論文93-30B-7-6

초기 마킹(Marking)과 접속행렬 비교에 의한 페트리 네트의 해석

(Petri Net Analysis by Comparing Initial Marking with Incidence Matrix)

禹 廣 俊*, 柳 昶 根*

(Woo Kwang Joon and Rhu Chang Keun)

要 約

페트리 네트는 다양한 시스템을 모델링할 수 있으며, 모델링된 시스템의 동작을 검증하기 위하여 페트리 네트를 해석해야 한다. 본 논문에서는 초기마킹과 변형된 접속행렬을 비교하여 페트리 네트를 해석하는 알고리듬을 제안한다. 제안된 알고리듬은 self-loop가 포함된 페트리 네트의 해석이 가능하고, 점화 순서에 따른 시퀀스도 확인할 수 있다.

Abstract

Petri nets are capable of modeling a large variety of systems, it is necessary to analyze the modeled system for verifying the behavior. In this paper, we propose an algorithm which analyzes Petri nets by comparing initial marking with modified incidence matrix. The proposed algorithm is able to analyze Petri net involving self-loop, and is able to verify the order of sequence according to that of firing.

Ⅰ. 서론

최근 공장자동화에서 자동 조립과정을 위한 시퀀스 제어 시스템의 제어프로그램은 다품종 소량 생산의 추이에 의해 높은 유연성(flexibility)과 유지보수성(maintenance)이 요구된다. 기존의 PC나 PLC는 어셈블리어나 ladder diagram을 이용하여, 시스템간의 동기와 배타조건을 나타내기 위해 많은 제어프래그가 포함되어 있어 제어프로그램의 이해가 어려울 뿐만 아니라 제어 사양의 변경에 따른 제어프로그램의 변경이 어렵다. 시퀀스제어는 시스템간의 동기와 배타조건을 유지하기 위해 병렬성을 갖는 비동기 시스템으로 표현하는 것이 바람직하다. 이러한 관점에서 GRAFCET과 페트리 네트가 제안되었다.[1-3]

페트리 네트는 많은 시스템에 적용할 수 있는 도식적이고 수학적인 모델링 도구로서 병행성(concurrent), 비동기, 분산, 병렬, AND/OR stochastic 시스템 등에 많이 이용되고 있다.[4] 페트리 네트는 도식적인 면에서 신호흐름도, 블럭선도와 비슷하나 토큰이 추가됨으로써 시스템이 동적 특성을 가질 뿐만 아니라 병행성인 경우도 표현이 가능하다.[5] [6] 또한 시스템을 상태방정식, 대수방정식, 그 밖에 다

*準會員, 檀國大學校 電子工學科
(Dept. of Elec. Eng. Dankook Univ.)
(※이 논문은 1991학년도 대학연구비 지원에 의한 연구 결과임.)
接受日字: 1992年 1月 1日

른 수학적인 모델로 표현할 수 있다. [7]

사양에 따라 페트리 네트로 모델링한 시스템은 완전성 여부를 확인할 수 있어야 한다. 이를 위한 기존의 해석방법은 가도달성(reachability) 트리나 행렬을 이용하는 방법이 있다. 가도달성 트리는 모든 페트리 네트에 적용 가능하나 같은 가도달성 트리에서 서로 다른 페트리 네트가 존재하는 경우가 발생한다. 또한 수작업에 의해 해석하기 때문에 복잡한 페트리 네트의 경우 오류를 범할 수도 있으며 해석하는데 많은 시간이 소요된다. [8]

행렬을 이용하는 방법은 페트리 네트를 해석하기 편리하나, self- loop의 표현이 불가능하여 접속행렬(incidence matrix) 안에서 정보를 상실하게 되며 [5], 시퀀스가 결정되지 못하고 행렬의 해가 존재해도 점화(firing)되지 않는 경우가 발생한다. [3]

본 논문에서는 행렬 표현법에 근거를 두고, self-loop의 표현이 가능할 뿐만 아니라 별도의 연산없이 점화 시퀀스 결정이 가능하여 이를 H/W나 S/W로 실제 시스템을 실현할 때 용이하고 컴퓨터에 의해 해석할 수 있는 알고리듬을 제안한다.

Ⅱ. 페트리 네트의 해석

1. 페트리 네트의 정의

페트리 네트는 다음과 같이 세 개의 구성요소로 된 집합으로 표기된다.

$n = (P, A, T)$

여기서 P는 장소(place)의 집합이며 원으로 표시한다.

$Pi = \{ p_1, p_2, p_3, \cdots, p_n \}$

T는 천이(transition)의 집합이며 바(bar)로 표시한다.

$T_j = \{ t_1, t_2, t_3, \cdots, t_m \}$

A는 아크(arc)의 집합이며 방향성선으로 표시한다.

$A_i = \{ a_1, a_2, a_3, \cdots, a_n \}$

그 밖에 M은 토큰(token)으로, 주어진 조건의 진(truth)을 의미하고 점으로 표시한다.

