DOI : 10.5392/JKCA.2010.10.11.341

경보-원인 경로 추적시스템 개발

Development of an Alarm-Cause Path Tracking System

류승필*, 김상훈*, 김은주*, 김정택**
세명대학교 컴퓨터학부*, 한국원자력연구원**

Sung-Pil Lyu(lsp415@semyung.ac.kr), Sang-Hoon Kim(kimsh@semyung.ac.kr),
Eun-Ju Kim(eunju@semyung.ac.kr), Jung-Taek Kim(jtkim@kaeri.re.kr)

요약

경보시스템은 원자력발전소의 안전을 위해 매우 중요하다. 운전원은 경보발생시 경보와 그 원인 간의 논리적 관계를 파악하기 위해 논리도면을 참조한다.

이 논문은 전산화된 월성원자력발전소 3, 4호기 경보 논리도면에서 경보-원인 경로를 추적하는 시스템을 제안한다. 또한 논리도면의 전산화를 위하여 논리도면을 2차원 문자배열로 표시하고 이를 검증할 수 있는 문법을 제시였다. 그리고 추적된 경보-원인 간 논리경로와 논리 상태를 표시하기위하여 ND 및 DC 연산을 제안하였다. 이 시스템은 현재 월성원자력발전소에서 운영 중에 있다.

■ 중심어 : | 경보원인경로추적 | 논리도면 전산화 |

Abstract

Alarm system is very important for the safety of nuclear power plant. When alarm, operator refers alarm logic diagrams to identify the logical relationship between the alarm and its causes.

This paper propose a system which tracks the logical path between alarm and its causes on the alarm logic diagrams of Wolsung nuclear power plant unit 3 & 4. And a grammar for the validation of logic diagrams expressed in 2 dimensional strings, and logical operations with 3 states to track alarm-cause paths and to display the state of logics are proposed. This system is on operation at Wolsung site.

■ keyword : | Alarm-cause Path Tracking | Computerization of Logic Diagram |

I. 서 론

화학에너지의 고갈과 지구 환경 오염에 대한 우려로 최근 원자력 발전소에 대한 관심이 매우 커지고 있다. 그러나 원자력발전소는 중대사고 발생 시 방사능 유출에 의한 환경과 인간에게 심각한 영향을 줄 수 있기 때문에 건설에 있어서 매우 조심스러운 접근을 하고 있다. 이로 인해 원자력발전소 건설 및 운영에 있어서 가장 중요한 것은 안전으로 간주되고 있으며 이에 대한 많은 연구들이 진행되고 있다.

발전소 운영 시에 안전을 위해 중요한 부분 중에 하나가 사고 직전 또는 직후에 일어나는 현상을 운전원에게 알려주고 이에 대한 조치를 취할 수 있게 하는 경보시스템이다. 그러나 원전 사고 전후에 발생하는 경보는 적게는 몇 개에서부터 많게는 몇 백 개까지 한꺼번에 발생하므로 짧은 시간에 경보의 원인을 파악하고 조치

접수번호 : #101020-009
접수일자 : 2010년 10월 20일
심사완료일 : 2010년 11월 02일
교신저자 : 류승필, e-mail : lsp415@semyung.ac.kr

를 취하는 것은 매우 어려운 작업 중에 하나이다. 그래서 많은 경보가 동시에 발생하면 이들 중에서 중요한 경보를 선택하여 보여주는 방법이나, 경보의 원인을 찾아주는 방법 등에 대한 연구가 활발히 진행되고 있다[1-13].

경보가 발생하면 운전원은 중요 경보와 그 원인, 원인과 경보간의 논리적 관계, 경보조치를 위한 절차 등에 대한 정보를 인지하기 위하여 논리도면 및 경보대응절차서를 찾게 되는데 비상 시에 수백 페이지에 이르는 논리도면이나 경보대응절차서에서 해당 경보원인을 찾는 것은 매우 어려운 일이라 할 수 있다. 이 연구는 이와 같은 정보들을 제공하는 월성원자력 발전소용 경보원인추적시스템(LogACTs: Logic Alarm Cause Tracking System)[9]의 일부로서 원인과 경보간의 논리적 관계 정보를 전산화된 논리도면에 표시해 주는 방법에 관한 연구이다.

경보논리도면을 출력하기위해서는 이를 위한 전산화가 필수적인데, 경보논리도면은 적어도 수백 페이지에서 많으면 수천 페이지에 이르는 방대한 양이며 이를 전산화하기위해서는 편집과 검증이 쉬운 도구가 요구된다. 특히 원자력 발전소에 있어서 안전을 위한 신뢰성은 거의 절대적이므로 논리의 오류를 막기 위한 검증은 필수적이라 할 수 있다. 논리도면을 전산화하고 이들 논리를 활성화 시키는 방법으로 문자형 논리도면을 이용한 방법이 이미 제안된 바 있다[10]. 문자형 논리도면은 상용 도구를 이용할 수 있으므로 편집이 자유롭고, 이를 객체화하거나, 또는 이를 메타 파일로 직접적인 객체 편집기를 설계할 수도 있다.

그러나 문자형 논리도면 또는 이를 객체화하여 논리도면으로 사용한 기존의 방법들[11-13]은 문자와 일대일 대응 또는 일부 문자열에 대해서 논리소자들이 객체로 변환되므로 논리소자간의 연결에 관련된 논리도면의 오류를 찾는 데에 어려운 점이 있다.

