SMV를 이용한 Structural Decision Table 명세의 정형검증

전승재° 지은경 차성덕
한국과학기술원 전자전산학과 전산학전공
{sjjeon° ekjee cha}@dependable.kaist.ac.kr

Formal Verification of Structural Decision Table Specification Using SMV

Seungjae Jeon°, Eunkyoung Jee, Sungdeok Cha
Div. of Computer Science, Dept. of EECS, KAIST and AITRC/IITRTC/SPIC

요 약

원자력 발전소의 제어 소프트웨어는 안전성이 중요시 되는 시스템이다. KNICS 컨소시엄의 APR-1400 RPS 개발 프로젝트에서는 시스템의 안전성과 품질을 높이기 위하여 요구사항을 NuSCR 정형명세로 기술하였다. 명세에 대한 분석을 위하여 SMV를 이용한 자동화된 정형검증 기법이 사용되는데, 본 논문에서는 테이블 형태의 명세인 SDT까지 그 범위를 확장하는 방법을 제안한다. 제안하는 방법의 효율성을 입증하기 위하여 실제 프로젝트에서 개발중인 시스템의 일부를 예제로 사용하였다.

1. 서론

원자력 발전소의 제어기는 안전성이 중요시되는 시스템으로서, 제어기에 사용되는 소프트웨어는 높은 수준의 품질을 요구한다. 최근 원자력 발전소의 시스템이 PLC (Programmable Logic Controller) 기반의 디지털 제어 장치로 대체되면서 소프트웨어의 안전성에 대한 중요성이 더욱 증가하고 있다. 따라서 개발 초기 단계에서부터 품질과 안전성을 보장하기 위한 정형검증 기법과 이를 지원하는 도구가 필요하다.

KNICS(원전계측제어시스템 개발사업단)의 APR-1400 (Advanced Power Reactor) RPS (Reactor Protector System: 노심보호시스템) 개발 프로젝트에서는 안전성을 향상시키기 위하여 정형명세 언어인 NuSCR[1]을 이용하여 요구사항 명세를 작성하고 있다. NuSCR로 기술된 시스템을 SMV 입력언어로 변환하여 정형검증을 자동화하는 분석 기법도 제시된바 있다[2]. 본 논문에서는 테이블 형태의 명세인 SDT (Structural Decision Table)까지 SMV 입력으로 변환하는 방법을 확장하고 활용 예제를 소개한다.

본 논문의 구성은 다음과 같다. 2장에서 NuSCR 언어와 SMV를 이용한 기존의 정형검증에 대한 연구를 간단히 소개한다. 3.1에서는 SDT를 SMV 입력으로 변환하는 방법을 설명한다. 3.1, 3.2에서 제안된 방법을 이용하여 ARP-1400 RPS 명세에 대해 완전성, 일관성 검사와 논리 검증을 수행하고 4장에서 결론을 맺는다.

2. 배경지식

2.1. NuSCR 언어

SCR (Software Cost Reduction) 방법론을 원자력 도메인에 적합하도록 확장한 정형명세 언어이다. NuSCR은 시스템의 명세를 효과적으로 기술하기 위하여 세 종류의 변수를 가진다. FSM (Finite State Machine)은 상태를 가지는 행위를 표현하기 위하여 쓰이고, TTS (Timed Transition System)는 여기에 시간적 제약 요소를 명세할 때 사용한다. SDT는 테이블 형태로 표현되어 조건에 따른 논리 연산을 기술한다. FOD (Function Overview Diagram)는 DFD (Data Flow Diagram)과 유사하며, 모든 변수들이 FOD 내부에서 사용되어 변수의 흐름과 입출력 관계, 명세의 계층 구조를 도식화한다.

2.2. NuSRS 도구

NuSRS는 연구 [2]에 대한 결과물로 구현된 도구로서, APR-1400 시스템의 NuSCR 정형명세 기술과 이에 대한 정형검증에 사용되고 있다. 명세 작성 기능, SMV를 이용한 정형검증 기능을 강화한 2.0 버전을 2005년 8월에 개발하였다.