$M_i = \{ m_0, m_1, m_2, \cdots, m_n \}$

특히 초기마킹(initial marking)은 M0 : P→{0, 1,2…..}이다.

마킹이라 함은 장소에 토큰을 주어 페트리 네트로 시스템의 상태를 나타내고 초기 마킹이라 함은 시스템의 초기 상태를 나타낸다. 시퀀스가 진행함에 따라 하나의 장소에 토큰이 계속 증가하는 페트리 네트를 무제한적(unbounded), 일정수 이내에 있는 경우를 제한적(bounded), 특히 하나의 장소에 하나의 토큰만이 존재하는 경우를 안전적(safe)이라 한다. 또한 데드록이 발생하지 않으면 생동적(live)이라 한다. 만일 각 입력장소의 천이에 이르는 아크의 수만큼 토큰을 포함한다면 천이는 점화가능(enable) 상태가 된다. 점화 가능한 천이는 점화(firing)될 수 있다.

2. 가도달성 트리를 이용한 해석방법

그림 1의 페트리 네트를 가도달성 트리로 해석한 결과가 그림 2이다. 여기서 마킹과 마킹 사이를 연결하는 천이의 집합을 점화 시퀀스라 하고 일반적으로 $\sigma = t_1, t_2, \cdots, t_m$ 으로 표기하며 그림 2의 경우 $\sigma = t_2, t_3, t_5, t_1, t_4, t_5$이다. 또한 초기토큰부터 점화 가능한 모든 천이의 점화 결과 새롭게 생성된 마킹의 집합을 가도달성 집합이라 하고 $R(M_0)$로 표기한다.

해석 결과 각 장소에 토큰수가 하나씩이므로 안전적이고, 천이 $t_j(j=1,2,3,4,5)$가 한 번 이상 점화됐으므로 생동적인 페트리 네트라는 것을 알 수 있다.

이 방법은 모든 페트리 네트에 적용이 가능하고 self-loop를 표현할 수 있으며, 시퀀스도 알 수 있으나 같은 가도달성 트리에서 서로 다른 페트리 네트가 존재하는 경우가 발생한다. 또한 수작업에 의해 해석하기 때문에 복잡한 페트리 네트의 경우 오류를 범할 수 있으며 해석하는데 많은 시간이 소요된다. [8]

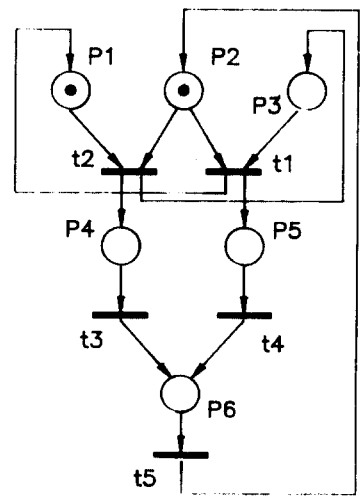

그림 1. 페트리 네트의 예 1
Fig. 1. Example 1 of Petri net.

110000
| t_2
001100
| t_3
001001
| t_5
011000
| t_1
100010
| t_4
100001
| t_5
110000

그림 2. 그림 1의 가도달성 트리
Fig. 2. Reachability tree of Fig. 1.

3. 행렬을 이용한 해석방법

이 방법은 페트리 네트를 행렬로 표현하여 해석하는 방법이다.

$D^-[j, i] = \#(P_i, I(t_j))$
$D^+[j, i] = \#(P_i, O(t_j))$.

여기서 D^-는 천이에 연결된 입력장소에 관련된 행렬이고 D^+는 천이에 연결된 출력장소에 관련된 행렬이다. 임의의 천이 t_j가 점화된 결과 각 장소의 토큰수를 표시하는 행렬 M'은 다음과 같이 정의한다.

$$M' = M + x.D$$

여기서 M 은 각 장소의 초기 토큰수를 표시하는 행렬, x는 천이의 점화 횟수를 의미하는 행렬, $D = D^+ - D^-$ 로 표시되는 행렬이다. 상기 식을 이용하여 해석한 페트리 네트의 예는 그림 3에서 보는 바와 같다.

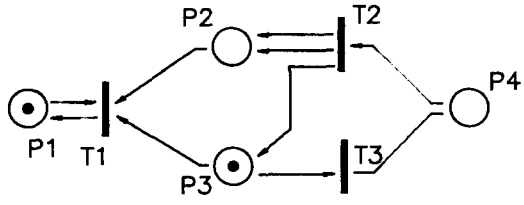

그림 3. 페트리 네트의 예 2
Fig. 3. Example 2 of Petri net.

$$D = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_2 & p_3 & p_4 \end{array} \\ \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & +2 & +1 & -1 \\ 0 & 0 & -1 & +1 \end{bmatrix} \end{array}$$

그림 4. 그림 2의 접속행렬
Fig. 4. Incidence matrix of Fig. 2.