또한 기존의 방법들은 논리소자의 입력 값이 반드시 모두 주어져야만 논리연산이 가능하다. 그러나 월성원자력 발전소의 경우 입력 신호는 기존 발전소 제어용 전산기 DCC 시스템으로부터 제공되고 이 시스템의 기능상 약 5초 간격으로 제한된 일부의 논리 값들이, 실제 발생하는 시각이 서로 다름에도 불구하고 같은 시각에 동시에 제공된다. 경보 논리는 입력 값의 발생순서에 따라 논리가 달라질 수 있고 모든 입력 값이 정해져야만 가능하므로 월성원자력발전소의 제한적인 환경으로 경보의 논리상태를 모두 운전원에게 보여주는 것은 매우 어렵다.

위와 같은 문제점들을 고려하여 본 연구에서는 문자형 논리도면을 2차원 언어로 간주하고 이를 위한 문법을 제시한다. 즉, 논리도면은 제안된 문법에 맞게 작성되어야하며 문법에 맞지 않은 도면을 찾아 낼 수 있도록 함으로써 이를 도면의 검증에 이용한다.

그리고, 문법에 따라 컴파일된 논리도면 정보로부터 논리소자 객체를 생성하여 이들 논리소자 간의 논리적 연결정보를 구축하는 방법과, 객체화된 논리소자의 상태를 3개의 상태로 나누어서 월성원자력 발전소와 같이 논리상태가 충분히 주어지지 않는 경우에도 논리상태에 대한 연산이 가능하고 경보-원인 경로를 추적할 수 있으며 관련 논리상태를 보여줄 수 있는 방법을 제시한다.

본 방법은 월성원자력발전소 경보논리 약 400페이지에 대한 시험을 마치고 운용 중에 있다.

II. 경보-원인경로추적 시스템

2.1 경보원인추적시스템의 구성

경보-원인경로추적시스템(FACTS : Fast Alarm-Cause path Tracking System)은 경보원인추적시스템(LogACTs)의 일부이다. LogACTs는 월성원자력발전소 3, 4호기의 경보 발생 시 원인을 논리적으로 추적하여 운전원이 상황을 파악하고 신속히 대응할 수 있도록 중요 원인 경보와 부수적인 경보를 구분하여 부수적인 경보 표시를 일시적으로 억제하거나, 제거하여 출력감발의 중요 원인을 신속히 추적할 수 있도록 지원하는 시스템이다.

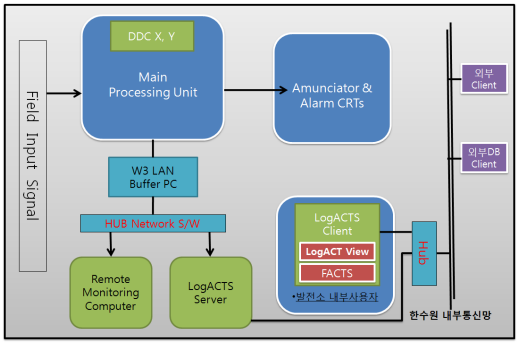


그림 1. 경보원인추적시스템의 시스템 구성도

LogACTs([그림 1]참조)은 크게 Server Module과 Client Module로 구성된다. LogACTs의 Server Module에서는 기존 발전소 제어용 전산기 DCC로부터 경보 및 상태 정보를 받아, 발전소의 상태에 따라 일시적이며 종속적으로 발생하는 경보를 억제하거나 경보들의 원인을 추적한다.

LogACTs의 Client Module은 LogACTs View부분과 FACTS로 구성된다. LogACTs View는 경보가 발생하면 해당되는 원인을 추적하고 경보에 관련되는 비상 또는 비정상 진입조건 및 해당 운전절차서 정보를 찾아 화면에 표시한다. 또한 경보 발생 중 자연적으로 수반되는 다른 경보들 중에서 운전원에게 필요한 중요 정보만 압축해서 보여주는 기능을 가지고 있다.

경보발생 시 운전원은 기기나 센서의 고장이 경보에 미치는 논리적 관계를 파악하고, 이의 영향을 완화시키거나 기능을 대신할 수 있는 방법을 찾기 위해 경보논리도면을 참조하게 된다. FACTS는 경보발생시 LogACTs View로부터 주어지는 경보와 그 원인사이의 논리경로를 추적하여 찾은 논리경로와 경로 상에 존재하는 논리도면과 함께 논리소자들의 상태를 화면에 표시한다.

2.2 경보-원인경로추적시스템

LogACTs에 의해 경보와 그 원인들의 조합이 만들어지고, 운전원들은 이들 원인과 결과 사이의 논리적인 관계를 파악하기 위해 논리도면을 참조하게 된다. 이때 논리도면으로부터 운전원에게 필요한 정보는 다음과 같이 요약할 수 있다.

(1) 경보원인과 경보 사이의 논리적인 관계(경보 경로) 및 논리도면

(2) 해당 경보관련 논리의 상태

위의 정보 (1)을 위해서는 경보 논리를 추적하여 원인에 이르는 경로를 찾아내고, 그 경로를 논리도면상에 표시하는 기능이 제공되어야 한다. 그리고 (2)를 위해서 경보결과 또는 경보원인을 포함하는 논리도면과 경보와 원인들 사이에 있는 논리소자들의 상태를 화면에 표시해야한다.

