한국시뮬레이션학회 논문지
Vol. 15, No. 1, pp. 1-10 (2006. 3)

DAPN과 인접행렬을 이용한 교착상태 회피에 대한 연구

송유진[1†] · 이종근[1]

The Study on the Deadlock Avoidance using the DAPN and the Adjacency Matrix

Yu-Jin Song · Jong-Kun Lee

ABSTRACT

The Flexible Management System (FMS) consists of parallel and concurrent machines, pieces of equipment, and carrying systems classified as buffers, tools, and routers, respectively. The concurrent flow of multiple productions in a system is competed with one another for resources and this resulting competition can cause a deadlock in FMS. Since a deadlock is a condition in which the excessive demand for the resources being used by others causes activities to stop, it is very important to detect and prevent a deadlock. Herein a new algorithm has been studied in order to detect and prevent deadlocks, after defining a relationship between the general places and resource share places in Petri nets like as DAPN: Deadlock Avoidance Petri Net. For presenting the results, the suggested algorithms were also adapted to the models that demonstrated FMS features.

Key words : Deadlock, DAPN(Deadlock Avoidance Petri Net), FMS(Flexible Manufacturing System), Adjacency Matrix, Algorithm

요 약

유연생산시스템은 많은 병렬적이고 동시 발생적인 기계들, 장치들, 수송 시스템들로 구성되며, 이러한 것들은 각각 버퍼들, 도구들, 라우터들로 나누어진다. 시스템에서 여러 생산품들의 병렬적인 흐름은 자원들을 이용하기 위해 서로 경쟁하게 되며, 유연생산시스템의 이러한 특징들로 인해 교착상태가 발생한다. 교착상태는 상대방이 점유하고 있는 자원을 서로 요구하는 과정에서 흐름이 멈추어진 상태를 말하므로, 이러한 교착상태의 탐지와 회피는 매우 중요한 문제이다. 이 사실에 근거하여 일반 플레이스와 자원공유플레이스 사이의 관계를 나타내는 DAPN(Deadlock Avoidance Petri Net)을 정의하여 교착상태를 탐지하고 회피하기 위한 새로운 알고리즘을 연구하였다. 또한 제안된 알고리즘을 유연생산시스템의 특징을 가진 모델에 적용하여 그 결과를 제시한다.

주요어 : 교착상태, 회피, 패트리넷, 유연생산시스템, 인접행렬, 알고리즘

1. 서 론

교착상태는 실시간으로 자원을 할당하는 시스템에서 매우 중요한 문제이다. 즉, 모든 프로세스들에 자원 할당의 순서가 완전하게 갖추어져 존재하느냐 아니냐의 문제는 그 시스템의 원활한 수행이 가능한지의 문제라고 볼 수 있다.

또한, 실시간 자원 할당은 자원을 다루는 시스템의 많은 형태에서 중요한 제어 책무를 가지고 있다. 자원 할당을 통해 시스템이 수행되는 많은 경우에서, 자원 프로세스는 시스템에 입력, 요구, 응답, 사용 그리고 요구된 자원들을 해제한 후 시스템에서 나가게 된다. 이 때 시스템의

* 본 연구는 과학기술부 목적기초연구(R08-2004-000-10029-0) 지원으로 수행되었음.

2005년 10월 18일 접수, 2006년 2월 28일 채택
[1] 창원대학교 컴퓨터정보통신공학부 컴퓨터공학과
주 저 자 : 송유진
교신저자 : 송유진
E-mail; syj@changwon.ac.kr

많은 형태들에서 교착상태에 걸리기 쉽다. 교착상태는 운영체제, 분산 데이터베이스, 자동 제어 시스템과 같은 많은 기술적인 분야에서 잘 알려진 문제로, 이러한 교착상태를 제어하기 위한 잘 알려진 정책들은 prevention, detection-resolution, avoidance이다. Prevention은 처음부터 프로세스들의 요구조건을 제한하여 교착상태를 불가능하게 하는 것으로 이것은 병렬 프로세스를 제한하기 때문에 지나치게 한정적이며, 자원사용이 불충분하다. Detection-resolution은 교착상태가 발생하는 것을 일단 허락하고, 그 다음에 마땅한 해결책을 찾는 방법이다. 이 방법은 자원할당 면에서 가장 큰 유연성을 가지고 있으나, 프로세스들의 중단, 시간 소비, 물리적인 장비들의 교체 등을 포함할 수도 있다. Avoidance는 프로세스 요구에 관한 지식과 함께 현재 상태 정보들을 사용하여 결국 교착상태가 일어나지 않도록 하는 방법이다. 이 방법은 할당 유연성 면에서 prevention보다는 유연성이 좋지만 detection-resolution 보다는 좋지 않은 중간 단계를 이룬다.[1,4,6,9-14,16-17] 따라서 avoidance는 교착상태에 관한 연구 분야에서 시스템 중단과 그 해결을 위한 비용 측면과 할당 유연성 측면을 모두 고려해 볼 때 효율적인 접근 방법이라 할 수 있다.

