공정의 안전 검증을 위한 PLC 모듈 개발

Development of PLC Modules for the Safety Verification of Chemical Processes

정 상 헌, 이 광 순, 문 일
(Sang Hun Jeong, Kwang Soon Lee and Il Moon)

Abstract : An automatic verification method has been studied to determine the safety and operability of programmable logic controller (PLC) based systems. For the systematic and efficient verification, we have developed a conversion method from relay ladder logic (RLL) to the verification system description. RLL is a common representation used to document PLC programs for the sequential logic of the system such as the safety interlocks and the startup/shutdown procedures. Once the modules are developed, complex RLLs can be represented by the combination of modules. As a result we can verify complex PLC systems using the verification method including RLL modules. The developed modules are used to verify alarm systems and show that the method is valid.

Keywords: verification, safety, PLC, RLL, SMV

I. 공정 안전 검증의 자동화

화학공장에서 유독성 물질이나 위험 물질을 다루는 공정들이 점차로 자동화됨에 따라 화학공정 시스템의 안전성이 더욱 중요한 문제로 대두되고 있다. 최근 들어 화학공정 시스템이 점점 복잡해짐에 따라 이들의 안전성을 완벽하게 검증하는 것은 더욱 어려워지고 있다. 따라서 공정의 설계와 운전 절차 등의 공정 정보로부터 공정의 운전이 안전하고 오류가 없다는 것을 입증해 줄 수 있는 자동 안전 검사 방법에 대한 연구가 필요하다. 본 연구에서는 공정의 안전성을 체계적이고 효율적으로 분석하고 검증하기 위하여, PLC를 사용한 공정에 대한 자동 논리 검증 방법을 개발하였다. 특히 논리 검증을 위한 공정의 입력 모델을 자동으로 생성할 수 있도록, 논리 검증기용 PLC 모듈들을 개발하였다.

PLC는 공정의 자동화에 널리 쓰이는 장치로 주로 공정의 안전 잠금장치(interlock)나 조업 시작과 조업 정지와 같은 순차적 논리, 또한 로컬 제어 등을 담당한다. 일반적으로 PLC에서 가장 많이 사용되고 있는 프로그래밍 언어는 RLL이다. RLL로 작성된 프로그램에 오류가 있으면 공정의 안전과 운전성에 큰 위험이 있을 수 있으므로, RLL이 제대로 프로그램 되었는지를 반드시 검증해 보아야 한다. 전통적으로 RLL에 대한 검증은 전문 설계가에 의해 수동으로 진행된다. 다른 방법으로는 PLC의 동작을 모사하는 모사기가 사용되기도 한다. 이 경우는 입력 변수가 증가하면 검사하여야 할 경우의 수가 급격하게 증가하기 때문에 가능한 모든 경우에 대하여 검사하는 것이 불가능하다. 논리 검증법은 체계적이고 자동화된 검사 방법이면서 현재로는 10^{120}의 상태까지 검사가 가능하여 기존 방법의 문제점을 해결한 검사 방법이다.

1986년에 Clarke 등[1]이 모델 검증 방법(model checking verification method)을 개발하였고, 통신 프로토콜과 VLSI 회로 설계의 검사에 이 방법을 사용하였다. 1992년에 McMillan 등[2]이 상태 공간을 기호를 이용하여 생성하고 검색 과정에서 필요한 상태만을 생성하는 방법을 연구하여, 검사시 상태의 수가 급격하게 늘어나는 것을 줄였다. 이들은 모델 검증 방법을 개선하여 부호 모델 검증법(symbolic model verification, SMV)을 개발하였다. Moon[3]은 이 부호 모델 검증법을 PLC 기반의 공정을 검사하는데 적용하였다. RLL의 기본 명령어들을 논리 검증기용 입력 언어로 변환하는 방법을 제시하고, 이를 이용하여 공정의 안전을 검증할 수 있음을 보였다. 이 연구에서는 부울(boolean) 논리를 논리 검증기용 입력 언어로 표현하는 방법을 보였으나 타이머나 카운터 등과 같은 RLL의 복합 명령어에 대해서는 변환 방식을 제시하지 않고 있다. 본 연구에서는 공정의 안전과 밀접한 관계에 있는 PLC의 RLL 동작을 자동으로 검증할 수 있도록, RLL에 대한 논리 검증기용 입력 모델을 일대일로 생성시켜 생성할 수 있는 방법을 개발하였다. 여러 기본 명령어들을 조합한 것과 같은 기능을 하는 RLL의 복합 명령어에 대한 논리 검증기용 모듈을 개발함으로써, 이를 이용하여 공정에 대한 입력 모델의 생성을 자동화 할 수 있는 길을 열었다. 아직까지는 모듈 개발의 한계로 인하여 완벽한 일대일 변환은 구현하지 못하였지만, 자동번역이 가능하도록 연구를 진행하고 있다.

II. 부호 모델 검증법

실제 시스템에 대한 모델을 준비하고 시스템의 중요한 성질을 기술함으로써, 논리 검증법은 다른 검사법으로는 찾아낼 수 없었던 시스템의 동작을 드러낼 수 있다. 논리 검증법의 목표는 시스템 안에 존재하는 오류를 가능한 한 완벽하게 그리고 효율적으로 찾아내는 것이다. 이를 통해 시스템이 완전하도록 설계를 변경하거나 운전 절차를 수정할 수 있다.

