페트리 네트와 퍼지 개념을 이용한 자동 조립 시스템 제어

고 인 선, 전 광 호

홍익대학교 전자공학과

1. 서 론

최신에 설계되는 생산 시스템은 제품 제조의 경비 절감과, 소비자의 다양한 욕구를 만족시키는 다품종 소량 생산 체제로의 전환과 제품 생산의 고속화를 추구하고 있다. 이것은 생산업계에 FA(Factory Automation)를 확산시켜 나가려는 움직임을 유발시켰으며, 컴퓨터 제어 응용으로 인하여 더욱 가속화 되었다. FA의 한 분야인 자동 조립 시스템은 다양한 종류의 로보트, 컨베이어, 무인 반송차(AGV) 등을 이용하여, 자동화된 조립 과정을 통해 인력, 경비와 작업 위험을 줄이고 생산 능률을 극대화 하는 것이 목표이다.

자동 조립 시스템은 이산 사건 시스템(Discrete Event System)으로 분류될 수 있다. 이산 사건 시스템은 시스템 구성 요소에서 발생되는 불연속적인 변화에 의해 상태가 변화한다. 이산 사건의 특징으로는 사건의 발생이 비동기, 병렬적이고, 사건의 발생이 다른 사건의 발생을 유도(Event Triggering)하는데 있다.

이산 사건 시스템을 분석하고 제어하기 위하여 여러 가지 방법들이 제안되었다. 페트리 네트(Petri Nets)는 시스템을 도식적, 계층적으로 모델링할 수 있고, 사건 발생의 동시성, 비동기성, 분산성, 병렬성 특징을 간단하게 표현할 수 있다. 페트리 네트로 시스템을 모델링하고 제어할 때 발생하는 문제점중 하나는 외부에서 시스템으로 들어오는 입력들의 표현이 어렵다는데 있다. 입력 변수들이 다양해지고, 입력 값들이 모호성을 지닌다면, 외부 시스템과 페트리 네트로 모델링된 시스템을 연결하는 것은 더욱 어렵게 된다.

일반적으로 생산 시스템은 복잡하고 크기 때문에, 완벽히 수학적으로 그 생산 시스템을 모델링하고, 생산 능률을 수학적으로 표현하여, 그 결과에 의하여 어떠한 제어 신호를 보낼 것인가는 결정 하는것은 매우 어렵고 경비가 많이 소요되는 작업이다. 모델링된 시스템에서 다수개의 사건을 발화시키는 것이 가능하고 그 중 하나를 골라 제어 신호를 보내야 한다고 가정할 때, 시스템의 생산성을 고려하여 그 중의 하나를 선택하여야 한다. 이럴 경우 그 생산 시스템의 전문가의 지식을 이용하여 제어하는 방법이 적절하다. 페트리 네트로 부터 제어 신호를 발생 시키려고 할때도 이러한 경우가 많이 발생한다. 이 현상을 충돌 현상(Conflict Situation)이라고 한다. 본 논문에서는 외부의 데이터를 모델링된 시스템으로 입력시키는 방법으로 퍼지 개념을 사용하여, 충돌 현상을 해결하였다. 이 방법의 이해를 돕기 위하여, 자동 조립 시스템의 예로, 모터를 조립하는 소규모의 자동화 시스템을 페트리 네트를 이용하여 제어하고, 이때 발생하는 충돌 현상을 퍼지 개념을 이용하여 해결하였다.

아래는 다음과 같은 순서로 구성되었다. II 장에서는 간단히 페트리 네트를 소개한다. III 장에서는 페트리 네트로 모델링된 시스템의 퍼지 개념이 어떻게 사용되었는지를 보인다. IV 장에서는 페트리 네트를 이용하여 모터 자동 조립 시스템을 제어하는데 발생하는 충돌 현상을 해결하는 과정을 제시한다. 마지막으로 결론을 맺는다.

2. 페트리 네트

페트리 네트는 플레이스(Place)와 트랜지션(Transition)이라는 두 종류의 노드(Node)와 호선(Arc)으로 구성되어 있다. 플레이스는 원으로, 트랜지션은 막대 또는 네모난 상자로 표현하고, 일반적인 의미는 표 1과 같다. 입력 함수는 각 트랜지션과 입력 플레이스의 관계, 출력 함수는 각 트랜지션과 출력 플레이스의 관계를 표현 한다.

표 1. 플레이스와 트랜지션의 전형적인 의미.

입력플레이스 (Input Place)	트랜지션 (Transition)	출력플레이스 (Output Place)
선행조건 (Precondition)	사건 (Event)	후행조건 (Postconditiom)
입력데이터 (Input Data)	연산 (Computation)	출력데이터 (Output Data)
입력신호 (Input Signal)	신호처리 (Signal Processor)	출력신호 (Output Signal)
⋮	⋮	⋮

호선은 트랜지션과 플레이스의 관계를 나타내므로, 플레이스에서 트랜지션으로 또는 트랜지션에서 플레이스로 향하는 호선이 존재하게 된다. 페트리 네트는 초기 마킹(Initial Marking) M_0라는 초기 상태를 지니는데, 플레이스 p가 양의 정수 k로 마킹되면 플레이스 p에는 k개의 토큰이 있다는 것이다. 마킹은 플레이스 내의 검은 점으로 표시 한다.

페트리 네트 PN은 다음과 같은 네개의 구성요소로 이루어진다.

