG를 생성할때 한 종류의 흐름객체를 대상으로, 이들이 공용으로 사용할 수 있는 공간적 자원을 노드로 한다. 만약, 작업물의 흐름이 정지되는 deadlock을 대상으로 한다면, 흐름객체를 작업물로 하여, 이들이 차지하는 공간적자원을 노드로 표현하고, 각 노드에 있는 작업물의 다음 routing을 아크로 표현한다. 이 작업물에 대한 그래프에서 한노드에 있는 여러 작업물중 한 작업물만 이동요청을 할 수 있다면, 각노드의 출력아크 수는 최대 하나가 될 것이다. 만약, 공장내의 자동반송차량의 움직임이 정지되는 deadlock을 대상으로 한다면, 흐름객체를 차량으로 하여, 이들이 있을 수 있는 공간적자



macro node	node	capacity
S_1	n_1, n_2	1
S_2	n_2, n_3	1
S_3	n_3, n_4	1
S_4	n_1, n_2, n_3	1
S_5	n_2, n_3, n_4	1
S_6	n_1, n_2, n_3, n_4	1

그림 1. Cycle Reduction Rule 1의 적용



macro node	node	capacity
S_1	n_3, n_4, n_6	2
S_2	n_1, n_2, n_3, n_4, n_5	4
S_3	n_1, n_2, n_3, n_4, n_5, n_6	4

그림 2. Cycle Reduction Rule 2의 적용

원을 노드로 하고, 각 노드에 있는 차량의 다음 path를 아크로 표현할 수 있을 것이다. 이러한 그래프 G에서 cycle을 이루는 어느 subgraph가 앞에서 설명한 정리 1을 만족하면 deadlock이 발생한다.

본 논문은 시스템 운용중에 발생할지 모르는 deadlock의 방지를 위한 방안을 도출하는데 있다. 그러므로, deadlock의 가능성을 미리 탐지하는 방법은 deadlock의 탐지보다 더욱 중요한 의미를 가진다. deadlock의 가능성을 탐지하기 위한 그래프의 생성은 한종류의 흐름객체에 대한 모든 공간적 자원을 노드로 표현하고, 각 노드에서의 출력아크는 모든 발생가능한 흐름객체의 이동(즉, 현재 또는 미래에 이 노드에 있을 수 있는 모든 흐름객체의 routing)을 고려한다. 예를 들어, FMS 내의 작업물을 흐름객체로 고려한다면, 작업물의 공간적 자원인 machining center, buffer등을 노드로 표현하고, 모든 작업물 종류에 대한 routing을 아크로 표현한다. 이러한 그래

프를 이용한 deadlock 가능성의 탐지는 cycle reduction rule 1과 2에 의해 모든 macro 노드의 용량계산을 필요로 한다. 만약, 어느 macro 노드에 있는 객체의 수가 계산된 용량을 초과했다면 deadlock 가능성이 있다고 할 수 있다. 결국, deadlock 가능성의 탐지에서 주어진 그래프(marking 제외)의 모든 macro 노드와 이 노드들의 용량(circular wait 상태를 허용하지않는 경우에서)을 계산하는 것이 중요하다. 이는 부록에 설명된 알고리듬에 의한다.

5. deadlock 방지

FMS에서 한 종류의 흐름객체에 대한 모든 공간적 자원을 노드로 표현하고, 흐름객체의 가능한 routing을 아크로 표현한 그래프를 통해 deadlock의 방지가 가능하다. 이 그래프에서 deadlock을 회피하는 방법은 자명하다. 어떠한 경우에도 deadlock의 가능성을 없애는 것이다. 즉, 그래프의 어느 cycle에 있는 흐름객체의 수가 어떠한 경우에도 cycle 축소를 통하여 미리 구해진 용량을 초과하지 않도록 한다.

그래프 G를 한 종류의 흐름객체에 대한 deadlock 가능성의 탐지를 위한 CDG라 하자. s_i와 c_i를 앞장에서 설명된 알고리듬을 통하여 구한 macro 노드의 노드 집합과 용량이라 하고, k를 macro 노드의 수라 하자.

Deadlock 방지 규칙 1:

시스템내의 흐름객체 수는 항상 다음 조건을 만족하도록 한다.