초기토큰 M = [1 0 1 0] 에서 t_3이 점화됐을 때 M'= [1 0 0 1] 이 되고, 초기토큰 M = [1 0 1 0] 에서 M'= [1 8 0 1] 의 도달 가능 여부를 확인하기 위해 x의 해를 구해보면 x = [0 4 5] 가 된다. 그러므로 t_2가 네 번, t_3가 다섯 번 점화되면 도달 가능하다는 것을 알 수 있다. 만약 x의 해가 존재하지 않으면 도달 가능하지 않은 경우이다.

그러나 이 방법에는 다음과 같은 문제점이 있다. 첫째, self-loop가 표현되지 못한다. 상기 예에서 D^- 행렬의 1행 1열과 D^+ 행렬의 1행 1열은 서로 상쇄되어 D 행렬의 1행 1열은 영이 되므로 페트리 네트를 완전하게 표현하지 못하여 경우에 따라서 이 방법이 적절하지 못한 경우가 있다.[4] 둘째, 시퀀스가 결정되지 못한다. 위 예에서 x = [0 4 5] 의 해가 존재하여 t_2, t_3의 점화 횟수는 알 수 있으나 점화 순서는 알 수가 없다. 셋째, 행렬의 해가 존재해도 점화되지 못하는 문제점이 있다.[3]

Ⅲ. 변형된 접속행렬의 표현

페트리 네트를 표현하기 위해 다음과 같이 변형된 접속행렬을 정의한다.

$D^-[i, j] = \#(p_i, I(t_j))$
$D^+[i, j] = \#(p_i, O(t_j))$.
$D = D^+ - D^-$.
$D = [d_{ij}]_{nxm}$.
여기서

$$d_{ij} = \begin{cases} s, \ \forall p_i, I(t_j) \cap O(tj) \neq \phi \\ I(tj) = O(tj) = a \\ \text{정수}, \ \forall p_i, I(tj) \cap O(tj) = \phi \end{cases}$$

변형된 접속행렬에 대한 점화규칙은 다음과 같이 정의한다.

1)하나의 천이 t_j는 다음 조건을 만족하면 점화 가능하다.

$M(pi) \geq |d_{ij}|$. 이 때 $d_{ij} \leq 0$ 이고 $s = -a$이다.

2)천이 t_j가 점화됐을 때 다음 마킹 M'은 $M \xrightarrow{t_j} M'$ 으로 표기하며, 다음과 같이 구한다.

$M'(p_i) = M(p_i) + d_{ij}$. 이 때 $s = 0$이다.

이상과 같이 정의된 변형된 접속행렬은 페트리 네트의 일반적인 특성인 데드록 (deadlock), 생동성 (liveness), 제한성(boundedness) 및 안전성 (safeness) 등의 해석이 가능하다.

무 데드록(deadlock freeness)

이 특성을 조사하기 위해 가도달성 집합 $R(M_0)$에 속하는 주어진 마킹에서 점화 가능한 천이가 최소한 한 개 이상 존재하는지 알아야 하므로 다음 조건을 만족하면 된다.

$\forall M \in R(M_0)$ $\exists j \in \{1,2,\cdots,n\}$

$M(p_i) \geq |d_{ij}|$. 이 때 $d_{ij} < 0$ 이고 $s = -a$이다.

마킹 M에서 t_j는 점화 가능하다.

여기서 행렬 d_{ij}에서 음수는 천이에 입력되는 아크의 가중치이므로 천이가 점화 가능하려면 입력장소에 있는 토큰의 갯수가 크거나 같아야 한다. 또한 입력아크와 출력아크가 서로 같은 self-loop에서는 아크의 가중치 a도 점화 조건으로 고려해야 하므로 s= -a로 놓아야 한다.

$\forall M \in R(M_0)$ $\exists j \{1,2,\cdots,n\}$ 인 천이에서 데드록이 없는 페트리 네트는 생동성(liveness)이 보장된다. [1] [3]

제한성(boundedness)

$\forall p_i \in P$, $M' \in R(M_0)$에 대하여 $M'(p_i) \leq k$ 이면 k-제한적이고 k = 1이면 안전적이므로 상기의 점화 규칙 $M'(p_i) = M(p_i) + d_{ij}$에 의하여 확인할 수 있다.

여기서 입력아크와 출력아크가 서로 같은 self-loop에서 점화전에 갖고 있던 토큰수가 점화 후에도 같은 수로 유지되어야 하므로 s = 0으로 놓아야 한다.

Ⅳ. 공간 복잡도와 시간 복잡도

페트리 네트의 행렬 표현은 다음과 같은 방법 등이 있다. [8]

방법 1 : $PN = \langle P, T, C^-, C^+, M0 \rangle$.

방법 2 : $PN = \langle P, T, C, M0 \rangle$.