교착상태 확인과 방지를 위하여, 그래프 이론,[18] 스케쥴링개념 이용,[19] 로직 모델,[20] 오토마타론 모델[21]과 패트리 넷 모델[2,3,5,8,15]들을 활용하였다. 그 중에서 패트리 넷 활용 모델이 FMS 분석에 가장 널리 활용되어진 연구로 PN을 이용한 교착상태 분석이 활발하게 이루어졌다. 특히 Siphon을 이용한 모델과[2,5,15,22] unfolding 모델 분석[6] 등이 주목할 만한 성과를 이루었다. Siphon 모델의 경우 패트리 넷의 siphon 성질을 이용하여 siphon 그룹에서의 마킹분석을 통하여 교착 상태 확인과 회피 모델을 제시하고 있으므로, siphon 성질이 나타나는 부문을 발견하고 이에 대한 교착 여부를 분석하여야하며, 복잡한 시스템의 경우 분석 시간과 알고리즘이 복잡한 단점이 발생한다. 또한 unfolding의 경우 시스템 전체적인 마킹의 흐름을 표시하므로, 교착 상태의 발견은 손쉬운 반면, 교착상태 회피 흐름을 찾아내는 데는 복잡한 알고리즘을 생성하여야하는 문제점과 큰 시스템의 경우 상태 폭발의 경우가 발생 가능하므로 이에 대한 해결 방법도 같이 제시되어야 한다.

본 연구에서는, 고정된 방식의 생산 시스템보다는 작업내용을 쉽게 변경할 수 있는 유연성을 가진 생산 시스템의 필요성이 증가하고 있는 시점에서, 효율적인 자원 할당이 요구되는 유연생산 시스템(FMS : Flexible Manufacturing System)의 상태를 *DAPN*(Deadlock Avoidance Petri Net)과 인접 행렬(Adjacency Matrix)을 이용하여 큰 시스템을 작업 단위별 Subnet으로 분리하여 이를 같은 차수의 행렬로 수정하는 방법을 사용함으로써 본래의 작업 순서를 그대로 유지하면서 Deadlock 상태의 탐지 후에 이를 회피할 수 있는 모델을 제시할 수 있는 알고리즘을 제안하였다. 또한, 이의 결과를 reachable tree와 Visual Object Net++(평가판1.44.2)을 사용하여 검증하고자 한다.

특히 본 연구에서 제안하는 알고리즘은 자원공유 플레이스에서의 마킹 변화가 교착 상태의 발생과 회피에 직접적인 관계이므로 인접 행렬에서의 자원공유 플레이스의 변화를 분석하여 자원 공유 마킹 플레이스에서의 마킹 변화에 따른 교착 상태가 확인 가능하게 되며, 또한 교착 상태 회피 모델의 제시가 용이하게 되는 장점을 가지고 있다.

이 연구의 구성은 다음과 같다.

먼저, 2장에서는 이 연구에서 제안하는 *DAPN*의 정의를 기술하고, 3장에서는 일반 플레이스와 자원 공유 플레이스간의 관계를 설명하는 행렬의 기본정의에 대해 기술하며, 4장에서 교착상태를 탐지하여 이를 회피하는 알고리즘을 예제 모델과 함께 제안한다. 5장에서는 제안된 알고리즘을 이용한 예제 모델의 교착상태 회피 결과에 대한 검증을 수행하고, 6장에서 결론을 맺는다.

2. DAPN(Deadlock Avoidance Petri Net)

기계, 이동 로봇, 부품과 같은 자원을 이용해 다양한 품종을 생산할 수 있는 생산시스템인 유연생산 시스템은 작업을 진행하는 복수개의 과정과 이러한 작업 프로세스들이 원활히 수행되도록 자원들을 분배해주는 자원의 수요공급을 담당하는 부분들로 구성되어 있다. 따라서 자원이 많으면 많을수록 생산의 효율이 높아질 것이다. 그러나 이런 자원은 한정된 양만을 이용할 수 있고, 또한 한 번에 하나의 작업을 하며 프로세스가 진행된다. 그러므로 사용가능한 자원은 복수 개이나 작업 단계에 있어서 한 순간 필요한 자원은 하나이다. 그래서 이러한 유연생산 시스템을 모델링함에 있어서도 작업의 단계 마다 에는 하나의 자원이 들어가 하나의 자원이 출력되어 진행된다.

기존의 PN(Petri Net)은 FMS의 특성상 작업을 하는 machine 부분과 부품을 의미하는 자원의 부분이 모두 하나의 플레이스 종류로 기술되어 있으므로 해서 교착상태의 최대의 문제점인 부품들의 분배와 취득의 부분을 명확히

표현해 낼 수 없었다. 따라서 본 연구에서는 작업 플레이스 P의 집합과 자원공유 플레이스 R의 집합으로 분리하여 표현하는 *DAPN*을 정의하여 모델링에 이용하고자 한다.

