CIM Testbed의 제어를 위한 Supervisor의 설계와 구현

Design and Implementation of Supervisors to Control of a CIM Testbed

손 형 일, 이 석
(Hyoung-Il Son and Suk Lee)

Abstract : A discrete event systems (DES) is a physical system that is discrete in time and state space, asynchronous (event rather than clock-driven), and in some sense generative(or nondeterministic). This paper presents the design of fifteen modular supervisors to control an experimental CIM testbed. These supervisors are nonblocking, controllable and nonconflicting. After verification of the supervisors by simulation, the supervisors for AGV system have been implemented to demonstrate their efficacy.

Keywords : discrete event systems, supervisory control theory, modular supervisor, manufacturing cells

I. 서론

Discrete Event Systems (DES)에 대한 모델링과 제어기의 설계는 미분방정식과 같은 전통적 제어이론으로는 많은 어려움이 있다. 그래서 이런 DES를 제어하기 위한 새로운 방법론이 필요하게 되었고, 근래에 새로운 모델링 기법과 제어 이론들이 많이 소개되고 있다. Petri net, formal language, controlled automata, min-max algebra, temporal logic 등이 그 대표적인 예라고 할 수가 있다. 그렇지만 이런 이론들은 각각 별개의 것이 아니라, logic, language, automata 이론을 바탕으로 서로 관련되어 있다[1]. 이런 여러 가지 방법론들 중에서 automata와 lattice 이론을 기반으로 하는 Ramadge와 Wonham의 supervisory control theory가 플랜트의 구속조건을 만족시키고 실제 일어날 수 있는 이벤트들을 최대한 허용시키는 슈퍼바이저를 설계할 수 있어서 많은 관심을 받고 있다[2].

본 논문에서는 CIM system을 모사하는 testbed의 제어를 위해 supervisory control theory를 이용하여 supervisor를 설계하고 이를 구현하였다. CIM testbed는 3대의 로봇, 2대의 AGV, NC 공작기계, 컨베이어 벨트 등으로 구성되어있다[3]. 이렇게 제작된 각각의 구성요소들을 플랜트로 간주하고 deterministic auto- maton으로 모델링하였고, CIM system의 운용규칙을 구속조건(behavior specification 또는 legal language)으로 하여 슈퍼바이저를 설계하였다. 그리고 슈퍼바이저의 controllability, nonblockingness를 검사하였다. 또 슈퍼바이저를 modular로 설계하여 modular 슈퍼바이저들이 nonconflictness를 만족하는지 검사하였다[4][5].

마지막으로 각각의 modular 슈퍼바이저들을 구현하기 위해서 Clocked moore synchronous state machine (CMSSM)으로 변환하였다[6][7]. 그리고 CMSSM을 회로설계 및 분석 프로그램으로 시뮬레이션하여 legal language에 맞게 플랜트가 작동하는지 검사하였다. 특히 AGV 충돌방지 슈퍼바이저는 실제 제작하여 AGV 시스템을 원활하게 제어하는 것을 확인하였다.

접수일자 : 1999. 10. 4., 수정완료 : 2000. 2. 24.
손형일 : 부산대학교 지능기계공학과
이 석 : 부산대학교 기계공학부

II. Supervisory Control Theory

1. Generator

DES를 모델링하기 위한 automaton은 다음과 같은 5개의 구성요소를 가지고 있다.

$$G=\{Q,\ \Sigma,\ \delta,\ q_0,\ Q_m\} \tag{1}$$

여기서 Q는 상태들의 집합, Σ는 이벤트의 집합, δ는 $\delta : Q\times\Sigma^*\rightarrow Q$ 인 천이함수, q_0는 초기 상태, Q_m은 작업의 완료를 나타내는 marked 상태들의 집합이다. 천이함수 δ에서 Σ^*는 이벤트의 sequence(string)를 나타낸다. 그리고 Σ는 controllable 이벤트 집합 Σ_c와 uncon- trollable 이벤트 집합 Σ_u로 나눌 수 있다.

이렇게 구성된 automaton이 생성해내는 언어는 $L(G)$로 나타내며 다음과 같이 정의된다.

$$s\in L(G) \Leftrightarrow s\in\Sigma^*,\ \delta(q_0,\ s)! \tag{2}$$

여기서 $\delta(q_0,\ s)!$는 q_0에서 스트링 s가 일어난 다음의 상태가 정의됨을 나타낸다. $L(G)$의 prefix closure를 $\overline{L(G)}$로 나타내고 정의는 다음과 같다.

$$\overline{L(G)}=\{t\in\Sigma^* \mid t\leq s \text{ for } some\ s\in L(G)\} \tag{3}$$

그리고 automaton G의 marked 언어를 $L_m(G)$로 나타내고

$$s\in L_m(G) \Leftrightarrow \delta(q_0,\ s)!\in Q_m\ ,\ L_m(G)\subseteq L(G) \tag{4}$$

와 같이 정의된다. 이때 G가 $\overline{L_m}(G)=L(G)$를 만족하 면 $L(G)$를 nonblocking하다고 말한다. 즉 G의 모든 상태에서 임의의 스트링이 일어난 후 marked 상태로 도달할 수 있음을 의미한다. 그리고 이는 supervisory control theory에서 proper 슈퍼바이저가 되기 위한 중요한 필요조건이

된다[5].