2.3. SMV (Symbolic Model Verifier)

상태들의 집합과 이들 간의 전이 관계를 BDD (Binary Decision Diagram)로 표현하여 모델 체킹을 수행하는 도구이다. 하드웨어 디자인의 정형검증을 위해 주로 사용된다. SMV 입력 언어로 모델링 된 시스템이 temporal logic으로 기술된 특정 속성을 만족하는지 분석할 수 있다. NuSCR로 작성된 명세도 SMV를 이용하여 정형검증을 수행하며, FSM, TTS, FOD를 SMV 입력으로 자동 변환하는 템플릿이 고안되어 있다[2].

3. SDT에서 SMV 입력으로의 변환

SDT를 SMV 입력으로 변환하기 위해서는, 먼저 상태기계 형태로 변환하고 이를 기반으로 SMV 입력을 생성하는 두 단계를 거친다. 표 1의 SDT를 변환하여 보자.

Condition	1	2	3
f_HI_LOG_POWER_Val_Out > k_HI_LOG_POWER_PV_Max ... ①	T	-	F
f_HI_LOG_POWER_Val_Out < k_HI_LOG_POWER_PV_Min ... ②	-	T	F
Action	1	2	3
f_HI_LOG_POWER_PV_Err := true	O	O	
f_HI_LOG_POWER_PV_Err := false			O

표 1. SDT 정형명세 예제: f_HI_LOG_POWER_PV_Err

상태기계 형태로의 변환은 그림 1에 따른다. 초기 상태 _init은 완전성, 일관성 검사를 위하여 필요하다. 나머지 상태들은 action의 수만큼 만들어지며, _true 상태에 있을 때는 true 값이 출력됨을 의미한다. 표 1에서 조건 ① 또는 ②가 만족되면 true 값이 출력되므로, 모든 상태로부터 _true 상태로의 전이를 만들어준다. 이때의 전이 조건은 ① or ②가 된다.

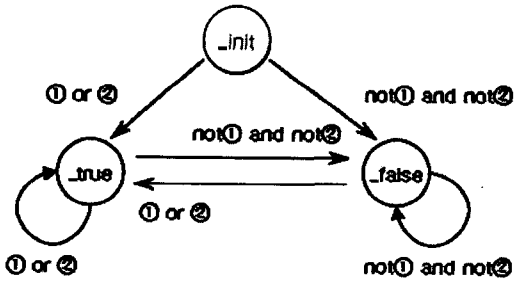

그림 1. 상태기계 형태로 변환된 모습

상태기계 형태에서 SMV 입력의 생성은 [2]에서 고안된 템플릿을 따른다. 일련의 변환 규칙은 NuSRS 2.0에 반영되어 있으며, 도구를 사용하여 f_HI_LOG_POWER_PV_Err를 SMV 입력으로 변환한 최종 모습은 부록에서 볼 수 있다.

3.2. 완전성, 일관성 (Completness, Consistency) 검사

정형명세는 완전하고 일관되어야 한다. SDT에서의 완전성과 일관성은 다음과 같이 정의된다[1].

완전성: 모든 condition들의 합집합이 모든 입력의 범위를 포함하여야 한다. 특정 입력은 적어도 어느 하나의 condition에는 속하게 된다.

일관성: 모든 condition 가운데 어느 둘도 교집합이 없어야 한다. 특정 입력은 하나의 condition에만 속하게 된다.

이를 그림 1에 대해서 다음과 같은 CTL 속성으로 표현할 수 있다:

완전성: *AG (in-_init -> AX ! in_init)*
어떤 경로를 취하더라도 _init 상태에 머물러 있는 경우는 없다. 어떠한 condition 중 하나에 속하게 되어 전이가 발생한다.

일관성: *AG ! (FROM-_init-TO-_true-taken && FROM-_init-TO-_false-taken)*
어떤 경우에도 _true, _false 상태로의 전이가 동시에 일어나지 않는다.

생성된 SMV 입력에 이 두 속성을 넣어 SMV 도구로 검증을 수행하였을 때 완전성과 일관성이 만족되면 true 결과를, 만족되지 않으면 반례를 보여준다. 그림 2는 f_HI_LOG_POWER_Val_Out 명세의 완전성 오류를 발견한 예제이다.