$PN=\langle P, T, A, M_0\rangle$
P : 플레이스의 유한집합 ; $P=\{p_1, p_2, p_3, \cdots, p_n\}$,
T : 트랜지션의 유한집합 ; $T=\{t_1, t_2, t_3, \cdots, t_m\}$,
A : 입력과 출력 호선들의 관계 매트릭스 ; $A\subseteq\{P\times T\} \cup \{T\times P\}$,
M_0 : 초기 마킹(Initial Marking)

특정 트랜지션의 입력 플레이스가 호선 수 이상의 토큰을 가지고 있을 때, 이 트랜지션은 활성화되었다고 하고, 이를 식으로 나타내면 아래와 같다.

$$\forall p\in {}^{\circ}t_i, \quad M(p) \geq A(p, t_i)$$

M은 각 플레이스의 토큰 수를 나타내는 n×1벡터(Vector)로서 $M(p)$는 플레이스 p_i내에 있는 토큰의 갯수이고, ${}^{\circ}t_i$는 트랜지션 t_i의 입력 플레이스의 집합이다. 활성화된 트랜지션은 호선으로 연결된 각 입력 플레이스의 토큰을 제거하고 출력 플레이스들로 연결된 호선의 수 만큼 토큰을 내보내는데 이를 발화(Firing)라고 한다. 자세한 내용은 참고문헌[1]을 참조하시오.

그림 1에서 플레이스 p는 트랜지션 t_1, t_2, t_5, t_6의 입력 플레이스이고, 각 트랜지션의 다른 입력 플레이스도 발화 조건을 만족하므로 트랜지션 t_1, t_2, t_5, t_6 중 하나만 선택하여 발화할 수 있다. 그러므로 이 중 어느 트랜지션을 선택하여 발화 시키는 Decision Making 문제가 발생하게 된다. 이

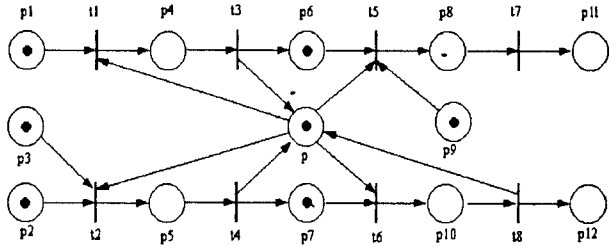


그림 1. 충돌 현상 1.

러한 현상을 충돌 현상(Conflict Situation)이라 하는데, 이는 플레이스 p가 트랜지션 t_1, t_2, t_5, t_6에 독립적이지 않기 때문이다. 조건(Condition) p가 사건(Event) t_1, t_2, t_5, t_6 중 하나만 발화 할수 있고, 사건 t_1, t_2, t_5, t_6를 동시에 발화시킬 수 없는 현상이다. 그림 1과 같은 충돌 현상을 해결하는 방법으로 단순 우선 순위 결정에 의해 트랜지션을 발화하는 방법이 쓰이고 있다. 그러나 시스템의 성능에 이 발화 순위가 적지 않은 영향을 미칠때, 단순 우선 순위 방법은 적절하지 않다.

3. 퍼지 개념을 이용한 충돌 현상 해결

제어 규칙은 If-Then 형태를 갖는 퍼지 조건문으로 표시된 언어적 표현을 사용한다. 일반적으로 If-Then 형태를 갖는 페트리 네트에서의 발화 제어 규칙은 다음과 같이 나타낼 수 있다.

"If(p_1 is true) and (p_2 is true) and ⋯ (p_n is true), Then one of possible firing transitions is fired."

위에서 true는 해당 플레이스에 토큰이 존재함을 나타낸다. 다수의 발화 가능한 트랜지션들이 있으면, 시스템은 충돌 현상하에 있게 된다. 이때에 미리 정해진 발화 우선 순위에 관한 퍼지 소속 함수값을 이용하여 가장 상황에 적합한(시스템의 성능을 최대한으로 하는) 트랜지션을 발화한다.

그림 1에서 페트리 네트의 일반적인 발화 규칙 일부는 아래와 같이 If − Then 형식으로 표현할 수 있다.

"If(p is true) and (p_1 is true), Then t_1 is fired."
" If(p is true) and (p_2 is true) and (p_3 is true), Then t_2 is fired."
"If(p is true) and (p_7 is true), Then t_6 is fired."
"If(p is true) and (p_6 is true) and (p_9 is true), Then t_5 is fired."

그림 1에서 초기에 p, p_1, p_2, p_3, p_9에만 토큰이 존재하면, 발화 규칙에 의하여 t_1과 t_2 중 어느 하나를 발화하여야 한다.

이와 같은 일반적인 충돌 현상의 발화 순위를 해결하기위해 충돌 현상에 관계하는 플레이스와 트랜지션으로부터 아래와 같은 규칙을 만들 수 있다.

"If(p is true) and (p_i is true)$_n$ Then one of t_j is fired."

where $i=1, 2, 3, 6, 7, 9$ and $j=1, 2, 5, 6$.

위와 같이 주어진 규칙에 의하여 표 2의 규칙 행렬을 만들 수 있다. 첫행에서 0.7, 0.8, 1, 0.9 는 t_1, t_2, t_5, t_6가 모두 동시에 발화 조건을 만족하였을 때의 우선 순위를 나타내고 있다. 일곱째 행의 0.7은 p_9에 토큰이 있으면 0.7 정도의 발화 가중치가 t_1에 첨가됨을 표시하고 있다. 이 우선 순위의 가중치들은 시스템 외부의 지식을 시스템 내부로 들여온 것이다. 빈칸은 0을 나타낸다.