위의 정보들을 위하여 논리도면을 화면에 표시하고 그 상태를 연산하기 위해서는 논리도면을 전산화하고 도면 내의 논리들이 동작시킬 수 있도록 하는 작업이 필수적이다. 그림 2는 논리도면을 전산화하고 앞서 언급한 정보들을 제공하는 FACTS의 구성이다.

2.3 논리도면의 표현과 객체화

2.3.1 논리도면 구성

원자력발전소에서 사용되는 경보논리는 여러 페이지의 경보논리도면(Diagram)으로 구성되어 있고, 각 논리도면에는 여러 개의 논리회로(Circuit)가 있을 수 있다.

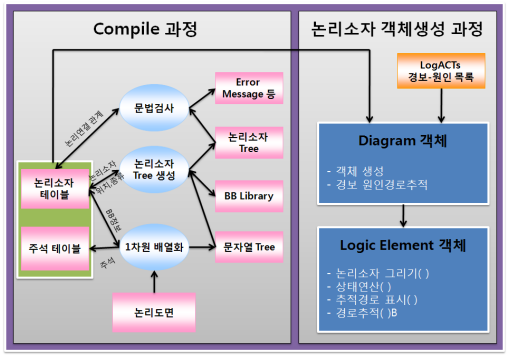


그림 2. 경보원인경로추적시스템(FACTS) 구성도

그리고 각각의 논리회로는 논리의 제어와 흐름에 관련되는 논리소자들이 방향성 그래프 형태로 연결되어 있다.

논리소자(Logic Element)는 크게 논리연산형 논리소자와 신호전달용 논리소자들로 나눌 수 있다. 논리연산

형 논리소자는 AND, OR, NOT, FLIPFLOP 등 Gate 종류와 Black box(이후 BB라 함) 형태의 논리모음용 논리소자가 있으며, 신호전달용 논리소자(Connector, 이후 CONN 이라함)는 도면 연결자(Drawing CONN : 도면들 사이 또는 도면 내부의 논리신호를 전달하는 논리소자, 이후 DWG_CONN이라함), 발전소 공정신호 전달 논리소자(External CONN : 외부 시스템으로부터 전달되는 신호를 주고받는 논리소자, 이후 EXT_CONN이라 함)와 도면 내에 논리소자들 사이에 신호를 전달하는 신호선(Line : 실선으로 표시) 등으로 구분된다([표 1] 참조). 그리고, 실제 논리도면 상에는 존재하지 않지만 BB 외부 또는 내부로 신호를 전달하는 Box CONN(이후 BOX_CONN이라 함)이 별도로 존재한다.

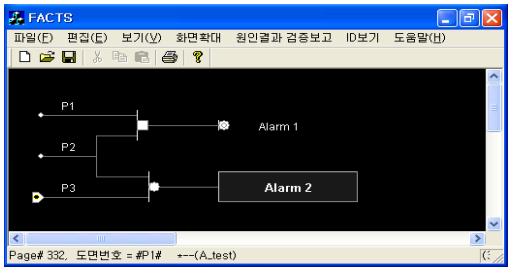


그림 3. 논리도면 표현의 예

표 1. Graphic and Text Symbols for Logic Elements

Logic Elements		Graphic Symbols	Text Symbols
Gate	AND		1〉
	OR		2〉
	NOT		3〉
	RS FLIP-FLOP		4〉
	DELAY		5〉
	LAMP		7〉
	2/3 LOGIC		8〉
	LIGHT		9〉
	COINCIDENCE MATRIX		A〉
	...	...	...
DWG_CONN			[ ]
EXT_CONN		•	*
BOX_CONN			( )
BB			{@ $}
Line			-, |, +

원자력발전소 경보논리도면에 사용되는 논리소자의 종류는 [표 1]에서와 같이 매우 제한적이고 규격화 되어 있어 대부분 문자로 대치하여 표현하여도 도면을 인지하는 것이 가능하다. 이와 같이 논리도면이 문자열로 표현 되면 이들에 대한 작성 규칙(문법)을 만들고 이를 이용하여 논리도면을 검증하는 데에 사용할 수 있다.

2.3.2 논리도면의 1차원 배열화

문자형 논리도면은 [표 1]에서 보여준 논리소자와 이들간의 연결 및 위치 정보를 가진 2차원 논리도면이다. 따라서 2차원 문자배열에 대한 문법적용은 어려우므로 문자형 논리도면을 1차원 문자열로 변환시키는 과정이 필요하다. 2차원 문자배열을 1차원 문자배열로 만들기 위해 문자형 논리도면을 다음과 같은 과정을 통해 1차원 문자열 트리를 만든다.

(1) 주석문 처리 : raster-scan 방향으로 #와 # 사이의 모든 문자열은 좌표와 함께 별도의 주석목록(Comment List)에 보관하고 문법처리를 위한 논리소자 대상에서 제외한다(도면에서 공백처리).

(2) BB 처리 : 도면 내에 '{@'를 직사각형 왼쪽 위의 모서리로 '$}'를 오른쪽 아래 모서리로 하는 모든 직사각형들 내의 문자열(시작점 및 끝점 좌표 포함)을 LET(Logic Element Table, 2.3.3절 [표 2] 참조)에 저장한다.