정의 1: *DAPN*

$DAPN=(P,R,T,I,O,M_0)$ 여기서,

$P=\{p_1,p_2,.....,p_n\}$

: 작업플레이스의 유한집합 $(n \geq 0)$

$R=\{r_1,r_2,.....,r_m\}$

: 자원 공유 플레이스의 유한집합 $(m \geq 0)$

$T=\{t_1,t_2,.....,t_k\}$

: 트랜지션의 유한집합 $(k \geq 0)$

$P \cap R=\varnothing, P \cap T=\varnothing, R \cap T=\varnothing$

$\forall p \in P \ \ |\bullet p|=1$ and $|p \bullet|=1$

$\forall r \in R \ \ |\bullet r| \geq 1$ and $|r \bullet| \geq 1$

$I: \{P \cup R\} \times T \rightarrow N$, I는 입력함수

$O: T \times \{P \cup R\} \rightarrow N$, O는 출력함수

$P \neq 0$ and $R \neq 0$ and $T \neq 0$

$M_0 \in M=\{M | M: \{P \cup R\} \rightarrow N\}$, M_0는 초기 마킹 함수

N : 양의 정수들의 집합

P는 유연생산시스템의 작업을 나타내기 위한 플레이스들의 집합이며, R은 자원들의 상태를 표현하는 자원공유 플레이스들의 집합을 나타낸다. 또한 T는 다른 작업을 위한 조건 여부를 체크하는 트랜지션들의 집합이고, I는 P나 R의 플레이스들에서 트랜지션으로 입력되는 입력함수, O는 트랜지션에서 P나 R의 플레이스들로 출력되는 출력함수를 말한다.

정의 2 : 점화

*DAPN*에 있는 $t_j \in T$는 $p_i \in P$와 $r_i \in R$이 enable될 때마다 점화할 수 있다. 즉, 점화는 트랜지션 자신에게로 입력되는 모든 플레이스들이 토큰을 보유하고 있는 상태, 다시 말하면 입력되는 모든 플레이스가 마킹된 상태여야 점화가 가능하다.

$M(p_i) \geq \#(p_i, I(T_j))$, $M(r_i) \geq \#(r_i, I(T_j))$

여기서,

$$\#(p_i, I(t_j))=\begin{cases}1 & \text{if } p_i \in I(t_j) \\ 0 & \text{if } p_i \notin I(t_j)\end{cases}$$

$$\#(r_i, I(t_j))=\begin{cases}1 & \text{if } r_i \in I(t_j) \\ 0 & \text{if } r_i \notin I(t_j)\end{cases}$$

정의 3 : 마킹

패트리넷에서는 플레이스들에 분산된 토큰의 분포를 마킹이라고 하는데, 이러한 마킹은 점화함으로써 다음 단계의 토큰분포, 즉 다음 단계의 마킹의 상태로 이행된다.

$\forall p_i \in P$, $\forall r_i \in R$ 인 경우에 대해, *DAPN*에 있는 $t_j \in T$의 점화 후 새로운 마킹의 결과를 M'이라고 한다.

$M'(p_i)=M(p_i)-\#(p_i, I(t_j))+\#(p_i, O(t_j))$,

$M'(r_i)=M(r_i)-\#(r_i, I(t_j))+\#(r_i, O(t_j))$

여기서,

$$\#(p_i, O(t_j))=\begin{cases}1 & \text{if } p_i \in O(t_j) \\ 0 & \text{if } p_i \notin O(t_j)\end{cases}$$

$$\#(r_i, O(t_j))=\begin{cases}1 & \text{if } r_i \in O(t_j) \\ 0 & \text{if } r_i \notin O(t_j)\end{cases}$$

정의 4 : Invariant matrix

플레이스와 트랜지션 사이의 관계를 나타내는 행렬을 Invariant matrix로 정의한다.

$C=\langle P', T, B^-, B^+ \rangle$

여기서,

C : *DAPN* 구조의 행렬

P' : 작업플레이스들의 집합 P와 자원공유 플레이스들의 집합 R의 합집합, 즉, $P \cup R$

T : 트랜지션들의 집합

$B^-=[i,j]=\#(P_i, I(t_j))$: 입력함수의 "행" 행렬

$B^+=[i,j]=\#(P_i, O(t_j))$: 출력함수의 "열" 행렬

정의 5 : Deadlock

$DAPN=(P,R,T,I,O,M)$에서 $P_S \subset \{P \cup R\}$인 P_S가 만약 $\forall t_j \in T$인 경우, 상태가 다음과 같으면 이를 Deadlock(교착상태)이라고 한다.

$\#(P_S, I(t_j)) \geq \#(P_S, O(t_j))$

3. 행 렬

이 장에서는 일반 플레이스와 자원 공유 플레이스간

의 관계를 설명하기 위한 행렬의 기본 정의를 설명하고자 한다.

정의 6 : 작업 플레이스와 자원공유 플레이스간의 행렬 M_{PR}

m행 n열의 행렬에서, B^-와 B^+를 각각 입력함수와 출력함수라고 할 때, 작업 플레이스와 자원 공유 플레이스간의 행렬 M_{PR}은 다음과 같이 정의한다.

$$M_{PR} = B^-(B^+)^T$$

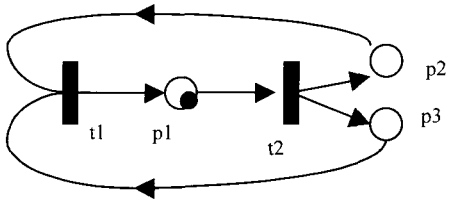

그림 1. 예제 모델
p_1 : 자원공유 플레이스,
p_2, p_3 : 작업 플레이스

$$M_{PR} = \begin{matrix} t_1\,t_2 \end{matrix} \quad \begin{matrix} p_1\,p_2\,p_3 \end{matrix} \quad \begin{matrix} p_1\,p_2\,p_3 \end{matrix}$$
$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix}$$

M_{PR}에서 플레이스 p_2와 p_3은 트랜지션 t_2에 의해 플레이스 p_1으로부터 토큰을 받는다. 마찬가지로 플레이스 p_1은 트랜지션 t_1을 통해 플레이스 p_2와 p_3으로부터 하나의 토큰을 받는다. 이러한 관계는 열의 관계가 플레이스 p_i로부터 보내는 토큰들의 합을 말하고, 행의 관계는 플레이스 p_i로 받아들이는 토큰들의 합이다. 그러므로 만약 플레이스 p_i의 행의 합이 플레이스 p_i의 열의 합보다 더 크면 이 플레이스는 dead가 아니다. 그러나 그렇지 않으면 이 노드는 점화하기에 충분한 토큰을 유지할 수 없으므로 dead가 될 것이다.