표 2. 규칙 행렬 R_1.

	p	p_1	p_2	p_3	p_6	p_7	p_9	t_1	t_2	t_5	t_6
p	1										
p_1		1									
p_2			1								
p_3				1							
p_6					1						
p_7						1					
p_9							1				
t_1	0.7	1					0.7	0			
t_2	0.8		1	1					0		
t_5	1				1		1			0	
t_6	0.9					1					0

규칙 행렬 R_1이 정해지면 출력 O는 다음과 같이 주어진다.

$$O=R_1\times U$$

입력 U는 현재 각 플레이스의 토큰의 수를 나타내고, 출력 O는 각 플레이스의 토큰의 수와 충돌 현상하의 각 트랜지션의 우선 순위를 나타낸다. 만약 그림 1에 나와 있는 시스템의 초기에 플레이스 p, p_1, p_2, p_3, p_9에 토큰이 존재한다면 입력 행렬 U_1은 다음과 같다.

$$U_1=[1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$$

U_1과 발화 규칙으로부터 t_1과 t_2가 발화 가능함을 알 수 있다. 출력 행렬 O_1을 구하면,

$$O_1=[1\ 1\ 1\ 1\ 0\ 0\ 1\ 2.4\ 2.8\ 2.0\ 0.9]$$

O_1에서 7열까지는 각 플레이스의 토큰 수를 나타내는 것이고 8열부터 11열까지는 트랜지션의 발화 우선 순위를 나타낸다. 따라서 8열부터 11열에서 원소의 값이 최대인 열을 취하면 2.8을 나타내는 9열이다. 9열은 t_2를 나타내므로, 이 상황에서 시스템은 t_2를 발화 시키는 제어신호를 보낸다. 이 경우는 단순히 t_1과 t_2의 발화 우선 순위를 비교한 것과 동일하다. 트랜지션 t_2가 발화된 후, 플레이스 p_5에 의해 트랜지션 t_4가 발화된다. 그 결과로 플레이스 p는 다시 충돌 현상에 놓이게 된다(발화 가능 트랜지션 : t_1, t_6). 이 때의 입력을 U_2라 하면 출력 O_2는 다음과 같다.

$$U_2=[1\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0]^T$$
$$O_2=[1\ 1\ 0\ 0\ 0\ 1\ 1\ 2.4\ 0.8\ 2.0\ 1.9]$$

O_2에 의해 트랜지션 t_1이 발화된다. 이 경우의 발화는 t_1, t_6의 발화 순위를 단순 비교한 것과는 다른 결과가 나온다. 이는 발화 가중치가 t_1에만 더해진 결과이다. p_4에 의해 t_3가 발화되고 그 결과로 플레이스 p는 다시 충돌 현상에 놓이게 된다(발화 가능 트랜지션 : t_5, t_6). 이 때의 입력 U_3와 출력 O_3는 다음과 같다.

$$U_3=[1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0]^T$$
$$O_3=[1\ 0\ 0\ 0\ 1\ 1\ 1\ 1.4\ 0.8\ 3.0\ 1.9]$$

O_3에 의해 플레이스 p는 트랜지션 t_5를 발화하게 된다. 이와 같은 방법으로, 페트리 네트로 모델링할 수 없는 전문가의 여러 정보들이 규칙 행렬을 통해서 시스템 제어에 이용된다.

외부의 정보들이 단순하게 규칙 행렬의 상수값으로 표현되기 어려울때 퍼지 개념을 이용하여 충돌 현상을 아래와 같은 방법으로 해결하였다.

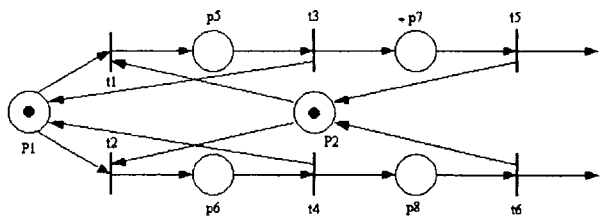


그림 2. 충돌 현상 2.

그림 2에서 p_1과 p_2는 충돌 현상하에 있다. p_1은 t_1과 t_2를 발화하는데 있어 우선 순위가 같다고 할 때, t_1과 t_2의 발화 결정은 p_2에 의해 이루어진다. p_2가 t_1과 t_2중 어느 것을 발

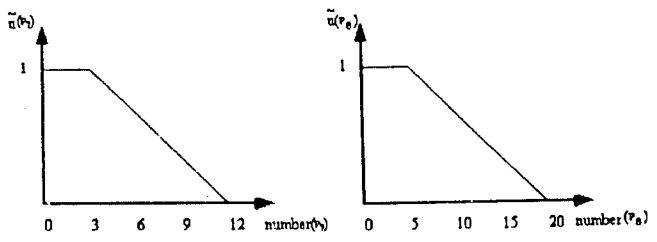
화하는 가는 p_7과 p_8의 토큰의 갯수에 의하여 결정될 경우를 취급하고져 한다. 이 때의 규칙 행렬은 표 3과 같다. 표 3에서 *는 토큰수에 의해 값이 변함을 뜻한다.

표 3. 규칙 행렬 R_2.

	P_1	P_2	p_3	p_4	t_1	t_2
P_1	1					
P_2		1				
p_3			1			
p_4				1		
t_1	1	*	1		0	
t_2	1	*		1		0

$m_o(p_7)=6$ and $m_o(p_8)=4$ 일때, 발화 우선 순위가 그림 3과 같이 퍼지 함수로 표시된다고 하면, p_7이 t_5를 발화하여 p_2에 보낸 정보 $-t_1$에 대한 발화 요구는 0.66 으로 결정되고, p_8이 t_6을 발화하여 p_2에 보낸 정보 $-t_2$에 대한 발화 요구는 1로 결정되어지므로 규칙 행렬의 원소 $R_2(5, 2)=0.66$, $R_2(6, 2)=1$로 주어지게 된다. 이 결과 그림 2의 충돌 현상에서는 t_2가 우선 순위가 높게 되므로 발화된다. t_2의 발화로 t_4가 발화되게 되고 이로 인해 p_8에 존재하는 토큰의 갯수가 15개로 늘어났다면 p_8이 t_6을 발화하여 p_2에 보낸 정보-트랜지션 t_2에 대한 발화 요구는 0.33으로 결정되어지므로 $R_2(6, 2)=0.33$ 이 된다. 따라서 규칙 행렬 R_2의 원소 *는 토큰수에 따라 변하고, 그 결과에 의해 충돌 현상에 놓여 있는 트랜지션들의 발화가 결정된다.

본 논문에서는 이와 같은 방법을 이용하여, 자동 조립 시스템을 페트리 네트로 제어할 때 발생하는 충돌 현상을 해결하였다.