(3) 문자열 트리 생성 : 문자열 트리의 각 노드는 문자열을 나타낸다. 도면의 좌측 상단부터 시작하여 우측 하단방향으로 읽으면서 문자 중 '*', '[', '(' 또는 '{'를 만나면 트리를 이루는 최초 문자열 노드의 첫 번째 문자를 구성한다. 그리고 개행문자를 만날 때까지 다음과 같이 트리를 만든다.

(3-1) 문자열은 도면 내에서 '+'를 만날 때까지 다음과 같이 문자열 노드를 생성한다.

(가) 문자열은 (나)의 경우를 제외하고 도면의 좌측에서 우측으로 연결되는 공백이 없는 문자열을 노드로 간주한다. 단, '['와 ']', '('와 ')' 및 '{'와 '}'사이의 문자열 내의 공백문자는 무시한다.

(나) '!'를 만나면 우측에 다른 문자가 나타날 때까지 위 또는 아래에 있는 '!'를 찾은 다음 (가)를 반복한다.

(3-2) '+'만나면 '+'와 연결된 문자열의 각 방향으로 자식 노드들을 만든 다음 (3-1)을 반복한다. 이 때 자식 노드의 첫 문자는 '+'의 위, 오른쪽 및 아래 쪽 세 방향으로 존재할 수 있다.

(3-3) 개행문자를 만나면 노드(문자열)의 시작 및 끝 좌표 등을 문자열 트리에 저장한 다음 (3)을 반복한다.

단계 (3-1)의 (나)와 (3-2)의 경우 문자열이 좌우로 연결되는 것이 아니라 상하로 연결될 수도 있으므로, 이때 각 노드 내의 문자열 사이 연결 관계는 기호 ↔ 와 ↕ 로 표현하고, ↔의 경우 생략될 수 있다.

A↔B : 문자열 A의 끝과 문자열 B의 시작이 수평방향으로 연결

A↕B : 문자열 A의 끝에 문자열 B의 시작이 위 또는 아래방향으로 연결

[그림 4](a)는 문자형 논리도면의 예이고, [그림 5]는 이에 해당하는 문자열 트리의 예를 보여준다.

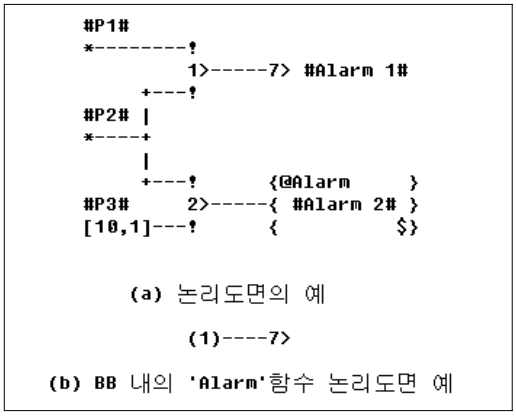


그림 4. 그림 3에 대한 문자형 논리도면의 예

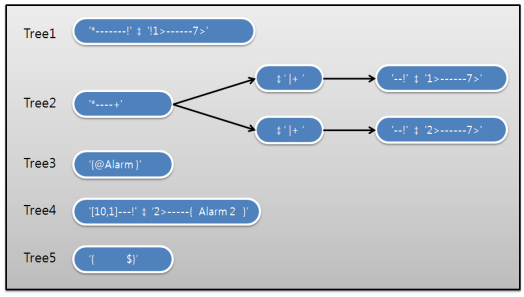


그림 5. 문자열 트리

2.3.3 논리소자 트리생성

문자열 트리의 각 노드는 논리소자들의 문자열로 이루어진다. 앞 절의 [표 1]에서처럼 논리도면은 논리소자들(Line, Gate, BB, DWG_CONN, EXT_CONN 등)의 조합으로 이루어진다. 따라서 각각의 문자열 노드는 문법적용을 위하여 논리도면의 최소 의미 단위인 논리소자들에 대한 리스트(트리)로 재구성이 필요하다. 각각의 논리소자는 다음과 같은 문자열 배열규칙을 가지므로 이를 이용하여 논리소자를 찾는다.

(1) Gate형 논리소자는 입력부분을 나타내는 '!'와 게이트 종류를 표시하는 숫자 또는 문자([표 1] 참조)가 수직방향으로 존재하고, 종류표시 문자의 왼쪽에 출력부를 나타내는 '>'로 구성된다. 즉 문자열 트리의 노드 문자열은 '!' ↕ '!' ↕...↕ '!' ↕ 'N>'의 형태로 표현된다(여기서 'N'은 Gate 종류를 나타내는 문자).

(2) BB는 '{@'와 '$}'사이에 존재하는 box영역 내의 문자열 이다([그림 4](a) 참조). 이때 '[@'다음의 문자열([그림 4](a)의 예에서 'Alarm')은 BB 함수명으로서 BB Library에 저장된 LET 형태의 함수를 의미한다.

(3) DWG_CONN과 BOX_CONN은 각각 '['와 ']' 사이 및 '['와 ']' 사이의 문자열로 이루어진다.

(4) EXT_CONN는 '*' 하나의 문자로 표현된다.

(5) Line은 '+', '-' 및 '|'들로 이루어진다. 다만 수직으로 연결되는 Line의 경우 '-'를 포함할 수 없다. 수평으로 연결되는 경우 [그림 4]와 같이 논리적으로 연결되지 않은 두신호선이 교차하는 경우

([그림 4](a)의 예)는 '-'들 사이에 '|'이 존재한다.