*DAPN*모델은 FMS의 job에 기반을 둔 작업 플레이스와 자원 공유 플레이스 사이의 모든 관계를 보여주기 위해 행렬로 표현된다. 이것은 job에 기반을 둔 서브넷들로 분해될 수 있다는 것을 의미한다.

정의 7 : Subnet

$DAPN' = (P', R', T', I', O')$,
$DAPN = (P, R, T, I, O)$ 일 때
만약, $P' \subseteq P$, $R' \subseteq R$, $T' \subseteq T$,
$I' = I \cap (\{P' \cup R'\} \times T')$,
$O' = (T' \times \{P' \cup R'\})$ 이면,
$DAPN' \subseteq DAPN$이고, 따라서 $DAPN'$은 $DAPN$의 subnet이다.

정의 8 : Sub Matrix

M_{PR}은 $DAPN$의 인접행렬이다.
M_{PRi}은 subnet $DAPN_i$의 인접행렬이다.
$DAPN_i \subseteq DAPN$
$\forall M_{PRi} \subseteq M_{PR}$, $(1 \le i \le n)$
그러므로, $\bigcup_i M_{PRi} = M_{PR}$

이제 우리는 job에 기반을 둔 작업 플레이스와 자원 공유 플레이스 사이의 모든 관계를 보여주는 행렬을 제안한다.

정의 9 : 확장된 인접행렬

M_{PRi}'은 M_{PRi}의 확장된 sub matrix이다.
$\#(p', M_{PRi}') = \mathrm{Lim}$
$\mathrm{Lim} = \max(\#(p, M_{PRi}), 1 \le i \le n)$
$p' \in M_{PRi}'$, $p \in M_{PRi}$
또한, $F^{M_{PRn}'}_{M_{PR1}'}$은 확장된 sub matrix M_{PR1}', M_{PR2}', ..., M_{PRn}'들의 연합이다.
즉, $F^{M_{PRn}'}_{M_{PR1}'} = \sum_{i=1}^{n} M_{PRi}'$
여기서, 작업 플레이스들의 수는,
$\#\left(p, F^{M_{PRn}}_{M_{PR1}}\right) = \max\left(\#(p, M_{PRi}), (1 \le i \le n)\right)$
자원공유 플레이스들의 수는,
$\#\left(r, F^{M_{PRn}}_{M_{PR1}}\right) = \max\left(\#(r, M_{PRi}), (1 \le i \le n)\right)$

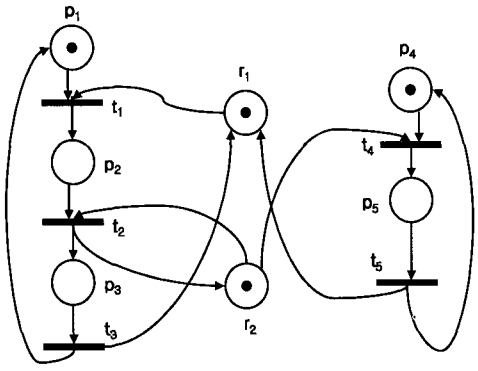

그림 2. DAPN 모델

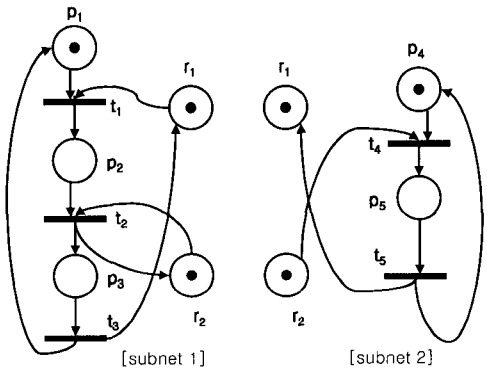

그림 3. 그림 2의 분할 *DAPN* 모델

정의 6과 7에 의해 분할된 *DAPN* 모델을 그림 3에 나타내었다.

또한 정의 8에 의해 우리는 다음과 같이 M_{PR}의 sub matrix M_{PR1}과 M_{PR2}를 얻을 수 있다.

$$M_{PR} = \begin{matrix} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ r_1 \\ r_2 \end{matrix} \begin{matrix} p_1\,p_2\,p_3\,p_4\,p_5\,r_1\,r_2 \\ \begin{bmatrix} 0100000 \\ 0010001 \\ 1000010 \\ 0000100 \\ 0001010 \\ 0100000 \\ 0010101 \end{bmatrix} \end{matrix}$$

$$M_{PR1} = \begin{matrix} p_1\,p_2\,p_3\,r_1\,r_2 \\ \begin{bmatrix} 01000 \\ 00101 \\ 10010 \\ 01000 \\ 00101 \end{bmatrix} \begin{matrix} p_1 \\ p_2 \\ p_3 \\ r_1 \\ r_2 \end{matrix} \end{matrix} \quad M_{PR2} = \begin{matrix} p_4\,p_5\,r_1\,r_2 \\ \begin{bmatrix} 0100 \\ 1010 \\ 0000 \\ 0100 \end{bmatrix} \begin{matrix} p_4 \\ p_5 \\ r_1 \\ r_2 \end{matrix} \end{matrix}$$

여기서, $\#(p, M_{PR1}) = 3$, $\#(p, M_{PR2}) = 2$ 이므로, 정의 9에 의해 $\#(p, M_{PR1}') = 3$, $\#(p, M_{PR2}') = 3$이다.