자동화된 화학공정 시스템의 안전성과 운전성을 검사하는 방법 중의 하나는 입력 가능한 공정 변수들의 조합에 대한 공정 출력 결과를 모사해 보는 것이다. 하지만, 공정의 규모가 보통인 경우에도 가능한 입력 변수 조합에 대하여 모두 검사하기 위해서는 엄청나게 많은 수의 모사가 요구된다. 이진 입력 변수의 수를 n이라고 했을 때, 검사 대상의 수는 2^n으로 늘어나게 된다. 따라서 공정의 가능한 상태를 모두 생성하여 검사하는 것은 유용한 방법이라고 할 수가 없다. 따라서 직접적인 표현 형식보다는 기호들을 사용하여 시스템을 표현하고, 검사하는 과정에서 필요한 상태들만을 생성시키는 방법을 도입하면, 규모가 큰 시스템을 다룰 때 요구되는 메모리 양을 줄일 수 있고 전체 모델 검사의 효율성을 높일 수 있다.

McMillan(1992)은 이전의 모델 검사 방법들보다 효율성을 높일 수 있는 논리 검증법을 제시하였고, 이 검증법을

접수일자 : 1995. 10. 26., 수정완료 : 1996. 2. 16.
정상헌 : 포항공과대학교 공정산업의 지능자동화연구센터
이광순 : 서강대학교 화학공학과
문 일 : 연세대학교 화학공학과
※ 본 연구를 지원하여 주신 연세대학교에 감사드립니다.

구현한 부호 모델 검증기(Symbolic Model Verifier, SMV)를 개발하였다[2]. SMV는 그림 1에서와 같이 '시스템 모델'과 '검증 질의어', '모델 검사기'의 세 구성 요소로 되어 있다.

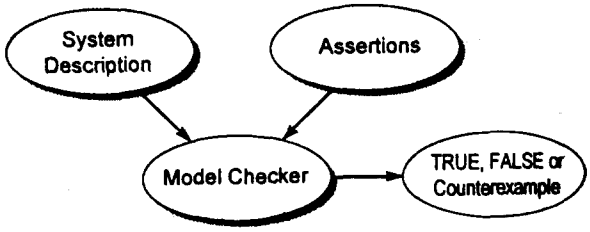

그림 1. 논리 검증법의 개요도[2,4].
Fig. 1. Overview of the automatic verification[2,4].

1. 시스템 모델

시스템 모델은 검사의 대상이 되는 시스템에 대해서 기술한 것으로, 이는 다시 공정 하드웨어 모델, 공정 소프트웨어 모델, 공정 운전 절차 모델의 세 부분으로 구성된다. 공정 하드웨어 모델은 공정을 구성하는 장치 자체에 대하여 기술한 것으로, PFD(process flow diagram)이나 P&ID (piping and instrumentation diagram) 도면 정보로부터 구성할 수 있다. 공정 소프트웨어 모델은 사용하는 자동화 프로그램에 대하여 기술한 것으로, PLC를 사용하여 자동화한 시스템의 경우에는 PLC 안에 실린 프로그램이 대상이 된다. 공정 운전 절차 모델은 조업자의 운전 행동에 대하여 기술한 것이다. 부호 모델 검증기는 자체 입력 언어를 가지고 있어서, 이 언어를 사용하여 시스템 모델을 기술한다. 입력의 형태는 사용되는 공정 변수들을 정의하고 이 공정 변수들 사이의 관계를 ~(not), &(and), |(or), next(one scanning time later), :=(define) 등의 연산자를 사용하여 표현하며, 화학공정용 범용 소프트웨어들에서 일반적으로 채택하고 있는 정의언어 방식의 사용자 인터페이스를 사용하고 있다.

2. 검증 질의어

검증 질의어는 시스템의 안전성 및 운전성을 검증하기 위한 질문들로서, 일반적으로 산업 표준 검사표(industrial standard checklists)나 공정 설계 사양, 또는 HAZOP(hazard and operability study)과 FTA(fault tree analysis)와 같은 방법들을 참조하여 구성한다. 검증 질의어는 시간에 따른 상태들의 관계를 나타낼 수 있어야 하므로, CTL (computation tree logic)을 사용하여 기술한다. CTL은 일반적인 이진 논리 개념에다 시간을 다룰 수 있는 연산자가 추가된 형태의 temporal logic이다. 이진 논리 연산자 ~(not), &(and), |(or), ->(imply) 등과, 시간에 관계된 연산자 G(globally), F(future), X(next time), U(until), A(all), E(some) 등의 연산자가 사용된다. 이중 G, F, X, U는 상태 한정자, 즉 시간의 경과에 대한 연산자이고, A, E는 경로 한정자, 즉 모든 경로에서 만족되어야 하는지, 아니면 최소한 한 경로에서만 만족되어도 되는지에 대한 연산자이다. 경로 한정자 하나와 상태 한정자 하나가 한 쌍을 이루어 하나의 연산자로 사용되는데, AX, EX, AU, EU 등의 형태이다[4]. 이 CTL 식들을 사용하여 주어진 상태에서 시작되는 일련의 공정 움직임에 대한 질문을 구성할 수 있다.

3. 모델 검사기

모델 검사기는 시스템 모델의 동작을 표현하는 상태 공간을 검색하고 주어진 검증 질의어에 대해 참, 거짓을 판정한다. 모델 검사기는 시스템 모델의 완전한 상태 전이 그래프를 생성하고, 효율적인 그래프 알고리듬을 이용하여 CTL 식의 파스 트리 내부를 검색한다. 모델 검사기는 CTL 연산자로 구성된 식의 참, 거짓을 결정할 수 있는 알고리듬들의 조합이다. 모델 검사기의 기능 중 하나는 거짓인 질의어에 대해 왜 거짓인지의 반례를 보여주는 것이다. 반례의 궤적은 공정이 어떠한 상태의 경로로 진행되면 대상 질의어가 거짓의 값을 갖는지를 보여주는 나열로, 이를 분석하면 검사의 대상이 되는 시스템에서 오류의 원인이 어디에 있는지를 찾는데 아주 유용하다.