방법 3 : $PN = \langle P, T, C, C^-, M0 \rangle$.

방법 4 : $PN = \langle P, T, C^r, M0 \rangle$.

방법 5 : $PN = \langle P, T, D, M0 \rangle$.

여기서 $C = C^+ - C^-$, $C^r = [C^b, C^p]$ 이고, 방법 5는 본 논문에서 제안한 방법이다.

공간 복잡도(space complexity, memory requirement)에 대하여 해석하기 위해 다음과 같은 파라미터를 정의한다. [8]

$m = |P|$.

$n = |T|$.

k = 가중치를 갖는 입력 아크의 총수.

r = 가중치를 갖는 출력 아크의 총수.

l = self-loop의 총수.

l_1 = 입력과 출력 아크의 가중치가 같은 self-loop의 수.

l_2 = 입력과 출력 아크의 가중치가 같지 않은 self-loop의 수.

RSVF(Reflexive Space Variant Factor).

RRSVF(Relative Reflexive Space Variant Factor).

$$RSVF = \frac{\text{self-loop를 포함할 경우 요구되는 메모리 공간}}{\text{self-loop를 포함하지 않을 경우 요구되는 메모리 공간}}$$

$$RRSVF = \frac{\text{특별한 방법에 대한 self-loop를 포함할 경우 요구되는 메모리 공간}}{\text{방법1에 의해 요구되는 메모리 공간}}$$

RSVF가 작을수록 self-loop가 포함된 페트리 네트를 표현하기 위해 소요되는 메모리 공간이 작고, RRSVF는 작을수록 표현이 효율적임을 의미한다.

UB는 상한경계(upper bounds)로 행렬에서 영이 아닌 원소의 최대 갯수를 의미한다.

그러므로

$UB(k) = mn$, $UB(r) = mn$, $UB(l) = \min(k, r)$이다.

LB는 하한경계(lower bounds)로 이론상 영이나, 이는 장소와 천이를 연결하는 아크가 없는 경우로 무의미하여 영이 될 수 없다.

방법 1 : c^-와 c^+ 행렬에서 영이 아닌 원소의 총수는 k+r이고, self-loop를 포함하는 페트리 네트에서 입력아크는 c^-에 출력아크는 c^+에 표현되어 정보를 상실하지 않으므로

$RSVF(1) = (k+r)/(k+r) = 1$,

$RRSVF(1) = (k+r)/(k+r) = 1$이 된다.

여기서 괄호안의 숫자는 해당방법을 의미한다. 방법 1은 RSVF와 RRSVF가 1이므로 다른 방법과 비교시 기준으로 삼는다.

방법 2 : $PN = \langle P, T, C, Mo \rangle$ 에서 $C = C^+ - C^-$에 의해 self-loop의 표현이 불가능하므로 의사

(dummy) 장소와 천이를 추가하여 self-loop를 제거한다. 그 결과 C = k + r + 2l 이 된다. 여기서 2l은 의사 장소와 천이 때문에 추가된 항이다.

RSVF(2) = (k+r+2l)/(k+r) = 1 + 2l/(k+r),

LB [RSVF(2)] = 1, l = 0일 경우.

UB [RSVF(2)] = 1 + 2UB(l)/(k+r), k와 r이 주어질 경우

$$= \begin{cases} 1 + 2/(1+r/k), & k \le r \\ 1 + 2/(1+k/r), & r \le k. \end{cases}$$

A U B [RSVF(2)] = 2, k = r일 경우.

RRSVF(2) = 1 + 2l/(k+r)

UB [RRSVF(2)] = UB [RSVF(2)]

LB [RRSVF(2)] = 1, l = 0일 경우.

여기서 A U B는 절대(absolute) 상한경계이며 최악의 경우 요구되는 메모리 공간이다.

방법 3 :

RSVF(3) = 1 - (2l1+l2)/(2k+r) = 1 - (l+l1) /(2k+r).

$$\text{LB [RSVF(3)]} = \begin{cases} 1 - 1/(1+r/2k), & k \le r \\ 1 - 2/(1+2k/r), & r \le k. \end{cases}$$

ALB [RSVF(3)] = 1/3, k = r일 경우. 여기서 ALB는 절대 하한경계이며 최선의 경우 요구되는 메모리 공간이다.

UB [RSVF(3)] = 1.

RRSVF(3) = 1 + (k-l-l1)/(k+r).

UB [RRSVF(3)] = 1 + k/(k+r), l = 0일 경우.

A U B [RRSVF(3)] ≅2, r ⟨⟨k.

$$\text{LB [RRSVF(3)]} = \begin{cases} 1 - 1/(1+r/k), & k \le r \\ 2 - 3/(1+k/r), & r \le k. \end{cases}$$

ALB [RRSVF(3)] = 1/2, k = r일 경우.