결국, $\#\left(p, F_{M_{PR1}'}^{M_{PR2}'}\right) = 3$와 $\#\left(r, F_{M_{PR1}'}^{M_{PR2}'}\right) = 2$를 얻을 수 있고, 따라서 $F_{M_{PR1}'}^{M_{PR2}'}$은 다음과 같다.

$$F_{M_{PR1}'}^{M_{PR2}'} = M_{PR1}' + M_{PR2}' =$$

$$\begin{bmatrix} 01000 \\ 00101 \\ 10010 \\ 01000 \\ 00101 \end{bmatrix} + \begin{bmatrix} 01000 \\ 10010 \\ 00000 \\ 00000 \\ 01000 \end{bmatrix} = \begin{bmatrix} 02000 \\ 10111 \\ 10010 \\ 01000 \\ 01101 \end{bmatrix}$$

그러므로,

$$F_{M_{PR1}'}^{M_{PR2}'} = \begin{matrix} p_{1,4}\,p_{2,5}\,p_3\,r_1\,r_2 \\ \begin{bmatrix} 02000 \\ 10111 \\ 10010 \\ 01000 \\ 01101 \end{bmatrix} \begin{matrix} p_{1,4} \\ p_{2,5} \\ p_3 \\ r_1 \\ r_2 \end{matrix} \end{matrix}$$

4. 교착상태 회피

이 장에서는 3장에서 제시된 행렬 표현들을 이용하여 모델이 교착상태를 가지고 있는지를 파악하여, 만약 교착상태가 있다면 그것을 회피하는 방법을 새로운 알고리즘을 통해 제안하고자 한다. 알고리즘은 그림 4와 같이 행렬 $F_{M_{PR1}'}^{M_{PR2}'}$를 사용한다.

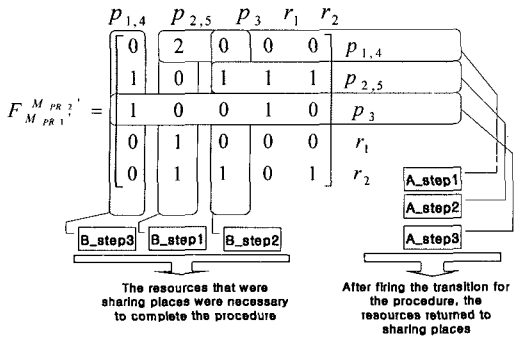

그림 4. 그림 3 모델의 알고리즘 적용을 위한 단계 표현

그림 4에서 B_step은 각 단계에 해당하는 작업 플레이스가 수행되기 위해 필요한 트랜지션의 점화전(Before) 자원의 상태이고, A_step은 각 단계에 해당되는 작업 플레이스가 수행되기 위해 필요한 트랜지션의 점화 후(After) 이루어지는 자원의 상태를 말한다. 또한 처음시작 단계는 초기 마킹이 되어 있는 작업 플레이스의 바로 다음에 있는 트랜지션을 중심으로 시작된다. 따라서 작업 플레이스가 동작하기 위해서 필요한 자원공유 플레이스의 자원들의 상태를 나타내는 테이블이 필요하다. 그래서 우리는 행렬 $F^{M_{PR2}'}_{M_{PR1}}$를 사용하여 B_step과 A_step의 값을 표 1에 나타낸다.

표 1. 그림 4의 B_step과 A_step의 자원공유 플레이스에 있는 자원의 수

Resources Sharing Places	B_step1	B_step2	B_step3	A_step1	A_step2	A_step3
r1	1	0	0	0	1	1
r2	1	1	0	0	1	0

각 단계의 자원의 값은 정의 10을 사용하여 계산될 수 있다.

정의 10 : 상태값(Status Value : SV)

SV_0 : 초기 상태의 자원공유 플레이스에 있는 자원들의 수

$SV_i = SV_{i-1} - \#(r, B-step((i+1)/2)),$

$1 \le i \le n,\ \{i | i\ is\ an\ odd\ number\}$

$SV_j = SV_{j-1} + \#(r, A-step(j/2)), \quad 2 \le j \le n,$

$\{j | j\ is\ an\ even\ number\}$

표 2. 그림 2의 자원 상태 값

Resources Sharing Places	SV0	SV1	SV2	SV3	SV4	SV5	SV6
r1	1	0	0	0	1	0	1
r2	1	0	0	-1	0	0	0

표 2에서, 만약 SV_i, $1 \le i \le n$, $\{i | i\ is\ an\ odd\ number\}$의 r1과 r2의 값이 음수이면, JOB1과 JOB2는 자동적으로 음수가 될 것이다.

이러한 가정은 JOB1과 JOB2의 음수 결과로 인해 점화하기 위해 필요한 자원들이 제공될 수 없다는 것을 나타낸다. 결국, 이 부분에서 교착상태의 원인이 된다.