Ⅲ. 공정의 안전 검증을 위한 PLC 모듈 개발

1. PLC 기반의 화학공정 시스템의 모델화

일반적으로 자동화된 화학공정 시스템은 자동화의 관점에서 크게 세 부분으로 구성되는데, 공정 하드웨어 부분과 공정 소프트웨어 부분, 공정 운전 절차 부분 등으로 나누어 생각할 수 있다[4,5]. 공정 하드웨어는 전체 공정을 구성하는 단위 공정들과 이 단위 공정들 사이의 흐름을 담당하는 연결 장치 등의 공정 장치들이고, 공정 소프트웨어는 공정의 자동 잠금장치나 조업 시작, 조업 정지 등의 순차적 논리와 제어 알고리듬 등을 의미하며, 공정 운전 절차는 조업자가 공정을 운전할 때 따라야 하는 운전 절차이다.

앞장에서 설명한 논리 검증법을 이용하여 화학공정 시스템의 안전성 및 운전성을 검사하기 위해서는 전체 공정에 대한 논리 검증기 입력 모델을 구성하여야 하는데, 먼저 각 부분에 대하여 논리 검증기 입력 모델을 구성하고, 이 모델들을 통합하여 전체 공정에 대한 입력 모델을 구성할 수 있다. 전체 공정 모델을 논리 검증기의 입력으로 사용하여 대상 공정 시스템의 안전도를 분석할 수 있다. 그림 2에 이러한 과정의 개요를 보였다.

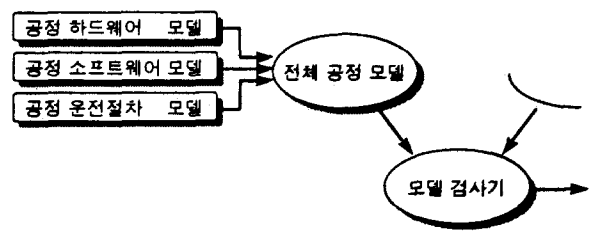

그림 2. 전체 공정 모델의 구성[4,5].
Fig. 2. Integration of system models[4,5].

공정 하드웨어, 공정 소프트웨어, 공정 운전절차 등의 동적 변화를 논리 수식으로 나열하여 모든 수식의 집합체를 형성한 다음, 이것을 SMV 입력 언어로 구현하면 전체 공정 모델이 구성된다. 추론 엔진인 모델 검사기는 McMillan[2]이 개발하였고, 본 연구에서는 화학공정의 안전 검증에 응용할 수 있도록 공정 모델을 개발하는 방법을 연구하였다.

2. PLC 소프트웨어

화학공정 제어시스템은 감지기에서 경영계획 단계까지 기능상으로 구분되는 여러 단계로 구성된 모델로 생각할 수 있다[6]. 하위 단계의 제어기는 개별 공정 설비의 제어를 담당하고, 상위 단계의 제어기는 하위 단계 제어기를 제어하고 운전 조건을 최적화하는 역할을 한다. 이들 중 제일 아래 단계인 안전 잠금장치(safety interlocking) 시스템은 이상 상황 발생시 안전을 위해서 조치되어야 하는 기능을 처리하는 부분으로, 조업자의 안전과 환경의 보호, 주요 설비의 보호 등을 담당한다. 실제로 안전 잠금장치 시스템은 대부분 PLC를 사용하여 구현된다. 또한 조업의 시작이나 정지와 같이 순차적으로 진행되는 절차나 로컬 제어 알고리듬 등의 논리도 PLC에 실리게 된다. 따라서 PLC 기반의 자동화된 화학공정 시스템의 안전성과 운전성을 분석하기 위해서는 먼저 PLC에 실린 안전 잠금장치, 조업 시작, 조업 정지 등의 프로그램에 대한 검증이 있어야 한다.

본 연구에서는 가장 일반적인 PLC용 프로그래밍 언어인

RLL을 사용하여 구현된 화학공정 시스템의 안전도를 논리 검증기를 이용하여 자동으로 검사할 수 있는 방법을 연구하였다. RLL의 프로그램은 일반적으로 전문가가 담당하지만 항상 오류를 포함할 수 있다. PLC가 많이 쓰이게 되면 될수록 RLL에 대한 검사의 중요성도 더욱 커지게 된다. 그러나 지금까지의 RLL 검사 과정은 대부분 설계자의 경험에 근거한 수동적인 검사 방법이나 논리 회로 모사기에 의한 검사 방법을 사용하고 있다. 수동적인 검사 방법에 의한 RLL의 검사는 검사 결과를 보증할 수 없다는 단점이 있으며, 특히 규모가 큰 공정에 대해서는 검사 결과의 신뢰도가 더욱 떨어지게 된다. 논리 회로 모사기에 의한 검사 방법의 경우는 검사할 수 있는 대상 공정의 규모가 상대적으로 작게 제한된다는 단점이 있다. RLL 입력 변수의 수는 임의로 변할 수가 있는데, 입력 변수의 수가 증가하면 검사하여야 할 시나리오의 수도 급속하게 증가하게 된다. 따라서 기존의 접근방법처럼 발생 가능한 모든 시나리오를 검사하는 방법으로는 검사할 수 있는 입력 변수의 수가 제한된다. 논리 검증기를 이용한 RLL의 검사는 이러한 문제들에 대한 합리적인 해결책을 제시한다.