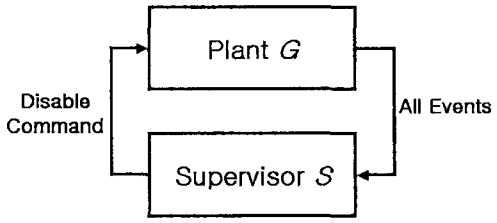


그림 1. 감독 제어 시스템.
Fig. 1. Supervisory control system.

2. Supervisor

슈퍼바이져 역시 다음과 같은 5개의 구성요소를 가진 automaton으로 표현된다.

$$S=\{X,\ \Sigma,\ \xi,\ x_0,\ X_m\} \tag{5}$$

G를 주어진 플랜트라고 하면 슈퍼바이져 S 아래에서의 G의 거동은 다음과 같이 나타난다.

$$S/G=\{X\times Q,\ \Sigma,\ \xi\times\delta,\ (x_0, q_0),\ X_m\times Q_m\} \tag{6}$$

그리고 G에 대한 $L(S)$의 controllability는 다음 조건으로 검사될 수 있다.

$$(\forall s,\ \sigma)s\in\overline{L}(S),\ \sigma\in\Sigma_u,\ s\sigma\in L(G)\Rightarrow s\sigma\in\overline{L}(S) \tag{7}$$

즉 슈퍼바이져 S가 플랜트 G에 대해서 controllable하다는 것은 다음과 같이 설명할 수 있다. 슈퍼바이져 S에서 허용되는 임의의 스트링 s가 있고 플랜트 G에서 발생할 수 있는 어떤 uncontrollable 이벤트 σ가 있다고 했을 때 스트링 $s\sigma$가 플랜트 G 에서 발생했을 때 슈퍼바이져 S도 스트링 $s\sigma$가 일어날 수 있도록 허용한다면 슈퍼바이져 S는 플랜트 G에 대해서 controll- able하다고 한다. 그리고 이렇게 controllable한 여러 슈퍼바이져 중에서 supremal language가 optimal 슈퍼바이져가 된다[4][5]. 그림 1에 supervisory control system의 구조를 간략하게 나타내었다[2].

3. Modular Supervisor

$$\overline{S}=\overline{S_1}\wedge\overline{S_2}\wedge\ldots \tag{8}$$

슈퍼바이져 S를 $S_i, i=1,2\ldots$로 설계할 때 이런 S_i를 modular 슈퍼바이져라 한다. 그리고 modular 슈퍼바이져는 (8)로 표현되는 nonconflictness 조건을 만족하여야한다[7]. Modular 슈퍼바이져가 nonconflictness를 만족하면, 모든 modular 슈퍼바이져들이 동시에 플랜트를 supervisory control하여 centralized 슈퍼바이져와 같은 작용을 할 수 있다. 만약 각 슈퍼바이져들이 conflicting하다면, modular 슈퍼바이져들은 centralized 슈퍼바이져가 허용하지 않는 스트링을 허용하게 되어 legal language를 만족시키지 못하게된다. 즉, 슈퍼바이져 감독 아래에서 발생되는 언어 중에서 marked 언어가 아닌 것이 발생할 수 있게된다. 그렇기 때문에 mo- dular 슈퍼바이져가 conflicting하다면 *nonblocking* 조건을 만족시키지 못하게된다.

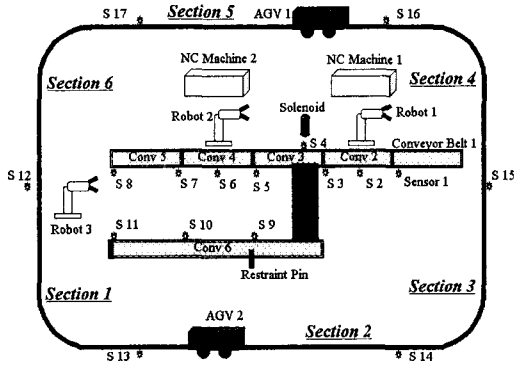


그림 2. CIM 실험시스템.
Fig. 2. CIM testbed.

III. CIM Testbed

1. 시스템 구성

본 논문에서는 CIM testbed로서 NC 공작기계 2대, 로봇 3대, 컨베이어 벨트, AGV 등으로 구성된 모형 CIM 시스템을 제작하였다. 모형 CIM 시스템은 두 종류의 제품을 생산한다고 가정하고 누적형과 비누적형의 두 개의 생산라인을 갖도록 설계하였다. 전체 시스템 구성은 그림 2와 같다.

2. Plant 모델링

전체 플랜트를 모델링한 automaton은 NC 기계, 로봇 등과 같은 각각의 시스템 구성요소들을 automaton으로 모델링한 후 각각의 automaton을 synchronous product[5]하면 된다. 본 논문에서는 플랜트를 모델링할 때 상태와 이벤트를 최소화하였다. 즉 슈퍼바이져가 관측할 필요가 없거나 관측할 수 없는 이벤트와 legal language에 특별한 영향을 주지 않는 이벤트들을 ε 이벤트로 projection[5]시켰다. 예를 들어 AGV를 모델링할 때 AGV의 속도가 변하는 것은 AGV automa- ton에 나타내지 않았다.

