J. Korea Saf. Manag. Sci. Vol. 17 No. 1 March 2015
http://dx.doi.org/10.12812/ksms.2015.17.1.179

ISSN 1229-6783(Print)
ISSN 2288-1484(Online)

생산자동화시스템 PLC 제어프로그램의 안전성 정형검증에 관한 연구

박 창 목*
*인덕대학교 테크노경영과

Formal Verification of PLC Program Safety in Manufacturing Automation System

Chang Mok Park*
*Dept. of Technology & Systems Management, Induk University

Abstract

In an automated industry PLC plays a central role to control the automation system. Therefore, fault free operation of PLC controlled automation system is essential in order to maximize a firm's productivity. A prior test of control system is a practical way to check fault operations, but it is a time consuming job and can not check all possible fault operation. A formal verification of PLC program could be a best way to check all possible fault situation. Tracing the history of the study on formal verification, we found three problems, the first is that a formal representation of PLC control system is incomplete, the second is a state explosion problem and the third is that the verification result is difficult to use for the correction of control program. In this paper, we propose a transformation method to reproduce the control system correctly in formal model and efficient procedure to verify and correct the control program using verification result. To demonstrate the proposed method, we provided a suitable case study of an automation system.

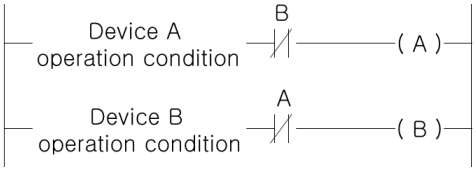
Keywords: PLC, Formal Verification, Safety, SMV, Automation

1. 서 론

생산자동화시스템이 PLC(Programmable Logic Controllers) 제어 프로그램은 작업장의 안전을 고려하여 설계되어야 한다. 만약 제어 프로그램이 완벽하게 설계되지 않은 경우 생산자동화시스템에 오동작이 발생할 수 있으며 이로 인해서 인명피해 및 고가 설비의 파괴 등 큰 손실이 발생할 수 있다. 이러한 자동화 생산 시스템의 오동작은 PLC 제어 프로그램에서 사용되는 특정신호들, 예를 들어 특정설비들을 구동하기 위한 출력신호들이 동시에 ON상태가 됨으로써 대부분 발생 된다. 따라서 자동화 생산 시스템의 오동작은 특정신호들이 동시에 ON되는 에러상황이 절대로 발생하지 않도록 PLC 제어 프로그램을 구현함으로써 방지될 수 있다.

†본 연구는 2013 인덕대학 학술연구비 지원에 의해서 수행되었음.
†Corresponding Author: Chang Mok Park,
Dept. of Technology & Systems Management, Induk University
M·P: 010-4928-6634, email: cmpark@induk.ac.kr
Received October 27, 2015; Revision Received March 17, 2015; Accepted March 17, 2015.

PLC 제어 프로그램은 생산자동화시스템의 설비들을 순차제어하는 기능과 에러상황을 방지하기 위한 인터록(Interlock) 기능을 포함한다. 인터록 기능은 안전에 관련된 문제이기 때문에 100% 완벽하게 구현되어야 한다. 예를 들어 인터록 기능은 특정 설비의 동작에 관련된 신호들이 동시에 ON상태가 되지 않도록 [Figure 1]에 도시된 바와 같이 프로그램되어 PLC 제어 프로그램에 포함된다.



[Figure 1] Interlock program

[Figure 1]을 참조하면 설비A구동조건이 만족되어도 B신호가 NOT으로 연결되어 있기 때문에 B신호가 ON상태이면 첫 번째 회로는 항상 OFF가 되어 A는 절대로 ON상태가 될 수 없다. 마찬가지로 설비B구동조건이 만족되어도 A신호가 NOT으로 연결되어 있기 때문에 A신호가 ON상태이면 두 번째 회로는 항상 OFF가 되어 B는 절대로 ON상태가 될 수 없다. 그러나 생산시스템의 PLC 제어 프로그램은 수천 개의 신호와 복잡한 로직으로 구성되어 있기 때문에 이렇게 특정신호들이 동시에 ON상태가 되는 에러 상황이 발생하지 않도록 하는 인터록 기능이 완벽히 구현되었는지를 PLC 프로그램 작성자 등이 일일이 확인하기가 용이하지 않다.

이러한 PLC 제어 프로그램의 안전성 검증을 위하여 시운전 및 가상 시뮬레이션을 사용하고 있으나[1] 모든 경우를 테스트하는 것을 불가능하므로 궁극적인 해결책이 될 수 없다. 최근 생산제어프로그램의 검증 방법으로 정형검증이 연구되고 있다.

정형검증 도구인 SMV(Symbolic Model Verifier)[2]는 상태탐색을 기반으로 하는 검증기법으로, 유한상태기계(FSM, Finite State Machine)로 표현된 시스템과 특성(Specification)이 주어지면, 모델체킹 알고리즘을 이용하여 주어진 시스템이 검증하고자 하는 특성을 만족하는지를 알아보기 위해서 전체 상태공간(state space)를 검사한다. 만일 특성을 만족하지 않는다면 반증예(counter example)를 제시하여 준다. SMV가 시간을 고려하지 않는 automata를 기본단위로 하는 반면, UPPAAL 은 timed-automata 를 사용하여 실시간 시스템의 정형검증이 가능하다[3]. 그러나 UPPAAL은 embedded system 의 검증용으로 만들어 졌으며, 비교적 규모가 큰 생산제어프로그램을 실시간 시스템으로 간주하여 검증하기에는 탐색공간에 너무 커서 현실적으로 적용이 어렵다.

최근 SMV를 활용한 제어 프로그램의 정형검증에 관한 다양한 시도들이 있다. PLC 제어프로그램을 정형모델로 변환하면 SMV를 사용한 검증이 이론적으로 가능하다[4]. 그렇지만 3가지 중요한 현안들이 해결되어야 한다. 첫째 제어프로그램 과 실제구동환경이 정형모델로 동일하게 표현되어야 한다. 둘째 규모가 큰 생산제어프로그램을 검증하기 위해서는 탐색 공간 축소를 위한 방법이 고려되어야 한다. 셋째 검증결과의 분석 및 활용이 가능해야 한다.

첫 번째 현안을 해결하기 위한 방법으로서, J. Lahtinen et al[5]은 제어 프로그램을 정형모델로 변환하기 위한 함수블록단위의 자동화된 프로세스를 정의하였다. O. Rossi 와 Ph. Schnoebelen[6]는 LD(Ladder Diagram)으로 작성된 프로그램을 정형모델로 표현함에 있어서 사이클을 가진 PLC구동 특성을 반영하였고, 타이머의 동작을 스위치로 추상화하여 사용하였다. 또한 Tord Alenljung와 Bengt Lennartson[7]은 고급언어인 Sensor Graphs로 표현된 PLC 프로그램을 정형모델로 변환하여 검증하였다.

두 번째 현안에 대한 연구로서, Doaa Soliman 과 Gerogy Frey[8]은 PLCOpen에서 제공된 SFB(Safety Function Blocks)단위로 구축된 프로그램을 검증하는 방법론을 제시하여, 전체 프로그램을 블록단위로 검증하여 검증 부하를 줄이기 위해 노력하였다. Min Zhou et al[9]은 IL 로 작성된 PLC 프로그램을 모듈로 분리하여 모듈 단위로 검증을 수행하여 탐색 공간을 감소시켰다.

세 번째 현안과 관련하여, 지은경 와 2인[10]은 원천계측제어시스템의 FBD(Function Block Diagram)를 검증하기 위하여 SMV를 기반으로 FBDVerifier 도구를 개발하였다. FBDVerfier 는 FBD를 Verilog 모델로 변환하여 SMV 모델체커를 이용해 정형검증을 수행하며, 반례(counter example)를 그래프형식으로 제공하여 오류분석을 쉽게 하도록 하고 있다.