방법4 :

RSVF(4) = 1 - l1/(k+r)

UB [RSVF(4)] = 1, l1 = 0일 경우

$$\text{LB [RSVF(4)]} = \begin{cases} 1 - 1/(1+r/k), & k \le r \\ 1 - 1/(1+k/r), & k \ge r \end{cases}$$

ALB [RSVF(4)] = 1/2, k = r일 경우.

RRSVF(4) = 1 - l1/(k+r)

UB [RRSVF(4)] = 1, l1 = 0일 경우.

LB [RRSVF(4)] = LB [RSVF(4)]

방법 5 : D 행렬은 임의의 천이 t_j에 대하여 $I(t_j) \cap O(t_j) \neq \phi$일 때는 s로 $I(t_j) \cap O(t_j) = \phi$일 때는 정수로 표시하므로 self-loop가 있을 때 k와 r에서 각각 한 원소가 없어지고 s로 대치되므로 영이 아닌 원소의 수는 k + r - l 이 된다.

그러므로

$$RSVF(5) = \frac{k + r - l}{k + r} = 1 - \frac{l}{k + r}$$

$$RRSVF(5) = \frac{k + r - l}{k + r} = 1 - \frac{l}{k + r}$$

이 되어 방법 4와 공간 복잡도가 같고 상한 경계와 하한 경계도 같다. 그러므로 방법 4와 5에서 요구되는 메모리 공간은 서로 같고 이 방법들이 1, 2 및 3에 비하여 항상 잇점이 있다.

각 방법에 대한 RSVF와 RRSVF 등의 계산 결과는 표 1과 같다.

표 1. 여러가지 표현법에 대한 공간 변수 계수 (a) RSVF (b) RRSVF

Table 1. Space variant factors for various methods of representations. (a) RSVF, (b) RRSVF.

방법	일반적 표현	RSVF 상한경계 $K^{U}_{s}B_{r}$	상한경계 $K^{U}_{\geq}B_{r}$	AUB	하한경계 $K^{L}_{s}B_{r}$	하한경계 $K^{L}_{\geq}B_{r}$	ALB
1	1	1	1	1	1	1	1
2	$1+\frac{2l}{K+r}$	$1+\frac{2}{1+\frac{r}{k}}$	$1+\frac{2}{1+\frac{k}{r}}$	2	1	1	1
3	$1-\frac{2l_1+l_2}{2k+}$	1	1	1	$1-\frac{1}{1+\frac{r}{2k}}$	$1-\frac{2}{1+\frac{2k}{r}}$	$\frac{1}{3}$
4	$1-\frac{l_1}{k+r}$	1	1	1	$1-\frac{1}{1+\frac{r}{k}}$	$1-\frac{1}{1+\frac{k}{r}}$	$\frac{1}{2}$
5	$1-\frac{1}{k+r}$	1	1	1	$1-\frac{1}{1+\frac{r}{k}}$	$1-\frac{1}{1+\frac{k}{r}}$	$\frac{1}{2}$

(a)

방법	일반적 표현	RRSVF 상한경계 $K^{U}_{s}B_{r}$	상한경계 $K^{U}_{\geq}B_{r}$	AUB	하한경계 $K^{L}_{s}B_{r}$	하한경계 $K^{L}_{\geq}B_{r}$	ALB
1	1	1	1	1	1	1	1
2	$1+\frac{2l}{k+r}$	$1+\frac{2}{1+\frac{r}{k}}$	$1+\frac{2}{1+\frac{k}{r}}$	2	1	1	1
3	$1+\frac{k-l-l_1}{k+r}$	$1+\frac{1}{1+\frac{r}{k}}$	$1+\frac{1}{1+\frac{r}{k}}$	2	$1-\frac{1}{1+\frac{r}{k}}$	$2-\frac{3}{1+\frac{k}{r}}$	$\frac{1}{2}$
4	$1-\frac{l_1}{k+r}$	1	1	1	$1-\frac{1}{1+\frac{r}{k}}$	$1-\frac{1}{1+\frac{k}{r}}$	$\frac{1}{2}$
5	$1-\frac{1}{k+r}$	1	1	1	$1-\frac{1}{1+\frac{r}{k}}$	$1-\frac{1}{1+\frac{k}{r}}$	$\frac{1}{2}$

(b)

시간 복잡도(time complexity)는 self-loop를 포함하는 다음 몇 가지 질문에 대하여 조사한다.[8] 방법 3이 방법 1과 2에 비해 계산적으로 효율적이므로 방법 3, 4 및 5에 대하여 시간 복잡도를 비교한다.

질문 1) 페트리 네트가 self-loop를 포함하는가 포함하지 않는가 ?

방법 3 :

성공한 검사

최선의 경우 비교횟수 = 2.

최악의 경우 비교횟수 = $3k + r - (2l_1 + l_2)$.

실패한 검사

최선과 최악의 경우 = $2k + r - (2l_1 + l_2)$.