시스템내의 흐름객체 수 $\leq \min(c_i, i=1, 2, \cdots, k)$

Deadlock 방지 규칙 1은 FMS 내의 흐름객체 수를 제한하는 정책이라고 할 수 있다. 만약, 흐름객체가 작업물이라면, 외부에서 FMS 내부로 들어오는 작업물을 제한하는 정책 이다. 이는 시스템내의 펠릿수를 미리 정하는 방안, 또는 loading 지역에서 deadlock 방지 규칙 1을 만족하는한 작업물을 load하는 방안등이 있을 수 있다. 만약, 흐름객체를 AGV라고 고려한다면, 시스템내의 AGV수를 미리 정할 수 있을 것이다. 이와 같이 deadlock 방지규칙 1에 의한 방법은 시스템내의 흐름객체수를 극도로 제한하여 설비의 이용율을 저하 시킬 여지가 있다. 시스템 운용중 한 종류의 흐름객체와 이를 위한 공간적 자원에 대한 실시간 정보를 갖고 있다면, deadlock 방지 규칙 1보다 설비 이용율을 향상 시킬 수 있는 deadlock 방지 규칙을 생각할 수 있다.

Deadlock 방지 규칙 2:

시스템 운용 중 이동을 원하는 흐름객체가 있을때 바로 다음 노드로의 이동이 수행됐다고 가정하고 각 노드의 $M(n_i)$를 계산한다. 이 $M(n_i)$들이 다음 조건을 만족하면 그 흐름객체의 이동을 수행하고, 만족치 않으면, 이동을 유보한다.

$$\sum_{n_i \in s_j} M(n_i) \leq c_j,\ j=1, 2, \cdots, k$$

어느 흐름객체가 시스템내의 다른 공간적 자원으로의 이동을 요청했다면, 이 객체의 이동여부는 deadlock 방지 규칙 2에 의해 결정될 수 있다. 즉, 이동요청한 객체가 원하는 공간적 자원으로 이동했다고 가정하여, 각 노드에서의 흐름객체수를 계산한다. 이는 실시

간 정보에 의해 현재 각 공간적 자원에 있는 객체의 수를 알면 가능할 것이다. 만약, 어느 macro 노드에 있는 흐름객체의 수가 미리 계산된 용량을 초과 한다면, deadlock의 가능성으로 인하여, 이동이 유보된다. 그렇지 않다면, 이동을 수행한다. Deadlock 방지 규칙 2에 의한 방안은 규칙 1에 의한 방법보다 시스템내의 흐름객체수를 제한하지 않으므로, 설비의 이용률을 증가 시킬 수 있다.

6. 예제: AGV 조정 시스템

둘이상의 차량(vehicle)을 운용하는 네트워크(network) 형태의 AGV 시스템에서 deadlock 상태를 방지하기 위한 AGV control 시스템의 설계를 예로 든다. 이러한 시스템에서 deadlock을 유발하는 주요소는 AGV path 상충(conflict)과 작업물 routing에 있다. 여러대의 AGV가 한 line을 동시에 차지하려고 한다면, AGV간의 상충이 발생한다. 만약, 이러한 상충이 해결되지 않았다면, AGV들의 충돌 또는 어느 AGV도 움직일 수 없는 deadlock 상태가 발생한다. 작업물들의 routing 또한 deadlock을 유발한다. 여러 종류의 작업물이 동시에 가공되는 FMS에서는 유연성을 갖기위하여 작업물 routing이 복잡해질 수 있다. 이러한 복잡성은 deadlock 상태를 쉽게 가져온다. 본 예에서는 흐름객체를 AGV와 작업물로 구분하여, AGV 이동에 의한 deadlock 과 작업물 이동에 의한 deadlock의 두가지 경우를 각각 고려한다.