본 연구는 위에서 언급한 첫 번째 현안을 위하여, 제어 프로그램 및 실제 구동 환경을 정형모델로 동일하게 표현하는 방법론을 제시한다. 그 중 기존연구들에서 고려하지 못한 프로그램 구동 순서 및 타이머 동작이 검증에 미치는 영향을 규명하고, 정형모델로 변환하는 방법론을 제시하려고 한다. 제어프로그램은 PLC 하드웨어 내부의 구동 특성으로 명령문들이 순차적으로 실행된다. 그러나 기존 연구에서는 제어프로그램을 변수들의 상태변환 시스템으로 간주하여 정형모델로 변환

ISSN 1229-6783(Print)
ISSN 2288-1484(Online)

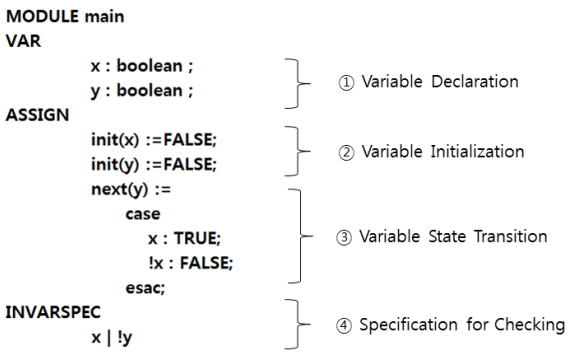
함으로써 실제와 다른 검증결과를 가져올 수 있다. 또한 기존 연구에서는 타이머를 생략하거나 단순한 스위치로 간략화 하여 검증하기 때문에 검증 신뢰도가 감소한다. 본 연구에서는 명령문들이 순차적으로 실행되는 것을 표현하는 automata 와 타이머를 이산상태를 가진 automata 로 표현하여 제어 시스템을 동일하게 정형 모델로 재현함으로써 검증 신뢰도를 향상하도록 하였다.

또한 본 연구에서는 두 번째 현안, 즉 탐색 공간 폭발문제를 극복하기 위한 릴레이 검증 및 수정 기법을 제안하여 효율적으로 대형 시스템을 검증하고 수정할 수 있도록 하였다. 릴레이 검증 및 수정에서는 신호의 의존관계 및 프로그램 구조를 이용하여 전체 검증 및 수정 문제를 단계적으로 실행하도록 하였다. 반면 기본 연구는 블록단위 검증만 수행하고 전체 검증에 대한 해법을 제시하고 못하였고, 특정 오류를 찾아내기만 하고 그 활용에 대한 방안은 제시하지 못하고 있다.

본 연구에서 사용한 SMV는 NuSMV 2.5.2버전을 사용하였다. NuSMV는 opensource 버전으로 nusmv.fdk.eu에서 다운로드 가능하다.

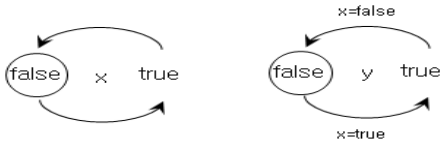
2. SMV를 이용한 정형검증

정형검증은 정형논리(Formal logic)을 이용하여 설계대상 시스템이 설계명세서(Specification)와 동일하게 동작하는 지를 검증하기 위하여 전체 상태 공간을 검사하는 방법이다. 이러한 상태공간은 BDD(Binary Decision Diagrams)로 표현되며, 상태공간을 줄이기 위해 BDD를 최소 크기로 줄인 후, 주어진 특성을 만족하지 않는 경로를 찾아내어 반증 예를 제시하여 준다. 반증 예는 특성이 만족되지 않는 시스템의 상태를 보여주는 것이다. SMV를 이용하여 특정 시스템을 검증하기 위해서는 [Figure 2]와 같은 4가지 구성요소가 제공되어야 한다.



[Figure 2] SMV elements

[Figure 2] SMV 예는 입력값 x에 따라 출력값 y가 결정되는 이산 상태 논리 시스템을 표현한 것이며, [Figure 3]과 같은 2개의 automata 로 표현할 수 있다. 이때 신호의 ON, OFF 상태는 true, false 로 표현된다.

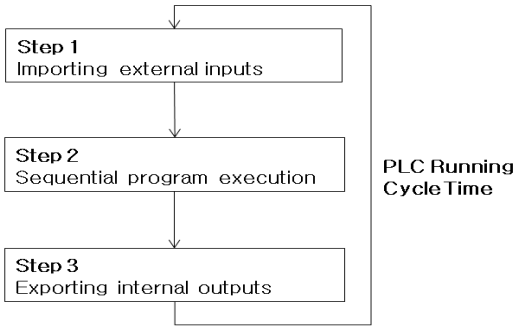


[Figure 3] A simple automata

원으로 표시된 상태는 초기상태를 나타내며, x의 상태변화는 임으로 발생된다고 간주고, y의 상태변화는 x값에 따른 상태전이특성으로 표현된다. 검증할 시스템 특성은 주어진 식이 항상 true 가 되는지를 체크하는 것이다. 만일 false가 존재한다면 이에 대한 예를 제시하게 된다. 위 시스템은 x, y 변수의 상태로 표현되는 이산 상태 시스템이며, 검증할 시스템 특성은 논리식 (x or !y)가 항상 true가 되어야 하는 것이다. 만약 x가 false 가 되고 y가 true가 되면 결과는 false 가 되므로 설계명세서에 위배되는 것으로 판단할 수 있다. 실제 위 시스템은 상태1(x=false, y=false) -> 상태2(x=true, y=false) -> 상태3(x=false, y=true) -> 상태4(x=false, y=false) 형태로 상태변화를 관측할 수 있으며 상태3에서 설계명세서가 위배되는 상황이 발생될 수 있다. 직관적으로 x에 의해서 y가 결정되기 때문에 상태3은 발생되지 않을 것처럼 보이지만, x의 변화가 y에 반영되기 직전의 상태에서는 발생될 수 있다.

3. PLC 구동환경을 고려한 SMV 변환

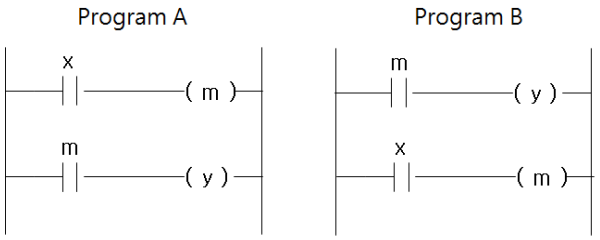
PLC 구동 시스템은 [Figure 4]에 도시된 바와 같은 사이클 동작을 반복 수행한다.



[Figure 4] PLC operation system

단계 1에서 외부 입력신호가 PLC 제어 프로그램의 내부입력메모리에 복사된다. 이에 따라 단계 1 이외의 단계에서 입력신호가 바뀌어도 내부입력메모리에는 반영되지 않는다. 단계 2에서 PLC 제어프로그램이 순차적으로 실행된다. 이때 실행순서에 따라 최종 결과가 달라지기 때문에 PLC 제어프로그램의 코드순서가 검증모델에서 고려되어야 한다. 순차적인 PLC 제어프로그램의 실행결과는 출력내부메모리에 저장된다. 마지막으로 단계 3에서는 출력메모리가 외부출력신호에 일괄적으로 복사된다. 이때 단계 3에서 출력신호에 대한 에러 상황이 체크 되어져야 한다.

입력신호(x), 내부 신호(m), 출력 신호(y)로 구성된 [Figure 5]의 프로그램 A와 B는 다른 상태 특성을 가진다.



[Figure 5] Two simple programs

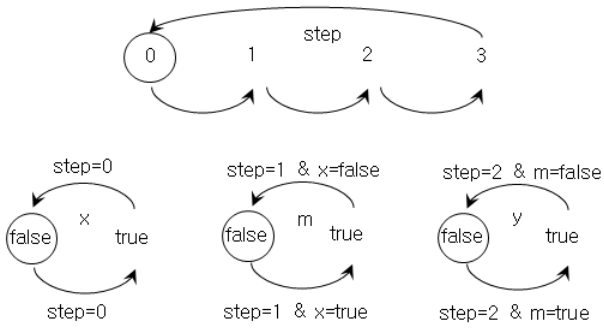
모든 신호의 초기상태가 false이고, 중간에 입력 신호 x가 true가 되는 경우 관측되는 상태 특성(단계 3에서 관측되는 상태)은 [Table 1]과 같이 요약할 수 있다.