위 (3)의 DWG_CONN의 경우, 괄호 안에 상대 DWG_CONN의 페이지 식별자(같은 페이지일 경우 생략될 수 있음)와 일련번호가 주어지므로 다른 도면과의 논리적인 연결이 가능하다. 그리고 BOX_CONN의 경우 소괄호안의 숫자는 입출력 신호의 순서를 나타내며 '(1)--'은 좌측 최상단으로부터 첫 번째 입력신호를 의미한다([그림 4](b) 참조).

이와 같이 문자열 트리내의 노드가 논리소자로 대치되면 [그림 6]과 같이 각각의 논리소자를 한 개의 노드로 하는 논리소자 트리로 재구성된다.

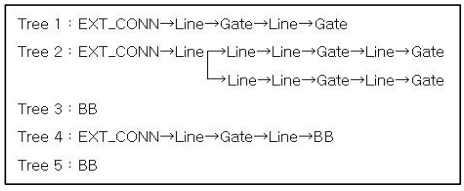


그림 6. 그림 5에 대한 논리소자 트리

표 2. 그림 5(a)에 대한 LET

일련번호	논리소자종류	함수종류	페이지	위치	입력소자번호	출력소자번호
1	BB	Alarm	1	(16,7)(28,9)		
2	EXT_CONN		1	(0,1)		3
3	Line		1	(1,1)(8,1)	2	4
4	Gate	1	1	(9,2)(1,3)	3,10	5
5	Line		1	(11,2)(15,2)	4	6
6	Gate	7	1	(16,2)(2,2)	5	
7	EXT_CONN		1	(0,5)		8
8	Line		1	(1,5)(5,5)	7	9,11
9	Line		1	(5,4)(5,3)	8	10
10	Line		1	(6,3)(8,3)	9	4
11	Line		1	(5,6)(5,7)	8	12
12	Line		1	(6,7)(8,7)	11	13
13	Gate	2	1	(9,8)(7,9)	12,16	14
14	Line		1	(11,8)(15,8)	13	17
15	DWG_CONN	10,1	1	(0,9)(5,9)		16
16	Line		1	(6,9)(8,9)	15	13
17	BOX_CONN	1	2	(0,0)(2,0)	14	18
18	Line		2	(3,0)(6,0)	17	19
19	Gate	7	2	(7,0)(0,0)	18	

논리소자 트리가 완성되면 LET에서 일련번호, 논리소자종류, 세부분류, 페이지 및 위치 항목이 채워진다. [표 2]는 LET를 나타내며 위치항목에서 Gate의 경우 첫 번째 항은 Gate의 종류를 나타내는 문자가 있는 논리소자의 위치이며, 두 번째 항목은 입력부분 y좌표의 시작과 끝을 나타낸다.

2.3.4 논리소자 문법

앞 절에서와 같이 논리소자 트리가 만들어지면 이들 논리소자의 연결 적합성을 검사하기 위해 다음과 같은 문법을 사용한다.

G={N,T,P,S}
N={Diagram, Circuit, InputCircuit, Processor, CONN}
T={Line, Gate, BB, DWG_CONN, EXT_CONN, BOX_CONN}
S : Diagram
P : Diagram ::= Circuit ◎ Circuit
Circuit ::= Input_Circuit↔Processor
| Input_Circuit↔CONN
Input_Circuit ::= CONN↔Line
| Input_Circuit ↔Processor↔Line
Processor ::= Gate | BB
CONN ::= DWG_CONN | EXT_CONN | BOX_CONN

여기서, ◎는 같은 도면(페이지)에 존재함을 나타내는 부호이다.

문법검사는 논리소자 트리의 root로부터 leaf에 이르는 각각 경로 상의 논리소자들에 대해서 수행하게 된다. 이와 같이 문법검사가 이루어지면 논리소자들 사이의 입출력 관계가 결정되므로 LET의 입력소자, 출력소자번호에 대한 목록이 완성된다.

LET는 논리소자의 종류, 위치, 입출력 연결관계 등 논리소자 객체의 모든 정보를 가지고 있으므로 객체 생성시 이용될 뿐만 아니라, 논리도면이나, BB Library 함수들의 논리를 저장하기 위한 정보로도 사용된다.

2.3.5 논리소자의 객체화

[표 2]와 같이 LET가 완성되면 문자형 논리도면에 대한 컴파일이 완료된다. LET를 이용하여 논리도면을 출력하고 그 논리 상태를 활성화하기 위해 크게 Diagram 객체와 논리소자 객체 등 두 종류의 객체가 필요하다.

Diagram 객체는 LET를 이용하여 논리소자 객체([그림 7]참조)를 생성하거나, LogACTs로부터 경보-

원인 경로추적 메시지를 처리한다. Diagram객체는 경로추적메시지 내의 경보-원인 쌍에 해당되는 논리소자가 있는지를 조사하고, 해당되는 논리가 있는 경우 관련 논리소자에게 경보-원인경로 추적 메시지를 보내는 역할을 한다.

한편, 논리소자들의 중요 기능은 경보-원인경로 추적과 논리 상태를 연산하여 화면에 그 결과를 보여주는 것이다.