(a) (b)

그림 3. 플레이스 p_7과 p_8에 있는 부품의 수량에 대한 소속 함수값.
(a) 플레이스 p_7의 토큰수에 대한 소속 함수값
(b) 플레이스 p_8의 토큰수에 대한 소속 함수값

Ⅳ. 자동 조립 시스템 제어

그림 4와 같은 모터를 조립하는 소규모의 지능적인 자동화 시스템을 제어하고져 한다. 시스템은 2대의 AGV (AGV-1, AGV-2)와 1대의 Conveyer System, 4대의 Robot Arm (고정용 2대, 조립용 2대), 2개의 Workcell, 그리고 2대의 지능 제어용 Workstation으로 구성 되어 있다. 그림 5와 같이, 모터의 부품은 모터 Frame, Stator Assembly, Rotor Assembly, Shaft, 2개의 Bearing (전 후 구별 없음), 2개의 End Bracket (Front, Rear의 구별이 있음)으로 구성 되어 있다. 부품들은 각각의 공정이 끝난 완전한 부품으로 구성 되어 있다고 가정한다. 이 후부터는 용어를 표 4의 약자로 표기 한다.

표 4. 시스템의 제원.

WS-1	Workstation (1)
WS-2	Workstation (2)
Robot-1	Robot Arm (조립용)
Robot-2	Robot Arm (고정용)
Robot-3	Robot Arm (조립용)
Robot-4	Robot Arm (고정용)

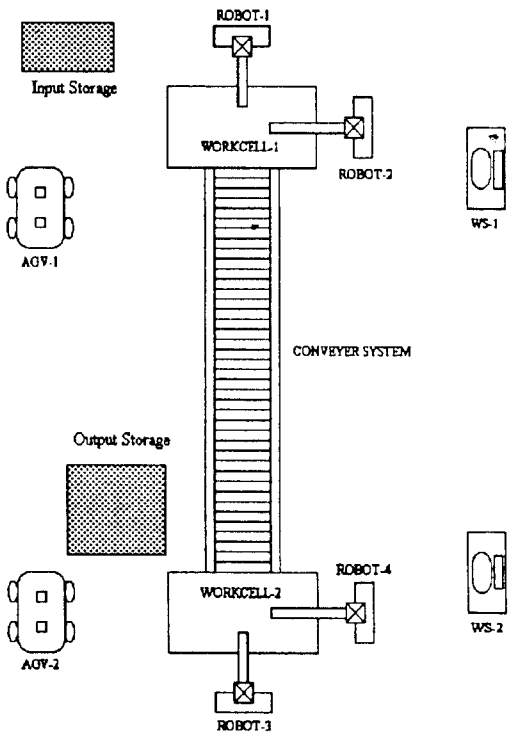
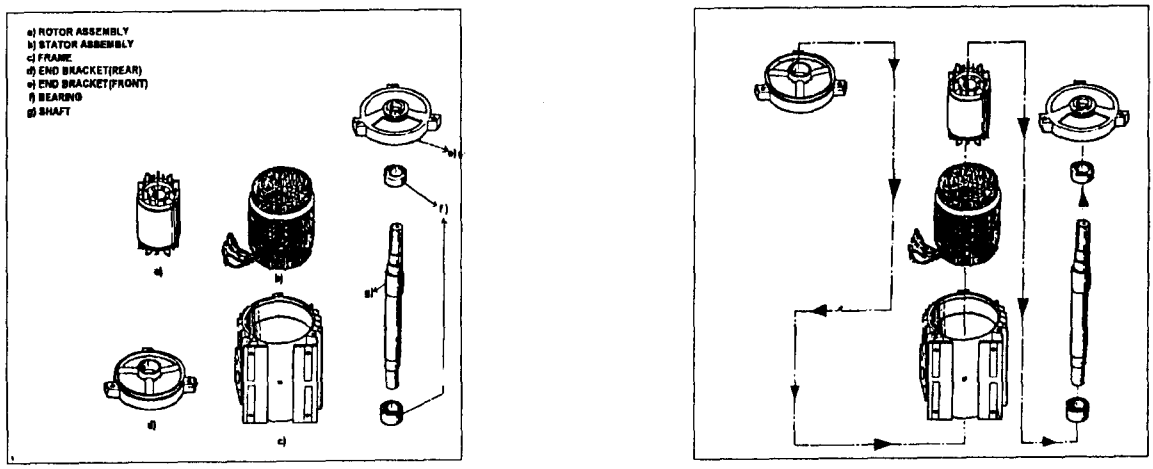


그림 4. 자동 조립 시스템.



(a) (b)

그림 5. Motor; (a)Motor의 부품, (b)조립도.

Workcell-1과 Workcell-2의 작업은 아래의 순서로 작동 한다.

Workcell-1

I-1) AGV-1은 부품 창고 (Input Storage)에서 Motor Frame이 담긴 Pallette를 Workcell-1의 위치로 운반한다. 그 후 AGV-1은 Workcell-1에 도달했음을 WS-1에 알린다.

I-2) WS-1은 Workcell-1에 설치되어있는 Robot-1으로 하여금 Workcell-1의 지정된 위치에 Motor Frame이 담긴 Pallette를 놓도록 지시한다.

I-3) Workcell-1에 설치되어있는 Robot-1은 Motor Frame이 담긴 Pallette를 지정된 위치에 놓는다.

I-1~3)의 작업을 End Bracket (Rear), Stator Assembly, Rotor Assembly에 대해서도 한다.

I-4) WS-1은 Robot-2으로 하여금 Motor Frame을 집어, Robot-1이 End Bracket (Rear)을 조립할 수 있게끔, 회전하게 한다.

I-5) WS-1은 Robot-1이 End Bracket (Rear)을 집어 Motor Frame에 조립을 하게 한다.

I-6) WS-1은 Robot-1이 Stator Assembly를 집어 Motor Frame에 조립을 하도록 지시한다.