<Table 1> Variable state transition
(true=1, false=0)

	Program A			Program B		
Variable	x	m	y	x	m	y
Initial state	0	0	0	0	0	0
x > TRUE						
state 1	1	1	1	1	1	0
state 2	1	1	1	1	1	1

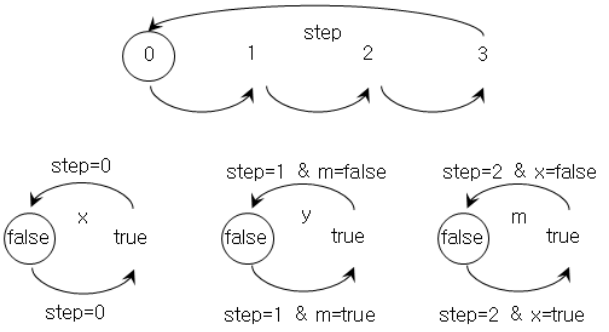
<Table 1>에서 보는 바와 같이 Program A에서는 (x=true and y=false) 상태가 관측되지 않지만, Program B 에서는 관측된다. 왜냐하면 Program A에서 x의 변화가 구동 사이클 내부에서 바로 y에 전달되지만, Program B에서는 x의 변화가 y에 전달되기 위해서는 사이클이 두 번 구동되어야 하기 때문이다.

SMV의 각 automata 는 언제든지 조건이 만족되면 상태변환하는 것을 가정하고 시스템 상태 특성을 탐색한다. 그러나 PLC 제어 로직은 앞에서 설명한 구동사이클을 가지고 있어서, 특성 상태가 발생되는지 체크는 항상 단계 3에서 수행되어야 하며, 로직 구동순서에 의존적이어야 한다. 본 연구에서 PLC제어로직을 automata로 변환할 때 step이라는 automata 를 삽입하여 PLC 구동 시스템의 사이클 특성 및 로직 구동순서를 구현하였다. Program A를 PLC 구동 환경을 고려하여 automata 로 표현 하면 [Figure 6]과 같다.



[Figure 6] Automata for program A

[Figure 6]에서 표현된 automata 시스템에서 (x=true and y=false) 상태가 관측되는지 여부를 체크하기 위해서는 (step=3 and x=true and y=false)가 되는 시스템 상태가 존재할 수 있는지 체크하면 된다. 출력 신호 y는 입력 신호 x에 로직으로 연결되어 있기 때문에 위 상태는 존재할 수 없게 된다. 반면 Program B를 PLC 구동 환경을 고려하여 automata로 표현 하면 [Figure 7]과 같다.



[Figure 7] Automata for program B

이러한 automata 시스템에서 (step=3 and x=true and y=false) 가 되는 시스템 상태는 존재하게 된다. 이 시스템에서 x의 변화가 논리적으로 y에 전달되기 위해서는 Step의 상태변화가 두 번 사이클 반복을 해야 하기 때문에 위 상태가 관측될 수 있다. 이러한

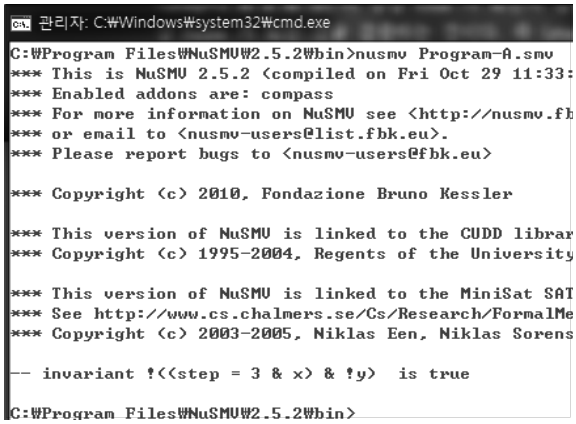
ISSN 1229-6783(Print)
ISSN 2288-1484(Online)

PLC 구동 환경을 고려하여 Program A를 검증하기 위한 SMV 변환코드는 결과적으로 [Table 2]와 같다.

검증 대상은 !(step=3 & x & !y) 이 항상 true 가 되는지 분석 하는 것이므로 (step=3 & x & !y)가 항상 false 가 되는 것을 검증하는 것이다. 즉 (step=3 & x & !y)가 true 되는 경우를 찾아보고, 만일 그런 경우가 생기지 않으면 이 시스템은 문제가 없는 것이다. 논리적으로 step=3 & x=true & y=false 인 상태를 찾아야 하는데 위 Program A는 그런 상태가 발생될 수 없다. Program A 에 대한 실제 SMV 분석결과는 아래 [Figure 8]과 같다.

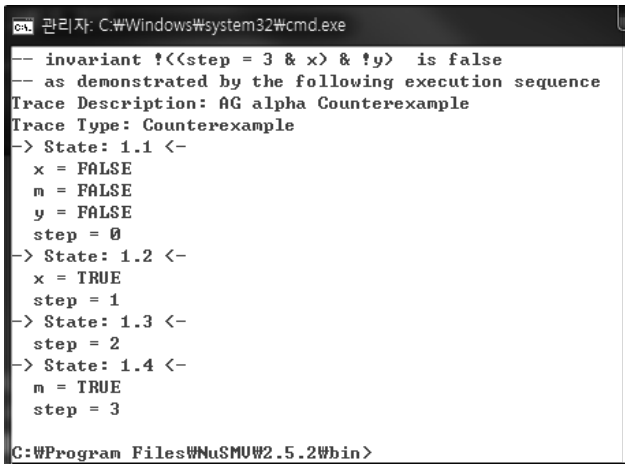
<Table 2> SMV code(Program A)

```
MODULE main
VAR
  x : boolean ;
  m : boolean;
  y : boolean ;
  step : 0..3;
ASSIGN
  init(x) :=FALSE;
  init(m) :=FALSE;
  init(y) :=FALSE;
  init(step) := 0;
  next(x) :=
    case
      step=0 : {TRUE,FALSE};
      step ] 0 & step [ 4 : x;
    esac;
  next(m) :=
    case
      step=1 & x : TRUE;
      step=1 & !x : FALSE;
      TRUE : m;
    esac;
  next(y) :=
    case
      step=2 & m : TRUE;
      step=2 & !m : FALSE;
      TRUE: y;
    esac;
  next(step) :=
    case
      step[ 3 : step+ 1;
      step=3 : 0;
      TRUE: step;
    esac;
INVARSPEC
        !(step=3 & x & !y)
```



[Figure 8] SMV verification (program A)

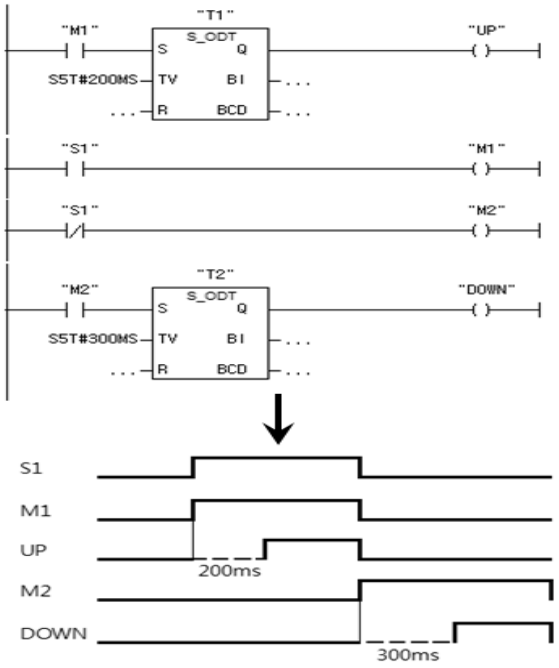
반면 Program B를 사용하여 되면 문제 상태가 발생된다. 실제 SMV 분석결과는 [Figure 9]와 같다.