위의 표 2를 통해 그림 2 의 $DAPN$모델은 SV_3의 음수 값으로 인해 교착상태가 생기게 됨을 알 수 있다. 이러한 교착상태를 회피하기 위해서는 다음과 같은 절차가 필요하다. 우선, 교착상태를 일으키는 음수 값을 없애야 하므로 그 이전 단계에서 자원을 공급하여 음수의 값이 나오지 않도록 하여야 한다. 그래서 결국, 표 2의 경우에서는 SV_3의 음수 값을 제거하기 위해 SV_i의 형태인 SV_2의 자원을 1로 수정하여야 한다. 이는 SV_j의 이전 값은 수정될 수 없기 때문이다. SV_j의 값이 수정될 수 없는 이유는 FMS의 특성상 제품을 만드는데 필요한 기계나 부품을 의미하는 자원은 임의로 변경할 수 없기 때문이다.

따라서 이와 같이 수정하면 테이블은 다음 표 3과 같이 수정된다.

표 3. 그림 2의 수정된 자원 상태 값(1)

Resources Sharing Places	SV0	SV1	SV2	SV3	SV4	SV5	SV6
r1	1	0	0	0	1	1	2
r2	1	0	1	0	1	1	1

즉, $\#(r_2, SV_2) = 1$ 로 수정된다.

표 3을 사용한 행렬을 다음과 같이 나타낸다.

$$F_{M_{PR1}}^{M_{PR2}'} = \begin{matrix} p_{1,4}\,p_{2,5}\,p_3\,r_1\,r_2 \\ \begin{bmatrix} 0\,2\,0\,0\,1 \\ 1\,0\,1\,1\,1 \\ 1\,0\,0\,1\,0 \\ 0\,1\,0\,0\,0 \\ 0\,1\,1\,0\,1 \end{bmatrix} \end{matrix} \begin{matrix} p_{1,4} \\ p_{2,5} \\ p_3 \\ r_1 \\ r_2 \end{matrix}$$

한편, 계속적인 프로세스 진행을 위해서는 자원의 상태 테이블의 마지막이 항상 초기 자원의 상태를 가지고 있어야 한다. 표 3의 마지막 단계인 SV_6은 이 조건을 만족하지 않는다. 즉, 계속적인 프로세스 진행을 위한 "boundedness" 속성을 만족하기 위해서는 자원의 초기 상태를 유지해야 하는데 이를 위해 표 3의 SV_6의 r_1 값을 1로 수정하여야 한다. 따라서 표 3은 다음과 같이 표 4로 수정된다.

즉, $\#(r_1, SV_6) = 2 - 1 = 1$ 로 수정된다.

표 4. 그림 2의 수정된 자원 상태 값(2)

Resources Sharing Places	SV0	SV1	SV2	SV3	SV4	SV5	SV6
r1	1	0	0	0	1	1	1
r2	1	0	1	0	1	1	1

표 4를 사용한 행렬은 다음과 같다.

$$F_{M_{PR1}}^{M_{PR2}'} = \begin{matrix} p_{1,4}\,p_{2,5}\,p_3\,r_1\,r_2 \\ \begin{bmatrix} 0\,2\,0\,0\,1 \\ 1\,0\,1\,1\,1 \\ 1\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0 \\ 0\,1\,1\,0\,1 \end{bmatrix} \end{matrix} \begin{matrix} p_{1,4} \\ p_{2,5} \\ p_3 \\ r_1 \\ r_2 \end{matrix}$$

결국, 행렬 $F_{M_{PR1}}^{M_{PR2}'}$은 다음 그림 5와 같이 수정된다.

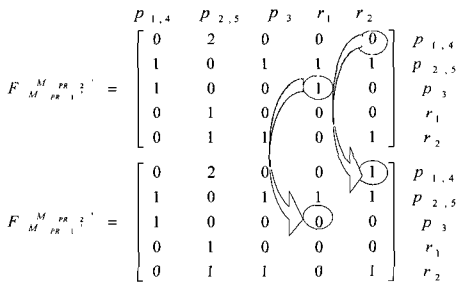

그림 5. 행렬 $F_{M_{PR1}}^{M_{PR2}'}$

그림 5 에서 우리는 r_1의 입력자원이 하나 제거되고, r_2의 입력자원이 하나 더해짐을 알 수 있다. 결국, 이 알고리즘을 통해 원래의 모델이 가지고 있던 교착상태의 위험을 탐지하고 이를 회피하여 새로운 모델을 생성할 수 있었다. 최종 모델은 그림 6과 같다.

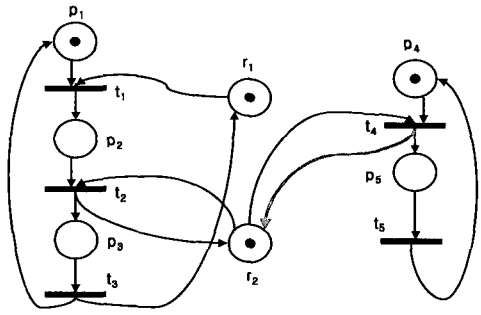

그림 6. 그림 2 모델에 교착상태 회피 알고리즘을 적용한 후의 수정된 모델

그림 2와 달리 그림 6의 상태는 더 이상 교착상태가 없으며, 또한 boundedness 성질을 만족한다.