I-7) WS-1은 Robot-1이 Rotor Assembly를 집어 Motor Frame에 조립을 하도록 지시한다.

I-4, 5, 6, 7)의 작업 도중 Pallette의 내용물이 비었을 시 Sensor가 감지하여 WS-1에 알리고 WS-1은 지정된 부품 Pallette를 AGV-1에게 가지고 올 것을 지시한다.

I-8) WS-1은 Robot-2로 하여금 조립이 완료된 Motor Frame을 Buffer-1에 저장하도록 한다.

I-9) WS-1은 I-4~8)의 작업을 반복한다.

I-10) Buffer-1에 반조립된 Motor Frame이 5개가 저장되면 WS-1에게 알린다.

I-11) WS-1은 I-10)과 II-2)의 동작이 완료되면 Conveyer System을 동작시킨다.

I-12) WS-1은 I-4~11)의 작업을 반복한다.

Workcell-2

II-1) AGV-2와 Robot-3은 Shaft, Bearing, End Bracket (Front)에 대해서 I-1~3)과 같은 작업을 Workcell-2의 위치로 한다.

II-2) WS-2는 WS-1에게 Workcell-2의 작업 준비가 끝났음을 알린다.

II-3) Conveyer System에 의해 Workcell-1로부터 운반되어진 반조립된 Motor Frame을 Sensor가 감지하여 WS-2에 알린다.

II-4) WS-2는 Robot-4로 하여금 Motor Frame을 집어 고정한다.

II-5) WS-2는 Robot-3로 하여금 Bearing을 집어 Motor Frame에 조립할 것을 지시한다.

II-6) WS-2는 Robot-3으로 하여금 Shaft를 집어 Motor Frame에 조립할 것을 지시한다.

II-7) WS-2는 Robot-3으로 하여금 Bearing을 집어 Motor Frame에 조립할 것을 지시한다.

II-8) WS-2는 Robot-3으로 하여금 End Bracket (Front)를 집어 Motor Frame에 조립할 것을 지시한다.

II-5, 6, 7, 8)의 작업 도중 Pallette의 내용물이 비었을 시 Sensor가 감지하여 WS-2에 알리고 WS-2은 지정된 부품 Pallette를 AGV-2에게 가지고 올 것

을 지시한다.

II-9) WS-2는 Robot-4로 하여금 완전히 조립된 Motor를 Buffer-2에 저장할 것을 지시한다.

II-10) WS-2는 AGV-2로 하여금 완성된 Motor를 창고 (Output Storage)로 운반할 것을 지시한다.

II-11) WS-2는 II-4~10)의 작업을 반복한다.

상기의 자동 조립 시스템을 페트리 네트로 모델링한 결과는 그림 6에 나타나 있고, 네모로 표시된 플레이스는 동일 플레이스를 나타낸다. 주요 플레이스와 트랜지션의 의미는 다음과 같다.

P_s	Sensor가 Available한 상태 P_{a-1} AGV-1이 Available한 상태
P_{a-2}	AGV-2가 Available한 상태 P_{r-1} Robot-1이 Available한 상태
P_{r-2}	Robot-2가 Available한 상태 P_{r-3} Robot-3이 Available한 상태
P_{r-4}	Robot-4가 Available한 상태 P_{w-1} WS-1이 Available한 상태
P_{w-2}	WS-2가 Available한 상태
$p_1 \sim p_8$	부품이 Input Storage에 있는 상태
t_{17}	Motor Frame에 End Bracket (Rear)를 조립 시작
t_{19}	Motor Frame에 Stator Assembly를 조립 시작
t_{21}	Motor Frame에 Rotor Assembly를 조립 시작
t_{38}	반조립된 Motor에 Bearing을 조립 시작
t_{40}	반조립된 Motor에 Shaft를 조립 시작
t_{42}	반조립된 Motor에 End Bracket (Front)를 조립 시작

그림 6에 나타난 것과 같이 Robot-1, Robot-2, Robot-3, Robot-4, AGV-1, AGV-2, WS-1, WS-2를 나타내는 플레이스 P_{r-1}, P_{r-2}, P_{r-3}, P_{r-4}, P_{a-1}, P_{a-2}, P_{w-1}, P_{w-2}는 충돌 현상하에 있다. 이는 각각의 제원들을 필요로 하는 작업 단계가 여러개 존재하기 때문이다. 따라서, 각각의 작업을 수행함에 있어, 이들 제원들의 적절한 배치가 이루어져야 한다.

Robot-1이 수행하여야 할 작업은 AGV-1이 부품창고에서 실어온 부품이 담긴 Pallette를 Workcell-1의 지정된 위치로 옮기는 것과 조립을 하는 것으로 나누어진다. Robot-1이 부품이 담긴 Pallette를 Workcell-1의 지정된 위치로 옮기는 작업을 나타내는 플레이스는 p_9, p_{12}, p_{15}, p_{18}이고 각각의 부품을 조립하는 작업을 수행하는 것을 나타내는 플레이스는 p_{30}, p_{32}, p_{36}이다. 플레이스 p_9, p_{12}, p_{15}, p_{18}, p_{30},