그림 3의 FMS는 단방향으로 움직일 수 있는 네트워크 형태의 path와 여러대의 AGV를 갖는 AGV 시스템을 포함한다. AGV들은 한번에 하나의 load를 이동시키는 unit load vehicle들이다. pick-up point, delivery point, intersection 에서의 path 합병점과 같은 20개의 control point들이 AGV 네트워크상에서 정의된다. AGV들은 이 control point에 있을때 control 컴퓨터로 부터 명령을 하달 받는다. 이 FMS는 7개의 machine cell, 하나의 load station, 그리고, 하나의 unload station등 총 9개의 workstation을 포함한다. 각 machine cell은 한 기계, 한 입력버퍼, 한 출력버퍼를 갖고있다. 그러나 machine cell 6(MC6)과 machine cell 7(MC7)은 한개의 기계 대신 두개의 동일한 기계를 갖는다. workstation과 차량간의 interface는 입, 출력버퍼에서 이루어지며, 모든 버퍼는 하나의 작업물을 저장할 수 있는 용량 1을 가진다. 차량은 출력버퍼에 있는 작업물을 pickup 하고, 이 작업물들은 다른 workstation의 입력버퍼에 배송된다. 표 2는 이 시스템에서 가공되는 작업물의 종류와 routing을 나타낸다.

6.1 Zone control logic 하에서 차량들에 의한 deadlock 방지

여러대의 차량을 운용하는 AGV 시스템에서 차량들 간의 충돌을 방지 하기 위하여, AGV path의 네트워크을 서로 중첩되지 않는 zone으로 나눈다. zone control logic은 어느 시점에서도 한 zone에는 최대 하나의 차량이 있도록 한다. zone은 control point들에 의해 나뉘어 지고, 차량은 control point에서 정지, 이동, 속도변환, path 선택, pickup, delivery등의 명령을 control 컴퓨터로 부터 받는다. 비록, 이러한 zone control logic이 차량간의 충돌과 상충을 해결할 수 있지만, 차량들의

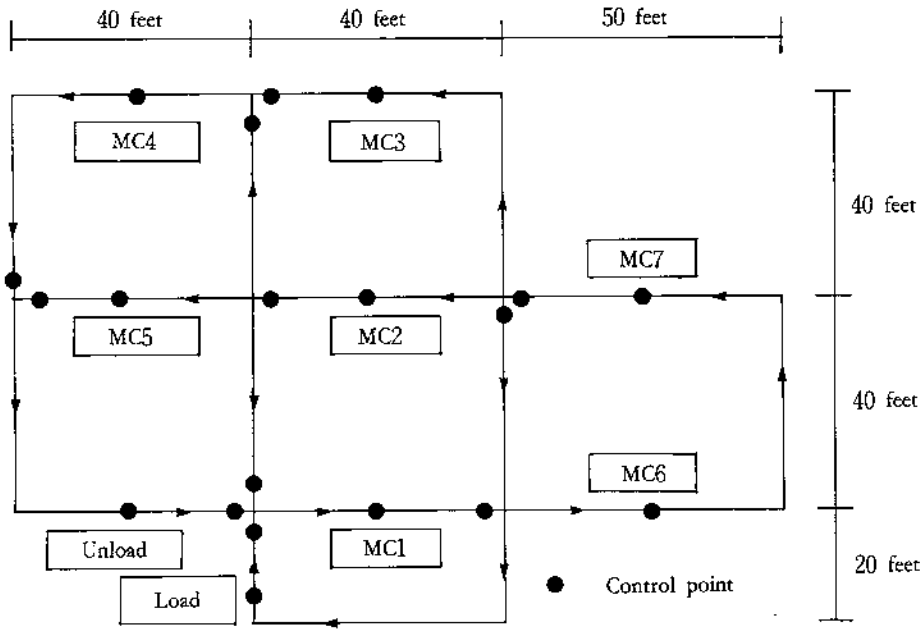


그림 3. AGC system을 갖는 FMS

표 2. FMS에서의 part routing

part type	routing[a]
1	1, 2, 3, 2, 3, 4, 5, 9
2	1, 5, 6, 7, 8, 6, 8, 9
3	1, 2, 3, 4, 3, 5, 7, 9
4	1, 2, 3, 4, 3, 5, 7, 9
5	1, 2, 3, 5, 3, 5, 7, 9
6	1, 5, 6, 5, 6, 7, 8, 9
	1, 5, 6, 8, 5, 6, 8, 9

a) routing 번호
1 : Load 2 : MCl 3 : MC2 4 : MC3 5 : MC4 6 : MC5 7 : MC6 8 : MC7 9:Unload