본 논문에서는 event-based로 supervisory control theory를 적용하였기 때문에 automaton의 상태에 상관없이 automaton에서 생성되는 language가 변하지 않도록 projection시켰다. 따라서 설계된 automaton이 projection 후 nondeterministic하게 변하면 subset con- struction[5]을 이용하여 deterministic하게 변환시켰다.

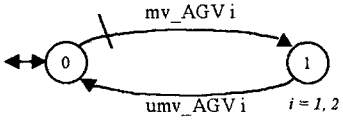
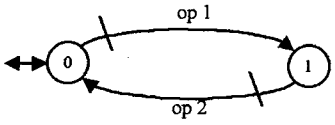
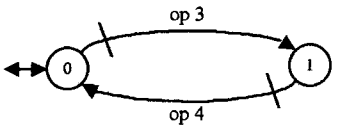


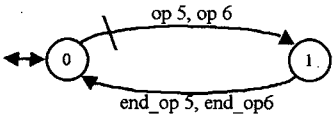
그림 3. AGV 모델링.
Fig. 3. AGV modeling.



(a)



(b)



(c)

그림 4. 로봇 모델링. (a) 로봇1, (b) 로봇2, (c) 로봇3.
Fig. 4. Robot modeling. (a) Robot1, (b) Robot2, (c) Robot3.

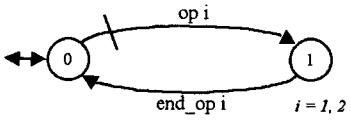


그림 5. NC 공작기계 모델링.
Fig. 5. NC machine modeling.

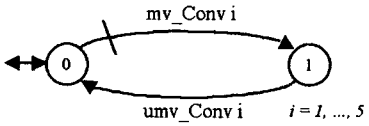


그림 6. 컨베이어 벨트 모델링.
Fig. 6. Conveyor belt modeling.

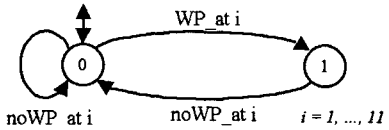
AGV 2대, 로봇 3대, NC 기계 2대, 컨베이어 벨트 5개, 센서 17개, 구속핀, 솔레노이드를 스테이트가 2개인 automaton으로 모델링하였다[3].

설계된 플랜트와 이벤트 목록이 그림 3에서 그림 9, 표 1에서 표 7에 각각 나타나있다. 그리고 전체 플랜트 automaton은 다음과 같이 TCT[5]의 SYNC 함수를 사용하여 구할 수 있다.

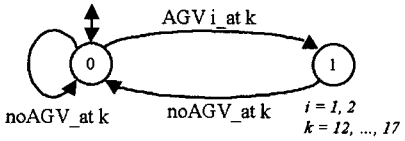
CIMPlant = **SYNC** (AGVs, Robots, NC Machines, Conveyor Belts, Sensors, Restraint Pin, Solenoid) (9)

3. Supervisor 설계

슈퍼바이저는 legal language를 세워, 플랜트에 대한



(a)



(b)

그림 7. 센서 모델링. (a) 컨베이어 벨트용 센서, (b) AGV용 센서.
Fig. 7. Sensor modeling. (a) Sensor for Conveyor belt, (b) Sensor for AGV.

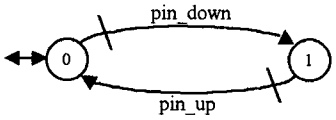


그림 8. 구속 핀 모델링.
Fig. 8. Restraint pin modeling.

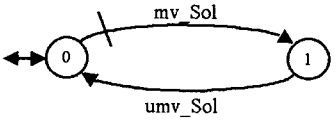


그림 9. 솔레노이드 모델링.
Fig. 9. Solenoid plant modeling.

표 1. AGV 이벤트 목록.
Table 1. AGV event list.

Events	
mv_AGV i	AGV *i*가 움직인다.
umv_AGV i	AGV *i*가 정지한다.

supremal controllable sublanguage를 구한 결과이다. 이렇게 얻어진 슈퍼바이저는 다음을 만족시킨다.

$$L(S/G) = L(S) \tag{10}$$

그렇지만 이렇게 설계된 슈퍼바이저는 플랜트에서 일어나는 이벤트들에 대해서 legal language보다 훨씬 많은 스테이트를 갖고있어 매우 복잡하게 된다. 그래서 실제 구현시 어려움이 생기기 때문에 다음을 만족하는 또 다른 슈퍼바이저를 구하는 게 슈퍼바이저의 구현시 간편하다.

표 2. 로봇 이벤트 목록.
Table 2. Robot event list.

Events	
op1	Conv 2에서 작업물을 집어 NC 공작기계1에 내려놓는다.
op2	NC 공작기계1에서 작업물을 집어 Conv 2위에 내려놓는다.
op3	Conv 4에서 작업물을 집어 NC 공작기계2에 내려놓는다
op4	NC 공작기계2에서 작업물을 집어 Conv 4위에 내려놓는다.
op5	Conv 5에서 제품을 집어서 AGV에 내려놓고 Conv 6에 위치한다.
op6	Conv 6에서 제품을 집어서 AGV에 내려놓고 Conv 5에 위치한다.
end_op5	op5의 완료.
end_op6	op6의 완료.