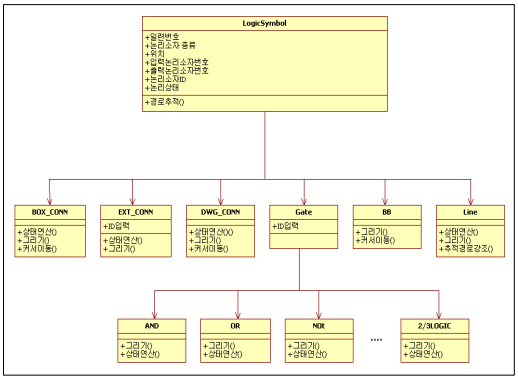


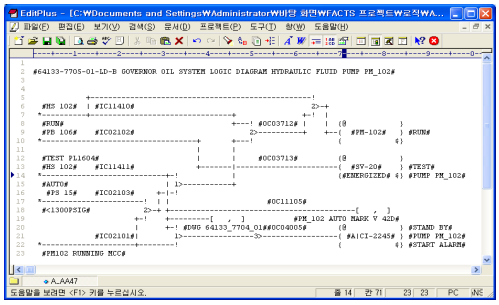
그림 7. 논리소자 객체 계층구조도

[그림 7]에서 커서이동() 함수는 논리도면의 연결 관계를 파악할 수 있도록 DWG_CONN와 BOX_CONN의 객체를 클릭하는 경우 상대 CONN이 있는 위치로 커서를 이동하며, BB를 클릭하는 경우 내부 논리도면을 출력한다.

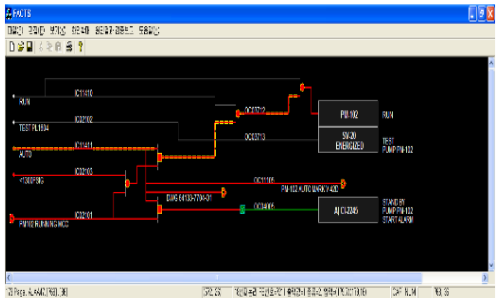
경보-원인경로 추적시 LogACTs로부터 주어지는 ID는 EXT_CONN과 Gate에만 존재하며 객체를 클릭하면 ID입력() 함수를 통해 ID를 입력하거나 수정할 수 있으며 이들 ID는 ID 파일에 저장된다. 이때 사용되는 ID는 발전소 현장에서 사용되는 고유 ID를 주로 사용한다.

그리고 Line객체의 경우 추적경로를 강조하여 표시하는 기능이 있다([그림 8](b) 참조).

[그림 8]은 문자형 논리도면으로부터 생성된 논리소자의 동작 예를 보여준다.



(a) 문자형 논리도면



(b) 객체화된 논리도면

그림 8. 월성 원전용 논리도면의 예

2.4 논리연산

이 연구에서 사용되는 논리연산은 가급적 많은 논리상태를 얻기 위해서 충분한 입력이 주어지지 않더라도 얻을 수 있는 필연적인 논리 상태를 구하는 ND 연산과, 경보원인 후보의 상태와 경보원인경로를 찾는 데에 필요한 DC 연산을 사용한다.

2.4.1 ND 연산

다음 절에서 설명할 논리추적이 완료되면 경보와 경보원인 사이의 경로 상에 존재하는 모든 논리소자는 주어진 경보와 경보원인의 상태 값에 의해서 필연적으로 정해지는 상태 값을 갖게 된다. 그러나 경로 상에 존재하지 않지만 서로 연결되어 있는 논리소자들의 논리상태는 입력 값이 충분히 주어지지 않으므로 일반적인 논리연산으로 결과를 얻을 수 없다.

발전소 운전원의 입장에서는 경로관련 논리소자의 상태정보를 많이 알수록 경보원인에 대한 논리적 이해가 보다 쉬워지므로 입력 값이 모두 주어지지 않더라도

논리소자의 상태가 필연적이면 그 상태를 계산해주는 것이 도움이 된다. 예를 들면, AND 연산에 오로지 입력 값 하나만 0(false)으로 주어지면 다른 입력 값에 관계없이 그 결과는 0으로 되는 것이 필연적이다. 이와 반대로 어떤 AND 논리소자의 출력 값이 1로 미리 주어져 있다면 그 입력 값은 모두 1이 될 수밖에 없다. 이와 같이 필연적인 귀결을 통해 입력 값 또는 출력 값을 추출해 낼 수 있다.

이와 같이 경보논리소자 중에서 두 개 이상의 입력을 필요로 하는 소자는 AND, OR 및 Flip-Flop 등은 부족한 입력 값에 대해서도 필연적인 결과를 가질 수 있지만, 입력 값이 불충분하여 연산결과를 결정할 수 없는 경우도 생긴다. ND 연산은 상태를 결정할 수 없는 경우를 처리할 수 있도록 논리소자는 1, 0, ND(Not Determined) 3가지 상태를 가진다.

2.4.2 DC 연산

경보와 원인사이의 경로에 있는 논리소자들은 LogACTs로부터 주어지는 논리 상태에 의해 결정된다.

두 개의 논리소자 A, B가 경보원인 경로 상에 존재하고 A가 B의 입력 소자라 하고, B의 출력 값 1, B가 OR Gate라 하면 A의 출력 값이 B 출력 값의 원인이 되기 위해서는 다른 입력 소자의 상태에 관계없이(DC : Don't Care) A는 1이 될 수밖에 없다.

