이산이벤트시스템의 고장진단

Failure Diagnosis of Discrete Event Systems

손 형 일, 김 기 웅, 이 석
(Hyoungil Son, Keewoong Kim, and Suk Lee)

Abstract : As many industrial systems become more complex, it becomes extremely difficult to diagnose the cause of failures. This paper presents a failure diagnosis approach based on discrete event system theory. In particular, the approach is a hybrid of event-based and state-based ones leading to a simpler failure diagnoser with supervisory control capability. The design procedure is presented along with a pump-valve system as an example.

Keywords : discrete event systems, supervisory control, failure diagnosis

I. 서론

산업시스템의 고장진단은 지난 몇십년 동안 많은 연구자들의 관심을 끌어온 연구분야였다. 특히, 반도체 생산시스템, 자동차 생산시스템, 화학공정 시스템, HVAC (Heating, Ventilation and Air Conditioning) 시스템, 발전소 등과 같이 대규모이고 복잡한 시스템의 고장진단을 위한 보다 조직적이고 체계적인 방법이 필요하게 되었다. 그래서 고장 트리(fault tree), analytical redundancy, 전문가 시스템(expert system), model-based reasoning, 이산이벤트 시스템(DES: Discrete Event System) 등을 이용한 많은 접근론들이 연구되고 있다. 이런 접근론들에는 각각 다음과 같은 제약이 있다. 먼저 고장 트리는 기본적으로 그것을 구성하기가 지나치게 어려워서 실질적인 사례에 적용하기가 쉽지 않다. 그리고 analytical redundancy는 계산 복잡도가 크고, 모델링 오차와 계측 노이즈로 인해 민감도(sensitivity)가 문제를 야기 시킬 수 있다. 전문가 시스템은 전문가의 지식을 추출하는 것이 어려워서 개발기간이 길어질 수 있는 단점이 있다. 마지막으로 model-based reasoning은 많은 연구가 이루어지고 있지만 아직까지 동적 시스템에 대한 적용이 어렵다는 단점이 있다. 그렇지만 DES 접근법은 일반적으로 고장진단 할 대부분의 산업시스템이 DES로 생각될 수 있어 시스템을 모델링하기가 비교적 쉽고 고장진단을 위한 보다 체계적인 방법을 제시하므로 고장진단을 위한 좋은 방법이 될 수 있다[1]-[4]. 이와 같은 이유로 본 논문에서는 DES 접근법을 이용하여 고장진단 시스템을 구성하는 기법을 제시하였다.

고장진단 시스템은 두 가지 기준으로 분류될 수 있다. 첫 번째 분류기준은 고장진단이 어떠한 상태에서 이루어지는가 이다. 먼저, 오프라인 고장진단 시스템은 시스템이 정상적인 작동을 하지 않는다고 가정하고 시스템을 작동시키기 전에 테스트 명령(test command)을 보내 시스템의 상태를 우선 검사하는 것이다. 이와 반대로 온라인 고장진단 시스템은 시스템이 정상동작을 한다고 가정하고 시스템에서 발생하는 이벤트들의 흐름에 따라 시스템의 상태를 나누고 현재 시스템이 정상상태인지 비정상상태인지 규정하는 것이다. 고장진단 시스템에 대한 두 번째 분류기준은 고장진단기(failure diagnoser)가 고장진단을 위하여 능동적인 행동을 취하는가 하는 것이다. 수동적인 고장진단 시스템은 시스템의 상태와 발생하는 사건의 관찰을 통하여 고장을 진단하는 시스템을 일컫는다. 이에 반해 능동적인 고장진단 시스템은 고장진단을 위하여 시스템의 작동을 능동적으로 변화시켜 수동적인 고장진단 시스템의 한계를 극복할 수 있는 가능성을 갖고있다.

DES 접근법에 의한 고장진단 시스템 구성은 크게 이벤트기반(event-based)[1][2] 접근법과 상태기반(state-based)[3] 접근법으로 연구되고 있다. 일반적으로 이벤트기반 접근법은 상태기반 접근법에 비해서 설계과정이 단순하지만 설계된 고장진단기의 상태 수가 많아지는 단점이 있다. 이와는 대조적으로 상태기반 접근법은 관리제어기(supervisor)의 상태를 고장진단기가 알아야하는 경우가 많아서 실질적인 구현에 어려움이 따른다. 이러한 어려움을 극복하기 위하여 본 논문에서는 두 가지 접근법을 혼합하여 온라인 능동 고장진단기(on-line passive diagnoser)를 설계하였다. 본 논문에서 제안한 고장진단기는 이벤트기반 접근법과 비교해서 상태 수를 감소시킨다. 그리고 관리제어기의 기능을 고장진단기와 통합시켜서 상태기반 접근법보다 구현이 용이하다. 이렇게 본문에서 제안한 방법론에 의해서 얻어지는 고장진단기를 고장진단적 관리제어기(diagnostic supervisor)로 명명하였고 고장진단적 관리제어기를 설계하기 위한 이론을 예제와 함께 제시하였다.

논문의 구성은 다음과 같다. 2장에서는 오토마타(automata)와 형식언어(formal language)에 대한 간단한 설명과 가제어성(controllability), 가관측성(observability)에 대해서 언급하였다. 3장에서는 DES의 모델링 방법과 그에 대한 고장진단기 즉 고장진단적 관리제어기를 설계하는 방법을 나타내었다. 그리고 가진단성(diagnosa-

접수일자 : 2000. 6. 7., 수정완료 : 2001. 2. 12.
손형일 : KAIST 기계공학과(chakurt@kaist.ac.kr)
김기웅, 이 석 : 부산대학교 기계공학과(kwkim@pusan.ac.kr/slee@pun.edu)

bility)의 정의와 진단가능하기 위한 필요충분조건을 제시하고 증명하였다. 또 예제로 본문에서 제시한 방법에 의해 펌프-밸브 시스템에 대한 고장진단적 관리제어기를 설계하였다. 마지막으로 4장에서는 본 논문의 결론을 내리고 있다.

Ⅱ. 배경이론

1. 오토마타와 형식언어

유한 상태 오토마톤(FSA: Finite State Automaton)은 다음과 같은 5개의 구성요소를 가지고 있다[5]-[8].

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

여기서 Q는 상태 집합, Σ는 이벤트 집합, δ는 $\delta: Q\times\Sigma^* \to Q$인 천이함수(transition function), q_0는 초기 상태, Q_m은 작업의 완료를 나타내는 목표 상태 집합(marked state set)이다. 천이함수 δ에서 Σ^*는 빈 이벤트(null event) ε을 포함하는 이벤트의 연속(문자열: sequence 또는 string)을 나타낸다. 이렇게 구성된 FSA가 생성해 내는 언어를 $L(G)$로 나타내며

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

으로 정의한다. 여기서 $\delta(q_0,\ s)!$는 q_0에서 문자열 s가 일어난 다음의 상태가 정의됨을 나타낸다. $L(G)$의 전치 닫힘 언어(prefix closure)를 $\overline{L(G)}$로 나타내고 정의는 다음과 같다.