표 3. NC 공작기계 이벤트 목록.
Table 3. NC machine event list.

Events	
op i	로봇에서의 event와 같다.
end_op i	NC 기계1, 2가 작업을 마친다.

표 4. 컨베이어 벨트 이벤트 리스트.
Table 4. Conveyor belt event list.

Events	
mv_Conv i	컨베이어 벨트 *i*가 움직인다.
unv_Conv i	컨베이어 벨트 *i*가 정지한다.

표 5. 센서 이벤트 목록.
Table 5. Sensor event list.

Events	
WP_at i	센서 *i*에 작업물이 있다.
noWP_at i	센서 *i*에 작업물이 없다.
AGV i_at k	센서 *k*에 AGV *i*가 있다.
noAGV_at k	센서 *k*에 AGV가 없다.

표 6. 구속 핀 이벤트 목록.
Table 6. Restraint pin event list.

Events	
pin_down	구속핀을 내린다.
pin_up	구속핀을 올린다.

표 7. 솔레노이드 이벤트 목록.
Table 7. Solenoid event list.

Events	
mv_Sol	솔레노이드가 움직인다.
unv_Sol	솔레노이드가 정지한다.

$$L(S'/G) = L(S) \tag{11}$$

즉, 슈퍼바이져 감독아래에서의 플랜트 거동 language는 S 감독아래에서의 플랜트 거동 language와 같지만 (11)을 만족시키면서 automaton은 S보다 간단한 S'을 설계할 수 있다. 이때 legal language를 K라고 하면, S는 최대 $K \wedge G$와 같은 스테이트 수를 가지지만, S'는 최대 K와 같은 스테이트 수를 가진다.

그리고 본 논문에서는 legal language를 $K_1, K_2, \ldots$로 설계하고 각각의 legal language를 만족시키는 슈퍼바이져를 $S_1, S_2, \ldots$으로 설계하였다. 이런 슈퍼바이져를 modular 슈퍼바이져라고 한다. 그리고 이렇게 설계된 슈퍼바이져가 optimal proper 슈퍼바이져인지는 다음의 조건을 만족하는지 검사함으로써 알 수가 있다.

정의 1 : Optimal proper 슈퍼바이져

다음의 조건을 모두 만족시키는 슈퍼바이져 S'_i를 플랜트 G에 대한 optimal proper 슈퍼바이져라 한다.

1) 슈퍼바이져 S'_i가 플랜트 G에 대해서 controllable 해야 한다.

2) $\overline{L_m(S'_i)} = L(S'_i)$ (12)

3) $\overline{L_m(G) \wedge L_m(S'_i)} = L(G) \wedge L(S'_i)$ (13)

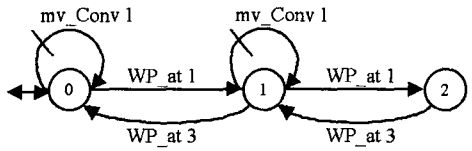
4) S_i를 K_i에 대한 supremal controllable sublanguage라 할때, $L(S'_i/G) = L(S_i)$를 만족해야한다. ■

정의 1의 1)은 설계된 슈퍼바이져가 플랜트에 대해 controllability를 만족해야됨을 의미하고, 2)는 슈퍼바이져가 nonblocking해야됨을 나타낸다. 그리고 3)은 슈퍼바이져가 플랜트와 nonconflict해야됨을 나타낸다. 즉 슈퍼바이져가 플랜트에 대해서 nonblocking해야됨을 말하고있다. 마지막으로 4)는 설계된 슈퍼바이져가 플랜트를 supervisory control 했을 때 supremal controllable sublanguage를 발생해야됨을 의미한다.

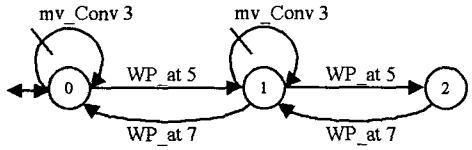
그리고 마지막으로 modular 슈퍼바이져가 플랜트에 대해서 centralized 슈퍼바이져와 같은 supervisory control을 하는지 검사하여야한다. 이는 S_i들이 서로 nonconflict한지 검사함으로써 알 수가 있다. 즉 다음을 만족시키면 된다.

$$\overline{L_m(S_1) \wedge L_m(S_2) \wedge \ldots} = L(S_1) \wedge L(S_2) \wedge \ldots \tag{14}$$

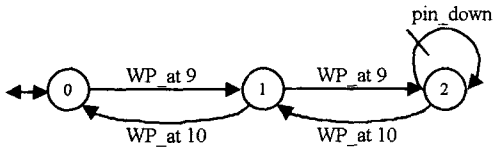
본 논문에서 설계한 플랜트 supervisory control testbed에는 아래와 같은 구속 조건을 주어 15개의 modular 슈퍼



(a)



(b)



(c)

그림 10. 버퍼크기 슈퍼바이져. (a) 컨베이어 벨트 2에 대한 버퍼크기 슈퍼바이져, (b) 컨베이어 벨트 4에 대한 버퍼크기 슈퍼바이져, (c) 컨베이어 벨트 6에 대한 버퍼크기 슈퍼바이져.