3. RLL 기본 명령어의 모델화

RLL은 PLC에서 사용하는 이진 논리 기반의 프로그래밍 언어이다. Moon[3]은 입력 접점들의 논리 연산만으로 구성되는 RLL 프로그램을 논리 검증기용 입력 언어로 기술된 모델로 변환하는 방법을 제시하였다. RLL의 가로단(rung)들은 연산자 &(and), |(or), ~(not), next(one scanning time later), :=(define) 등을 사용하여 각각 논리 검증기용 입력 언어로 기술할 수 있다. RLL 가로단의 입력과 출력 사이에는 시간 차이가 있는데, 연산자 next를 사용하여 그 시간 차이를 나타낸다. 이 연산자들을 이용하면, RLL의 명령어 중 접점과 출력, 입력의 직렬 연결, 입력의 병렬 연결, 래치 등을 논리 검증기용 입력 모델로 일대일로 변환할 수가 있다.

4. RLL 복합 명령어의 모델화

실제 RLL의 명령어에는 이와 같은 기본적인 명령들 뿐 아니라 타이머, 카운터, 계산값 이동, 계산값 비교, 사칙연산, 논리 연산 등과 관련된 복합 명령어들이 있다. 복합 명령어는 기능적으로는 기본 명령어의 조합에 의해 구현될 수 있는 기능을, 사용하기에 용이하도록 하나의 함수로 만든 명령어이다. 복합 명령어를 포함하여 PLC에서 사용되는 명령어들은 PLC 제조 회사에 따라 서로 다른데, 기능적으로 보면 기본적으로 입력, 출력, AND, OR, XOR, NOT, 메모리 관련 명령어, 시간 관련 명령어, 특수 명령어 등으로 구성된다[7].

논리 검증기를 이용하여 RLL의 논리 검증을 자동화하기 위해서는, RLL의 각 가로단들을 논리 검증기의 입력 모델로 일대일 대응으로 변환시킬 수 있어야 한다. 따라서 기본 명령어만으로 구성되는 가로단의 일대일 변환 뿐 아니라 복합 명령어를 포함한 가로단에 대해서도 일대일로 논리 검증기용 입력 모델을 기술할 수 있어야 한다. 즉 RLL의 타이머나 카운터 명령어와 같은, 단위 명령어들의 조합 형태인 복합 명령어들도 그에 대응하는 논리 검증기용 단위 입력 모델이 개발되어야 한다. 논리 검증기 입력용 RLL 단위 모델들이 개발되어 있다면, RLL을 논리 검증기 입력 모델로 일대일 변환하는 것이 가능하게 되어, 주어진 RLL로부터 논리 검증기 입력 언어로 구현된 시스템 모델을 자동으로 생성하는 것도 구현 가능하게 된다.

본 연구에서는 RLL의 복합 명령어에 대응되는 논리 검증기용 단위 입력 모델을 모듈화된 함수의 형태로 개발하였다. 이 절에서는 이 중에서 ONTMR와 DCAT의 예를 통하여 논리 검증기용 RLL 단위 모델을 개발하는 방법을 보이겠다.

4.1 ONTMR (ON delay timer) 모델

ON 지연 타이머(ONTMR)는 입력조건이 성립된 상태가 설정치 시간만큼 경과된 후에 내부 코일이 켜지는(on) 타이머로, RLL로 나타내면 그림 3과 같다.

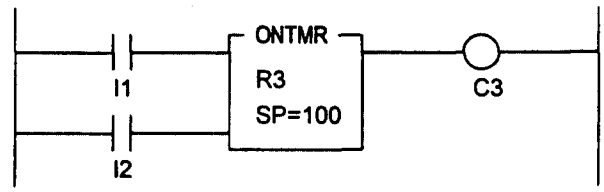

그림 3. ONTMR (ON 지연 타이머).
Fig. 3. ONTMR (ON delay timer).

그림 3에서 R3은 타이머의 레지스터이고, 이 경우 타이머의 설정치는 100이다. 입력 접점 I1이 켜지면(on) 타이머가 작동하여 R3의 경과치가 증가한다. 타이머 작동 중 입력 접점 I2(재설정 신호)가 켜지면 R3이 0으로 재설정된다. 타이머 작동 중 입력 접점 I1이 꺼지면(off) 그 시점에서의 R3 경과치는 그대로 유지된다. 후에 입력 접점 I1이 다시 켜지면 R3 경과치도 다시 증가한다. R3 경과치가 증가하여 설정치와 같아지면 내부 코일 C3이 켜지며, 그 후 입력 접점 I2가 켜지면 내부 코일 C3이 꺼지고 R3 경과치는 0으로 재설정된다. C3이 켜져 있는 경우는 R3 경과치가 설정치로 되어 있으므로 입력 접점 I1이 꺼지거나 켜져도 타이머에는 변화가 없다.

그림 3의 ONTMR에 대응되는 논리 검증기 입력 모듈을 그림 4에 보였다. 모듈 ONTMR는 네 개의 인자 I1, I2, SP, R을 가진다. I1은 모듈의 설정 조건이고, I2는 모듈의 재설정 조건이다. SP는 타이머의 설정치이고, R은 이 ONTMR 모듈을 사용하는 모듈 내에 정의되어 있는 타이머 레지스터이다. 레지스터의 동작이 논리 검증기 입력 모듈의 3번째 줄에서 8번째 줄까지에 정의되어 있다. 모듈의 출력은 10번째 줄에서 13번째 줄까지에 정의되어 있다.

```
1   MODULE ONTMR( I1, I2, SP, R )
2   ASSIGN
3     init(R) := 0;
4     next(R) :=  case
5                   I2 : 0;
6                   !I2 & I1 & R<SP : R+1;
7                   1 : R;
8                 esac;
9   DEFINE
10    output :=   case
11                  R=SP : 1;
12                  1 : 0;
13                esac;
```

그림 4. ONTMR의 논리 검증기 입력 모듈.
Fig. 4. SMV module for ONTMR.