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

만약 $L(G)=\overline{L(G)}$이면 $L(G)$는 전치 닫힘(prefix-closed)이라고 한다. 그리고 FSA G의 목표 언어(marked language)를 $L_m(G)$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 만족하면 $L(G)$를 비막힘(nonblocking)이라고 말한다.

(1)과 같이 표현할 수 있는 FSA G_1, G_2에 대해서 G_1의 이벤트 집합을 Σ_1, G_2의 이벤트 집합을 Σ_2라 하고 $\Sigma=\Sigma_1\cup\Sigma_2$로 나타낼 때 $P_i: \Sigma^* \to \Sigma_i^*$인 투영함수(projection) P_i를

$$\begin{aligned} &P_i(\varepsilon)=\varepsilon \\ &P_i(\sigma)=\begin{cases}\varepsilon & \text{if } \sigma\notin\Sigma_i \\ \sigma & \text{if } \sigma\in\Sigma_i\end{cases} \\ &P_i(s\sigma)=P_i(s)P_i(\sigma),\ \forall s\in\Sigma^*,\ \forall\sigma\in\Sigma\end{aligned} \tag{5}$$

와 같이 정의한다. 그러면 G_1, G_2의 동기적 합성(synchronous product)은

$$G_1\times G_2=P_1^{-1}\{L(G_1)\}\cap P_2^{-1}\{L(G_2)\} \tag{6}$$

으로 정의할 수 있고 G_1, G_2의 공통적 합성(meet product)은 다음과 같이 정의한다.

$$G_1\wedge G_2=L(G_1)\cap L(G_2) \tag{7}$$

2. 부분관측하의 관리제어

2.1 가제어성과 가관측성

(1)에서 정의한 FSA G의 이벤트 집합 Σ를 $\Sigma=\Sigma_c\cup\Sigma_{uc}=\Sigma_o\cup\Sigma_{uo}$으로 서로소(disjoint)[1]하게 나눌 수 있는데 여기서 Σ_c는 제어가능 이벤트 집합(controllable event set), Σ_{uc}는 제어불가능 이벤트 집합(uncontrollable event set), Σ_o는 관측가능 이벤트 집합(observable event set), 마지막으로 Σ_{uo}는 관측불가능 이벤트 집합(unobservable event set)을 나타낸다. 그리고 (8)과 같이 자연 투영함수(natural projection) $P: \Sigma^* \to \Sigma_o^*$를 정의한다.

$$\begin{aligned} &P(\varepsilon)=\varepsilon \\ &P(\sigma)=\begin{cases}\varepsilon & \text{if } \sigma\notin\Sigma_o \\ \sigma & \text{if } \sigma\in\Sigma_o\end{cases} \\ &P(s\sigma)=P(s)P(\sigma),\ \forall s\in\Sigma^*,\ \forall\sigma\in\Sigma\end{aligned} \tag{8}$$

위와 같은 정의에 대하여 $E\subseteq G$인 임의의 FSA E의 G에 대한 가제어성, 가관측성를 정의 1, 정의 2에 나타내었다[5][6][9][10].

정의 1 : $E\subseteq G$인 E가 다음을 만족하면 E는 G에 대해서 제어가능하다고 한다.

$$\forall s\in\overline{L(E)},\ \sigma\in\Sigma_{uc},\ s\sigma\in L(G) \Rightarrow s\sigma\in\overline{L(E)}$$

정의 2 : $E\subseteq G$, $\Sigma_c\subseteq\Sigma_o$[2]에 대해서 E가 다음을 만족하면 E는 G에 대해서 관측가능하다고 한다.

$$\forall s, s',\ \sigma\in\overline{L(E)},\ P(s)=P(s'),\ s\sigma\in\overline{L(E)},$$

$$s'\sigma\in L(G) \Rightarrow s'\sigma\in\overline{L(E)}$$

2.2 최대 제어가능 및 관측가능 부분언어

정의 1, 정의 2에 따라서 G에 대한 E의 최대 제어가능 부분언어(supremal controllable sublanguage), 최대 관측가능 부분언어(supremal observable sublanguage)를 구하면 (9), (10)과 같이 나타난다[10]. 그리고 G에 대한 E의 최대 제어가능 및 관측가능 부분언어(supremal controllable and observable sublanguage)는 $\Sigma_c\subseteq\Sigma_o$라는 가정아래서 (11) 와 같다[10].

$$supC(E)=E-\{(G-E)/\Sigma_{uc}^*\}\Sigma^* \tag{9}$$

$$supO(E)=E-P^{-1}\{P(G-E)\}\Sigma^* \tag{10}$$

$$supCO(E)=G\wedge P^{-1}[supC\{P(E)\}] \tag{11}$$

---

1) Set A, B가 A≠B이고 A∩B=이 ∅ 면 A, B는 서로disjoint하다고 한다.

2) $\Sigma_c\subseteq\Sigma_o$라는 가정에 의해서 normality와 observability의 개념이 같아진다[10].

2.3 부분관측하의 감독

(11)에서 G를 플랜트, E를 정상 언어(legal language)라 두면 $supCO(E)$가 우리가 구하고자 하는 관리제어기가 된다. 이렇게 구한 관리제어기는 (S, ϕ)로 구현될 수 있다. 여기서 S는 FSA $S=\{X, \Sigma, \xi, x_0, X_m\}$를 나타내며 X는 상태 집합, Σ는 이벤트 집합, ξ는 $\xi: X \times \Sigma \rightarrow X$인 천이함수, x_0는 초기 상태, X_m는 목표 상태 집합을 의미한다. 그리고 ϕ는 $\phi: X \rightarrow 2^{\Sigma}(\supseteq \Sigma_{uc})$로 나타나는 제어 함수(control map)이다.