Fig. 10. Buffer size supervisor. (a) Buffer size supervisor for conveyor belt 2, (b) Buffer size supervisor for conveyor belt 4, (c) Buffer size supervisor for conveyor belt 6.

바이저를 만들어냈다.

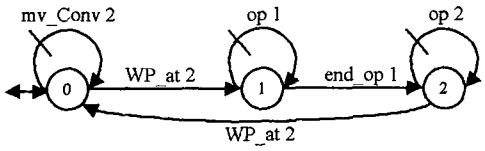
구속조건 : Behavior specification

1) 컨베이어 벨트 2, 4, 6의 버퍼 크기를 2로 한다.

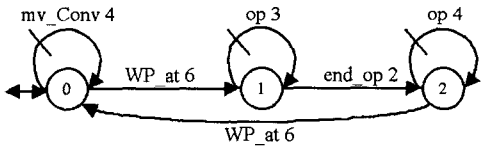
2) 로봇 1이 작업물을 집어 NC 공작기계 1에 옮긴다음 NC 공작기계 1이 가공을 완료하면 로봇 1이 작업물을 집어 컨베이어 벨트 위에 올려놓는다.

3) 로봇 2가 작업물을 집어 NC 공작기계 2에 옮긴다음 NC 공작기계 2가 가공을 완료하면 로봇 2가 작업물을 집어 컨베이어 벨트 위에 올려놓는다.

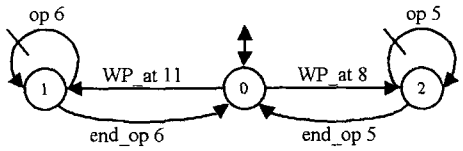
4) 두 생산라인에서 가공완료된 제품을 로봇 3이 두대의 AGV에 구분하여 싣는다.



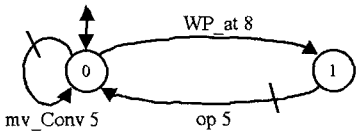
(a)



(b)



(c)



(d)

그림 11. 라우팅 슈퍼바이져. (a) 로봇 1, NC 공작기계 1에 대한 라우팅 슈퍼바이져, (b) 로봇 2, NC 공작기계 2에 대한 라우팅 슈퍼바이져, (c) 로봇 3, 생산라인 1, 2에 대한 라우팅 슈퍼바이져, (d) 로봇 3, 컨베이어 벨트 5에 대한 라우팅 슈퍼바이져.

Fig. 11. Routing supervisor. (a) Routing supervisor for robot 1 and NC machine 1, (b) Routing supervisor for robot 2 and NC machine 2, (c) Routing supervisor for robot 3 and process line 1, 2 (d) Routing supervisor for robot 3 and conveyor belt 5.

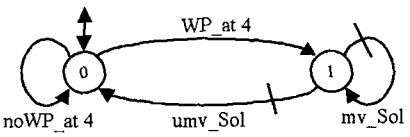


그림 12. 작업물 분류 슈퍼바이져.

Fig. 12. Workpiece selection supervisor.

5) 솔레노이드가 작업물을 분류시킨다.

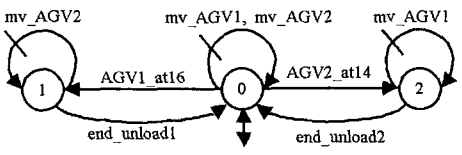


그림 13. 하역작업 슈퍼바이져.

Fig. 13. Unloading supervisor.

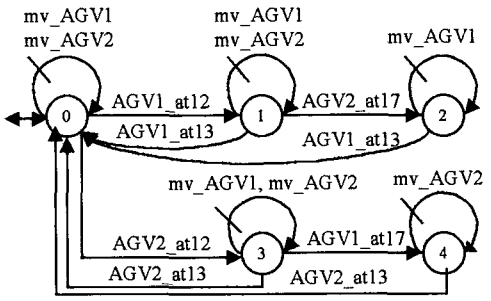


그림 14. 구간 1에 대한 충돌방지 슈퍼바이져.

Fig. 14. Collision protection supervisor for section 1.

6) 두 대의 AGV는 각각의 위치에 제품을 하역한다.

7) AGV는 서로 충돌하지 말아야한다.

3.1 생산라인 슈퍼바이져

위의 구속조건 1)~5)에 맞는 legal language를 만들어 정리 1을 만족하는 8개의 modular 슈퍼바이져를 만들었고 이들을 그림 10, 11, 12에 나타내었다[3].

그림 10(a)의 컨베이어 벨트 2에 대한 버퍼크기 슈퍼바이져를 예로 들어서 슈퍼바이져가 어떻게 플랜트를 supervisory control하는지 알아보자. 컨베이어 벨트 2에 대한 버퍼크기 슈퍼바이져의 제어 데이터(control data)는 초기 상태와 상태 1에서는 모든 이벤트를 enable하다가 상태 2에서 이벤트 mv_Conv1을 disable시키는 것이다. 즉 스트링 ε^*mv_Conv1* ε^*WP_at1 ε^*

mv_Conv1* ε^*WP_at1 ε^*이 일어나면 버퍼크기 슈퍼바이져 1은 이벤트 mv_Conv1이 일어나지 못하게 한다.