circular wait 상태에 의한 deadlock을 해결하지는 않는다. 그림 4에 있는 예를 보자. Control point CP3에 있는 차량이 zone 4로 이동을 원하고 있고, control point CP1, CP2, CP4에 있는 차량들은 각각 다음 zone으로 이동하려고 할때 circular wait 상태가 발생한다. 즉, AGV 네트워크의 zone을 CDG로 표현한 그래프에서 marking된 노드들 z1, z2, z3, z4가 cycle을 이루어, 정리 1에 의해 deadlock이 발생한다. 이러한 deadlock을 없애기 위해서는 앞에서 제안한 deadlock 방지 규칙들을 이용할 수 있다. 규칙 1을 이용한 해결방안은 시스템내의 차량수가 이 CDG에서 어느 cycle의 용량보다 적도록, 네트워크를 많은 zone으로 나누거나, 또는 차량수를 제한하는 것이다. 즉, 예제 FMS의 AGV 네트워크를 나타내는 CDG(그림 5)에서, cycle 축소를 통한 macro 노드들의 수는 91이었다. 이 중 최소 용량을 갖는 macro 노드는 노드 8, 9, 19, 20으로 구성된 것으로 용량은 3이다. 그러므로, 차량의 수가 3이하이면 deadlock을 피할 수

있다. 또는, AGV에게 명령을 내릴때 control 시스템에서 deadlock 방지 규칙 2를 고려한 명령을 내린다. 현실적으로 deadlock 방지 규칙 1에 의한 방안이 실현가능하면, 이 방안은 규칙 2에 의한 방안 보다 더욱 바람직하다. AGV에 명령을 내릴때 마다 요구되는 deadlock 방지를 위한 특별한 절차를 피할 수 있기 때문이다.

는 process 설비를 노드로 표현하여, 그 설비에 있을 수 있는 작업물의 수를 용량으로 한다. 그리고, 모든 종류의 작업물에 대한 routing을 아크로 표현한다. 이 CDG에서 현재 각 process 설비에 있는 작업물의 수를 알 수 있다면, deadlock 가능성 탐지가 가능하다. 뿐만 아니라 deadlock 방지 규칙 1 또는 2를 AGV dispatching 시스템에 포함 시킴으로써

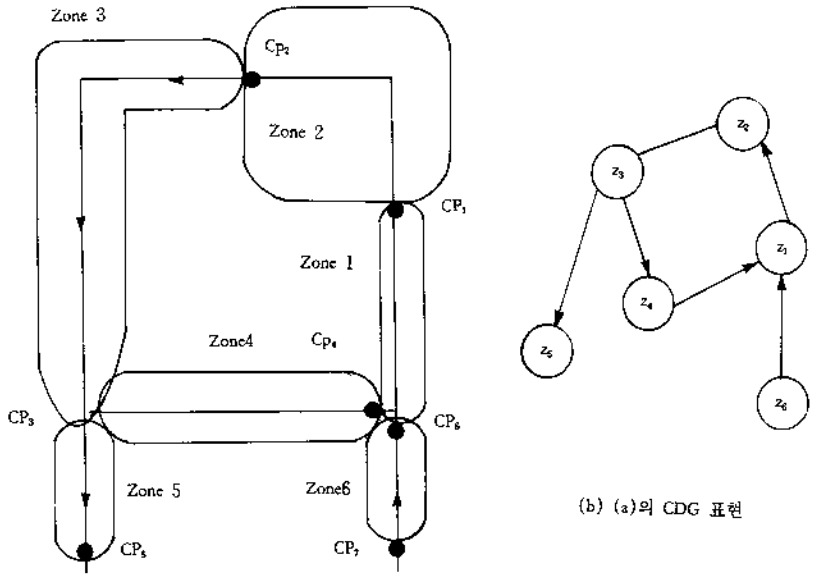


그림 4. AGV network 예

6.2 작업물 이동에 의한 deadlock 방지

FMS에서 작업물들의 이동이 circular wait 상태에 있으면, deadlock이 발생한다. 이러한 circular wait 상태는 확률적 특성을 가지고 있어, 시스템 운영전의 계획단계에서 deadlock의 발생을 정확하게 예측할 수 없다. 그러나, 시스템 운영단계에서 deadlock 가능성의 예측은 가능하다. 즉, 작업물의 routing에 나타나 deadlock의 방지가 가능하다. 앞에서 설명했듯이 시스템내의 재공품에 대한 실시간 정보를 가질 수 있다면, deadlock 방지 규칙 2가 설비 이용률 측면에서 더욱 바람직하다. 본 예제에서는 deadlock 방지규칙 2에 의한 방안을 설명한다.