Ⅲ. 고장진단기 설계

1. DES 모델링

일반적으로 고장진단 할 DES는 몇 개의 시스템 구성요소, 즉 제어대상 플랜트와 이를 제어하는 관리제어기로 구성되어 있다고 가정할 수 있다. 먼저, n개의 시스템 구성요소를 유한 상태 오토마톤 $\overline{G_i}=\{\overline{Q_i}, \overline{\Sigma_i}, \overline{\delta_i}, \overline{q_{0,i}}, \overline{Q_{m,i}}\}, i=1, \ldots, n$으로 나타낼 수 있다. $\overline{G_i}$의 구성요소는 (1)에서와 같이 정의된다. 그리고 만들어진 n 개의 FSA $\overline{G_i}$를 병렬적 합성하여 전체 플랜트의 FSA $\overline{G}=\{\overline{Q}, \overline{\Sigma}, \overline{\delta}, \overline{q_0}, Q_m\}$를 만들 수 있다. 여기서 $\overline{G}$의 각 구성요소도 (1)의 정의와 같다. 그리고 이벤트 집합 $\overline{\Sigma}$는 제어가능 이벤트 집합 $\overline{\Sigma_c}$와 제어불가능 이벤트 집합 $\overline{\Sigma_{uc}}$로 서로소하게 나눌 수 있다. 또 관측가능 이벤트 집합 $\overline{\Sigma_o}$와 관측불가능 이벤트 집합 $\overline{\Sigma_{uo}}$로도 서로소하게 나누어진다. 즉, $\overline{\Sigma}=\overline{\Sigma_c} \cup \overline{\Sigma_{uc}}=\overline{\Sigma_o} \cup \overline{\Sigma_{uo}}$가 된다[5][6].

플랜트에 대한 관리제어기를 앞장에서 정의한 (S, ϕ)로 표현하면 고장진단 해야 될 DES는 $\overline{G}$와 S를 공통적 합성한 유한 상태 무어 오토마톤(FSMA: Finite State Moore Automaton) $G=\{Q, \Sigma, \delta, q_0, Q_m, Y, \lambda, C, \gamma\}$로 나타낼 수 있다.[3] FSMA G에서 Q는 $Q=\overline{Q} \times X$인 상태 집합, Σ는 $\Sigma \subseteq \overline{\Sigma}$인 이벤트 집합, δ는 $\delta: Q \times \Sigma \rightarrow 2^{Q}$인 천이함수, $q_0=(\overline{q_0}, x_0)$인 초기 상태, Q_m은 목표 상태 집합이다. 그리고 Y는 센서 출력 집합(sensor output set), λ는 $\lambda: Q \rightarrow Y$인 센서 출력 함수(sensor output map), C는 $C \subseteq \Sigma_c$인 제어 명령 집합(control command set), γ는 $\gamma: Q \rightarrow 2^{C}$인 제어 명령 함수(control command map)이다. 여기서 센서 출력은 시스템에 있는 계측센서의 값을 말하고, 제어 명령은 관리제어기의 허용가능 이벤트(enabled event)중에서 다른상태로의 천이를 실제로 발생시키는 이벤트를 의미한다.

예제 : 그림 1과 같이 펌프, 밸브, 관리제어기로 구성된 간단한 시스템을 생각하자. 그리고 계측센서로는 유량계(flowmeter)가 사용되고 이 유량계의 값은 noflow를

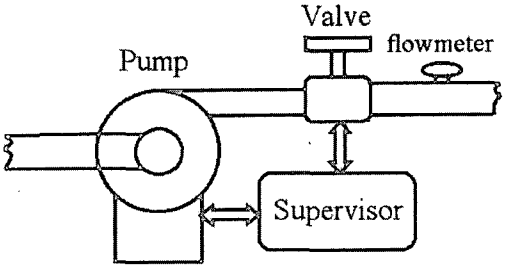
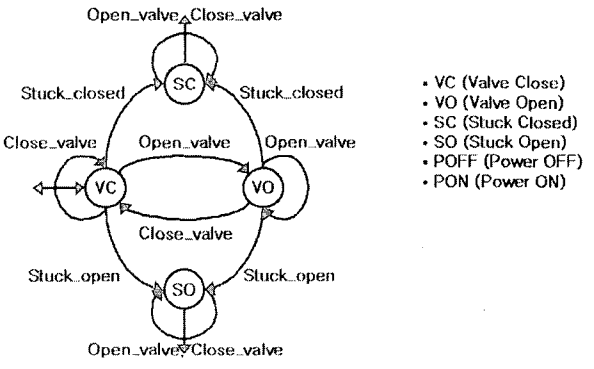


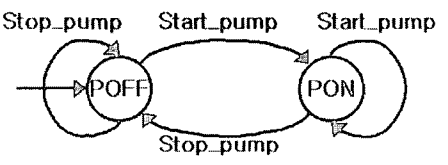
그림 1. 펌프-밸브 시스템.

Fig. 1. Pump-Valve system.

나타내는 NF와 flow를 나타내는 F 두 개가 있다고 하였다. 따라서 센서 출력 집합은 Y={NF, F}이다. 시스템에서 펌프는 고장이 발생하지 않고, 밸브에서만 고장이 일어난다고 가정한다. 펌프와 밸브의 FSA는 그림 2와 같이 나타낼 수 있는데 밸브의 상태 VC와 VO에서 밸브가 닫힌 상태에서 밸브가 더 이상 열리지 않는 고장 Stuck_closed와 밸브가 열린 상태에서 밸브가 더 이상 닫히지 않는 고장 Stuck_open의 고장이 일어날 수 있다고 가정하였다. 밸브와 펌프의 초기 상태는 각각 VC, POFF로 잡았다. 그리고 밸브와 펌프의 목표 상태 집합은 역시 각각 {VC}, {POFF}로 둬야겠지만, 이때 밸브의 경우 상태 SC, SO에서 막힘(blocking)이 일어나므로 (12)에 의해서 정상 언어 E의 $supCO(E)$가 존재하지 않게 된다. 그러므로 밸브의 경우 목표 상태 집합을 {VC, SC, SO}로 정의하였다.



(a) $\overline{G_{valve}}$



(b) $\overline{G_{pump}}$

그림 2. 구성요소의 FSA $\overline{G_i}$. (a) 밸브, (b) 펌프.

Fig. 2. FSA $\overline{G_i}$ of components. (a) Valve, (b) Pump.

3) 일반적으로 $L(S/\overline{G})=L(S)$이므로[5] S에 λ, Y만을 추가시키면 G가 된다.

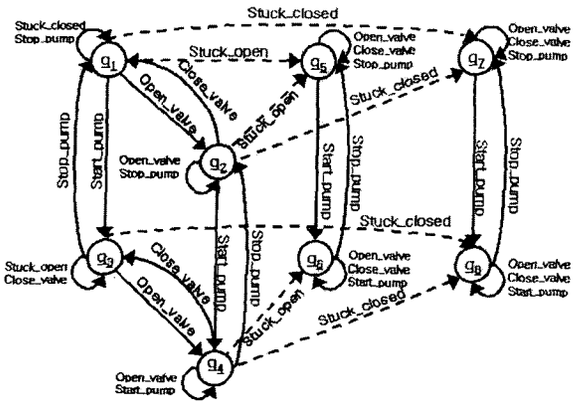


그림 3. 플랜트의 FSA $\overline{G}$.

Fig. 3. FSA $\overline{G}$ of plant.