3.2 AGV 슈퍼바이져

구속조건 6), 7)과 정리 1을 만족하는 7개의 modular 슈퍼바이져를 설계하였다[3]. 구속조건 7)의 legal language는 AGV 라인을 그림 2와 같이 6개의 구간으로 나누어서 각 구간에 대한 슈퍼바이져 6개를 설계하였다. 그림 14에 구간 1에 대한 충돌방지 슈퍼바이져를 나타내었다. 다른 구간에 대한 충돌방지 슈퍼바이져는 상태 천이 이벤트만 각 구간의 센서 신호에 맞게 바꾸면 된다.

구간 1에 대한 충돌방지 슈퍼바이져 1은 상태 2에서 이

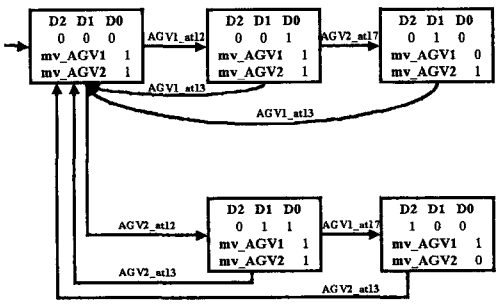


그림 15. 구간 1에 대한 충돌방지 슈퍼바이져의 CMSSM.

Fig. 15. CMSSM of collision protection supervisor for section 1.

벤트 mv_AGV2를 disable시키고, 상태 4에서 mv_AGV1을 disable시키는 제어 데이터를 가지고 있다. 즉, 스트링 ε^*(mv_AGV1+mv_AGV2)* ε^*AGV1_at 12 ε^*(mv_AGV1 +mv_AGV2)* ε^*AGV2_at17 ε^*mv_AGV1* ε^*이 발생하면 이벤트 mv_AGV2를 disable시키고, 스트링 ε^*(mv_AGV1+mv_AGV2)* ε^*AGV2_at12 ε^*(mv_AGV1+ mv_AGV2)* ε^*AGV1_at17 ε^*mv_AGV1* ε^*이 발생하면 이벤트 mv_AGV1을 disable시킨다.

Ⅳ. Implmentation

1. CMSSM 변환

본 논문에서는 설계된 modular 슈퍼바이져를 실제구현하기 위해서 CMSSM으로 변환하였다. CMSSM이란 현재의 상태, 입력 그리고 clock에 대해서 특정한 출력값을 가지는 machine을 말한다[7]. CMSSM으로 변환된 슈퍼바이져는 PLC나 디지털 회로로 구현될 수 있다[6][8]. 그림 15에 구간 1에 대한 충돌방지 슈퍼바이져의 CMSSM을 나타내었다.

그림 15에서 D2~D0는 CMSSM의 상태를 나타내고 mv_AGV1, mv_AGV2는 각 상태에서의 출력을 나타낸다. 그리고 상태 출력은 현재 입력에대한 edge trigger 방식으로 처리된다.

여기서 슈퍼바이져를 CMSSM으로 변환시킬 때 고려되어야할 점은 다음과 같다. 센서가 신호를 발생시키는 구간이 CMSSM clock보다 긴 경우에 하나의 사건을 여러개의 사건이 일어난 것으로 오인하는 것을 방지하여야 한다는 것이다.

그림 10(a)의 컨베이어 벨트 2에 대한 버퍼크기 슈퍼바이져의 초기 상태에서 이벤트 WP_at1이 발생하면 두번째 상태로 이동하게되고 또다시 WP_at1이 발생하면 세 번째 상태로 바뀌게된다. 그런데 CMSSM에서는 그림 16과 같이 WP_at1의 발생시간이 CMSSM의 clock 주기보다 길게 되면 CMSSM에서는 WP_at1이 여러번 발생한 것으로 인식하고 WP_at1이 한번 발생하더라도 초기상태에서 세 번째 상태로 이동하게 된다. 그러므로 슈퍼바이져를 CMSSM으로 변환시킬 때 같은 이벤트가 상태를 연속으로 바꿀때는 그런 이벤트들을 구분해야된다. 즉, 여기서는 초기 상태에서 일어나는 WP_at1와 첫 번째 상태에서 일어나는 WP_at1을 구분해서 CMSSM을 만들어야된다. 버퍼크기 슈퍼바이져 CMSSM에서는 후자의 WP_at1을 WP_at1'으로 나타내었고 이를 그림 17에 보이고 있다. 그리고 이는 실제 구현시 이벤트 WP_at1'을 위한 또다른 센서가 추가되어야함을 의미한다.

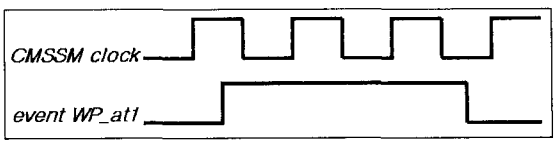


그림 16. 이벤트 발생 신호.

Fig. 16. Event occurrence signal.

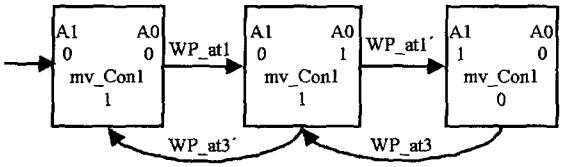


그림 17. 컨베이어 벨트 2에 대한 버퍼크기 슈퍼바이저의 CMSSM.