AGV dispatching 시스템은 작업물의 이동에 차량을 할당하기 위한 두가지 기본적인

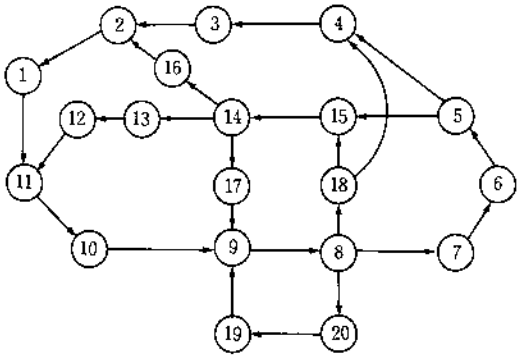


그림 5. AGV network(그림 3)의 CDG 표현

기능 (vehicle selection, workstation selection)으로 구성되어 있다. 한 workstation에 있는 작업물이 다른 workstation에로의 이동을 원하고 있다면, 여러개의 쉬고 있는(작업물 수송의 명령을 기다리고 있는) 차량중에서 하나의 차량을 선택하여 그 작업물을 이동시키기 위한 명령을 하달한다. 한 차량이 쉬고 있다면, 다른 workstation으로의 이동을 원하는 작업물을 가지고 있는 여러개의 workstation 중에서 한 workstation을 선택하여 작업물 이동 명령을 차량에게 하달한다. 이러한 기본 기능은 참고문헌 [2]에서 work-center-initiated rule 과 vehicle-initiated rule로 불린다. 작업물의 이동을 수행하기 위하여 이 두가지 기본기능들은 결합되어, 쉬고 있는 차량과 이동을 원하는 작업물의 현 workstation들에 우선순위를 할당하고, 이동이 가능하면 선택된 차량에게 선택된 작업물의 이동에 따른 명령을 하달한다. 이 할당 과업은 차량, 버퍼, 기계, 그리고, 작업물등의 시스템 상태가 변할 때 마다 수행된다.

작업물 이동에 의한 deadlock을 방지하기 위하여, deadlock 방지 규칙들은 workstation selection 기능에 포함되어야 한다. 즉, workstation selection에 의해 선택된 작업물의 이동을 고려할때 deadlock 방지 규칙에 의해 이동 여부를 결정한다. 만약, 이동이 허락된다면 이동명령을 AGV에 하달하고, 그렇지 않으면, 이동을 유보한다. 유보된 작업물의 이

표 3. 그림 6의 macro node 계산결과

Macro node	Combination	No. of nodes	Capacity	Nodes
s_0		2	5	2 3
s_1		2	5	3 4
s_2		2	5	3 5
s_3		2	5	5 6
s_4		2	6	6 8
s_5		3	7	3 4 5
s_6		3	8	5 6 8
s_7		3	10	5 7 8
s_8		3	9	6 7 8
s_9		4	11	5 6 7 8
s_{10}	s_0+s_1	3	7	2 3 4
s_{11}	s_0+s_2	3	7	2 3 5
s_{12}	s_0+s_5	4	9	2 3 4 5
s_{13}	s_2+s_3	3	7	3 5 6
s_{14}	s_2+s_6	4	10	3 5 6 8
s_{15}	s_2+s_7	4	12	3 5 7 8
s_{16}	s_2+s_9	5	13	3 5 6 7 8
s_{17}	s_3+s_5	4	9	3 4 5 6
s_{18}	s_5+s_6	5	12	3 4 5 6 8
s_{19}	s_5+s_7	5	14	3 4 5 7 8
s_{20}	s_5+s_9	6	15	3 4 5 6 7 8
s_{21}	$s_{11}+s_3$	4	9	2 3 5 6
s_{22}	$s_{11}+s_6$	5	12	2 3 5 6 8
s_{23}	$s_{11}+s_7$	5	14	2 3 5 7 8
s_{24}	$s_{11}+s_9$	6	15	2 3 5 6 7 8
s_{25}	$s_{12}+s_3$	5	11	2 3 4 5 6
s_{26}	$s_{12}+s_6$	6	14	2 3 4 5 6 8
s_{27}	$s_{12}+s_7$	6	16	2 3 4 5 7 8
s_{28}	$s_{12}+s_9$	7	17	2 3 4 5 6 7 8