그림 4의 모듈이 그림 3의 RLL 명령어의 모든 상태를 그대로 재현할 수 있는 지를 확인하기 위하여, 그림 3의 명령어가 가질 수 있는 모든 경우에 대하여 하나하나 그림 4의 모듈에 적용하여 양쪽의 결과를 비교하였고, 그 동치성을 증명하였다. 다른 RLL 명령어에 대한 모듈들에 대하여도 마찬가지 방법으로 확인하였다.

4.2 DCAT (discrete control alarm timer) 모델

이산 제어 경보 타이머(DCAT)는 하나의 입력과 두 개의 피드백(feedback)을 가지고 있는 장치에 사용하기 위한 RLL 복합 명령어로, 그림 5에 보인 것과 같다.

DCAT 블록의 입력으로는 장치의 상태를 결정하는 논리

값이 사용되며, 출력은 장치를 제어하는데 사용된다. 이 복합 명령어는 장치가 움직이는데 걸리는 시간을 측정하여 오류 상태인 경우 경보를 발생하도록 하는 역할을 하는 것으로, 여러 개의 가로단들로 구성되는 기능을 대체할 수 있다.

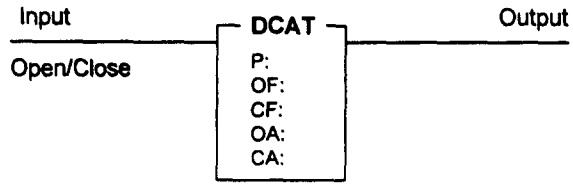

그림 5. DCAT (이산 제어 경보 타이머).
Fig. 5. DCAT (discrete control alarm timer).

그림 5에서 P는 장치를 열거나 닫고자 할 때 허용되는 시간이다. OF(open feedback)는 제어되는 장치가 특정 위치까지 열렸음을 감지하는 장치로부터 들어오는 입력이고, CF(close feedback)는 장치가 특정 위치까지 닫혔음을 감지하는 장치로부터 들어오는 입력이다. OA(open alarm)는 DCAT의 입력이 켜져(on) 있을 때, DCAT 타이머 허용시간 전에 OF 입력이 들어오지 않으면 켜지는 제어 릴레이이다. CA(close alarm)는 DCAT의 입력이 꺼져(off) 있을 때, DCAT 타이머 허용시간 전에 CF 입력이 들어오지 않으면 켜지는 제어 릴레이이다.

DCAT의 개폐 입력신호가 꺼져있다가 켜지면 다음의 동작들이 일어난다.

· 시간 지연이 P 값으로 설정되고, 두 경보 출력 OA와 CA는 꺼지고, DCAT의 출력은 켜짐.

· DCAT의 개폐 입력신호가 켜져있는 동안, 타이머는 OF 입력이 켜지거나 허용 시간을 초과할 때까지 작동됨.

· OF 입력이 허용 시간 전에 켜지면, 시간 지연은 0으로 설정되고, OA는 계속 꺼져 있음.

· OF가 허용 시간 전에 켜지지 않으면, OA가 켜짐.

· OF가 허용 시간 전에 켜졌으나 개폐 입력신호가 켜져있는 동안 OF가 다시 꺼지면, OA가 켜짐. 그 후 OF가 다시 켜지면 OA는 꺼짐.

DCAT의 개폐 입력신호가 켜져있다가 꺼지면 다음의 동작들이 일어난다.

· 시간 지연이 P 값으로 설정되고, 두 경보 출력 OA와 CA는 꺼지고, DCAT의 출력도 꺼짐.

· DCAT의 개폐 입력신호가 꺼져있는 동안, 타이머는 CF 입력이 켜지거나 허용 시간을 초과할 때까지 작동됨.

· CF 입력이 허용 시간 전에 켜지면, 시간 지연은 0으로 설정되고, CA는 계속 꺼져 있음.

· CF가 허용 시간 전에 켜지지 않으면, CA가 켜짐.

· CF가 허용 시간 전에 켜졌으나 개폐 입력신호가 꺼져있는 동안 CF가 다시 꺼지면, CA가 켜짐. 그 후 CF가 다시 켜지면 CA는 꺼짐. 단, OF와 CF가 모두 켜지면, OA와 CA가 모두 켜지며, 이것은 장치의 개폐 위치를 감지하는 감지기가 고장난 경우라고 생각할 수 있다.

그림 5의 DCAT에 대응되는 논리 검증기 입력 모듈을 그림 6에 보였다. 모듈 DCAT는 일곱 개의 인자 INPUT, P, R, OF, CF, OA, CA를 가진다. INPUT은 개폐 입력신호이고, P는 장치 개폐 허용 시간으로 타이머의 설정치이다. R은 이 DCAT 모듈을 사용하는 모듈 내에 정의되어 있는 타이머 레지스터이다. OF, CF, OA, CA는 각각 위에서 설명한 것과 같다. 5번째 줄과 6번째 줄의 old_input은 바로 이전 스캐닝 시간에서의 INPUT의 값을 저장하는 변수이다. 허용 시간 안에 장치의 개폐 동작이 완료되었는지를 알려주는 변수 flag가 7번째 줄에서 12번째 줄까지에 정의되어 있다. 타이머 레지스터의 동작은 13번째 줄에서 18번째 줄까지에 정의되어 있고, 19번째 줄에서 36번째 줄까지는 장치의 개폐 실패에 따른 경보를 구현한 것이다. 이 모듈 DCAT의 출력은 38번째 줄에서 정의한 것과 같이 항상 입력과 같다.

```
1   MODULE DCAT( INPUT,P,R,OF,CF,OA,CA )
2   VAR
3     old_input, flag : boolean;
4   ASSIGN
5     init(old_input) := INPUT;
6     next(old_input) := INPUT;
7     init(flag) := 0;
8     next(flag) :=case
9           old_input = !INPUT : 0;
10          !flag&R>0&((INPUT&OF)|(!INPUT&CF)) : 1;
11          1 : flag;
12    esac;
13    init(R) := P;
14    next(R) := case
15                  old_input = !INPUT : P;
16                  !flag & R>0 : R-1;
17                  1 : 0;
18                esac;
19    init(OA) := 0;
20    next(OA) :=case
21                  OF & CF : 1;
22                  old_input = !INPUT : 0;
23                  !INPUT : OA;
24                  flag : !OF;
25                  !flag & R=0 : 1;
26                  1 : OA;
27                esac;
28    init(CA) := 0;
29    next(CA) :=case
30                  OF & CF : 1;
31                  old_input = !INPUT : 0;
32                  INPUT : CA;
33                  flag : !CF;
34                  !flag & R=0 : 1;
35                  1 : CA;
36                esac;
37  DEFINE
38    output := INPUT;
```

