실시간 모델 체커를 이용한 월성 SDS2의 명세 및 검증

지은경[+] ○ 홍형석[+] 차성덕[+]
[+]한국과학기술원 전산학과

Specifying and Verifying Wolsong SDS2 Using Real-Time Model Checker

Jee, Eun-Kyoung[+] ○ Hong, Hyoung-Seok[+] Cha, Sung-Deok[+]
[+]Dept. of Computer Science, KAIST

요 약

실시간 시스템은 그 특성상 시스템 내에서의 오류가 매우 위험하고 때론 치명적일 수 있다. 따라서, 시스템의 정확성과 안전성을 보장하는 것은 매우 중요한 문제가 된다. 이러한 문제를 해결하기 위해 정형기법을 사용한 명세와 검증을 통해 실시간 시스템의 안전성을 보장하려는 시도들이 많이 이루어져 왔다. 이 논문에서는 월성 원자로 운전 중지 시스템의 기존 명세의 문제점을 보완하여, 실시간 요구사항들을 타임드 오토마타(timed automata)로 기술할 것을 제안한다. 또한 명세를 실시간 모델 체커 UPPAAL의 입력으로 넣어서, 모델 체킹 기법을 이용해 자동적으로 시간 제약 속성을 검증한다.

1. 서론

비행기 제어기, 원자력 발전소 제어기, 산업 로봇 등 반응시간이 정확도에 중요한 영향을 미치는 시스템들을 실시간 시스템이라고 한다. 실시간 시스템들은 근본적 성질 때문에, 시스템 내에서의 오류가 매우 위험할 수 있으며, 때로는 치명적일 수 있다. 따라서, 복잡한 실시간 시스템의 정확성을 보장하는 것은 중요하며, 이를 위해서 정형기법(formal method)을 사용하여 시스템의 명세단계부터 시스템의 안전성을 확인하려는 노력들이 이루어지고 있다.

모델 체킹(model checking)은 병행 시스템(concorrent system)의 행위를 상태 전이 그래프(state transition graph)로 표현하고, 속성(property)은 시제논리(temporal logic) 식으로 기술하여 이들을 자동적으로 비교하는 방법이다. 모델 체킹은 유한 상태 시스템들의 자동 검증에 대한 가장 성공적인 기술 중의 하나로 여겨져 오고 있으며, SPIN[1], SMV[2]등의 여러 도구들도 개발되었다.

실시간 시스템들은 보통 어려운 시간 제약들을 만족해야만 한다. '원자력 발전소의 정지신호가 언젠가는 나온다.'고 말하는 것만으로는 충분하지 않다. 그 이벤트가 다른 이벤트들과 관련해서 언제 일어날 수 있는 지에 대한 상한과 하한이 있다. 반응 시스템 (reactive system)에 대한 시제 증명에 대한 전통적인 방법들은 시간에 대한 정성적인 성질들만 유지한 채,(예, 언젠가는 p가 만족한다.) 정량적인 시간에 대해서는 추상화했다. 그러나 후에 R.ALur의 몇 명의 저자들에 의해 실수 변역을 가지는 정량적인 시간 모델에 대한 모델 체킹 알고리즘이 제안되고 그 알고리즘이 결정적임이 증명되었다 [3]. 실수 변역을 가진 시간상에서의 모델 체킹은 상태 전이 그래프와 정량적인 시간 정보를 가진 논리식을 결합하는 것이다.

이 논문에서는 안전성이 중요한 실시간 시스템인 월성 원자력 발전소의 원자로 운전 중지 시스템(Shutdown System)을 명세하는데 있어서, 기존의 테이블 형태의 명세의 단점을 보완하여 시간 관련 요구사항들은 타임드 오토마타 (Timed Automata)[5]를 이용하여 명확히 기술할 것을 제안한다. 또한, 실시간 시스템에 대한 모델체킹 알고리즘을 사용하여 구현된 실시간 모델체커를 사용하여 시간 제약 속성을 검증한다. 실시간 모델체커로는 Aalborg 대학의 BRICS와 Uppsala대학의 Department of Computing Systems와의 협력 연구로 개발된 UPPAAL[4] 도구를 사용한다.

2. 예제 : 월성 제 2 원자로 운전 중지기

원자력 발전소 원자로 운전 중지기 시스템에서는 압력이나 파워등 입력 값들이 비정상적이거나 시스템의 주기적인 체크에서 이상이 생기면 그 결과로 트립(trip)신호를 내보내게 된다. 트립 신호를 내보냄으로써 핵 연쇄반응을 재빨리 종결시키고 반응기를 위험하지 않은 상태로 유지시키려는 목적이다. 월성 제 2 원자로 운전 중지기 시스템(Shutdown System 2)은 두개의 독립적인 원자로 운전 중지기 중 두 번째 것이다.