[Figure 9] SMV verification (program B)

5. 타이머 시스템의 SMV 변환

타이머를 포함한 제어 프로그램을 검증하기 위해서는 실시간 시스템을 다룰 수 있는 정형모델이 필요하며, 이러한 경우 검증에 많은 부하를 주기 때문에 대규모의 생산시스템의 검증에 실시간 모델을 사용하는 것은 불가능하다. 그렇기 때문에 기존 연구에서는 타이머를 단순한 스위치로 간주하고 시간에 대한 부분을 생략하고 있다. 그러나 시간을 고려하지 않는 경우 검증 신뢰성을 손상시킬 수 있다.

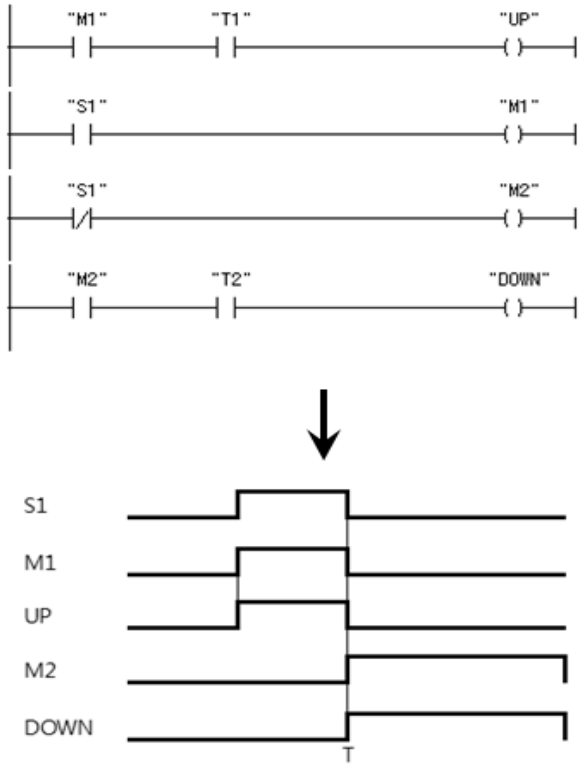


[Figure 10] PLC program with a timer

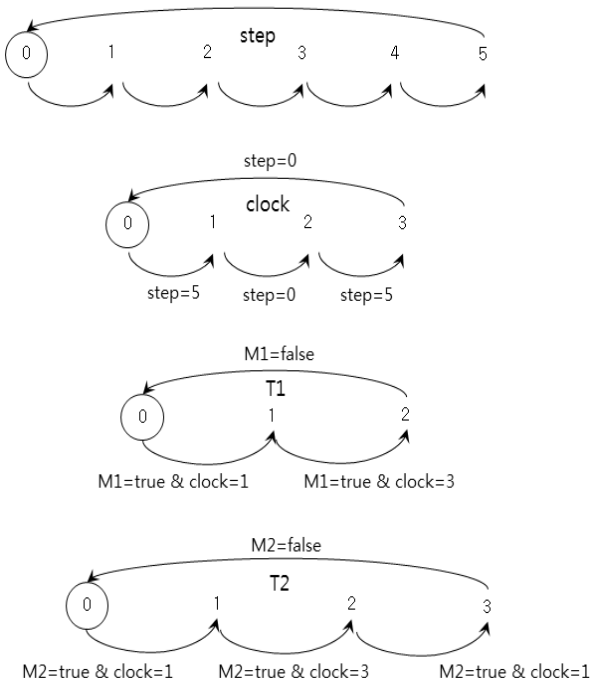
[Figure 10]의 예제 프로그램에서 타이머를 고려하는 경우에는 타임신호차트에서 보는 바와 같이 UP 및 DOWN 신호가 동시에 ON 되는 경우는 발생하지 않는다. PLC 프로그램은 SIEMENS STEP 7에서 작성된 것이며, S_ODT 함수블럭은 S신호가 ON되면 특정 시간이후에 Q신호를 ON 시켜주는 기능을 한다. [Figure 10] 프로그램에서 T1은 200ms, T2는 300ms 후에 작동한다.

[Figure 11]에서 보는 바와 같이, 기존 연구에서 사용한 방법인 타이머를 단순한 스위치로 간주한 경우 타임신호 상에서 UP 및 DOWN 신호가 동시에 ON 되는 경우가 시간 T 시점에서 잠깐 발생한다. 검증 시스템은 이 지점에서 특정 상황이 발생된다고 잘못된 검증 결과를 제시 하므로 사용자가 이러한 것들을 모두 체크해야 하므로 검증의 효율성을 떨어뜨린다.

본 연구에서는 타이머를 이산상태 모델로 표현하였으며, 타이머의 동작은 PLC 내부 클럭에 따라 변하도록 하고 내부클럭을 표현하는 clock 이라는 automata를 사용하였다. 다수의 타이머가 clock에 따라 동작하므로 타이머간의 동기화가 가능하다. 본 연구에서 기본 시간은 100ms 로 설정하였으며, 일반적으로 PLC 구동 사이클은 100ms 이내 이므로, 결과적으로 위 프로그램의 step, clock 및 타이머 T1과 T2에 대한 상태 변환 모형은 [Figure 12]와 같이 구현 가능 하다.



[Figure 11] Timer conversion into switch

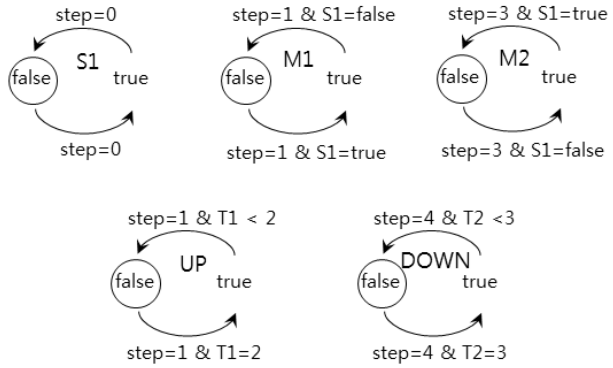


[Figure 12] Automata for step, clock, T1 and T2

위 상태 변환 모형에서 step의 상태가 마지막 step 5가 되면 100ms 가 지난 것으로 가정하고 이에 따라 clock 에서는 0 에서 1로 변환과 2에서 3으로 상태 변화는 각각 100ms 시간경과를 표현하게 된다. 타이

ISSN 1229-6783(Print)
ISSN 2288-1484(Online)

머 T1, T2는 구동신호와 clock에 따라 상태 변환이 발생된다. 결과적으로 [Figure 10]의 PLC 프로그램에서 입출력 변수에 대한 상태변환 모델은 [Figure 13]처럼 표현할 수 있다.



[Figure 13] Automata for PLC program with timers

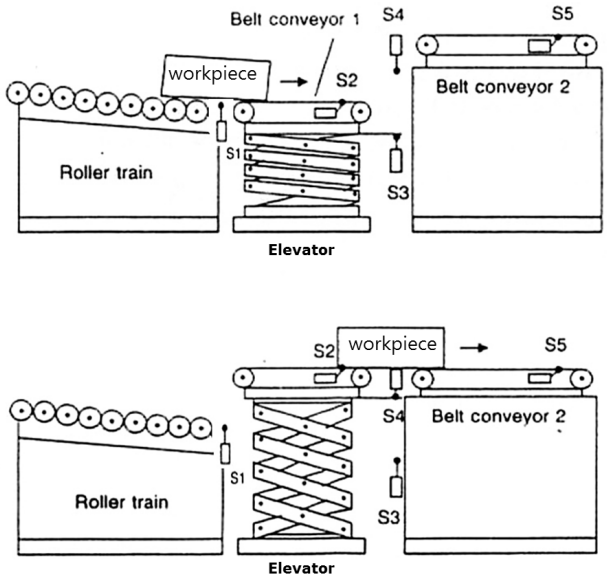
[Figure 13]에서 표현된 automata 의 상태변화 특성은 [Table 3]과 같으며, 보는 바와 같이 UP 신호와 DOWN 신호는 동시에 true 상태가 되지 않는다.