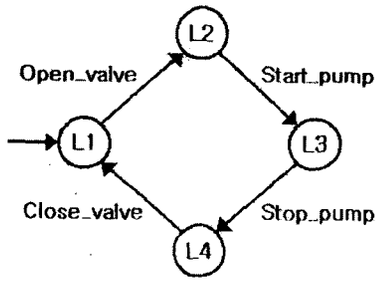


그림 4. 정상 언어의 FSA E.

Fig. 4. FSA E of legal language.

그림 2의 FSA에서 $\overline{\Sigma_{uc}}$={Stuck_closed, Stuck_open}가 되고 고장은 당연히 관측불가능 이벤트이기 때문에 $\overline{\Sigma_{uc}}=\overline{\Sigma_{uo}}$이다. 그림 2의 밸브와 펌프의 FSA를 병렬적 합성한 결과가 그림 3에 나와있다. 그림 3에서 관측불가능 이벤트는 상태천이를 점선으로 나타내고 관측가능 이벤트는 실선으로 나타내었다. 그리고 그림 4에서 스템의 관리제어기를 구하기 위한 정상 언어 E를 나타내었다. 여기서 제어 명령 집합은 C={Open_valve, Close_valve, Start_pump, Stop_ pump}이다.

이제 정상 언어에 대한 최대 제어가능 및 관측가능 부분언어 즉, 관리제어기 S를 구하는 것이 필요하다. 이를 위하여 (11)를 사용해야 하는데 이는 DES 설계를 위한 소프트웨어 TCT[5]를 사용하였고 구해진 관리제어기를 그림 5에 나타내었다. 그리고 관리제어기의 제어 함수 ϕ는 표 1에 나타내었다. 마지막으로 DES의 FSMA를 구하기 위해서 그림 3의 $\overline{G}$와 그림 5의 S를 공통적 합성하여 G를 구하면 상태 천이가 그림 5와 동일하게 얻어진다. 즉, $L(G)=L(S/\overline{G})=L(S)\bigcap L(\overline{G})=L(S)$가 된다. 따라서 G는 그림 5의 S에서 상태만을 $x_i, i=1,\ldots,12$에서 $q_i, i=1,\ldots,12$으로 일대일 대응시키면 된다. 그리고 DES G의 제어 명령 함수 γ 역시 관리제어기의 제어 함수 ϕ에서 상태를 $x_i, i=1,\ldots,12$에서 $q_i, i=1,\ldots,12$으로 일대일 대응시키면 쉽게 얻어

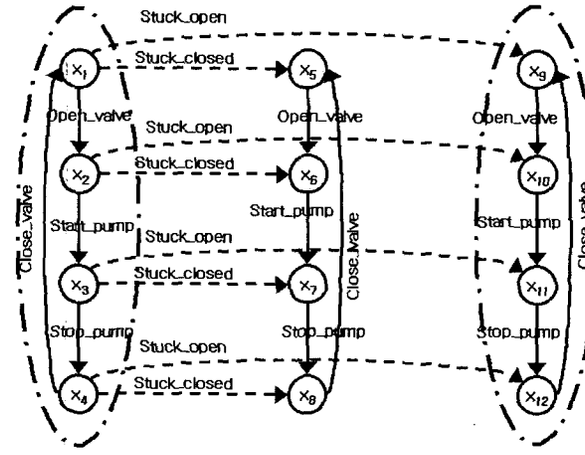


그림 5. 관리제어기의 FSA S.

Fig. 5. FSA S of supervisor.

표 1. 예제의 제어 함수.

Table 1. Control map ϕ of example.

State	Control Command C
x_1	Open_valve
x_2	Start_pump
x_3	Stop_pump
x_4	Close_valve
x_5	Open_valve
x_6	Start_pump
x_7	Stop_pump
x_8	Close_valve
x_9	Open_valve
x_{10}	Start_pump
x_{11}	Stop_pump
x_{12}	Close_valve

표 2. 예제의 센서 출력 함수 λ.

Table 2. Sensor output map λ of example.

State	Sensor Output Y
q_1	NF
q_2	NF
q_3	F
q_4	NF
q_5	NF
q_6	NF
q_7	NF
q_8	NF
q_9	NF
q_{10}	NF
q_{11}	F
q_{12}	NF

지므로 FSA G와 제어 명령 함수 γ는 생략하였다. 최종적으로 우리가 구하고자 하는 펌프-밸브 시스템의 FSMA G는 위에서 구한 FSA G, 제어 명령 함수 γ와 표 2에 나타낸 센서 출력 함수를 추가시키면 된다.

2. 고장진단기

먼저 DES의 상태 조건(state condition)을 정의하기 위해서 이벤트 집합 Σ를 정상 이벤트 집합(normal event set) Σ_N과 고장 이벤트 집합(failure event set) Σ_F로 나누었다. 그리고 Σ_F는 m개의 고장 이벤트 f_i , $i=1,\ldots,m$에 대해서 $\Sigma_F=\{f_1,\ldots,f_m\}$라 가정하였다. G의 임의의 상태 q에서 고장 이벤트 f_i가 발생하여 도달한 상태 q', 즉 $q'=\delta(q,f_i)$인 q'를 q_{f_i}로 나타내고 또, $F_1,\ldots,F_n$을 고장 모드(failure mode)라 하고, $K=\{N,F_1,\ldots,F_n\}, n\le m$을 DES의 상태 조건 집합(state condition set)으로 나타내었다. 고장 모드는 Σ_F를 n개로 서로소 분할(partition)한 것이다. 즉, Σ_F를 같은 부류로 취급할 수 있는 고장의 종류로 다시 나눈 것이다. 그리고 Q_N, Q_{F_i}를 (12), (13)와 같이 각각 정의하면 DES의 상태 집합 Q 역시 상태 조건에 따라서 $Q=Q_N\cup Q_{F_1}\cup\ldots\cup Q_{F_n}$으로 서로소하게 나눌 수 있다. 마지막으로 K에 따라 (14)와 같이 DES의 상태 조건 함수(state condition map) $x:Q\to K$를 정의하였다.

$$Q_N=\{q' \mid \forall q,\sigma\in\Sigma_N, q'=\delta(q,\sigma)\} \tag{12}$$

$$Q_{F_i}=\{q_{f_i} \mid \forall f_i, f_i\in F_i, q_{f_i}=\delta(q,f_i)\} \tag{13}$$

$$x(q)=\begin{cases} N & \text{if } q\in Q_N \\ F_i & \text{if } q\in Q_{F_i} \\ \textit{undefined} & \textit{elsewhere}\end{cases} \tag{14}$$