Condition	1	2	3	4
f_HI_LOG_POWER_MT_Query = 1	T	F	F	F
f_HI_LOG_POWER_AT_Query = 1	F	T	F	F
f_HI_LOG_POWER_PT_Query = 1	F	F	T	
Action	1	2	3	4
f_HI_LOG_POWER_Val_Out := f_HI_LOG_POWER_MT_Val	O			
f_HI_LOG_POWER_Val_Out := f_HI_LOG_POWER_AT_Val		O		
f_HI_LOG_POWER_Val_Out := f_HI_LOG_POWER_PT_Val			O	
f_HI_LOG_POWER_Val_Out := f_HI_LOG_POWER_PV				O

표 2. f_HI_LOG_POWER_Val_Out

	1	2
\FROM-_init-TO-0-enabled	0	-
\FROM-_init-TO-0-taken	0	-
\FROM-_init-TO-1-enabled	0	-
\FROM-_init-TO-1-taken	0	-
\FROM-_init-TO-2-enabled	0	-
\FROM-_init-TO-2-taken	0	-
\FROM-_init-TO-3-enabled	0	-
\FROM-_init-TO-3-taken	0	-
STATE	_init	_init
f_HI_LOG_POWER_AT_Query	1	
f_HI_LOG_POWER_MT_Query	0	
f_HI_LOG_POWER_PT_Query	1	
\in-_init	1	1

그림 2. 완전성 속성에 대한 반례

3.3. 논리 검증

APR-1400 시스템의 명세는 여러 개의 SDT, FSM, TTS가 모여서 이루어져 있다. SMV의 입력으로 변환된 각각을 하나의 SMV 프로그램에 포함시키면 여러 모듈에 걸쳐 있는 논리를 검증할 수 있다[2]. 다음은 FOD g_ChA_Pre_Perm_Gen에 대한 검증 예제이다.

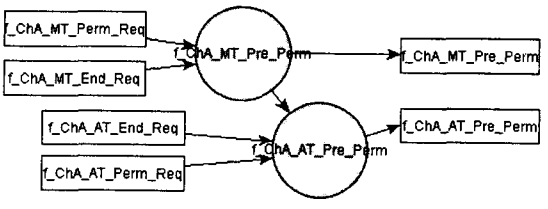

그림 3. g_ChA_Pre_Perm_Gen

Condition	2	3
f_ChA_MT_Pre_Req=1 & f_ChA_MT_End_Req=0	T	F
Action	2	3
f_ChA_MT_Pre_Perm := 1	O	
f_ChA_MT_Pre_Perm := 0		O

표 3. f_ChA_MT_Pre_Perm

Condition	2	3
f_ChA_AT_Perm_Req=1 & f_ChA_AT_End_Req=0 & f_ChA_MT_Pre_Perm=0	T	F
Action	2	3
f_ChA_AT_Pre_Perm := 1	O	
f_ChA_AT_Pre_Perm := 0		O

표 4. f_ChA_AT_Pre_Perm

그림 3에서 두 개의 SDT가 시스템을 구성하고 있음을 볼 수 있다. 각 세부명세는 표 3, 표 4에 기술되어 있다.

' f_ChA_MT_Pre_Perm 출력이 나온 직후 주기에는 f_ChA_AT_Pre_Perm이 나오지 않는다' 는 논리를 시스템이 만족하는지 분석하기 위해서는 다음 CTL 속성에 대해 SMV 검증을 수행한다.

AG (f_ChA_MT_Pre_Perm = 1
-> AX ! (f_ChA_AT_Pre_Perm = 1))

모듈의 수가 많아지고 복잡해지면 인스펙션을 통한 확인 & 검증이 어려워지는데 비해 자동화된 정형 검증을 이용하면 시스템이 특정 논리를 만족하는지 빠르고 정확하게 분석할 수 있다.

4. 결론

이 논문에서는 NuSCR 언어의 SDT 명세를 SMV 입력으로 변환하여 정형검증을 수행하는 방법을 제안하였다. 그 결과로 SDT 형태 정형명세의 완전성과 일관성을 검사하였고, 여러 모듈로 이루어진 시스템이 특정 속성을 만족하는지 검증할 수 있었다.

제안된 방법은 NuSRS 2.0에 구현되어 KNICS APR-1400 RPS 정형명세의 검증에 활용되고 있다. 분석을 거친 NuSCR 정형명세로부터 PLC 코드를 생성하고 검증하는 연구가 진행 중이다[3].