<Table 3> Automata state transition

state	step	clock	S1	M1	T1	UP	M2	T2	DOWN
1	0	0	1	0	0	0	0	0	0
2	1	0	1	1	0	0	0	0	0
3	2	0	1	1	0	0	0	0	0
4	3	0	1	1	0	0	0	0	0
5	4	0	1	1	0	0	0	0	0
6	5	1	1	1	1	0	0	0	0
7	0	2	1	1	1	0	0	0	0
8	1	2	1	1	1	0	0	0	0
9	2	2	1	1	1	0	0	0	0
10	3	2	1	1	1	0	0	0	0
11	4	2	1	1	1	0	0	0	0
12	5	3	1	1	2	0	0	0	0
13	0	0	1	1	2	0	0	0	0
14	1	0	1	1	2	1	0	0	0
15	2	0	1	1	2	1	0	0	0
16	3	0	1	1	2	1	0	0	0
17	4	0	1	1	2	1	0	0	0
18	5	1	1	1	2	1	0	0	0
19	0	2	0	1	2	1	0	0	0
20	1	2	0	0	0	0	0	0	0
21	2	2	0	0	0	0	0	0	0
22	3	2	0	0	0	0	1	0	0
23	4	2	0	0	0	0	1	0	0
24	5	3	0	0	0	0	1	0	0
25	0	0	0	0	0	0	1	0	0
26	1	0	0	0	0	0	1	0	0
27	2	0	0	0	0	0	1	0	0
28	3	0	0	0	0	0	1	0	0
29	4	0	0	0	0	0	1	0	0
30	5	1	0	0	0	0	1	1	0
31	0	2	0	0	0	0	1	1	0
32	1	2	0	0	0	0	1	1	0
33	2	2	0	0	0	0	1	1	0
34	3	2	0	0	0	0	1	1	0
35	4	2	0	0	0	0	1	1	0
36	5	3	0	0	0	0	1	2	0
37	0	0	0	0	0	0	1	2	0
38	1	0	0	0	0	0	1	2	0
39	2	0	0	0	0	0	1	2	0
40	3	0	0	0	0	0	1	2	0
41	4	0	0	0	0	0	1	2	0
42	5	1	0	0	0	0	1	3	0
43	0	2	0	0	0	0	1	3	0
44	1	2	0	0	0	0	1	3	0
45	2	2	0	0	0	0	1	3	0
46	3	2	0	0	0	0	1	3	0
47	4	2	0	0	0	0	1	3	1
48	5	3	0	0	0	0	1	3	1

6. 인터록 검증

[Figure 14]는 SMV를 이용한 PLC 제어프로그램의 인터록 기능 검증방법을 적용하기 위한 자동화 생산 시스템 도면이다.

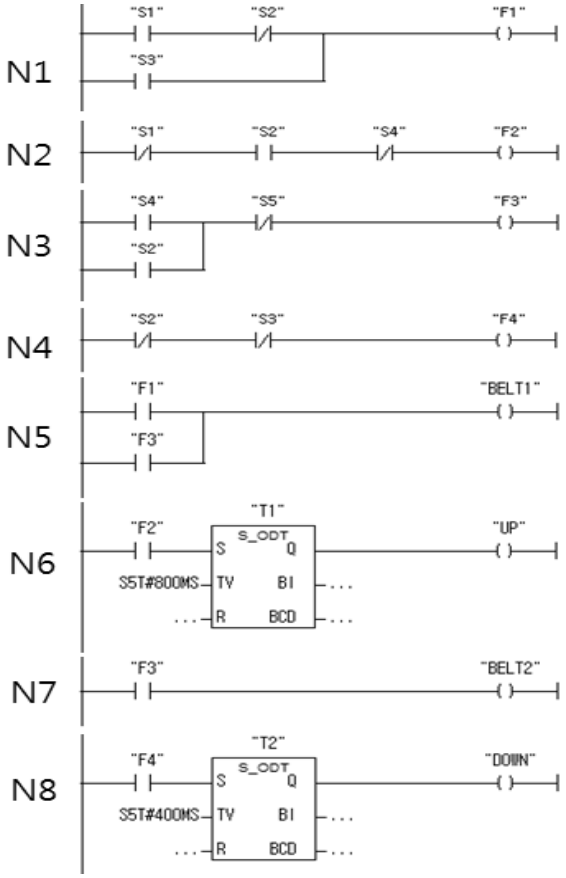


[Figure 14] An automation system

자동화 생산 시스템은 아래 순서(구동 1 -> 구동 2 -> 구동 3 -> 구동 4 -> 구동 5)의 동작을 반복적으로 수행한다. 즉 자동화 생산 시스템은 아래에 설명된 바와 같은 구동 순서를 갖도록 PLC 제어 프로그램에 의해 동작된다.

구동1: 롤러 트레인(Roller train)에 접근하는 작업물이 센서 S1에 감지되면 벨트 컨베이어 1(Belt conveyor 1)을 구동한다.
구동2: 작업물이 벨트 컨베이어 1(Belt conveyor 1)의 오른쪽 끝에 도달하여 센서 S2가 이를 감지하면 800ms 후에 엘리베이터(Elevator)를 상승한다.
구동3: 엘리베이터(Elevator)가 끝까지 상승하여 센서 S4가 이를 감지하면 엘리베이터(Elevator)의 상승을 정지한다.
구동4: 엘리베이터(Elevator)의 상승이 끝나면 벨트 컨베이어 1(Belt conveyor 1)과 벨트 컨베이어 2(Belt conveyor 2)를 동시에 구동하여 작업물을 벨트 컨베이어 2(Belt conveyor 2)의 오른쪽 끝으로 이동시킨다.
구동 5: 작업물이 사라지게 되어 센서 S2로부터 작업물 감지 신호가 없으면 400ms 후 엘리베이터(Elevator)를 하강한다.

위 시스템의 PLC 제어프로그램은 [Figure 15]와 같다.



[Figure 15] PLC program

위 PLC 제어프로그램은 센서 상태를 나타내는 센서 입력신호들(S1, S2, S3, S4, S5)와 자동화 생산 시스템의 상태를 나타내는 내부신호들(F1, F2, F3, F4), 마지막으로 자동화 생산 시스템의 설비들을 구동하기 위한 출력신호들(UP, DOWN, BELT1, BELT2)로 구성되어 있다.

내부신호의 상태 변화는 아래와 같다.

- 내부신호 F1은 작업물을 롤러 트레인(Roller train)에서 벨트 컨베이어 1(Belt conveyor 1)의 오른쪽 끝까지 이동시키는 자동화 생산 시스템의 상태를 표현하고, [(S1=ON AND S2=OFF) OR S3=ON] 조건을 만족할 때 온(ON) 상태가 된다(N1).
- 내부신호 F2는 엘리베이터(Elevator)를 이용하여 작업물을 위로 올리는 상태를 나타내며 [S1=OFF AND S2=ON AND S4=OFF] 조건을 만족할 때 온(ON) 상태가 된다(N2).
- 내부신호 F3는 두 개의 벨트 컨베이어 1, 2를 동시에 구동하여 작업물을 벨트 컨베이어 2(Belt conveyor 2)의 오른쪽으로 보내주는 상태를 표현하며,

ISSN 1229-6783(Print)
ISSN 2288-1484(Online)

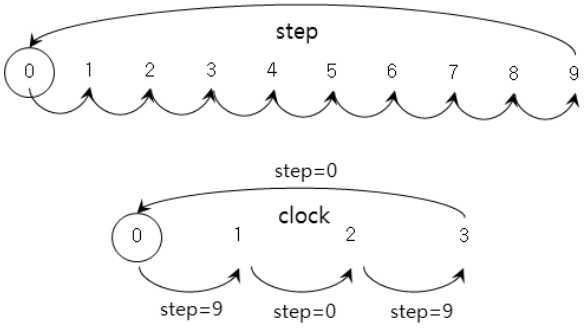
[(S4=ON OR S2=1) AND S5=OFF]조건을 만족할 때 온(ON) 상태가 된다(N3).
- 내부신호 F4는 엘리베이터(Elevator)를 하강시켜주는 상태를 정의하며, [S2=OFF AND S3=OFF]조건을 만족할 때 온(ON) 상태가 된다(N4).

출력 신호의 상태 변화는 아래와 같다.