이러한 교착상태의 탐지와 회피 과정을 알고리즘으로 나타내면 다음과 같다.

```
Algorithm:
int main()
{
n=the number of resource sharing places;
Define value of resource state table, SV, according
to the definition 11;

While(exist values of SV_i)
{
result=Detection(the values of SV_i, n);
if (result==0){
    i++;
    continue;
else
    Avoidance1(the value of SV_i, result);
}
If (value of initial resource sharing places in table !=
value of final resource sharing places in table) {
Cout << "This net is UNBOUNDEDNESS";
```

```
  Avoidance2();
}
}
Detection(the values of SV_i, n)
{
for(r=0; r<= n; r++)
   if (SV_i of the r < 0) {
        cout << "This net is deadlock"
        return r,
        }
}
Avoidance1(the values of SV_i, r)
{
#(r,SV_{i-1}) = #(r,SV_{i-1}) - #(r,SV_i);
Compute values of resource state table according to
Definition 11 again;
}
Avoidance2()
{
value of final resource sharing places =
value of initial resource sharing places;
}
```

5. 검 증

이 장에서는 reachable tree와 Visual Object Net++을 통해 제안된 알고리즘을 검증하고자 한다. 먼저, 교착상태가 있는 예제 모델에 제안된 알고리즘을 적용하여 회피된 결과를 reachable tree로 나타내면 다음 그림 7과 같다.

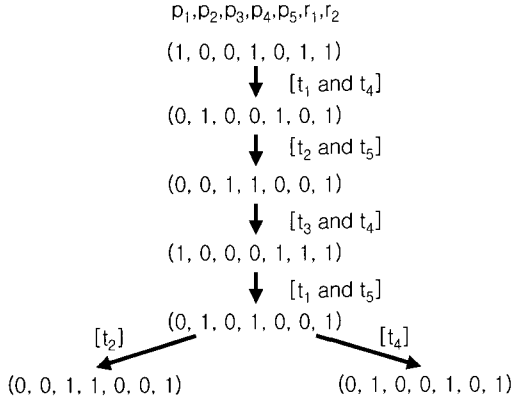

그림 7. 그림 6의 Reachable Tree

그림 7에서 우리는 수정된 모델이 교착상태가 더 이상 없고, bounded 됨을 발견할 수 있다.

또한, 우리는 Visual Object Net++을 사용하여 수정된 모델이 더 이상 교착상태가 아님을 검증하였다. 다음 그림 8은 Visual Object Net++을 이용한 화면이다.

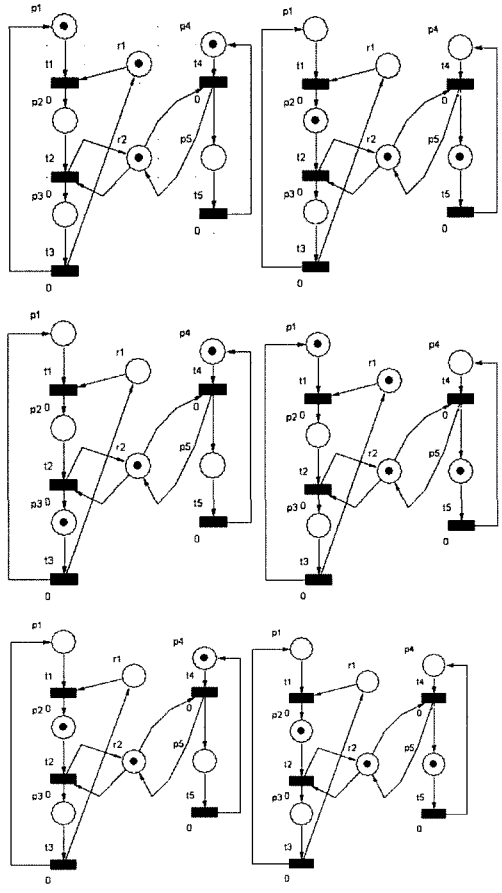

그림 8. Visual Object Net++을 이용한 수행 결과 화면

6. 결 론

이 연구에서 우리는 기계, 이동 로봇, 부품과 같은 자원을 이용해 다양한 품종을 생산할 수 있는 생산 시스템인 유연생산 시스템에서의 교착상태 회피를 위한 알고리즘을 인접행렬과 *DAPN*을 이용하여 제안하였다.

또한, 이 연구에서 제안된 알고리즘을 이용해 출력되는 결과물은 교착상태를 벗어날 수 있는 수행 가능한 모델이

므로 이러한 수정 모델을 통해 교착상태가 없는 적절한 작업을 진행할 수 있는 장점이 있다.

한편, 제안된 알고리즘을 사용하여 교착상태를 해결한 모델을 reachable tree와 Visual Object Net++을 사용하여 검증함으로써 본 연구의 신뢰성을 높였다.

이제 앞으로의 연구는 좀 더 실제적인 모델들에 적용하여 알고리즘의 효율성 및 경제성을 분석하고자 한다.