이제 DES G에 대한 고장진단기를 FSMA $D=\{Z,H,\zeta,z_0,Z_m,\overline{K},\overline{x}\}$로 정의하면 FSMA D에서 Z는 $Z=2^Q-\varnothing$인 상태 집합, H는 $H=Y\times C$인 이벤트 집합, z_0는 $z_0=\{q_0\cup q_0'\}$인 초기 상태, Z_m은 $Z_m\supseteq Q_m$인 목표 상태 집합, $\overline{K}$는 $\overline{K}=2^K-\varnothing$인 상태 조건 집합, $\overline{x}$는 $\overline{x}:Z\to\overline{K}$인 상태 조건 함수이다. 이렇게 정의한 고장진단기는 관리제어기의 제어 함수를 유지하므로 이런 고장진단기를 고장진단적 관리제어기라고 명명하였다. 그리고 z_0에서 q_0'은 다음과 같이 정의하였고 이를 개념적으로 그림 6에 나타내었다.

$$q_0'=\bigcup\{q' \mid q'=\delta(q_0,F_i), \forall F_i\in\Sigma_F\} \tag{15}$$

초기 상태를 이렇게 정의한 이유는 고장진단기의 초기 상태를 위와 같이 z_0로 둠으로써 DES의 초기 상태가 정상인지 비정상인지 고려할 필요가 없기 때문이다. 즉 고장진단기가 초기화되기 전에 고장이 발생했던지 초기화되고 난 후 고장이 발생하던지 고장진단기의 작동에

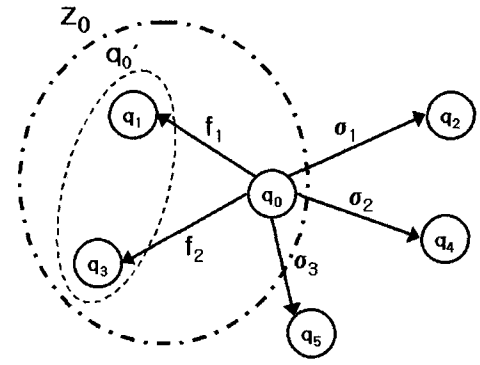


그림 6. 고장진단기의 초기 상태.
Fig. 6. Initial state of diagnoser.

는 아무런 문제가 없다. 이와 다르게 [1][2]에서는 시스템이 정상 상태에서 시작한다고 가정을 하였는데 이는 현실적으로 불확실하므로 고장진단기의 초기 상태를 위와 같이 정의를 하는 것이 보다 일반적이라고 할 수 있다. 또 [3]에서는 초기 상태를 모든 상태 집합 또는 정상 상태 집합으로 가정하였는데 이 역시 고장진단적 관리제어기는 관리제어기의 초기 상태에서 시작해야 되므로 적절하지 못하다.

DES로부터 고장진단기를 설계하기 위해서는 고장 이벤트 등과 같은 관측불가능 이벤트들, 그리고 관측가능 이벤트지만 관리제어기의 제어 명령 집합에는 포함되지 않는 이벤트들을 제외한 이벤트만으로 발생하는 상태 천이를 구성해야된다. 정의 3에 이를 정의하였다.

정의 3 : 다음을 만족시키는 DES G의 모든 상태 천이 (q,σ,q')을 제어 명령 천이 시스템(CCTS: Control Command Transition System)이라 한다.

$$\forall\sigma\in C,\ q'=\delta(q,\sigma)$$

CCTS는 DES에서 발생하는 상태 천이 중에서 관리제어기에 의해서 허용가능 천이(enable transition)만을 나타낸다. 그림 6을 예로 들면 상태 천이 (q_0,f_1,q_1), (q_0,f_2,q_3) 등은 그림 6의 DES에 대한 CCTS에 포함되지 않는다.

고장진단기의 천이 함수 ζ는 아래와 같이 정의한다. 단, $n(\cdot)$은 $\cdot$의 개수를 의미한다.

$$\zeta(z_k,\gamma_k)=\begin{cases} z_{k+1} & \text{if } n(y_{k+1})=1 \\ z_{k+1+i},\ m-1\ge i\ge 0 & \text{if } n(y_{k+1})\ge 2, \\ & y_{k+1}=\{y_{k+1}^0,\ldots,y_{k+1}^{m-1}\} \\ \textit{undefined} & \textit{elsewhere}\end{cases} \tag{16}$$

그리고 그림 7에 천이 함수 ζ를 개념적으로 나타내었다.

(16)이 의미하는 것은 다음과 같다. 고장진단기의 상태 z_k에서 관리제어기가 γ_k인 제어 명령을 발생시키면 고장진단기의 상태는 z_{k+1}로 천이된다. 그런데 만약 제어 명령 γ_k가 발생한 후 2개 이상의 센서 출력이 정의될 수 있으면 센서 출력에 따라 z_{k+1}을 만들어 준다. 이

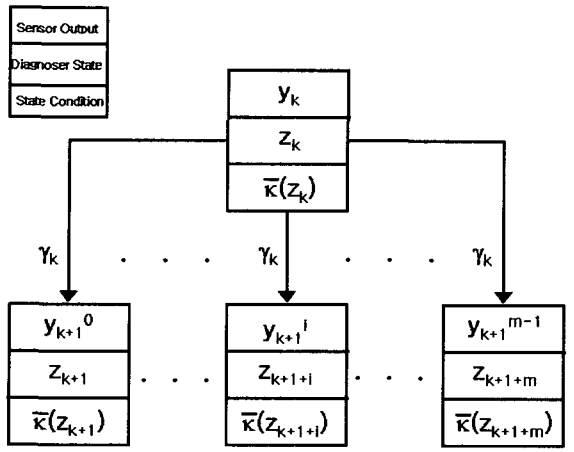


그림 7. 천이함수 ζ.
Fig. 7. Transition function ζ.

는 각각의 센서 출력에 따라서 고장진단기의 상태 조건을 정의하기 위해서이다. 즉 ζ는 고장진단기의 현재 상태 z_k에서 제어 명령 γ_k가 발생한 후 센서 출력 값에 따라서 고장진단기의 다음 상태 z_{k+1}을 결정하는 것이다. 따라서 CCTS에서 센서 출력이 다른 상태들을 서로 구분해주면 된다.

본 논문에서 제시한 천이 함수 ζ는 [1]-[3]과 비교하여 다음과 같은 이유로 다르며 더욱 효과적이다. 먼저 [1][2]에서는 관측가능 이벤트가 발생할 때마다 고장진단기의 상태 조건을 수정(update)해야 되지만 본문서 제시한 방법으로는 센서 출력 값이 바뀌거나 관리제어기가 새로운 제어 명령을 내릴 때만 상태 조건을 수정하면 된다. 그러므로 본 논문에서 제시한 고장진단기의 상태 개수는 [1][2]의 경우보다 적거나 같게된다. 그리고 [3]에서는 관리제어기의 상태를 고장진단기의 상태 출력에 추가하여야 될 경우가 있는데, 이때 고장진단기는 항상 관리제어기와 동기화 되어 서로 상태 정보를 공유해야된다. 그렇지만 본 논문에서 제안하는 고장진단기는 고장진단기 자체가 제어와 고장진단을 동시에 하는 고장진단적 관리제어기이기 때문에 [3]보다 더욱 간단해진다.