이와 같이 어떤 논리소자가 경로의 일부가 되기 위해 필연적으로 가져야하는 논리상태를 연산하는 것을 DC 연산이라 한다. DC 연산은 각각의 논리소자에 대해 정해진 함수를 사용한다. DC 연산(함수명 : DC_OP(A,A_S, B) : 논리소자 A의 출력 A_S에 대해 A의 입력 논리소자 B의 상태를 DC 연산) 결과가 주어진 원인 상태와 상충되는 경로가 발생하면 그 경로는 경보원인 경로후보에서 제외된다(2.5절 참조).

2.5 경보원인경로추적

경보원인과 결과 사이의 경로추적은 객체화된 논리소자들의 입출력 논리소자들 간의 논리연결 정보를 이용하여 경로를 추적한다. 경로 추적을 위한 자료흐름은 다음과 같다.

(1) LogACTs는 경보원인 추적결과를 원인-결과 논리소자 ID와 그 상태(1 or 0)를 보내 FACTS의 Diagram 객체에게 경보원인경로추적을 요구한다.

(2) Diagram 객체는 논리소자 ID를 확인하여 존재와 위치를 파악하고, 경보와 그 원인의 쌍을 만든 다음 각 쌍에 대해서 원인경로추적 메시지를 보낸다.

경보논리에서 같은 신호들이 여러 논리도면에 사용될 수 있으므로 하나의 경보와 원인에 대해서 여러 개의 쌍(ID는 같고 논리도면 번호나 일련번호가 다른 쌍)으로 주어질 수 있다. 이 모든 쌍에 대해서 FACTS는 경로를 찾고 DC 연산을 이용하여 상태가 일치하는 경로를 찾는다. 다음은 각각의 객체 내에서 수행하는 경보원인경로추적(ACPT) 알고리즘이다.

```
bool ACPT(C, C_S, N_S) {
  // C : Cause Node
  // C_S : the state of C
  // N_S : the state of the current node N
  // FoundPath : Found path or not
  // Path_List : Alarm-Cause Path List
  // N: Current logic element itself
  if( N==C) {
    if(N_S==C_S) FoundPath=true;
    else FoundPath=false;
  }
  else { // N!=C
    FoundPath=false;
    for {i=0; i<n; i++) { //좌측노드 Li(i=1,2,...n)
      if(ACPT(C, C_S, DC_OP(N, N_S, Li))){
        FoundPath=true;
        store the pair (Li, N) into the Path_List;
      }
    }
  }
  return FoundPath;
}
```

[그림 8] (b)에서 굵은 점선은 추출된 경보-원인 경로

를 보여주고 있다. 그리고 나머지 논리소자는 red, green, white 등 3가지 색으로 논리소자의 3 state를 보여준다.

추출된 경로 외의 red 또는 green으로 표시된 논리소자는 ND 연산에 의해 추출된 상태를 가지는 논리소자들 이다.

Ⅲ. 적용

FACTS는 월성 원자력발전소 3, 4호기에 대해서

- 경보 신호 수 : 2000개
- 논리도면
  - NOP(정상운전) 경보관련 논리도면 : 367페이지
  - EOP(비상운전), AOP(비정상)용 경보관련 논리도면 : 40페이지
- 논리도면 내의 총 논리소자 수 : 약 39,000

논리 도면 및 논리소자를 생성하여 수행하였다.

[그림 9]은 경보발생 시 LogACTs([그림 9]의 흰색 바탕화면)에 의해 경보원인이 감지되어 그 결과를 출력하는 화면을 보여주고 있다. 이때 운전원은 경보발생시 Logic Diagram 박스를 클릭하여 FACTS를 호출할 수 있다. FACTS 호출 시, LogACTs는 좌측의 경보와 그 원인들의 정보를 FACTS에 제공하고 FACTS는 경보 관련 논리도면 상에 경보-원인간의 논리 경로가 출력된다(그림 9의 우측 검은 바탕 화면).

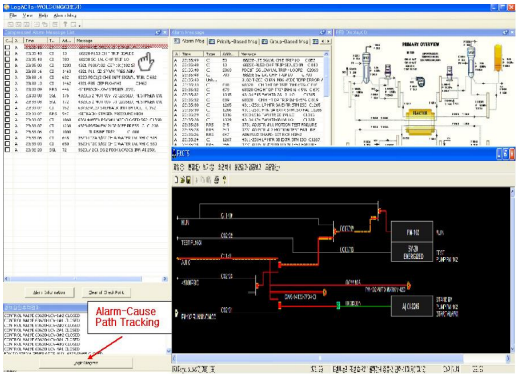


그림 9. LogACTs 출력감발 원인추적 결과

Ⅳ. 결론

경보발생 시 운전원의 신속한 조치를 위해 경보관련 논리도면을 논리 상태와 함께 신속하게 제공하는 것은 매우 중요한 일이다. 이러한 기능을 수행하기 위해 논리도면의 전산화와 경보원인경로추적 상태를 보여주는 작업이 필요하다.

이 연구에서는 논리도면 작성을 위해 2차원 배열 문자형 논리도면을 도입하고 이에 대한 문법을 제안하였다. 이러한 문법은 도면작성 시 오류를 줄여 줌으로써 논리도면에 대한 검증에 도움이 되도록 하였다.