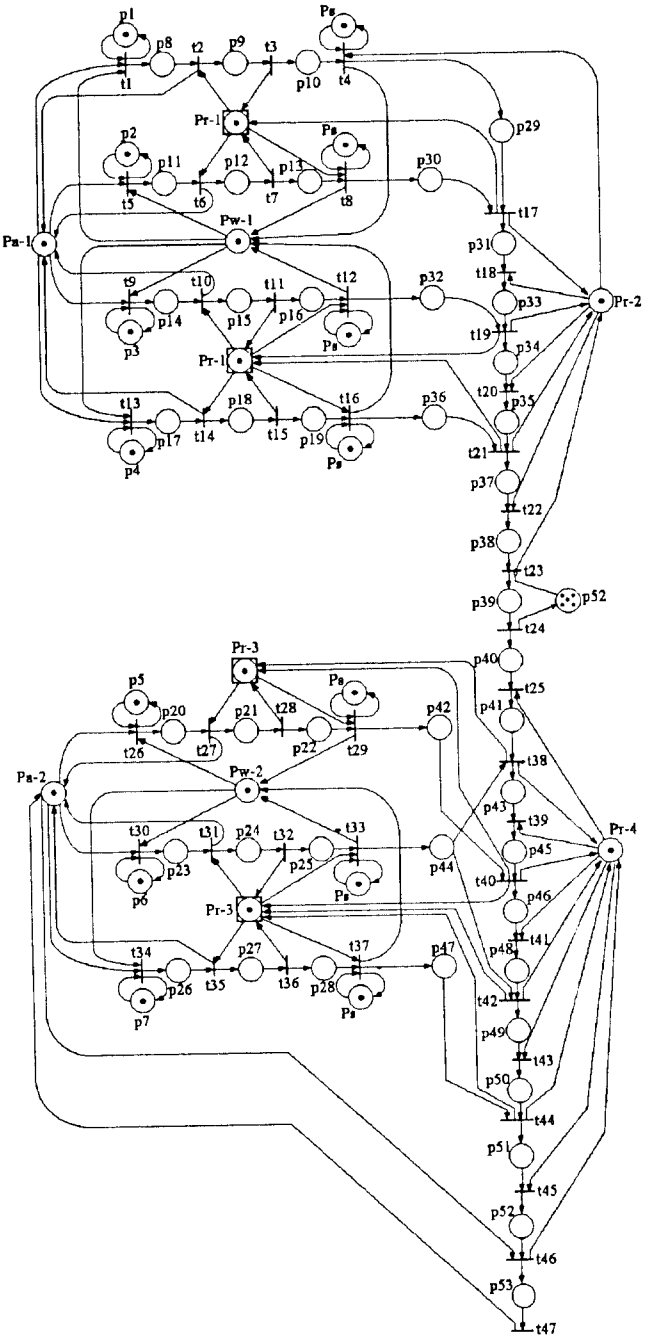


그림 6. 자동 조립 시스템의 페트리 네트.

p_{32}, p_{36}에 토큰이 있다는 것은 그 플레이스가 나타내는 작업을 Robot-1이 수행하고 있다는 것이다. 따라서 플레이스 p_9, p_{12}, p_{15}, p_{18}, p_{30}, p_{32}, p_{36}에는 동시에 토큰이 존재 할 수 없다. 또한 AGV-1이 부품을 실어나르는 작업을 나타내는 플레이스는 p_8, p_{11}, p_{14}, p_{17}이므로 이 플레이스들도 동시에 토큰이 존재할 수 없다. 이러한 전제조건에 의해 다음과 같은 규칙을 세울 수 있다.

"If (AGV-1이 부품이 담긴 Pallette를 실어온다)

Then (Robot-1은 부품을 Workcell-1의 지정된 위치로 옮긴다)"

"If (p_i is true $i=8,11,14,17$) Then (t_k is fired $k=2,6,10,14$)" ;

"If (Workcell-1의 지정된 위치에 부품이 있다) Then (Robot-1은 부품을 조립한다)"

"If (p_j is true $j=13,16,19$) Then (t_l is fired $l=8,12,16$)" ;

또한 Robot-1은 Robot-2의 작업수행에 있어서도 영향을 받기 때문에 다음과 같은 Rule이 존재하게 된다.

"If (Robot-2가 조립을 위해 Motor Frame을 집는다) Then (Robot-1은 필요한 부품을 집어 조립을 한다)"

"If (p_n is true $n=29,33,35$) Then (t_h is fired $h=8,12,16$)" ;

따라서 Robot-1에 대한 규칙은 다음과 같이 나타낼 수 있다.

"If (p_i is true $i=8,11,14,17$) AND (p_j is true $j=13,16,19$) AND (p_n is true $n=29,33,35$) AND (P_{r-1} is true) Then (t_m is fired $m=2,6,14,8,12,16$)"

위의 규칙에 의해 Robot-1에 대한 규칙 행렬 R_1은 표 5와 같다.

표 5. Robot-1에 대한 규칙 행렬 R_1.

	Pr-1	p_8	p_{11}	p_{14}	p_{17}	p_{10}	p_{13}	p_{16}	p_{19}	p_{29}	p_{33}	p_{35}	t_2	t_6	t_{10}	t_{14}	t_8	t_{12}	t_{16}
Pr-1	1																		
p_8		1																	
p_{11}			1																
p_{14}				1															
p_{17}					1														
p_{10}						1													
p_{13}							1												
p_{16}								1											
p_{19}									1										
p_{29}										1									
p_{33}											1								
p_{35}												1							
t_2	0.4	1				0.5							0						
t_6	0.5		1				0.5							0					
t_{10}	0.6			1				0.5							0				
t_{14}	0.7				1				0.5							0			
t_8	0.8						1			0.7							0		
t_{12}	0.9							1			0.7							0	
t_{16}	1								1			0.7							0

같은 방법으로 Robot-2에 대해서도 규칙 행렬 R_2를 정의 할 수 있다. Robot-2가 수행하여야 할 작업은 부품을 조립하는 작업과 반조립된 Motor를 Conveyer System에 실는 작업으로 나누어진다. Robot-2가 조립을 위해 Motor Frame을 집는 작업을 뜻하는 플레이스는 p_{29}, p_{33}, p_{35}이고, Workcell-2로 반조립된 부품을 옮기기 위해 Conveyer System으로 나르는 작업을 뜻하는 플레이스는 p_{38}이다. 즉, p_{29}, p_{33}, p_{35}, p_{38}는 동시에 토큰이 존재할 수 없다. 따라서 다음과 같은 규칙을 세울수 있다.