참 고 문 헌

1. Corbett JC, Evaluating Deadlock detection methods for concurrent software, IEEE tr. Software Engineering, Vol.22(3), 1996.
2. Ezpleta J., Colom JM. And Martinez J., A Petri net based on deadlock prevention policy for flexible manufacturing systems, IEEE tr. Robotics and Automation, Vol.11. No.2, 173-184, 1995.
3. Damasceno BC. and Xie X., Petri nets and deadlock-free scheduling of multiple-resource operations, IEEE Conf. on Systems, Man and Cybernetics, 878-883, 1999.
4. L. Ferrarini and M. Maroni, A Control Algorithm for Deadlock-Free Schedulings of Manufacturing Systems, IEEE Conf. on Systems, Man and Cybernetics, 3762-3767, 1997.
5. Feng Chu. and Xiao-Lan X., Deadlock Analysis of Petri Nets Using Siphons and Mathmatical Programming, IEEE tr. Robotics and Automation, Vol.13. No.6. 1997.
6. Melzer S. and Romer S., Deadlock checking using net Unfoldings, In Proc. of the Conf. on Computer-Aided Verfication, CAV'97, 1997.
7. Murata T.(1989). Petri Nets: Properties, Analysis an Applications, Proceedings of the IEEE, 77(4), IEEE, USA, 541-580, 1989.
8. Xiong HH. and Zhou MC., Deadlock free scheduling of an automated manufacturing system based on Petri nets, In IEEE ICRA'97, 945-950, 1997.
9. Yoon H. and Lee D., Deadlock-Free Scheduling for Automated Manufacturing cells, Proceeding of International Conference on Control, Automation Robotics and Vision, 2000.
10. M. A. Lawley, Deadlock avoidance for production system with flexible routing, IEEE tr. Robotics and Automat., vol. 15, no. 3, pp. 497-509, 1999.
11. M. p. Fanti, B.Maione, S.Mascolo, and B. Turchiano, Event-based feedback control for deadlock avoidance in flexible production systems, IEEE tr. Robotics and Automat., vol. 13, no.3, pp.347-363, 1997.
12. Jonghun P. and Spyros A. Reveliotis, Deadlock Avoidance in Sequential Resource Allocation Systems With Multiple Resource Acquisitions and Flexible Routings, IEEE tr. Automatic Control, vol. 46, No. 10, pp. 1572-1583, 2001.
13. Sulistyono W., Lawley M.A.. Deadlock Avoidance for manufacturing system withPartially ordered Process Plans, In: IEEE Trans. on Robotics and automation, Vol. 17, No. 6, pp. 819-832, 2001.
14. Wu N.,Zhou MC., Avoiding Deadlock and reducing Starvation and Blocking in Automated manufacturing System, In: IEEE Trans. on Robotics and automation, Vol. 17, No. 5, pp. 658-669, 2001.
15. A.Giua,et elc., Observer-Based state-feedback control of timed Petri nets with deadlock recovery, In: IEEE Trans. on Automatic control, Vol. 49, No. 1, pp. 17-29, 2004.
16. F. Hsieh and S. Chang, Dispatching-Driven Deadlock Avoidance Controller Synthesis for Flexible manufacturing Systems, In: Trans. on Robotics and automation, Vol. 10, No. 2, pp. 196-209, 1994.
17. Mark L. and Spyros R., Deadlock Avoidance for Sequential Resource Allocation Systems: Hard and Easy Cases, The International Jounal of Flexible Manufacturing Systems, 13, pp.385-404, 2001.
18. T.K.Kumaran,W.Chang,H.Cho et al., A structured approach to deadlock detection, avoidance and resolution in flexible manufacturing systems, Intl/ Journal of Production Research, 32(10), pp.2361-2379, 1994.
19. S. E. Ramaswamy, S. B. Jpshi, Deadlock-free schedules for automated manufacturing workstations, IEEE Tr. on Robotics and Automation, 12(3), pp.391-400, 1996.
20. V.H.Garmhausen, E.M. Clarke,.campos, Deadlock prevention in flexible manufacturing systems using symbolic model checking, Proceeding IEEE Intl. Conf. on Robotics and Automation, pp.527-532, April, USA, 1996.
21. S.A.Reviolis and P.M.Ferreira, Deadlock avoidance policies for automated manufacturing cells, IEEE Tr. on Robotics and Automation, 12(6), pp.845-857, 1996.
22. K. Barkaoui, A. Chaoui, B. Zouari, Supervisory control of discrete event systems using structure theory of Petri nets, 1997 IEEE Intl. Conference on System, Man, and Cybernetics, pp. 3750-3755, Oct. Orlando,Florida, USA, 1997.

송유진 (syj@changwon.ac.kr)

1992 창원대학교 컴퓨터정보통신공학부 컴퓨터공학과 학사
1995 창원대학교 컴퓨터정보통신공학부 컴퓨터공학과 석사
2003 창원대학교 컴퓨터정보통신공학부 컴퓨터공학과 공학박사
1995~2002 창원대학교 강사
2003~현재 창원대학교 초빙교수

관심분야 : 패트리넷, 성능분석, 알고리즘, 정보보호, 스케줄링

이종근 (jklee@changwon.ac.kr)

1974 숭실대학교 전자계산학과 학사
1976 숭실대학교 전자계산학과 석사
1978 고려대학교 경영대학원 경영학 석사
2002 LCGI / Ecole Centrale Paris 전산학, 공학박사
1987~1990 LSI / Univ. de Montpellier II 연구원 역임
1983~현재 창원대학교 컴퓨터정보통신공학부 컴퓨터공학과 교수

관심분야 : 패트리넷, 성능분석, 정보보호, 스케줄링