Fig. 17. CMSSM of buffer size supervisor for conveyor belt 2.

이렇게 CMSSM이 만들어지면 CMSSM의 입력, 출력에 대해서 logic을 만들수가 있다[7]. 컨베이어 벨트 2에 대한 버퍼크기 슈퍼바이저 CMSSM에 대해서는 센서 신호 WP_at1, WP_at1', WP_at3, WP_at3'가 입력이 되고, 컨베이어 벨트1의 제어 신호 mv_Conv1이 출력이 된다.

마지막으로 그림 15에 대한 CMSSM의 논리식을 (15)~(19)에 나타내었다.

$$D2new = (D1 \wedge D0 \wedge AGV1\_at17 \wedge \sim AGV2\_at13) \vee (D2 \wedge \sim AGV2\_at13) \quad (15)$$

$$D1new = (\sim D1 \wedge D0 \wedge AGV2\_at17 \wedge \sim AGV1\_at13) \vee (D1 \wedge \sim D0 \wedge \sim AGV1\_at13) \vee (\sim D2 \wedge \sim D1 \wedge \sim D0 \wedge AGV2\_at12) \vee (D1 \wedge D0 \wedge \sim AGV1\_at17 \wedge \sim AGV2\_at13) \quad (16)$$

$$D0new = \{(\sim D2 \wedge \sim D1 \wedge \sim D0) \wedge (AGV1\_at14 \vee AGV2\_at12)\} \vee (\sim D1 \wedge D0 \wedge \sim AGV2\_at17 \wedge \sim AGV1\_at13) \vee (D1 \wedge D0 \wedge \sim AGV1\_at17 \wedge \sim AGV2\_at13) \quad (17)$$

$$mv\_AGV1 = \sim D2 \quad (18)$$

$$mv\_AGV2 = \sim D1 \vee D0 \quad (19)$$

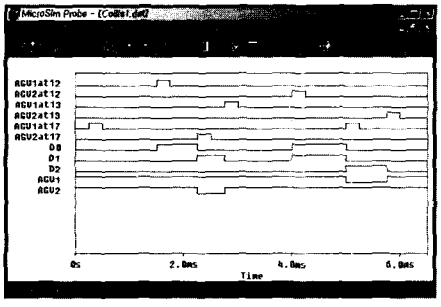


그림 18. 구간 1에 대한 충돌방지 슈퍼바이저의 시뮬레이션 결과.

Fig. 18. Simulation result for collision protection supervisor for section 1.

2. Simulation

앞에서 설계된 CMSSM이 원하는 제어신호를 생성하는지를 검증하기 위해 회로설계 및 분석 프로그램인 PSpice로 CMSSM을 시뮬레이션 하였다. 시뮬레이션은 각각의 CMSSM의 입력신호를 임의로 주고, 출력신호를 검사하였다. 설계된 모든 슈퍼바이저들에 대해서 시뮬레이션을 하였으며, AGV 라인 구간 1에 대한 충돌방지 슈퍼바이저의 CMSSM에 대한 PSpice 시뮬레이션 결과를 그림 18에 나타내었다. 그림 18에서 AGV 1at12, AGV2at12, AGV1at13, AGV2at13, AGV1at17, AGV2at17이 입력으로 쓰인 센서신호이고, AGV1, AG V2가 출력을 나타내는 AGV 제어 신호이다. 그리고 D0~D2가 CMSSM의 상태를 나타낸다. 시뮬레이션 결과를 분석해보면 먼저 D0~D2가 모두 0으로 초기상태로 되어있다. 이때 AGV1at17 이벤트가 발생하게 된다. 그렇지만 CMSSM의 상태는 바뀌는 않는다. 왜냐하면 AGV1at17은 초기상태에서 selfloop 이벤트이기 때문이다. 그리고 AGV1at12가 발생하면 CMSSM은 D0가 1로 바뀌면서 상태 1로 변화하고 이때 AGV2at17이 발생하게되면 D0는 0으로 바뀌고 D1은 1로된다. 따라서 상태는 2가되고 AGV2를 disable시키게된다. 그리고 A GV1at13이 발생하면 상태 0으로 돌아가서 다시 AGV2를 enable시킨다. 즉, 먼저 어떤 A GV가 구간 1로 들어오고 그 AGV가 그 구간을 빠져나가기 전에 다른 AGV가 그 구간으로 들어온다면 나중의 AGV를 첫 번째 AGV가 그 구간을 빠져나가기 전까지 정지시키게 되는 것이다. 위의 시뮬레이션 결과를 보면 AG V2가 먼저 구간 1로 들어가는 경우, 즉 상태 0에서 이벤트 AGV2at12가 발생할 때도 같은 결과를 보인다.

3. Supervisor 제작

AGV 충돌 방지 슈퍼바이저를 그림 19과 같은 구조로 실제 제작하여 시험하였다. AGV 충돌방지 시스템은 다음과 같이 작동하게 된다. AGV 라인에 있는 각각의 센서들이 AGV 1, AGV 2를 감지하여 그 센서 신호를 6개의 충돌방지 modular 슈퍼바이저로 보내게 된다. 그리고 각 슈퍼바이저들은 그 센서신호를 입력으로 받아들여 미리 입력된 논리회로에 따라 feedback 과정을 거쳐 AGV 모터 구동회로로 출력신호를 보내게 되는 것이다. 그림 20은 실제 제작된 AGV 충돌방지 시스템이다.