- 내부신호 F1 혹은 F3가 온(ON) 상태가 되면 벨트 컨베이어 1(Belt conveyor 1)을 구동하기 위한 BELT1 출력신호가 온(ON) 상태로 된다(N5).
- 내부신호 F2가 온(ON) 상태가 되고 800ms 후에 엘리베이터(Elevator)를 상승하기 위한 UP출력신호가 온(ON) 상태로 된다(N6).
- 내부신호 F3가 온(ON) 상태가 되면 벨트 컨베이어 2(Belt conveyor 2)를 구동하기 위한 BELT2 출력신호가 온(ON) 상태로 된다(N7).
- 내부신호 F4가 온(ON) 상태가 되고 400ms 후에 엘리베이터(Elevator)를 하강하기 위한 DOWN 출력신호가 온(ON) 상태로 된다(N8).

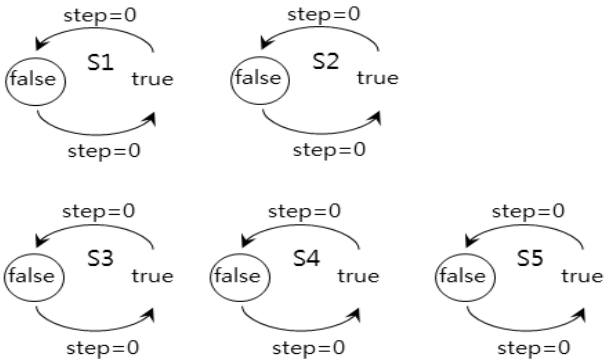
상기 UP 출력신호 및 BELT1 출력신호가 동시에 온(ON) 상태가 될 경우 작업물이 벨트 컨베이어 2(Belt conveyor 2)의 베이스에 부딪히기 때문에 이를 에러상황으로 정의할 수 있다. 따라서, 에러상황은 UP 출력신호 및 BELT1 출력신호가 동시에 온(ON)이 되는 경우이고, UP=ON AND BELT1=ON으로 나타낼 수 있다.

위 PLC 제어 프로그램을 SMV의 상태변환 시스템으로 변환하기 위하여 먼저 구동 사이클을 표현하는 step automata를 구성한다. 전체 프로그램이 8개의 네트워크로 구성되어 있으므로 [Figure 16]에서 보는 바와 같이 step은 0부터 9까지 총 10개의 상태로 구성되어 진다. clock은 100ms 시간 흐름을 나타내는 0 -> 1 상태 변화, 2 -> 3 상태 변화로 표현되어 진다.



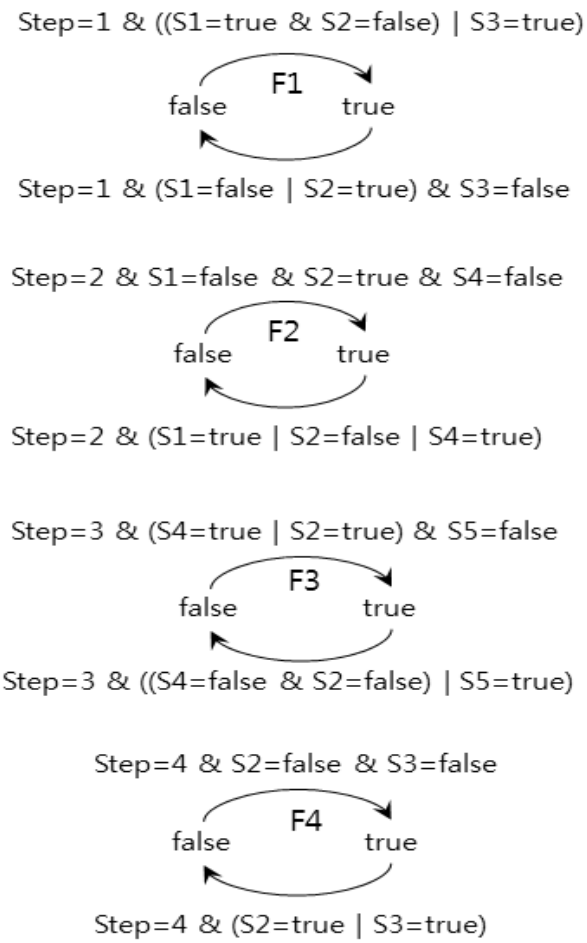
[Figure 16] Automata for step and clock

입력 센서는 step이 상태 0일 때, 임의로 변경될 수 있으므로 [Figure 17]과 같은 상태변환 시스템을 가지게 된다.



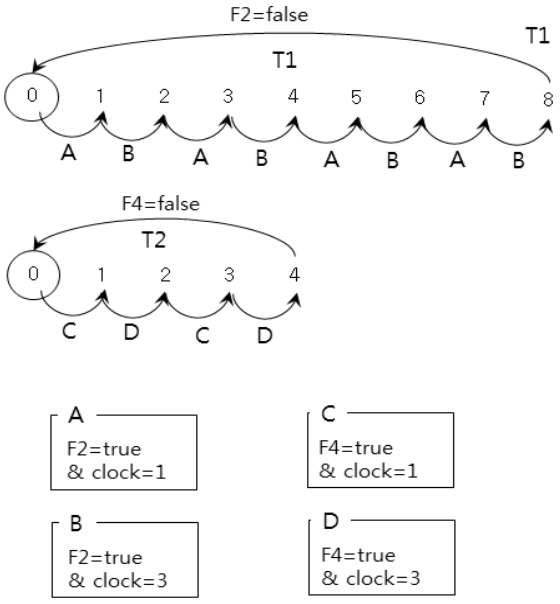
[Figure 17] Automata for input signal transition

전체 시스템 상태를 표현하는 상태변수 F1,F2,F3,F4 는 [Figure 18]과 같이 입력 센서의 상태에 의하여 결정된다.



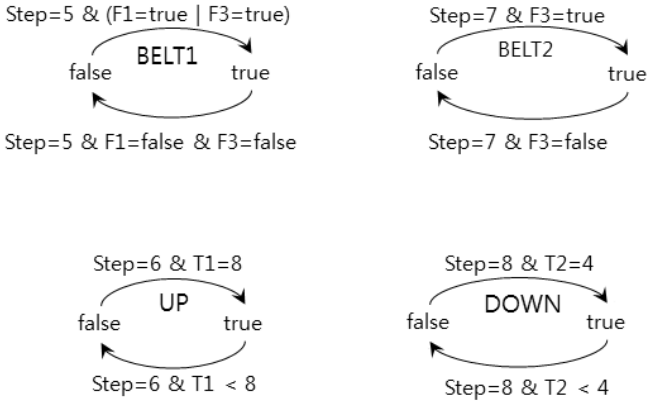
[Figure 18] Automata for internal signal transition

타이머 변수 T1, T2는 [Figure 19]와 같은 automata 로 표현될 수 있다.



[Figure 19] Automata for timer transition

출력변수의 변화는 [Figure 20]과 같은 특성으로 표현된다.



[Figure 20] Automata for output signal transition

예제 제어 시스템을 SMV 포맷으로 표현한 것은 <Table 4>와 같다.

<Table 4> SMV conversion for PLC program

```
-- Function name=FC 1
MODULE main
VAR
-- input symbols
 S1 : boolean;
 S2 : boolean;
 S3 : boolean;
 S4 : boolean;
 S5 : boolean;

-- output symbols
 BELT1 : boolean;
 BELT2 : boolean;
 UP : boolean;
 DOWN : boolean;
 F1 : boolean;
 F2 : boolean;
 F3 : boolean;
 F4 : boolean;
 T1 : 0..8;
 T2 : 0..4;

-- system symbols
 STEP : 0..9;
 CLOCK : 0..3;
 SPEC0 : boolean;

DEFINE
ASSIGN
 init(STEP) := 0;
 init(CLOCK) := 0;
 init(SPEC0) := FALSE;
 init(BELT1) := FALSE;
 init(BELT2) := FALSE;
 init(DOWN) := FALSE;
 init(F1) := FALSE;
 init(F2) := FALSE;
 init(F3) := FALSE;
 init(F4) := FALSE;
 init(T1) := 0;
 init(T2) := 0;
 init(UP) := FALSE;
 next(STEP) :=
  case
   STEP [ 9 : STEP+ 1;
   STEP = 9 : 0;
   TRUE : STEP;
  esac;
 next(CLOCK) :=
  case
   CLOCK=0 & STEP = 9 : 1;
   CLOCK=1 & STEP = 0 : 2;
   CLOCK=2 & STEP = 9 : 3;
   CLOCK=3 & STEP = 0 : 0;
   TRUE : CLOCK;
  esac;
 next(S1) :=
  case
   STEP=0 : {TRUE,FALSE};
   STEP ] 0 & STEP [= 9 : S1;
   TRUE : S1;
  esac;
 next(S2) :=
  case
   STEP=0 : {TRUE,FALSE};
   STEP ] 0 & STEP [= 9 : S2;
   TRUE : S2;
  esac;
 next(S3) :=
  case
   STEP=0 : {TRUE,FALSE};
   STEP ] 0 & STEP [= 9 : S3;
   TRUE : S3;
  esac;
 next(S4) :=
  case
   STEP=0 : {TRUE,FALSE};
   STEP ] 0 & STEP [= 9 : S4;
   TRUE : S4;
  esac;
```