참고문헌

[1] 유준범, " Synthesis of Functional Block Diagram from NuSCR Formal Specification" , doctoral thesis, 2005

[2] 조재명, " NuEditor: An Environment for NuSCR Specification and Verification" , master' s thesis, 2003

[3] 신모범, 유준범, 차성덕, " VIS 검증기를 이용한 FBD 명세의 정형검증" , 한국정보과학회 한국컴퓨터종합학술대회 2005 논문집(B), pp.427-429, 2005

[4] Junbeom Yoo, Sungdeok Cha, Changhwoi Kim, Duckyong Song, " Synthesis of FBD-based PLC design from NuSCR formal specification" , Reliability Engineering and System Safety, 2003

부록. SMV 입력 변환 결과: f_HI_LOG_POWER_PV_Err

```
MODULE main
VAR
 f_HI_LOG_POWER_PV_Err : boolean;
 f_HI_LOG_POWER_Val_Out : 0..100;
STATE : {_init, _true, _false};
ASSIGN
init(STATE) := _init;
next(STATE) := case
 FROM-_init-TO-_true-taken : _true;
 FROM-_true-TO-_true-taken : _true;
 FROM-_false-TO-_true-taken : _true;
 FROM-_init-TO-_false-taken : _false;
 FROM-_true-TO-_false-taken : _false;
 FROM-_false-TO-_false-taken : _false;
 1 : STATE;
esac;
init(f_HI_LOG_POWER_PV_Err) := 0;
next(f_HI_LOG_POWER_PV_Err) := case
 FROM-_init-TO-_true-taken : 1;
 FROM-_true-TO-_true-taken : 1;
 FROM-_false-TO-_true-taken : 1;
 FROM-_init-TO-_false-taken : 0;
 FROM-_true-TO-_false-taken : 0;
 FROM-_false-TO-_false-taken : 0;
 1 : f_HI_LOG_POWER_PV_Err;
esac;
DEFINE
k_HI_LOG_POWER_PV_Max := 100;
k_HI_LOG_POWER_PV_Min := 0;
in-_init := STATE = _init;
in-_true := STATE = _true;
in-_false := STATE = _false;
FROM-_init-TO-_true-enabled := in-_init &
(( ( f_HI_LOG_POWER_Val_Out > k_HI_LOG_POWER_PV_Max ) ) ||
( ( f_HI_LOG_POWER_Val_Out < k_HI_LOG_POWER_PV_Min ) ));
FROM-_true-TO-_true-enabled := in-_true &
(( ( f_HI_LOG_POWER_Val_Out > k_HI_LOG_POWER_PV_Max ) ) ||
( ( f_HI_LOG_POWER_Val_Out < k_HI_LOG_POWER_PV_Min ) ));
FROM-_false-TO-_true-enabled := in-_false &
(( ( f_HI_LOG_POWER_Val_Out > k_HI_LOG_POWER_PV_Max ) ) ||
( ( f_HI_LOG_POWER_Val_Out < k_HI_LOG_POWER_PV_Min ) ));
FROM-_init-TO-_false-enabled := in-_init &
(!( f_HI_LOG_POWER_Val_Out > k_HI_LOG_POWER_PV_Max )
& !( f_HI_LOG_POWER_Val_Out < k_HI_LOG_POWER_PV_Min ));
FROM-_true-TO-_false-enabled := in-_true &
(!( f_HI_LOG_POWER_Val_Out > k_HI_LOG_POWER_PV_Max )
& !( f_HI_LOG_POWER_Val_Out < k_HI_LOG_POWER_PV_Min ));
FROM-_false-TO-_false-enabled := in-_false &
(!( f_HI_LOG_POWER_Val_Out > k_HI_LOG_POWER_PV_Max )
& !( f_HI_LOG_POWER_Val_Out < k_HI_LOG_POWER_PV_Min ));
FROM-_init-TO-_true-taken := FROM-_init-TO-_true-enabled;
FROM-_true-TO-_true-taken := FROM-_true-TO-_true-enabled;
FROM-_false-TO-_true-taken := FROM-_false-TO-_true-enabled;
FROM-_init-TO-_false-taken := FROM-_init-TO-_false-enabled;
FROM-_true-TO-_false-taken := FROM-_true-TO-_false-enabled;
FROM-_false-TO-_false-taken := FROM-_false-TO-_false-enabled;
```