동은 다음 의사결정 시점(시스템 상태가 변할때)에서 다시 이동여부가 결정된다. 그림 6은 예제 FMS에서 가공되는 작업물들의 routing 을 CDG로 표현한 것이다. 이 CDG에서 node 2, 3, 4, 5, 6 은 용량 3을 갖고, 7, 8은 용량 4를 갖는다. 노드 1과 9는 무한개의 용량을 갖는다고 가정한다. 표 3은 이 CDG를 macro 노드의 계산 알고리듬에 의해 구한 macro 노드와 용량을 나타낸다. Deadlock 방지 규칙2에 의한 AGV control은 표 3의 macro 노드와 용량을 이용하여, 각 macro 노드에 있을 작업물의 수를 조사, 어느 작업물의 이동여부를 결정한다.

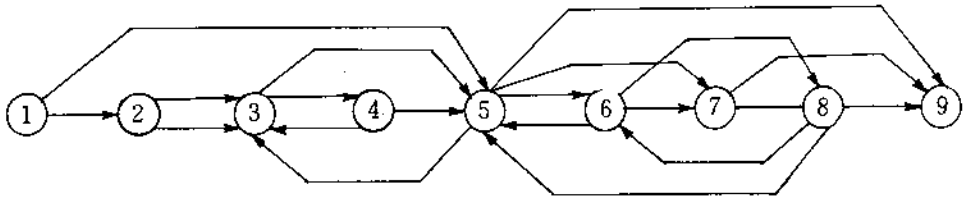


그림 6. Part routing(표 2)의 CDG 표현

7. 결론

본 연구에서는 FMS를 운용할때 발생하는 deadlock의 탐지 와 방지를 위한 방안을 제시하였다. 특히 FMS 내에서 작업물, 공구, 차량등과 같은 흐름객체들이 공간적 자원들을 사용하는데서 발생하는 deadlock을 대상으로 하였다. 제시된 방안의 기본적인 배경이 되는 것은 이들 흐름객체와 공간적 자원을 CDG로 표현한 것이다. CDG의 특성을 이용하여, 그래프상에서의 circular wait 상태의 가능성을 탐지하는 알고리듬을 제시 하였고, 두가지 deadlock 방지 규칙을 제시하였다. 이러한 방안들이 AGV 시스템을 운영하는 FMS에서 어떻게 이용될 수 있는지를 보였다.

제안된 두 deadlock 방지 규칙은 FMS 의 deadlock으로 부터 회복시키는 기존의 방안과 상이하다. 이들 방법 과의 성능비교 또는 결합에 대한 연구가 요구된다. 예를 들어, central 또는 common 버퍼를 이용한 deadlock 회복 방법과는 어떠한 성능차이가 있는지, 그리고, 시스템 성능을 증가시키기 위하여 이들 방법과 제시된 방안과의 결합은 가능한지를 연구할 필요가 있다. 이들 연구들은 앞으로의 과제에 포함되어야 할 것이다.

참 고 문 헌

[1] Banaszak, Z. A. and Krogh, B. H., "Deadlock Avoidance in Flexible Manufacturing Systems with Concurrently Competing Process Flows", *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 6, pp. 724-734, 1990.

[2] Best, E. and Thiagarajan, P. S., "Some Classes of Live and Safe Petri Nets", in *Concurrency and Nets: Advances in Petri Nets* (Edited by Hartman, J, Genrich, K. V., and Rosenberg, G.), Springer-Verlag;

Berlin, 1987.

[3] Coffman, Jr., E. G., Elphick, M. J., and Shoshani, A., "System Deadlocks", *Computing Surveys*, Vol. 3, No. 2, pp. 67-78, 1971.