ISSN 1229-6783(Print)
ISSN 2288-1484(Online)


```
next(S5) :=
 case
  STEP=0 : {TRUE,FALSE};
  STEP ] 0 & STEP [= 9 : S5;
  TRUE : S5;
 esac;
next(BELT1) :=
 case
  STEP=5 &  ( F1 | F3 )  : TRUE;
  STEP=5 & !( F1 | F3 )  : FALSE;
  TRUE : BELT1;
 esac;
next(BELT2) :=
 case
  STEP=7 &  F3  : TRUE;
  STEP=7 & !F3  : FALSE;
  TRUE : BELT2;
 esac;
next(UP) :=
 case
  STEP=6 & ( T1=8 )  : TRUE;
  STEP=6 & ( T1[8 )  : FALSE;
  TRUE : UP;
 esac;
next(DOWN) :=
 case
  STEP=8 & ( T2=4 )  : TRUE;
  STEP=8 & ( T2[4 )  : FALSE;
  TRUE : DOWN;
 esac;
next(F1) :=
 case
  STEP=1&((S1&!S2)|S3):TRUE;
  STEP=1&!((S1&!S2)|S3):FALSE;
  TRUE : F1;
 esac;
next(F2) :=
 case
  STEP=2&(!S1&S2&!S4):TRUE;
  STEP=2&!(!S1&S2&!S4):FALSE;
  TRUE:F2;
 esac;
next(F3) :=
  case
   STEP=3&((S4|S2)&!S5):TRUE;
   STEP=3&!((S4|S2)&!S5):FALSE;
   TRUE : F3;
  esac;
next(F4) :=
  case
   STEP=4&(!S2&!S3):TRUE;
   STEP=4&!(!S2&!S3):FALSE;
   TRUE : F4;
  esac;
next(T1) :=
  case
   F2&T1[8&T1mod2=0&CLOCK=1:T1+ 1;
   F2&T1[8&T1mod2=1&CLOCK=3:T1+ 1;
   !(F2) : 0 ;
   TRUE : T1;
  esac;
next(T2) :=
  case
   F4&T2[4&T2mod2=0&CLOCK=1:T2+ 1;
   F4&T2[4&T2mod2=1&CLOCK=3:T2+ 1;
   !(F4)  : 0 ;
   TRUE : T2;
  esac;

-- spec.
next(SPEC0) :=
 case
  STEP=9&BELT1=TRUE&UP=TRUE: TRUE;
  STEP=9&!(BELT1=TRUE&UP=TRUE): FALSE;
  TRUE : SPEC0;
  esac;
INVARSPEC
 !SPEC0
```

위 검증 모델을 SMV 로 실행한 결과 [Figure 21] 처럼 에러 상황이 발생되는 경우를 알 수 있다.

```
C:\Program Files\NuSMV\2.5.2\bin>nusmv fc1.smv
*** This is NuSMV 2.5.2 (compiled on Fri Oct 29
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nu
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <nusmv-users@fbk.eu>

*** Copyright (c) 2010, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD
*** Copyright (c) 1995-2004, Regents of the Univ

*** This version of NuSMV is linked to the MiniS
*** See http://www.cs.chalmers.se/Cs/Research/Fo
*** Copyright (c) 2003-2005, Niklas Een, Niklas

-- invariant !SPEC0  is false
```

[Figure 21] SMV verification of PLC program

검증 결과에서 S1=false, S2=true, S3=true 인 경우 BELT1 신호와 UP 신호가 동시에 true가 되는 것을 확인할 수 있었다.

7. 릴레이 검증 및 수정 기법

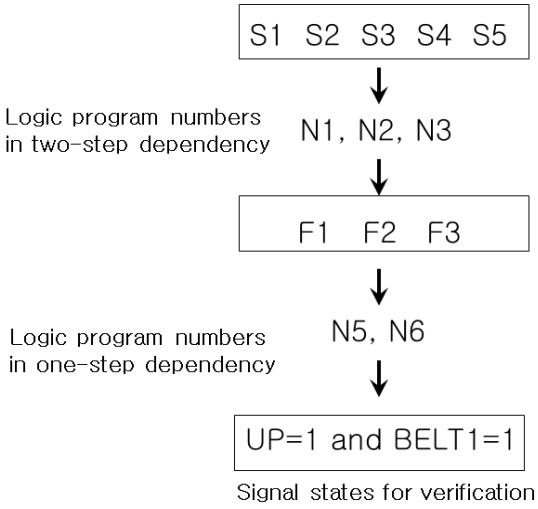
인터록 검증 예에서 사용된 PLC 제어 프로그램에서 BELT1 출력신호와 UP 출력신호가 동시에 온(ON) 상태가 되는 상황은 6가지 경우가 있으며, 이를 [Table 5]에 정리하였다.

<Table 5> 6 error cases

Signal	S1	S2	S3	S4	S5
CASE1	0	1	1	0	0
CASE2	0	1	1	1	0
CASE3	0	1	1	0	1
CASE4	0	1	1	1	1
CASE5	0	1	0	0	0
CASE6	0	1	1	0	0

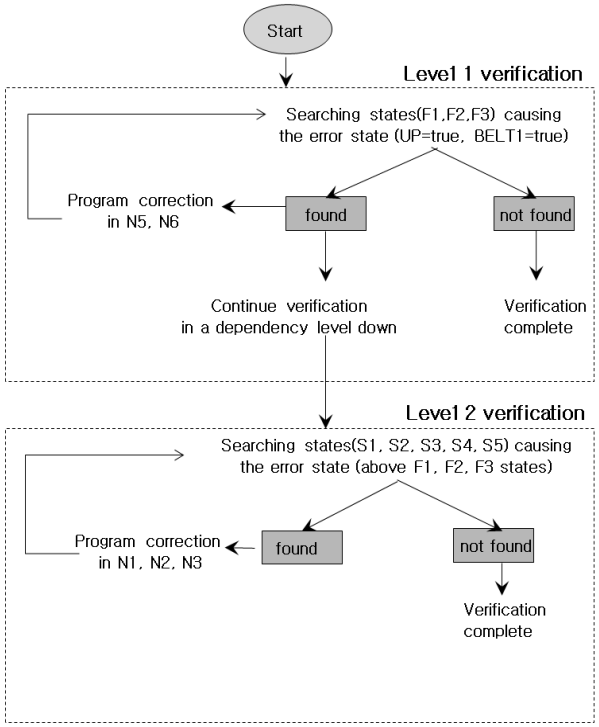
<Table 5>를 참조하면 BELT1, UP 출력신호가 동시에 온(ON) 상태가 되는 상황은 센서입력신호의 상태로 표현될 수 있다. 실제 SMV 도구를 이용하여 이러한 경우를 모두 찾아내는 것은 대형 PLC 제어 프로그램에서는 매우 어렵다. 설사 이러한 경우를 모두 찾아도 이것을 활용하여 PLC 제어 프로그램을 수정하는 것은 더욱더 어려운 일이다. 즉 [Table 5]에서 표현된 6가지 상황이 발생할 때에 BELT1 출력신호 및 UP 출력신호가 동시에 온(ON) 상태가 되지 않도록 PLC 제어프로그램을 수정해야 하는 것은 매우 어려운 일이다. 왜냐하면 PLC 제어프로그램의 어느 부분을 수정해야 하는지 알 수 없으며, 또한 수정할 수 있는 방법 또한 여러 가지가 있을 수 있기 때문이다. 이러한 어려움은 대형 자동화 생산시스템의 PLC 제어 프로그램 일수록 더욱 심해진다. 이에 따라서 본 연구에서는 SMV를 이용하여 에러 상황을 발생하는 출력신호들을 동시에 온(ON) 상태가 되게 하는 신호들의 상태들을 찾고, 신호 의존관계 계층구조와 논리회로 번호를 이용하여 에러 상황이 발생하지 않도록 PLC 제어 프로그램을 정정하는 방법론을 제시한다.