그림 6. DCAT의 논리 검증기 입력 모델.
Fig. 6. SMV module for DCAT.

Ⅳ. PLC 모듈 적용 예

RLL의 복합 명령어에 대응하는 논리 검증기용 단위 입력 모델을 개발하면, 기본 명령어 뿐 아니라 복합 명령어가 모두 포함된 일반적인 RLL에 대하여 논리 검증기용 입력 모델로 일대일로 변환하는 것이 가능하게 된다. 다음은 파이프라인에 연결되어 있는 솔레노이드 밸브의 개폐를 제어하는 공정을 대상으로 그 과정을 보인 예이다. 대상 공정에서 파이프라인의 직경으로 인해 밸브를 여닫는데 30초가 걸린다고 설정했다. 그림 7에 개요도를 보였다. 전기 솔레노이드 Y7에 전기를 보냄으로써 밸브를 여닫을 수 있다. 상한 위치 감지기 X17은 밸브가 열렸음을 알려주는 피드백이고, 하한 위치 감지기 X18은 밸브가 닫혔음을 알려주는 피드백이다.

이 밸브 공정에서는 밸브 상태에 대한 피드백을 통해서 밸브가 열렸는지, 닫혔는지, 개폐가 진행중인지, 개폐에 실패했는지, 감지기가 고장났는지 등에 대한 정보를 얻을 수 있다. 밸브의 상태를 쉽게 알 수 있도록 다음과 같은 경보 시스템을 설계한다고 하자. 밸브가 열리는데 실패했을 때에는 경보 Y1이 켜지고, 밸브가 닫히는데 실패했을 때에는 경보 Y2가 켜진다. 위치 감지기로부터의 두 피드백 신호가 모

두 있으면, 감지기가 고장났음을 알려주는 경보 Y3이 켜진다. 밸브가 두 위치 감지기 사이에서 움직이고 있을 때에는 이동중임을 표시하는 표시기 Y4가 켜진다.

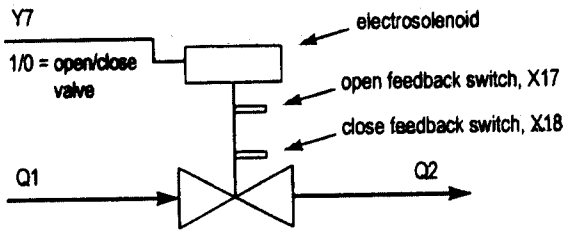

그림 7. 솔레노이드 밸브.
Fig. 7. Solenoid valve.

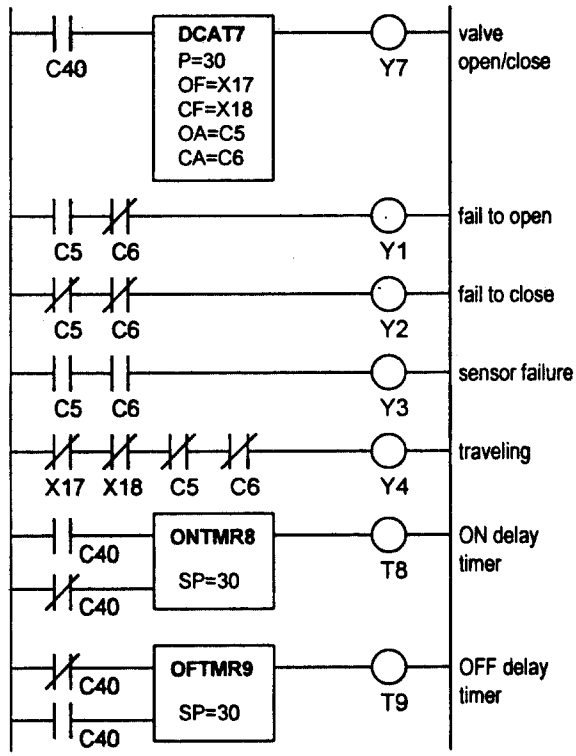

그림 8. 밸브 경보 시스템을 구현한 RLL.
Fig. 8. RLL for the alarm system of the solenoid valve process.

이 밸브 공정에 대하여 경보 Y1에서 Y4까지를 RLL로 구현한 것이 그림 8이다. 제어 릴레이 C40을 켜거나 끔으로써 밸브의 상태 Y7을 제어하게 된다. Y7은 DCAT 명령어에 상관없이 C40의 상태에 따른다. 그림 8에서 ONTMR와 OFTMR(OFF delay timer)는 각각 밸브의 개폐 허용시간을 재기 위하여 사용한 것으로, 논리 검증기를 이용하여 경보기가 제대로 구현되었는지를 검사할 때 사용할 질의어를 구성하기 위한 것이다. OFTMR는 입력조건이 성립되지 않은 상태가 설정치 시간만큼 경과되면 내부 코일이 꺼지는 타이머에 대한 모듈로, 그림 4의 ONTMR와 같은 방법으로 구현한다.