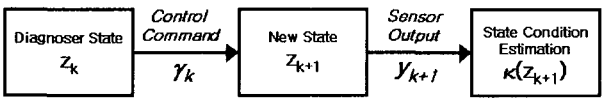


그림 8. 온라인 고장진단.
Fig. 8. On-line failure diagnosis.

DES에 대한 온라인 고장진단은 그림 8과 같이 이루어진다. 그림 8에서 고장진단기의 현재 상태 z_k에서 제어 명령 γ_k가 발생하면 새로운 상태 z_{k+1}로 간다. 그리고 z_{k+1}의 센서 출력 y_{k+1}을 읽어들이고 z_{k+1}의 상태 조건을 추정함으로써 DES를 온라인 고장진단 한다. 따라서 시스템 작동 전에 DES의 모든 상태에 대해서 상태 조건을 추정해야되는 오프라인 고장진단보다 현재 고장진단기의 상태 조건만을 추정하면 되므로 오프라인 고장진단보다 간단하고 계산 복잡도가 줄어드는 이점이 있다.

예제(계속) : 앞에서 구성한 DES에 대해서 Stuck_closed 이벤트를 고장 모드 F_1, Stuck_open 이벤트를 고장 모드 F_2로 두자. 그러면 상태 조건 집합은 $K=\{N,F_1,F_2\}$이 된다. 그리고 상태 조건 함수 x는 표 3과 같이 정의된다.

고장진단기를 만들기 전에 먼저 고장진단기의 초기 상태 z_0를 구하면 $q_5=\delta(q_1,F_1)$, $q_9=\delta(q_1,F_2)$이므로 $z_0=\{q_1,q_5,q_9\}$가 된다. 그리고 예제의 DES Q에 대한 CCTS를 정의 3에 의해서 만들면 표 4와 같이 나타난다.

표 4는 다음과 같은 과정에 의해서 만들어 졌다. DES

표 3. 예제의 상태 조건 함수 x.
Table 3. State condition map x of example.

State	State Condition x
q_1	N
q_2	N
q_3	N
q_4	N
q_5	F_1
q_6	F_1
q_7	F_1
q_8	F_1
q_9	F_2
q_{10}	F_2
q_{11}	F_2
q_{12}	F_2

표 4. 예제의 CCTS.
Table. 4. CCTS of example.

q	$\sigma=\gamma(q)$	$q'=\delta(q,\sigma)$	$\lambda(q)$
q_1	Open_valve	q_2, q_6, q_{10}	NF
q_2	Start_pump	q_3, q_{11}	F
		q_7	NF
q_3	Stop_pump	q_4, q_8, q_{12}	NF
q_4	Close_valve	q_1, q_5, q_9	NF
q_5	Open_valve	q_6	NF
q_6	Start_pump	q_7	NF
q_7	Stop_pump	q_8	NF
q_8	Close_valve	q_5	NF
q_9	Open_valve	q_{10}	NF
q_{10}	Start_pump	q_{11}	F
q_{11}	Stop_pump	q_{12}	NF
q_{12}	Close_valve	q_9	NF

Q의 상태 q_1에서 발생하는 상태 천이는 $(q_1, \text{Open\_valve}, q_2)$, $(q_1, \text{Stuck\_closed}, q_5)$, $(q_1, \text{Stuck_open}, q_9)$이다. 그렇지만 이벤트 Stuck_open, Stuck_closed는 제어 명령 집합 C에 속하지 않으므로 transition $(q_1, \text{Stuck\_closed}, q_5)$, $(q_1, \text{Stuck_ope}, q_9)$는 CCTS에서 제외되고 상태 천이 $(q_1, \text{Open\_valve}, q_6)$, $(q_1, \text{Open_valve}, q_{10})$이 포함된다. 나머지 상태에 대해서도 같은 과정으로 CCTS를 만들면 된다.

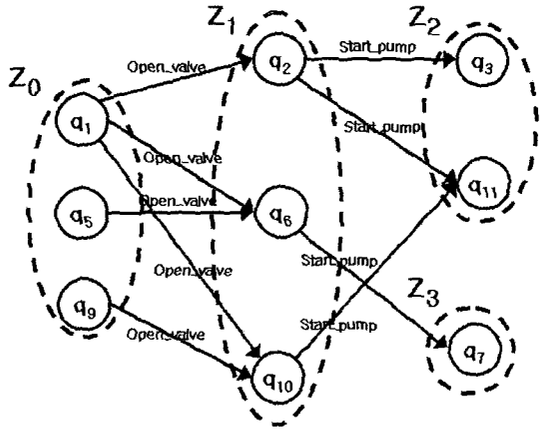


그림 9. 고장진단기 설계과정의 일부.
Fig. 9. A part of diagnoser design procedure.

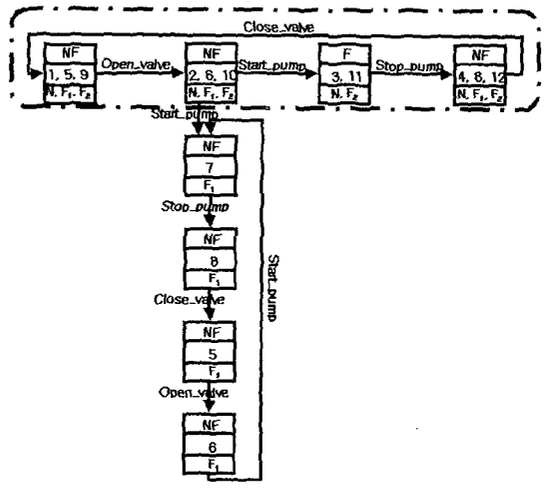


그림 10. 펌브-밸프 시스템에 대한 고장진단기.
Fig. 10. Diagnoser of Pump-Valve system.

CCTS와 위에서 정의한 천이 함수 ζ에 의해서 펌프-밸브 시스템의 고장진단기를 만드는 과정을 그림 9에 나타내었다. 초기 상태 $z_0=\{q_1, q_5, q_9\}$에서 제어 명령 Open_valve가 발생하면 DES Q의 천이 함수 δ에 의해서 상태 q_2, q_6, q_{10}으로 천이 된다. 이는 표 4의 CCTS로 쉽게 확인된다. 그리고 $\lambda(q_2)=\lambda(q_6)=\lambda(q_{10})$=NF이므로 즉, q_2, q_6, q_{10}의 센서 출력이 하나이기 때문에 (17)에 의해서 $z_1=\{q_2, q_6, q_{10}\}$이 된다. $z_1=\{q_2, q_6, q_{10}\}$에서는 표 4의 CCTS에서 알 수 있듯이 제어 명령 Start_pump가 발생하면 상태 q_3, q_7, q_{11}로 천이된다. 그런데 $\lambda(q_3)=\lambda(q_{11})$=F, $\lambda(q_7)$=NF이므로 $z_2=\{q_3, q_{11}\}$, $z_3=\{q_7\}$로 분리된다. 이런 과정을 거쳐서 최종 설계된 고장진단기를 그림 10에 나타내었다.