한편 경보원인경로추적 시 경로 상에 존재하는 논리소자의 상태와 관련 논리소자의 상태를 보여주기 위해 DC 및 NC 연산을 이용한 경보원인경로 추적 방법을 제안하였다. ND 연산은 충분하지 않은 논리상태 값을 3가지 상태로 표현하여 필연적인 논리 상태를 찾아냄으로 해서 운전원에게 보다 많은 경보관련 논리 상태를 제공하였다.

현재 이 시스템은 월성 원자력발전소 3, 4호기에서 사용 중에 있으며, 사용자의 편의를 위하여 문자형 논리도면을 기반으로 그래픽 논리도면 편집기를 개발 중이다.

이 시스템은 경보원인경로 추적뿐만 아니라 논리설계, 논리기능 확인 및 논리도면관리 등 여러 분야에 적용해서 사용가능할 것으로 생각된다.

참 고 문 헌

[1] I. S. Kim, "Computerized Systems for On_Line Management of Failures: A State-of-Art Discussion of Alarm Systems and Diagnostic Systems Applied in the Nuclear Industry", *Reliability Engineering and System Safety*, 1994.

[2] I. K. Hwang, "An Object-Oriented implementation to improve Annunciation", *IAEA Specialists Meeting on EXperience and*

Improvements in Advanced Alarm Annunciation Systems in Nuclear Power Plants, 1996.

[3] K. Zhao, B. R. Upadhyaya, "Adaptive fuzzy inference causal graph approach to fault detection and isolation of field devices in nuclear power plants", *Progress in Nuclear Energy*, Vol.46, No.3-4, pp.226-240, 2005.

[4] S. J. Lee and P. H. Seong, "A Dynamic Neural Network Based Accident Diagnosis Advisory System For Nuclear Power Plants", *Progress in Nuclear Energy*, Vol.46, No.3-4, pp.268-281, 2005.

[5] E. Zio and G. Gola, "Neuro-fuzzy pattern classification for fault diagnosis in nuclear components", *Annals of Nuclear Energy,* 33, pp.415-426, 2006.

[6] K. Mo, S. J. Lee, P and H. Seong, "A Dynamic Neural Network Aggregation Model for Transient Diagnosis in Nuclear Power Plants", *Progress in Nuclear Energy, 49.* pp.262-272, 2007.

[7] E. Zio, "A Fuzzy Decision Tree Method for Fault Classification in the Steam Generator of a Pressurized Water Reactor", *Annals of Nuclear Energy,* 36 (2009), pp.1159-1169, 2009.

[8] A. M. Aboshosha, "Neurofuzzy Computing aided Fault Diagnosis of Nuclear Power Reactors", *Proceedings of the 7th ICEENG Conference*, 25-27 May, 2010.

[9] J. W. Lee, "LogACTs(Logic Alarm Cause Tracking System) for Nuclear Power Plant Operation", *Sixth American Nuclear Society International Topical meeting on NPIC&HMIT*, 2009.

[10] S. P. Lyu, "A Study of Logic Display to Monitor the States of Logic Elements", *American Nuclear Society International Topical on NPIC&HMIT*, 1996.

[11] K. C. Kwon, "The Real-time Function test facility for advanced instrumentation and control in nuclear power plant," *IEEE Trans. on Nuclear Science,* Vol.48, No.2, 1999.

[12] S. P. Lyu, "An Identification of Alarm Cause by Tracking Logic Diagram", *2002 Spring Conference, Korea Nuclear Society*, 2002.

[13] J. T. Kim, "An Application on Alarm Root Cause Tracking System(ACTs)" *Joint 8th Annual IEEE Conference on HFPP and 13th Annual Workshop on HPRCT*, 2007.

저 자 소 개

류 승 필(Sung-Pil Lyu) 정회원

- 1979년 2월 : 서울대학교 전자공학과 학사
- 1980년 ~ 1993년 : 한국원자력연구소 계측제어연구실 선임 연구원
- 1987년 2월 : 충남대학교 전자공학과(석사)
- 1991년 8월 : 충남대학교 전자공학과(박사)
- 1993년 ~ 현재 : 세명대학교 컴퓨터학부 교수

<관심분야> : 시뮬레이션, 패턴인식, 인공지능

김 상 훈(Sang-Hoon Kim) 정회원

- 1989년 2월 : 동국대학교 컴퓨터공학과(석사)
- 1996년 8월 : 동국대학교 컴퓨터공학과(박사)
- 1997년 ~ 현재 : 세명대학교 컴퓨터학부 부교수

<관심분야> : 프로그래밍언어, 소프트웨어공학

김 은 주(Eun-Ju Kim) 정회원

- 1997년 2월 : 세명대학교 전자계산학과(학사)
- 1999년 8월 : 세명대학교 전자계산교육(석사)
- 2009년 8월 : 세명대학교 전산정보학(박사)

<관심분야> : 시뮬레이션, 전산교육, 인공지능

김 정 택(Jung-Taek Kim) 정회원

- 1984년 2월 : 한양대학교 원자력공학과(학사)
- 1986년 2월 : 한양대학교 원자력공학과(석사)
- 1986년 ~ 현재 : 한국원자력연구원 계측제어인간공학연구부 책임 연구원

<관심분야> : 경보처리, 고장진단, 인공지능