[4] Commoner, F., Holt, A. W., Even S., and Pnueli A., "Marked Directed Graphs", *Journal of Computer and System Sciences*, Vol. 5, pp. 511-523, 1971.

[5] Egbelu, P. J. and Tanchoco, J. M. A., "Characterization of AGV Dispatching Rules", *International Journal of Productions Research*, Vol. 22, No. 3, pp. 359-374, 1984.

[6] Murata, Tomohiro and Komoda, Norihisa, "Liveness Analysis of Sequence Control Specifications Described in Capacity Designated Petri net using Reduction", *IEEE Conference on Robotics and Automation*, pp. 1960-1965, 1987.

[7] Okogbaa O. Geoffrey and Huang Jiansheng, "A Simulation Study of Deadlock Avoidance in FMS", *1st Industrial Engineering Research Conference*, pp. 197-201, 1992.

[8] Peterson, J. L and Silberschatz, A. "Operating System Concepts", Addison-Wesley: MA, 1985 (2nd ed.).

[9] Sabuncoglu, I. and Hommertzheim, D. L., "An Investigation of Machines and AGV Scheduling Rules in a FMS", *Proceedings of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems*, pp. 261-265, 1989.

[10] Viswanadham, N., Narahari, Y., and Johnson, T. L., "Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturing Systems Using Petri Net Models", *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 6, pp. 713-723, 1990.

부록 : Circular wait 상태를 허용치 않을 경우 macro 노드의 용량 계산을 위한 알고리듬

MACRO—NODE(N,A,C)

Input: CDG G=(N,A), 각 노드의 용량 $C(n_i)$

Output: macro 노드의 노드 집합 $s_i(i=1,2,\cdots,k)$ 와 용량 c_i

Process:

Step 1: 1.1 G에서 모든 cycle의 노드 집합 $s_i(i=1,2,\cdots,ns)$를 구하고, s_i 의 노드수 (m_i) 와 용량($c_i=m_i-1$)을 구한다.

1.2 만약 ns 〈= 1 이면 중지.

1.3 s_i들을 m_i 의 증가 순서로 정렬한다.

Step 2: 2.1 /* 한 s_i 가 다른 s_j 를 포함할 경우 c_i 재계산 */

for i=2 to ns

$s_t = s_i$; $c_t = c_i$

for j=1 to i-1

만약 $s_t \in s_j$ 이면

$s_t = s_t - s_j$; $c_t = c_t + c_j - \sum_{nj \in sj} C(n_j)$

end

$c_i = c_t$

end

k = ns; last = ns

2.2 /* s_i 와 s_j 가 일부 노드들을 공유할 경우 새로운 macro 노드 생성 */

for i=2 to ns

for j=1 to i-1

SHARE—MACRO(s_i,s_j,s_t,c_i,c_j,c_t)

만약 s_t가 NULL이 아니고 유일하다면

k = k + 1; $s_k = s_t$; $c_k = c_t$

end

end

만약, k=last 이면 중지

start = last; last = k

Step 3: /* 새로운 macro 노드와 기존의 cycle 들이 일부 노드들을 공유 할 경우 새로운 macro 노드 생성 */

for j=start+1 to last

for j=1 to ns

SHARE—MACRO(s_i,s_j,s_t,c_i,c_j,c_t)

만약, s_t 가 NULL이 아니고 유일하다면

k = k+1; $s_k = s_t$; $c_k = c_t$

end

end

만약, k=last 이면 중지.

start = last

last = k

Step 3를 반복한다.

SHARE—MACRO(s_1,s_2,s_3,c_1,c_2,c_3)

/* 두 cycle이 노드의 일부를 공유할 경우 새로운 macro 노드 생성 */

Input: 두 cycle의 노드 집합 (s_1,s_2) 과 각 cycle의 용량 (c_1,c_2)

Output: 새로운 macro 노드 s_3 와 용량 c_3

Process:

만약, s_1 과 s_2가 노드의 일부를 공유 한다면

$$s_3 = s_1 + s_2;\ c_3 = c_1 + c_2 - \sum_{n_i \in s_1 \cap s_2} C(n_i)$$

그렇지 않다면

$$s_3 = \text{NULL};\ c_3 = 0$$