그림 10을 보면 상태 $\{q_2, q_6, q_{10}\}$에서 제어 명령 Start_pump가 발생한 후 센서 출력 값이 F가 되면 고장진단기의 상태는 $\{q_3, q_{11}\}$가 되고, NF가 되면 $\{q_7\}$가 된다. 즉 관리제어기에서는 하나의 제어 명령을 보내지만 고장진단기에서는 센서 출력에 따라서 상태를 구분한다.

설계된 고장진단기의 목표 상태 집합 Z_m은 $\{q_1, q_5, q_9\}$이다. 그런데 상태 $\{q_7\}$ $\{q_8\}$,, $\{q_5\}$, $\{q_6\}$에서는 $\{q_1, q_5, q_9\}$로 상태 천이가 일어나지 않는다. 즉 막힘이 발생한다. 그러므로 고장진단적 관리제어기의 막힘 해결 문제가 아직 남아있다.

3. 가진단성

앞 절에서는 주어진 DES에 대해서 고장진단기를 설계하는 방법을 제시하였다. 본 절에서는 설계된 고장진단기가 어떻게 DES의 상태를 규정하여 고장을 검출하는지를 설명한다.

고장진단기의 가진단성를 정의하기 전에 먼저 고장진단기의 상태 집합 Z를 상태 조건에 따라서 정의 4, 5, 6과 같이 나눌 수 있다.

정의 4 : $\overline{x(z)}=\{N\}$이면 상태 z를 정상(normal)이라 한다.

정의 5 : $\overline{x(z)}\supset\{F_i\}$, $\overline{x(z)}\not\subset\{F_i\}$이면 상태 z를 F_i-불확정(F_i-uncertain)이라 한다.

정의 6 : $\overline{x(z)}=\{F_i\}$이면 상태 z를 F_i-확정(F_i-certain)이라 한다.

그림 10의 고장진단기를 예로 들면 상태 $\{q_1, q_5, q_9\}$, $\{q_2, q_6, q_{10}\}$, $\{q_4, q_8, q_{12}\}$는 F_1, F_2-불확정, $\{q_3, q_{11}\}$은 F_2-불확정이고 상태 $\{q_7\}, \{q_8\}, \{q_5\}, \{q_6\}$은 F_1-확정이다. 그리고 정상인 상태는 존재하지 않는다.

정의 7에 고장진단기의 가진단성을 정의하였다.

정의 7 : 고장진단기가 초기화되고 임의의 고장 모드 F_i가 발생한 후 임의의 이벤트들이 N_i번 발생한 후 고장진단기의 상태가 F_i-확정이 되면 고장진단기는 F_i-진단가능(F_i-diagnosable)하다고 한다. 그리고 모든 고장 모드 F_i에 대해서 F_i-진단가능하다면 고장진단기는 진단가능(diagnosable)하다.

예제(계속) : 고장진단기의 상태 $\{q_7\}, \{q_8\}, \{q_5\}, \{q_6\}$가 F_1-확정이므로 펌프-밸브 시스템의 고장진단기는 F_1-진단가능하다. 그렇지만 F_2-확정인 상태가 없으므로 F_2-진단가능하지는 않다.

마지막으로 고장진단기가 F_i-진단가능하기 위한 필

요충분조건을 만들기 위해 다음의 몇 가지를 정의하자.

정의 8 : DES에서 $q_{i+1}=\delta(q_i,\sigma_i)$, $i=1,\ldots,n$ 그리고 $q_1=\delta(q_{n+1},\sigma_{n+1})$인 상태 집합 $\{q_1,q_2,\ldots,q_{n+1}\}$은 사이클(cycle)을 이룬다고 한다.

정의 9 : 고장진단기에서 F_i-불확정인 상태 집합 $\{z_1,z_2,\ldots,z_{n+1}\}$가 사이클을 이룬다고 하자. 이때 다음을 만족시키는 사이클 $\{q_1^N,q_2^N,\ldots,q_{k+1}^N\}$, $\{q_1^{F_i},q_2^{F_i},\ldots,q_{l+1}^{F_i}\}$가 DES에 존재한다면 고장진단기의 상태 집합 $\{z_1,z_2,\ldots,z_{n+1}\}$가 F_i-비결정 사이클(F_i-indeterminate cycle)을 이룬다고 한다.

1) $x(q_m^N)=N$, $m=1,\ldots,k+1$, $q_m^N\in z_r$, $m=1,\ldots,k+1$, $r=1,\ldots,n+1$, $k\le n$ 그리고 $\{q_1^N,q_2^N,\ldots,q_{k+1}^N\}\subseteq\{z_1,z_2,\ldots,z_{n+1}\}$인 DES의 상태 집합 $\{q_1^N,q_2^N,\ldots,q_{k+1}^N\}$가 사이클을 이룬다. 이런 사이클을 N-사이클이라 한다.

2) $x(q_{m'}^{F_i})=F_i$, $m'=1,\ldots,l+1$, $q_{m'}^{F_i}\in z_r$, $m'=1,\ldots,l+1$, $r=1,\ldots,n+1$, $l\le n$ 그리고 $\{q_1^{F_i},q_2^{F_i},\ldots,q_{l+1}^{F_i}\}\subseteq\{z_1,z_2,\ldots,z_{n+1}\}$인 DES의 상태 집합 $\{q_1^{F_i},q_2^{F_i},\ldots,q_{l+1}^{F_i}\}$가 사이클을 이룬다. 이런 사이클을 F_i-사이클이라 한다.

정리 1 : 고장진단기에 F_i-비결정 사이클이 존재하지 않는다는 것은 고장진단기가 F_i-진단가능하기 위한 필요충분조건이다.

증명 : 고장 모드 F_i가 발생한 후 고장진단기의 상태는 정상, F_i-불확정, F_i-확정 중의 하나가된다.

1) 정상일 경우 : 충분조건) F_i-비결정 사이클이 존재하지 않으므로 임의의 이벤트 발생 후 상태 조건이 F_i가 되는 상태로 가는 경로가 존재한다. 따라서 결국에는 고장진단기의 상태는 F_i-확정이 되어 F_i-진단가능하게 된다.

필요조건) 고장진단기가 F_i-진단가능하므로 F_i-확정인 상태로 가는 경로가 존재한다. 그러므로 F_i-불확정인 상태들의 사이클인 F_i-비결정 사이클은 존재하지 않는다.

2) F_i-불확정일 경우 : 정상일 경우와 같다.