질문 1)에 대한 평균 시간 복잡도는 $O(\max(k,r))$이다.

방법 4 :

성공한 검사와 실패한 검사의 비교횟수 = 1.

질문 1)에 대한 시간 복잡도는 O(1)이다.

방법 5 : 이 방법에서는 행렬 D에 s가 있으면 self-loop가 존재하므로 성공한 검사와 실패한 검사의 비교횟수 = 1,

질문 1)에 대한 시간 복잡도는 O(1)이다.

질문 2) 장소 pi와 천이 tj사이에 self-loop가 있는가 ?

방법 3 :

성공한 검사

최선의 경우 비교횟수 = k.

최악의 경우 비교횟수 = $3k + r - (2l_1 + l_2)$.

실패한 검사

최선의 경우 비교횟수 = k.

최악의 경우 비교횟수 = $3k + r - (2l_1 + l_2)$.

질문 2)에 대하여 최악의 경우 시간 복잡도는 $O(\max(k,r))$이다.

방법 4 :

성공한 검사와 실패한 검사의 비교횟수 = 1.

질문 2)에 대한 시간 복잡도는 O(1)이다.

방법 5 : 행렬 D에 s가 존재하면 p_i와 t_j 사이에 self-loop가 존재하므로

성공한 검사와 실패한 검사의 비교횟수 = 1,

질문 2)에 대한 시간 복잡도는 O(1)이다.

질문 3) 페트리 네트에 몇개의 self-loop가 있는가 ?

방법 3 :

최대 비교횟수 = $2k + r - (2l_1 + l_2)$.

질문 3)에 대하여 최악의 경우 시간 복잡도는 $O(\max(k,r))$이다.

방법 4 : 행렬 C_p에 있는 영이 아닌 원소의 수가 self-loop의 수이다.

질문 3)에 대한 시간 복잡도는 O(1)이다.

방법 5 : 행렬 D에 있는 s의 수가 self-loop의 수이다.

질문 3)에 대한 시간 복잡도는 O(1)이다.

시간 복잡도의 비교 결과 방법 4와 5의 시간 복잡도는 서로 같고, 이들은 방법 1, 2 및 3에 비하여 항상 계산면에서 효율적이다. 공간 복잡도와 시간 복잡도에 대한 자세한 해석은 Sajal[8] 등에 있다.

Ⅴ. 초기마킹과 접속행렬 비교에 의한 해석

1. 알고리듬

단계 1) 초기토큰 행렬 M_0과 변형된 접속행렬 $D = [d_{ij}]_{mxn}$을 구한다.

여기서 $d_{ij} = \begin{cases} s, & p_i,\ I(t_j) \cap O(t_j) \neq \phi, \\ & I(t_j) = O(t_j) = a \\ 정수, & p_i,\ I(tj) \cap O(tj) = \phi. \end{cases}$

단계 2) d_{ij}의 음수 혹은 s를 포함하는 각 열에 대하여 다음을 실행한다.

$M(p_i)$ d_{ij}. 이 때 $d_{ij} < 0$ 이고 $s = -a$이다.

$M'(p_i) = M(p_i) + d_{ij}$. 이 때 $s = 0$이다.

점화 가능한 천이 번호 t_j를 기록한다. 여기서 M'을 제 1세대 행렬이라 한다.

단계 3) M'를 M으로 대치하고 단계 2에서 구한 t_j의 열에서 원소가 양수인 p_i행을 구한 후, 이 p_i행에서 음수나 s를 포함하는 또 다른 t_j열을 구하여 단계 2를 반복한다.

단계4) M'이 초기 토큰 행렬과 같으면 알고리듬은 끝나게 된다.

단계 1은 변형된 접속행렬을 구하는 단계이고 단계 2는 점화가능 여부를 판단하여 새로운 마킹을 생성하는 단계로 장에서 논리적 타당성이 입증되었으며 단계 3과 4는 다음에 의해 논리적 타당성을 갖는다.

페트리 네트는 장소(p_i) → 천이(t_j) → 장소(p_{i+b}) 혹은 천이(t_j) → 장소(p_{i+b}) → 천이(t_{jc})의 구조이므로 $-d_{ij} = \#(p_i, I(t_j)) = \#(t_j, O(p_i))$, $+d_{i+b\ j} = \#(p_{i+b}, O(t_j)) = \#(t_j, I(p_{i+b}))$, 그리고 $-d_{i+b\ jc} = \#(p_{i+b}, I(t_k)) = \#(t_{jc}, O(p_{i+b}))$이다. 여기서 $\#(p_i, I(t_j))$는 장소 p_i에서 천이 t_j로 입력되는 아크의 가중치이고 $\#(t_j, O(p_{i+b}))$는 천이 t_j에서 출력장소 p_{i+b}로 입력되는 아크의 가중치이다.