그림 9는 그림 8의 RLL을 논리 검증기 입력 언어로 표현한 것으로, 전체 공정에 대한 논리 검증기 입력 모델의 일부이다. 그림 9에서 ASSIGN 다음에 나오는 7개의 줄은 그림 8의 각 가로단들과 일대일로 대응된다. 이미 모듈 형태로 개발한 RLL의 복합 명령어 DCAT와 ONTMR, OFTMR를 사용하여, 해당하는 블록을 하나의 변수로 선언하고 각 가로단을 한 줄의 입력 언어로 변환하였다. 밸브의 경우도 solenoid_valve 모듈을 개발하여 그 기능을 모사하였다. DEFINE 부분에서 밸브 출력 유량의 유무를 나타내는 변수 Q2와 밸브의 상한/하한 위치 감지기 변수 X17, X18을 solenoid_valve 모듈의 출력 변수들로 정의하여 공정 변수들을 연결하였다.

```
VAR
  DCAT7 : DCAT( C40, 30, R1, X17, X18, C5, C6 );
  ONTMR8 : ONTMR( C40, !C40, 30, R2 );
  OFTMR9 : OFTMR( !C40, C40, 30, R3 );
  Valve1 : solenoid_valve( Y7, 25, R4, Q1 );
DEFINE
  Q2  := Valve1.Fout;
  X17 := Valve1.Zopen;
  X18 := Valve1.Zclose;
ASSIGN
  next(Y7) := DCAT7.output;
  next(Y1) := C5 & !C6;
  next(Y2) := !C5 & !C6;
  next(Y3) := C5 & C6;
  next(Y4) := !X17 & !X18 & !C5 & !C6;
  next(T8) := ONTMR8.output;
  next(T9) := OFTMR9.output;
```

그림 9. 그림 8의 RLL에 대한 시스템 모델.
Fig. 9. System description for RLL in Fig. 8.

이와 같이 전체 공정에 대한 시스템 모델을 구성하고, 경보 시스템을 구현한 RLL의 동작을 검사할 질의어를 준비하였다. 첫 번째 질의어는 다음과 같다.

AG((C40 & T8 & Q1 & !Q2) -> AX Y1)

이 질의어는 밸브 열기 실패 경보 Y1이 제대로 동작되도록 구성되었는지에 대한 질문으로, 그 의미는 "밸브를 열라는 신호(C40)가 밸브 이동 시간 동안 유지된 후(T8)에, 밸브 입력 유량이 있음(Q1)에도 불구하고 밸브 출력 유량이 없으면(!Q2), 즉시 밸브 열기 실패 경보(Y1)가 켜지는가"이다. 이 질의어에 대하여 논리 검증기는 TRUE의 결과를 보였고, 이는 경보 Y1의 동작이 제대로 설계되었다는 것을 보장한다.

두 번째 질의어는 공정의 상태에 대한 질문으로, 밸브 닫기 실패 경보 Y2와 밸브 위치 감지기 고장 경보 Y3, 그리고 밸브의 입출력 유량사이의 관계 등에 의한 복합적인 상황이 설계한 대로 동작되고 있는 지에 대한 질의어이다.

AG((!C40 & !T9 & Q1 & !Y3 & Y2) ->
A[(Q2 & !C40 & !T9) U (!Q1 | (!Y3 & !Y2))])

즉, "밸브를 닫으라는 신호(!C40)가 밸브 이동 시간 동안 유지된 후(!T9)에, 밸브 입력 유량이 존재(Q1)하고 감지기가 고장나지 않은 상태(!Y3)에서 밸브 닫기 실패 경보(Y2)가 켜지면, 밸브 닫기 신호가 그대로 유지되는 동안에는 밸브 입력 유량이 없어지거나 밸브 닫기 실패 경보가 꺼지기 전까지는 밸브 출력 유량이 계속 존재하는가"라는 질문으로, 밸브 닫기가 실패되면 밸브의 출력 유량은 계속 존재하게 되는 특성을 확인하는 것이다. 그런데 논리 검증기를 이용하여 이 질의어로 검사한 결과는 FALSE 였다. Y2나 Y3 또는 밸브의 동작에 대한 모듈의 설계 중 어느 부분이 잘못 설계된 것이다. 논리 검증기는 이 질의어에 대하여 FALSE가 나오게 된 경로도 제시하는데, 이를 분석하면 문제의 원인을 찾아낼 수 있고, 시스템을 다시 설계할 수 있다. 오류의 원인을 분석한 결과, Y2의 설계가 잘못된 것이었고, 그림 10과 같이 Y2에 대한 RLL 가로단을 수정하여야 한다.

수정된 RLL에 대하여 두 번째 질의어를 사용하여 다시 논리 검증기로 검사한 결과는 TRUE 였다. 즉, Y2와 Y3, 밸브의 입출력에 대한 설계가 모두 제대로 되었음을 알 수

있다. 단, 이 방법은 질문하지 않은 시스템의 동작에 대해서까지 보장하지는 못한다.

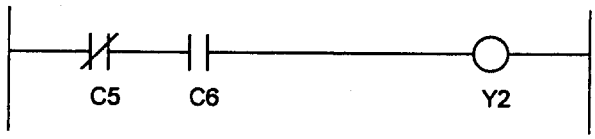

그림 10. 수정된 RLL 가로단.
Fig. 10. Revised RLL rung.