3) F_i-확정일 경우 : 현재 상태가 F_i-확정이므로 당연히 F_i-진단가능하고 F_i-확정인 상태로 가는 경로가 존재하므로 역시 F_i-비결정 사이클이 존재하지 않는다.

예제(계속) : 그림 10의 고장진단기를 보면 고장 모드 F_1에 대해서는 비결정 사이클이 존재하지 않지만, F_2에 대해서는 비결정 사이클이 존재한다. 즉, $\{\{q_1,q_5,q_9\},\{q_2,q_6,q_{10}\},\{q_3,q_{11}\},\{q_4,q_8,q_{12}\}\}$가 F_2-비결정 사이클을 이룬다. 왜냐하면 그림 4에서 $\{q_1,q_2,q_3,q_4\}$가 상태 조건 N으로 사이클을 이루고있고, 상태 집합 $\{q_9,q_{10},q_{11},q_{12}\}$ 역시 상태 조건 F_2로 사이클을 이루고 있기 때문이다. 즉, N-사이클, F_2-사이클이 존재한다. 따라서 F_2-비결정 사이클이 존재하므로 F_2-진단가능하지 않다. 그렇지만 고장 모두 F_1에 대해서는 상태 집합 $\{\{q_7\},\{q_8\},\{q_5\},\{q_6\}\}$가 상태 조건 F_1으로 사이클을 이루고 있으며 이는 시스템에서 F_1-사이클만을 이루고 있으므로 F_1-비결정 사이클은 존재하지 않는다. 그러므로 F_1-진단가능하게 된다.

이것의 물리적인 의미는 다음과 같다. 그림 10의 초기 상태 $\{q_1,q_5,q_{10}\}$에서 만약 고장 이벤트 Stuck_ closed가 발생하고 Open_valve 명령이 내려지더라도 고장진단기의 상태는 아직 F_1-불확정을 나타낸다. 이때 제어 명령 Start_pump가 일어났을 때 센서 출력이 NF가 되면 F_1-확정이 되는 것이다. 즉, 펌프를 작동시켰는데도 유량계가 NF를 가리키므로 밸브가 닫혀있음을 d고장진단기는 알 수 있는 것이다. 따라서 고장진단기는 Stuck_closed 고장 이벤트가 발생했다고 판단한다. 하지만 고장 이벤트 Stuck_open이 발생했을 때는 유량계의 값이 정상적으로 나타나므로 고장진단기는 고장을 알지 못하는 것이다. 이는 고장진단기가 유량계의 값만으로는 N-사이클, F_2-사이클을 구분하지 못하여 F_2-비결정 사이클에서 빠져 나오지 못하게됨을 의미한다. 따라서 Stuck_open을 검출하기 위해서는 추가적인 센서가 필요함을 나타낸다.

Ⅳ. 결론

본 논문에서는 DES 접근론적 고장 진단 문제를 다룸에 있어서 기존에 제시되었던 이벤트기반 접근법과 상태기반 접근법을 결합하여 관리 제어와 고장 진단을 동시에 수행하는 온라인 수동 고장진단적 관리제어기를 설계하는 방법을 제시하였다. 그리고 이 방법은 모든 관측가능 이벤트에 대해서 상태 조건을 업데이트할 필요가 없으므로 이벤트기반 접근법보다 간단한 고장진단기를 구성하며, 관리제어기의 기능이 고장진단기에 포함되어 있으므로 상태기반 접근법보다 구현시 용이하다는 장점이 있다. 또 가진단성를 정의하고 진단가능하기 위한 필요충분조건을 제시하여 주어진 시스템이 진단가능한지 오프라인 상태에서도 확인할 수 있게 하였다. 진단가능하기 위한 조건에 의해서 진단가능한 시스템에 불필요한 센서가 있는지, 진단가능하지 않는 시스템에 어떤 센서가 필요한지도 간접적으로 제시해준다.

추후 연구과제로는 임의의 고장 이벤트에 대해서 가진단성을 만족하지 않는 DES를 적절한 제어 신호에 의해서 가진단성을 만족시키게 만드는 능동 고장 진단 문제를 다루는 것을 들 수 있겠다. 그리고 시스템이 진단가능하기 위한 최적의 센서 개수와 위치를 찾는 것과 본문에서 언급한 고장진단적 관리제어기의 막힘 해결 방안도 앞으로 다루어야할 문제이다.

참고문헌

[1] Meera Sampath, *A Discrete Event Systems Approach to Failure Diagnosis*, Ph.D. Thesis, The University of Michigan, December, 1995.

[2] M. Sampath, S. Lafortune, and D. Teneketzis, "Active diagnosis of discrete-event systems," *IEEE Trans. Automat. Control*, vol. 43, no. 7, pp. 908-929, July, 1998.
[3] Shahin Hashtrudi Zad, *Fault Diagnosis in Discrete-Event and Hybrid Systems*, Ph.D. Thesis, The University of Toronto, 1999.
[4] Yongseok Park, *Model-based Monitoring of Discrete Event Systems*, Ph.D. Thesis, The Purdue University, May, 1996.
[5] W. M. Wonham, *Notes on Control of Discrete Event Systems*, Department of Electrical and Computer Engineering, University of Toronto, 1998.
[6] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, pp. 81-98, Jan., 1989.
[7] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addision-Wesley, 1979.
[8] B. A. Davey and H. A. Priestley, *Introduction to Lattices and Order*, Cambridge Universit Press, 1990.
[9] W. M. Wonham and P. J. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Control and Optimization*, vol. 25, no. 3, May, 1987.
[10] R. D. Brandt, V. Garg, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham, "Formulas for calculating supremal controllable and normal sublanguages," *Syst. Contr. Lett.*, vol. 15, no. 2, pp. 111-117, Aug., 1990.

손 형 일

1976년 2월 16일생. 1998년 부산대학교 생산기계공학과 졸업. 부산대학교 지능기계공학과 석사(2000). 2000년~현재 한국과학기술원 기계공학과 박사과정. 관심분야는 DES의 분산관리 제어 및 고장진단, 반도체 공정 시스템 자동화, 필드버스.

이 석

1961년 12월 11일. 1984년 서울대 기계공학과 졸업. 1985년 Pennsylvania State Univ. 석사, 동대학원 박사(1990년), 1990년~1993년 신시내티 대학교 기계공학과 조교수, 1993년~현재 부산대학교 기계공학부 부교수. 관심분야는 자동화네트워크(Profibus, Fieldbus Foundation), 차량용 네트워크(CAN, LIN, KWP2000), DES, 자율주행.

김 기 웅

1972년 11월 29일. 1996년 부산대 생산기계공학과 졸업. 동대학원 석사(1998), 1998년~현재 동대학원 박사과정. 관심분야는 자동화 및 제어용 프로토콜(CAN, LIN, KWP2000), 자동차용 운영체제, 실시간제어, 이산사건시스템(DES).