따라서 t_j 열의 $-d_{ij}$에서 다음 장소의 입력인 $+d_{i+b\ j}$로 그리고 $+d_{i+b\ j}$행에 포함된 $-d_{i+b\ jc}$의 천이 t_{jc}는 점

화 가능한 천이가 되고 초기 토큰은 시스템의 초기상태로 되돌아 가면 같은 시퀀스가 반복되므로 단계 3과 4는 논리적 타당성을 갖는다..

2. 해석 예

그림 1을 제안된 알고리듬으로 해석한다.

단계 1) 그림 6과 같다.

단계 2) b열이 조건을 만족하므로 M' = 001100이 된다.

단계 3) b열에서 원소가 양수인 행은 p3과 p4 행이다. 먼저 p3행에서 음수를 포함하는 열은 a열이나 조건을 만족하지 못하므로 단계 2가 실행되지 못한다. 다음 p4 행에서 음수를 포함하는 열은 c열이므로 단계 2를 실행하면 M' = 001001이 된다. c열에서 p6 행이 양수이므로 p6 행의 음수를 구하면 e열이 되어 위 과정을 반복하면 그림 7과 같은 결과를 얻는다.

단계 4) 그림 7의 loop 6은 초기 토큰과 같은 상태이므로 알고리듬은 끝나게 된다.

그 결과 가도달성 트리로 실행한 결과인 그림 2와 일치함을 알 수 있다. 여기서 loop 1은 제 1세대 행렬, loop 2는 제 2세대 행렬 등을 의미하며 a, b 및 c 는 천이를, 이들의 나열은 점화 시퀀스를 의미한다. 그림 7의 loop 6에서 6= bceade는 가도달성 트리를 이용하여 얻은 점화 시퀀스 = $t_2, t_3, t_5, t_1, t_4, t_5$와 일치한다.

페트리 네트로 기술된 시스템을 H/W로 실현하려면 점화 시퀀스와 이에 따른 토큰 상태를 알아야 하는데 제안된 알고리듬은 점화 시퀀스 6과 새로운 토큰 상태 M'을 별도의 연산없이 바로 구할 수 있으므로 H/W로 실현할 때 용이할뿐만 아니라 데이터 흐름 경로를 바로 결정할 수 있어 S/W로 실현할 때도 용이하다.

M matrix

p1	1
p2	1
p3	0
p4	0
p5	0
p6	0

D matrix

	a	b	c	d	e
p1	1	-1	0	0	0
p2	-1	-1	0	0	1
p3	-1	1	0	0	0
p4	0	1	-1	0	0
p5	1	0	0	-1	0
p6	0	0	1	1	-1

그림 6. 그림 1의 변형된 접속행렬

Fig. 6. Modified incidence matrix of Fig. 1.

Loop 1 b 0, 0, 1, 1, 0, 0

Loop 2 bc 0, 0, 1, 0, 0, 1

Loop 3 bce 0, 1, 1, 0, 0, 0

Loop 4 bcea 1, 0, 0, 0, 1, 0

Loop 5 bcead 1, 0, 0, 0, 0, 1

Loop 6 bceade 1, 1, 0, 0, 0, 0

그림 7. 제안된 알고리듬에 의한 그림 1의 해석 결과

Fig. 7. Analysis result of Fig. 1 by proposed algorithm.

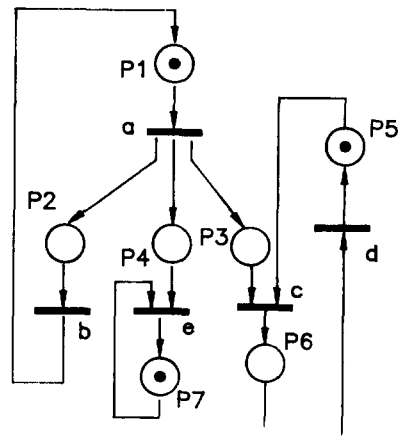

그림 8. 페트리 네트의 예 3

Fig. 8. Example 3 of Petri net.

	a	b	c	d	e
p1	-1	1	0	0	0
p2	1	-1	0	0	0
p3	1	0	-1	0	0
p4	1	0	0	0	-1
p5	0	0	-1	1	0
p6	0	0	1	-1	0
p7	0	0	0	0	s

그림 9. 그림 8의 변형된 접속행렬

Fig. 9. Modified incidence matrix of Fig. 8.