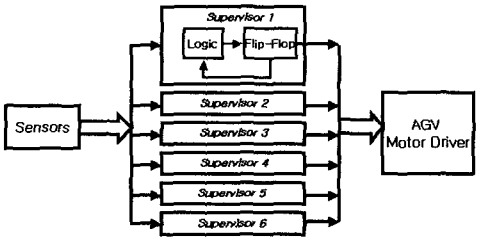


그림 19. AGV 충돌방지 시스템의 블록 다이아그램.

Fig. 19. Block diagram of AGV collision protection system.

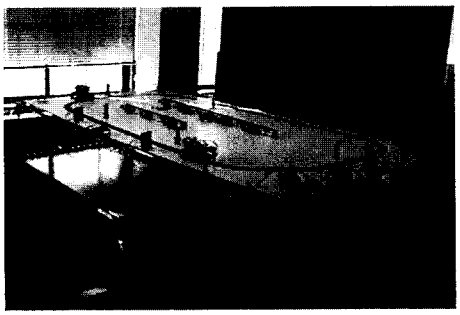


그림 20. AGV 충돌방지 시스템.
Fig. 20. AGV collision protection system.

V. 결론

본 연구에서는 CIM testbed의 제어를 위하여 supervisory control theory에 기초한 modular supervisor를 설계하고 구현하였다. Supervisor를 modular로 설계함으로써 legal language의 변화에 따른 supervisor의 수정 및 보완이 단순하며 controllability의 검사를 위한 계산량도 훨씬 줄어드는 장점이 있다. 그리고 설계된 supervisor들을 CMSSM으로 변환하고 구현을 위한 회로설계와 회로의 시뮬레이션을 수행하여 각각의 modular supervisor들이 주어진 legal language를 생성하는 것을 확인하였다. 또한 AGV 시스템을 위한 supervisor들을 실제 제작하여 AGV 시스템을 specifi- cation에 따라 제어하는 것을 관찰할 수 있었다. 이러한 과정을 통하여 supervisory control theory가 제조 시스템의 제어를 위하여 가장 제약이 적은 supervisor를 체계적으로 설계할 수 있는 방법임을 확인하였다.

본 논문에서 다룬 legal language를 low-level legal language로 두고, 플랜트의 보다 효과적인 제어를 위해 임의의 high-level legal language를 정의하여 이에따른 high-level supervisor를 구할 수 있을 것이다. 즉 hierarchical supervisory control 시스템을 구성할 수 있다. 이와 같은 시스템에서는 high-level supervisor가 low-level 플랜트의 모든 이벤트들을 관측할 필요가 없으므로 어떤 projection 함수를 두어 새로운 플랜트, 즉 high-level 플랜트를 만들 수 있다. 이렇게 함으로써 시스템을 보다 분산적이고 계층적으로 supervisory control할 수 있을 것이다. 그리고 또한 이벤트들의 발생 시간을 고려한 timed DES를 플랜트로 삼는 것도 보다 효과적인 감독 제어 시스템을 개발하는데 많은 도움을 줄 것이다. 마지막으로 AGV의 경로에 대한 의사결정, 작업순서 등에 high-level 슈퍼바이저의 설계와 AGV의 이동속도를 고려한 timed DES를 설계하는 것도 추후 연구분야로 삼을 수 있다.

참고문헌

[1] P. J. Ramadge, and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control and Optimization*, vol. 25, no. 1, January, 1987.

[2] P. J. Ramadge, and W. M. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81-98, January, 1989.

[3] 손형일, 김철수, 이석, "제조셀의 제어를 위한 DES 슈퍼바이저의 설계", '99 춘계 정밀공학회 학술회의 논문집, pp. 721-724, 1999.

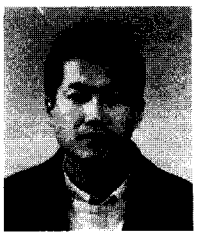
[4] W. M. Wonham, and P. J. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Control and Optimization*, vol. 25, no. 1, January 1987.

[5] W. M. Wonham, "Notes on control of discrete- event systems," Department of Electrical and Computer Engineering, University of Toronto, 1998,

[6] R. J. Leduc, "PLC implementation of a DES supervisor for a manufacturing testbed," MACs Thesis, Department of Electrical and Computer Engineering, University of Toronto, 1996.

[7] J. Wakerley. *Digital Design Principles*. Prentice-Hall, Inc., 1990.

[8] B. A. Brandin "The real-time supervisory control of an experimental manufacturing cell," *Systems Control Group Report No.9404*, Department of Electrical and Computer Engineering, University of Toronto, 1994.



손 형 일

1998년 부산대학교 생산기계공학과 졸업. 1998년 ~ 현재 부산대학교 지능기계공학과 석사과정. 관심분야는 Supervisory Control, Failure Diagnosis.

이 석

제어 · 자동화 · 시스템공학 논문지 제5권, 제5호, 참조.