"If (Workcell-1의 지정된 위치에 Motor Frame이 있다) Then (Robot-2는 부품을 집는다)"

"If (p_i is true $i=10,31,34$) Then (t_k is fired $k=4,18,20$)" ;

"If (Motor가 반조립되었다) Then (Robot-2는 반조립된 Motor를 Conveyer System에 실는다)"

"If (p_{37} is true) Then (t_{22} is fired)" ;

Robot-2도 Robot-1의 작업수행에 있어서 영향을 받기 때문에 다음과 같은 규칙이 존재하게 된다.

"If (Robot-1이 조립을 위해 부품을 집는다) Then (Robot-2는 Motor Frame을 집는다)"

"If (p_n is true $n=30,32,36$) Then (t_l is fired $l=4,18,20$)" ;

Robot-2에 관한 규칙은 다음과 같이 나타낼수 있다.

"If (p_i에 is true $i=10,31,34$) and (p_{37} is true) and (p_n is true $n=30,32,36$) and (P_{r-2} is true) Then (t_m is fired $m=4,18,20,22$)"

규칙 행렬 R_2는 표 6과 같다.

표 6. Robot-2에 대한 규칙 행렬 R_2.

	Pr-2	p_{10}	p_{31}	p_{34}	p_{37}	p_{30}	p_{32}	p_{36}	t_4	t_{18}	t_{20}	t_{22}
Pr-2	1											
p_{10}		1										
p_{31}			1									
p_{34}				1								
p_{37}					1							
p_{30}						1						
p_{32}							1					
p_{36}								1				
t_4	0.4	1				0.8			0			
t_{18}	0.6		1				0.9			0		
t_{20}	0.8			1				1			0	
t_{22}	1				1							0

Robot-3과 4에 대한 규칙 행렬도 같은 방법으로 만들수 있다. 이를 사용하여 자동 조립 시스템의 Robot의 충돌 현상에 관한 제어를 할 수 있다.

AGV-1을 나타내는 플레이스 P_{a-1}의 발화 우선 순위 결정은 충돌 현상2에서 사용한 추론을 이용한다. Workcell-1에서 부품의 수량을 체크하는 센서를 나타내는 플레이스는 P_s이다. 플레이스 P_s에서 부품의 수량을 체크하여 WS-1으로 정보를 보내면 WS-1은 퍼지추론을 하여 부족한 부품이 담긴 Pallette를 AGV-1으로 하여금 Input Storage로부터 가져오게 한다.

"If (P_s is true) and (P_{a-1} is true) and (p_n is true $n=1,2,3,4$) Then (t_m is fired $m=1,5,9,13$)"

하나의 Pallette에 담긴 부품의 수량은 다음과 같다고 하자.

Motor Frame : 12　　End Bracket(Rear) : 20
Stator Assembly : 8　　Rotor Assembly : 10

현재 Workcell-1의 지정된 위치(p_{10}, p_{13}, p_{16}, p_{19})에 Motor Frame이 6, End Bracket (Rear)의 수량이 25, Stator Assembly의 수량이 5, Rotor Assembly의 수량이 5개가 있다고 하면 부품의 수량에 대한 멤버쉽 값이 그림 7과 같이 나타내어질 때, 각각의 부품 수량에 대한 소속 함수 값은 그림 7에 의해 아래와 같이 나타내어진다.

$$\text{INPUT} = \begin{array}{c} \\ \text{NB} \\ \text{NS} \\ \text{M} \\ \text{PS} \\ \text{PB} \end{array} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.8 & 0 & 0.75 & 1 \\ 0.2 & 0.5 & 0.25 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \begin{array}{c} \\ \text{NB} \\ \text{NS} \\ \text{M} \\ \text{PS} \\ \text{PB} \end{array} = \begin{bmatrix} 1 \\ 0.9 \\ 0.8 \\ 0.7 \\ 0.6 \end{bmatrix}$$

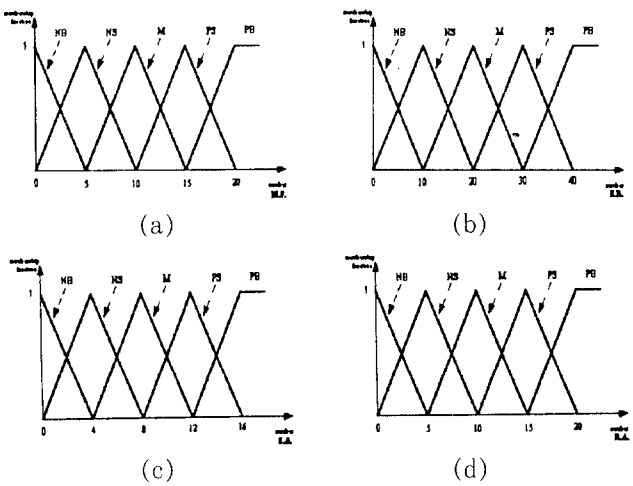


그림 7. Workcell-1에 있는 부품의 수량에 대한 멤버쉽값.
(a) Motor Frame　(b) End Bracket
(c) Stator Assembly　(d) Rotor Assembly.

위의 행렬에서 각 행은 Workcell-1에서의 부품의 수량에 대한 멤버쉽값을 나타낸다. 1 행은 Motor Frame의 수량이 6개 이므로 그림 7로 부터 $[0\ 0.8\ 0.2\ 0\ 0]^T$로 나타내어진다. 또한 같은 방법으로 2 행은 End Bracket, 3 행은 Stator Assembly, 4 행은 Rotor Assembly에 대한 멤버쉽 값을 나타낸 것이다.

행렬의 각 열에서, 원소의 최대를 취하면 NB : Negative Big 열에서는 각 원소의 값들이 0으로 같고, NS : Negative Small 열에서는 Rotor Assembly를 나타내는 원소가 Max이다. PB 보다는 PS가, PS 보다는 M이, M는 NS가, NS보다는 NB가 Weight가 크므로 NS 열의 최대인 부품 Rotor Assembly를 나타내는 원소 값 1에 의해 AGV-1은 Rotor Assembly를 Workcell-1으로 옮기게 된다. 즉, 결과에 의해 트랜지션 t_{13}이 발화가 된다. AGV-2의 우선 순위 결정도 AGV-1과 같은 방식으로 결정할 수 있다.