그림 8은 self-loop를 포함한 경우로 접속행

Loop 1 a 0,1,1,1,1,0,1

Loop 2 ab 1,0,1,1,1,0,1
ac 0,1,0,1,0,1,1
ae 0,1,1,0,1,0,1

Loop 3 aba 0,1,2,2,1,0,1
abc 1,0,0,1,0,1,1
abe 1,0,1,0,1,0,1
acb 1,0,0,1,0,1,1
acd 0,1,0,1,1,0,1
ace 0,1,0,0,0,1,1
aeb 1,0,1,0,1,0,1
aec 0,1,0,0,0,1,1

Loop 4 abab 1,0,2,2,1,0,1
abac 0,1,1,2,0,1,1
abae 0,1,2,1,1,0,1
abab 1,0,2,2,1,0,1
abca 0,1,1,2,0,1,1
abcd 1,0,0,1,1,0,1
abce 1,0,0,0,0,1,1
abea 0,1,2,1,1,0,1
abab 1,0,2,2,1,0,1
abec 1,0,0,0,0,1,1
acba 0,1,1,2,0,1,1
acbd 1,0,0,1,1,0,1
acbe 1,0,0,0,0,1,1
acdb 1,0,0,1,1,0,1
acde 0,1,0,0,1,0,1
aceb 1,0,0,0,0,1,1
aced 0,1,0,0,1,0,1
aeba 0,1,2,1,1,0,1
aebc 1,0,0,0,0,1,1
aecb 1,0,0,0,0,1,1
aecd 0,1,0,0,1,0,1

Loop 5 ababa 0,1,3,3,1,0,1
ababc 1,0,1,2,0,1,1
ababe 1,0,2,1,1,0,1
abacb 1,0,1,2,0,1,1
abacd 0,1,1,2,1,0,1
abace 0,1,1,1,0,1,1
abaeb 1,0,2,1,1,0,1
abaec 0,1,1,1,0,1,1
abaee 0,1,2,0,1,0,1
abcab 1,0,1,2,0,1,1

그림 10. 그림 8를 제안된 알고리듬으로 해석한 결과

Fig. 10. Analysis result of Fig. 8 by proposed algorithm.

있는 s는 self-loop를 의미한다. 그림 8의 해석 결과는 그림 10과 같다. 여기서 시퀀스가 진행함에 따라서 토큰수가 계속 증가하므로 이 페트리 네트는 무제한적임을 알 수 있다.

Ⅵ. 결 론

최근 공장자동화에서는 비동기성 및 병렬성을 갖는 시스템을 모델링하기 용이한 언어가 요구되고 있는데 그중 대표적인 것이 페트리 네트이다.

본 논문에서는 페트리 네트로 모델링한 시스템의 완전성 여부를 확인하기 위해서 해석하는 새로운 알고리듬을 제안하였다. 이 방법은 self-loop의 표현이 가능할 뿐만 아니라 Sajal [8] 등이 제안한 해석 방법과 공간 복잡도와 시간 복잡도 면에서는 같으나 별도의 연산과정 없이 점화 시퀀스의 결정이 가능하여 이를 S/W 나 H/W로 실현할 때 용이하다.

參 考 文 獻

[1] T. Murata, N.Komoda, et al., "A Petri net-based controller for flexible and maintainable sequence control and it's applications in factory automation," *IEEE Trans.* Ind. Elec., vol.IE-33, no. 1, pp. 1-8, 1986.

[2] M. Lioyd, "GRAFCET - graphical function chart programming," Proceeding Conference Programmable Controllers, London, pp.51-56, 1985.

[3] J. L. Peterson, "Petri net theory and the modeling of systems," Prentice-hall, 1981.

[4] T. Murata, "Petri nets: properties analysis and application," Proceedings of the IEEE, vol. 77, no. 4, pp. 541-580, 1989.

[5] S.M. Chen, J.S.Ke and J.F. Chang, "Knowledge representation using fuzzy Petri nets," IEEE Trans. on Knowledge and Data Engineering, vol.2, no. 3, pp. 311-319, 1990.

[6] M. Llyas and H. Khalil, "Modeling of communication protocols by using Petri nets," Proceedings of 3th Annual Conference on Computers and Industrial

Engineering, vol.1, pp. 547-551, 1986.
[7] J. Prock, "A new technique for fault detection using Petri nets," Automatica, vol. 27, no. 2, pp. 239-245, 1991.
[8] S. K. Das and V. K. Agrawal, et al., "Reflexive incidence matrix(RIM) representation of Petri nets," *IEEE Trans.* on Software Engineering, vol. se-13, no. 6, pp. 643-654, 1987.

著 者 紹 介

禹 廣 俊(正會員)

1946年 11月 8日生. 1974年 2月 한양대학교 전자공학과 졸업(학사). 1977年 2月 한양대학교 대학원 전자공학과 졸업(석사). 1983年 1月 Institut National Polytechnique de Grenoble(박사). 1983年 ~ 현재 단국대학교 전자공학과 교수. 주관심 분야는 Brushless D.C. motor servo control, 공장자동화 등임.

柳 昶 根(正會員)

1956年 2월 19日生. 1981年 2月 단국대학교 전자공학과 졸업(학사). 1983年 8月 단국대학교 대학원 전자공학과 졸업(석사). 1993年 8月 단국대학교 대학원 전자공학과 박사과정 졸업예정. 주관심 분야는 페트리 네트, 공장자동화 등임.