이상과 같이 솔레노이드 밸브 공정에 대한 경보 시스템을 예로 공정의 RLL과 공정 장치로부터 논리 검증기용 시스템 모델을 개발하는 방법과 질의어를 준비하여 설계의 오류를 검사하는 방법 등을 살펴보았다. RLL의 복합 명령어에 해당하는 모듈을 개발하여 사용하면, RLL로부터 시스템 모델로의 변환이 일대일로 가능하게 됨을 보였다.

V. 결론

본 연구는 논리 검증기를 이용하여 PLC를 사용한 공정의 안정성과 운전성을 검사하는 방법을 더욱 개선하고 체계화하였다. RLL 복합 명령어의 기능을 모듈 형태로 구현한 RLL 단위 모델들을 개발하였고, ONTMR 모듈과 DCAT 모듈을 예로 하여 그 개발 방법을 설명하였다. 실례로 솔레노이드 밸브의 개폐 제어 공정에 대한 경보 시스템을 대상으로 하여, RLL 단위 모델을 이용하면 시스템 모델의 개발이 가로단별로 가능해짐을 보였다.

PLC는 안전 잠금장치나 조업 시작 절차, 조업 정지 절차 등과 같은 순차적 논리들을 처리하는데, 주로 공정의 안전과 밀접한 관련이 있기 때문에, 공정의 동작이 안전한지 그리고 설계한 대로 동작하는지를 검사하기 위해서는, PLC의 RLL 동작에 오류가 없음을 확인하여야 한다. 논리 검증기를 이용하면 RLL의 동작을 검증하고 오류를 찾을 수 있으므로, 공정의 안전을 체계적으로 검사할 수 있다. 특히 RLL 단위 모델의 개발을 통하여 일반적인 RLL을 대상으로 논리 검증기를 사용할 수 있도록 확장하였다. RLL로 구현된 공정에 대하여 논리 검증기용 시스템 모델을 일대일로 개발할 수 있으므로 시스템 모델을 자동으로 생성할 수 있는 가능성을 열었다.

향후에는 RLL에서 논리 검증기용 입력 언어로의 자동 번역이 가능하도록 자동 번역 시스템에 대한 연구가 진행되어야 하고, 사용의 편리성을 높이기 위하여 그림운영 방식의 사용자 인터페이스에 대한 연구도 필요하다.

참 고 문 헌

[1] E. M. Clarke, E. A. Emerson and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Trans. Programming Lang. and Sys.*, vol. 8, no. 2, pp. 244-263, April 1986.

[2] K. L. McMillan, *Symbolic Model Checking - An approach to the state explosion problem*, Ph.D Thesis, Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, 1992.

[3] I. Moon, "Modeling programmable logic controllers for logic verification," *IEEE Control Systems*, vol. 14, no. 2, pp. 53-59, April 1994.

[4] I. Moon, G. J. Powers, J. R. Burch and E. M. Clarke, "Automatic verification of sequential control systems using temporal logic," *AIChE J.*, vol. 38, no. 1, pp. 67-75, January 1992.

[5] 정상헌, 이산 화학공정의 안전도 분석을 위한 논리 검증기 개발, 박사학위논문, 서강대학교 화학공학과, 1994.

[6] T. G. Fisher, *Batch Control Systems: Design, Application, and Implementation*, ISA, USA, 1990.

[7] ISA-S5.2, *Binary Logic Diagrams for Process Operations*, ISA, North Carolina, 1981.

[8] S. Jeong, K. Lee and I. Moon, "Safety analysis of fuel cell processes using symbolic model checking," *ISICIMS '94 Proceeding*, Seoul, November 1994.

[9] S. Probst and G. J. Powers, "Automatic verification of control logic in the presence of process faults," *1994 AIChE Annual Meeting*, San Francisco, 1994.

[10] I. Moon, *Automatic Verification of Discrete Chemical Process Control Systems*, Ph.D Thesis, Dept. of Chemical Engineering, Carnegie Mellon University, Pittsburgh, 1992.

정 상 현

1963년생. 1986년 서강대학교 화학공학과 졸업 (공학사). 1989, 1995년 동 대학원 화학공학과 졸업 (공학석사, 공학박사, 공정제어전공). 1996년 ~ 현재 포항공과대학교 공정산업의 지능자동화연구센터 Post-doc. 주요 관심 분야는 공정 안전, 신경회로망 응용, 지능형 제어, 컴퓨터 제어 시스템 등이다.

문 일

1960년생. 1983년 연세대학교 화학공학과 졸업 (공학사). 1985년 한국과학기술원 화학공학과 졸업 (공학석사). 1992년 미국 Carnegie Mellon대 화학공학과 졸업 (공학박사). 1993년 ~ 연세대학교 화학공학과에 임용되어 현재 부교수. 1985 ~ 1987년 한국과학기술연구소 연구원. 1993년 영국 Imperial College 연구원. 주요 관심 분야는 공정 설계, 운전, 제어에 관한 기초 이론 및 응용 등이다.

이 광 순

1955년생. 1977년 서울대학교 화학공학과 졸업 (공학사). 1979년 한국과학기술원 화학공학과 졸업 (공학석사). 1983년 한국과학기술원 화학공학과 졸업 (공학박사, 공정제어전공). 1983년 서강대학교 화학공학과에 임용되어 현재 정교수. 1986년 캐나다 Univ. of Waterloo 방문교수. 1995년 미국 Auburn Univ. 방문교수. 1991년 ~ 현재 포항공과대학교 부설 공정산업의 지능자동화센터(ARC) 연구원으로 회분식공정 자동화 분야 담당. 주요 관심 분야는 batch, repetitive, transient 공정의 학습제어 및 MPC 응용 등이다.