이상과 같은 퍼지 개념을 이용하여, 제원(Resource)을 나타내는 플레이스에서 발생하는 충돌 현상을 해결하였다. 각 제원에 대한 단순한 우선 순위 결정을 벗어나, 그 상황에 적합한 우선 순위를 결정하였다.

5. 결　　론

본 연구에서는 페트리 네트를 사용하여 모델링된 이산 사건 시스템의 제어시 발생하는 충돌 현상을 해결하기 위하여 퍼지 개념을 사용하였다. 이를 통하여 페트리 네트로 시스템을 제어 할 경우 발생하는, 큰 문제점인 외부 시스템과의 데이터의 입출력 설정의 어려움을 해결하는 방법을 보였다. 또한, 제시된 규칙 행렬의 단순성으로부터 쉽게 충돌 현상하의 우선 순위를 변화시킬 수 있다. 시스템을 제어하는 전문가의 지식이 모호하여 단순히 상수값으로 우선 순위를 표현할 수 없는 경우에는 퍼지 개념을 이용하여 해결하였다. 이러한 방법들은 소규모의 모터 자동 조립 시스템을 제어하는데 부품의 수량, 작업의 대기 상태를 퍼지화하여 규칙 행렬을 만들어 제어 신호를 발생 시켰다. FMS, CIM을 제어할 때 발생하는 Scheduling 문제도 본 논문의 방법을 사용하면 해결할 수 있다고 본다.

참 고 문 헌

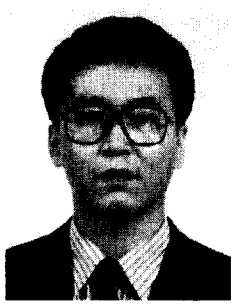
[1] J. L. Peterson, "Petri Net theory and the modeling of systems", Prentice-Hall, 1981.

[2] H.-J. Zimmermann, "Fuzzy Set Theory and Its Applications", Kluwer Academic Publishers, 1991.

[3] M. M. Gupta, J. B. Kiszka and G.M. Trojan, "Multivariable structure of fuzzy control system", IEEE Trans. Syst. Man, Cyber. vol. 16, no. 5, pp.

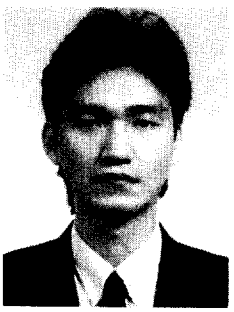
638-655, 1986.
[4] Daniel Tabak and Alexander H. Levis, "Petri Net Representation of Decision Models", IEEE Trans. Syst. Man, Cyber. vol. SMC-15, no. 6, pp. 812-818, 1985.
[5] J. Maiers and Y. S. Sherif, "Applications fo Fuzzy Set Theory", IEEE Trans. Syst. Man, Cyber. vol. SMC-15, no. 1, pp. 175-186, 1985.
[6] Luca Ferranrini, "An Incremental Approach to Logic Controller Design with Petri Nets", IEEE Trans. Syst. Man, Cyber. vol. 22, no. 3, pp. 461-473, 1992.
[7] Carl G. Looney and Abdulrdah A. Alfize, "Logical Controls via Boolean Rule Matrix Transformations", IEEE Trans. Syst. Man, Cyber. vol. SMC-17, no.6, pp. 1077-1082, 1987.
[8] Klaus Peter Brand and Jurgen Kopainsky, "Principles and Engineering of Process Control with Petri Nets", IEEE Trans. Automatic Control, vol. 33, no. 2, pp. 138-149, 1988.
[9] Madan M. Gupta, Georgen M. Trojan and Ierzy B. Kiszka, "Controllablity of Fuzzy Control System", IEEE Trans. Syst. Man, Cyber. vol. SMC-16, no. 4, pp. 576-582, 1986.
[10] Ghanssan M. Abdelnour, Chir-Ho Chang, Feng-Hsin Huang and Y. Cheung, "Design of a Fuzzy Controller Using Input Mapping Factors", IEEE Trans. Syst. Man, Cyber. vol. 21, no. 5, pp. 952-960, 1991.
[11] C. P. Pappis and Ebrahim H. Mamdani, "A Fuzzy Controller for a Traffic Junction", IEEE Trans. Syst. Man, Cyber. vol. SMC-7, no. 10, pp. 707-717, 1977.
[12] C. C. Lee, "Fuzzy Logic in control system : Fuzzy logic controller.-Part I", IEEE Trans. Syst. Man Cybern., vol. 20, pp. 404-418, 1990.

저 자 소 개



고 인 선

1955년 7월 31일생. 1979년 서울대학교 전자공학과 졸업(B.S.). 1987년 Marquette University, Dept. of ECE 졸업(M.S.). 1991년 Rensselaer Polytechnic Institute, Dept. of ECSE 졸업(Ph. D.) 1981-1985 : 대우전자 근무 1991-1992 : 대우전자 근무 1992-현재 : 홍익 대학교, 전자공학과 조교수 주 관심 분야 : 이산 사건 시스템 제어, 공장 자동화, 지능 제어, CIM, Petri Net 응용.
(121-791) 서울 마포구 상수동 72-1
TEL)(02)320-1697 / FAX)(02)320-1119



전 광 호

1967년 9월 9일생. 1993년 2월 홍익대학교 전자공학과(공학사). 1995년 2월 홍익대학교 전자공학과 석사과정(공학석사). 1995년 3월 ~ 현재 홍익대학교 전자공학과 박사과정. 주관심 분야는 이산 사건 시스템 및 퍼지 제어.
(121-791) 서울 마포구 상수동 72-1