PLC 제어 프로그램은 설계특성상 다양한 조건 및 시퀀스(sequence)가 복합적으로 혼합되어 작성된다. 이러한 특성 때문에 특정 상태가 발생하는 것을 체크하기 위해서는 신호의 상호의존관계를 이용한 Tree구조를 만들고 이것을 바탕으로 릴레이 검증을 수행하면 전체 문제를 여러 개의 단위문제로 분할하여 검증하는 효과를 거둘 수 있다. 이러한 방법론은 상태폭발문제(탐색해야할 탐색공간이 너무 큼)에서 자유로워질 수 있게 한다.



[Figure 22] Dependency structure of program

[Figure 22]는 BELT1 출력신호 및 UP 출력신호가 동시에 온(ON) 상태인 것을 검증하기 위하여 BELT1 출력신호 및 UP 출력신호를 시작으로 의존관계 계층구조를 표시한 것이다. UP 출력신호 및 BELT1 출력신호와 연관된 로직은 N5, N6이며, 연관된 내부신호는 F1, F2 및 F3이다. 또한 내부신호 F1, F2, F3과 연관된 로직은 N1, N2 및 N3이며, 연관된 센서입력신호는 S1, S2, S3, S4 및 S5 이다. 이러한 구조를 가지는 PLC 제어프로그램의 검증은, UP 출력신호 및 BELT1 출력신호가 동시에 온(ON) 상태가 되는 경우를 방지하도록 프로그램을 검증하고 수정하는 것으로 이뤄질 수 있으며, [Figure 23]에서 도시된 알고리즘을 이용하면 PLC 제어 프로그램 검증 및 수정을 단순화할 수 있다.



[Figure 23] Relay verification procedure

검증절차는 다음과 같다.

[Figure 23]에서 레벨 1 검증방식은 제어 프로그램의 1차 의존관계만을 이용한 검증방식을 나타낸다.

1.에러 상황(UP=1 & BELT1=1)을 발생하는 내부신호 F1, F2, F3 상태조합을 SMV를 이용하여 찾아낸다.
2.[F1=1, F2=1, F3=0]일 경우 에러 상황이 발생하는 것으로 SMV가 알려준다.
3.프로그램 작성자가 프로그램 N5, N6을 수정하여 [F1=1, F2=1, F3=0]일 경우에 에러 상황이 발생하지 않도록 한다.

4.에러 상황을 발생하는 내부신호 F1, F2, F3 상태 조합이 발생하지 않을 때까지 위 절차를 반복한다.

UP, BELT1 출력신호의 상태를 결정하는 것은 내부신호 F1, F2, F3 신호이므로 PLC 제어프로그램 N5, N6에서 에러상황을 방지하도록 처리하는 것이 효율적이고 관리하기 쉽다. 실제 프로그램 작성자가 설계하는 PLC 제어프로그램은 이러한 형태를 가지는 것이 일반적이다. 그러나 만일 N5, N6 프로그램의 수정으로 이러한 에러 상황 방지가 불가능하거나 원천적인 에러 상황 방지가 목적인 경우 다음과 같은 검증 절차를 이용하여 종속관계를 레벨 다운하여 검증할 수도 있다. 즉, 레벨 2의 검증방식은 제어 프로그램의 2차 의존관계만을 이용한 검증방식이다.

1.레벨 1 검증에서 찾은 내부신호 상태 [F1=1, F2=1, F3=0]을 발생하는 센서입력신호 S1, S2, S3, S4, S5의 상태조합을 SMV를 이용하여 찾아낸다.

2.[S1=0, S2=1, S3=1]일 경우 [F1=1, F2=1, F3=0]이 발생하는 것으로 SMV가 알려준다.

3.프로그램 작성자가 프로그램 N1, N2, N3을 수정하여 센서 입력신호 [S1=0, S2=1, S3=1]일 경우에 [F1=1, F2=1, F3=0]이 발생하지 않도록 한다.

4.[F1=1, F2=1, F3=0] 상태가 발생되는 센서입력신호 S1, S2, S3, S4, S5의 상태가 발견되지 않을 때까지 위 절차를 반복한다.

8. 결론

본 연구에서는 생산시스템 안전성 진단에 활용할 수 있는 제어프로그램 정형검증 방법론을 제시하고 있다. 실제 생산라인 제어프로그램의 안전성 테스트를 수행하는 것은 비용과 시간이 많이 들고 완벽하지 않다. 이러한 단점을 극복하기 위하여 자동화된 검증 방법론이 필요하다.

본 연구에서는 유한상태시스템을 수학적으로 검증하는 SMV를 사용하여 생산시스템의 제어프로그램을 검증하는데 활용하였으며, 기존 연구가 제어 코드의 SMV 변환에 초점을 맞추고 있지만, 본 연구는 PLC 구동 특성 및 타이머 동작을 동일하게 SMV으로 근사화하는 방법론을 제시하여 신뢰도 높은 검증이 가능하도록 하였다. 또한 실제 산업에 활용되기 위해서 릴레이 검증 및 수정 기법을 제시하여 검증의 효율성을 높이고, 검증 부하를 줄일 수 있는 방법을 개발하였다.

9. References

[1] Lock-Jo Koo, Chang Mok Park, Chang Ho Lee, SangChul Park & Gi-Nam Wang(2011), "Simulation framework for the verification of PLC programs in automobile industries." International Journal of Production Research, vol 49(16), 4925-4925

[2] "NuSMV Tutorial." http://nusmv.fbk.eu/

[3] "A Tutorial on UPPAAL 4.0." http://www.uppaal.com/

[4] M. B. Younis and G. Frey(2003), "Formalization of existing PLC programs: A survey." Proc. Computing Eng. in Systems Applications, Lille, France, paper No. S2-R-00-0239.

[5] J. Lahtinen, J. Valkonen, K. Björkman, J. Frits, I. Niemelä, K. Heljanko(2012), "Model checking of safety-critical software in the nuclear engineering domain." Reliability Engineering & System Safety, vol 105, 104-113

[6] O. Rossi, Ph. Schnoebelen(2000), "Formal modeling of timed function blocks for the automatic verification of ladder diagram programs." Proc. 4th Int. Conf. Automation of Mixed Processes: Hybrid Dynamic Systems (ADPM), 177-182

[7] Tord Alenljung, Bengt Lennartson(2009), "Formal verification of PLC controlled systems using sensor graphs." IEEE conference on Automation science and engineering, 164-170

[8] Doaa Soliman, Georg Frey(2011), "Verification and validation of safety applications based on PLCopen safety function blocks." Control Engineering Practice vol 19(9), 929-946

[9] Min Zhou , Fei He, Ming Gu, Xiaoyu Song(2009), "Translation-based model checking for PLC programs." 33rd Annual IEEE International Computer Software and Applications Conference, 553-562

[10] 지은경, 전승재, 차성덕(2009), "함수 블록 다이어그램으로 구현된 PLC 프로그램에 대한 정형 검증 기법." 정보과학회 논문지 15(3), 211-215

저 자 소 개

박 창 목

아주대학교 산업공학과에서 학사, 석사, 박사학위를 취득하였고, 지식시스템 선임연구원, 유디엠텍 대표이사를 거쳐 현재는 인덕대학교 테크노경영학과 조교수로 재직 중이다. 관심분야는 생산최적화, 생산자동감시제어, 데이터마이닝 등이다.

주소 : 서울시 노원구 월계동 인덕대학교 테크노경영학과