트립 파라미터에는 PDLTrip, PZLTrip, SLLTrip 의 세 가지가 있으며, 각각은 압력, 파워등의 입력값을 읽어 들여서 연산을 수행한 후, Trip또는 NotTrip의 결과값을 내보낸다. 트립 파라미터들 중, PZLTrip과 SLLTrip은 시간 제약을 포함하지 않지만, PDLTrip은 시간 제약을 포함한다. 이 논문에서는 월성 SDS2(Shutdown System 2)에서 특히 PDLTrip 부분을 중심으로 보고자 한다.

월성 SDS2의 요구 명세는, 기능 요구 명세를 자연어로 기술한 PFS(Program Functional Specification)와 소프트웨어 요구 명세를 테이블 형태로 기술한 SRS(Software Requirements Specification) 두 가지가 있다.

월성 SDS2의 요구사항을 크게 두 가지로 나눈다면 실시간 요구사항과 비실시간 요구사항으로 나눌 수 있는데, 먼저 비실시간 요구사항에 대한 부분 중 일부인 f_Flog(접두어 f_는 fuction을 의미한다)에 대한 부분을 보자. 자연어로 기술된 PFS에서 f_Flog부분을 간략히 보면 다음과 같다.

- Real the ion chamber log power signal for ϕ_{LOG}
- If the signal is irrational, set ϕ_{LOG} to a default value of 100%FP(Full Power)(default value : 4000mV)

이 부분에 대한 SRS명세는 표 1.에 보여진다.

CONDITION STATEMENTS			
w_FlogRange[m_Flog]	a	b	c
ACTION STATEMENTS			
f_Flog=4000	X		X
f_Flog=m_Flog		X	

표 1: f_Flog

Condition Macros : w_FlogRange[m_Flog]

a : m_Flog $\leq$ 200
b : 200 < m_Flog < 4088
c : m_Flog $\geq$ 4088

비실시간 요구사항의 명세는, 예에서 보는 것처럼 테이블을 통해 수학적 함수로서 기술되고 있다. w_FlogRange[m_Flog]가 a이면 즉, m_Flog의 값이 200이하이면 f_Flog함수의 값은 디폴트 값인 4000이 되고, m_Flog의 값이 200과 4088사이이면, f_Flog의 값은 m_Flog의 값으로 정해진다. 시간 제약을 포함하지 않은 비실시간 요구사항의 경우는 이렇게 수학적 함수 시맨틱을 가진 테이블 명세로 정확하게 기술할 수 있다.

이번엔 실시간 요구사항중의 일부인 PDLTrip에 대한 부분을 보자. PDLTrip은 트립을 일으킬 수 있는 세 가지 경우의 함수값을 입력으로 받는데, 그 중에서 시간 제약에 관계된 PDLDly에 대한 부분만 보도록 한다. PDLDly는 입력값을 기준치와 비교한 후, 어느 정도의 딜레이를 가진 후에, 다시 체크하도록 하는 트립 요소이다. PFS에서 PDLDly에 대해 기술한 부분을 보면 다음과 같다.

If the pressure is the below the setpoint and the power exceeds 70%FP,

- Continue normal operation for [2.7,3.0] seconds.
- If the power still exceeds 70%FP, open the trip
- Once the delayed parameter trip has occur, keep the parameter trip D/O open for one second(± 0.1 seconds)
- Close the parameter trip D/O once all pressure signals are above the delayed trip setpoint or power is below 70%FP

SRS에서 이와 같은 부분을 기술한 것은 다음과 같다.

CONDITION STATEMENTS				
t_Trip	T	F	F	F
s_PDLDly = k_InDlyTrip	-	T	T	F
ANY(i=1 . . . 4, f_PDLSnrDly[i]=k_SnrTrip)	-	T	F	-
ACTION STATEMENTS				
f_PDLDly=k_InDlyTrip	X	X		
f_PDLDly=k_InDlyNorm			X	X

표 2: f_PDLDly

t_Trip은 시간 함수로서 t_Trip은 그 안에 t_Wait,t_Pending과 같은 또 다른 시간 함수를 포함한다. 시간 함수들은 기본적으로 t_Wait 함수에 기반하는데, 이들 시간 함수들은 다소 반정형적(semi-formal)으로 정의되었다. 예를 들어, 'A조건이 만족하면 2.7초에서 3.0초간 그 상태에 머물러 있어야 한다.'를 표현하기 위해서 '2.85초 전에 A라는 조건이 만족되는 순간이 있으면 현재 참이다.' 는 의미의 식을 정의하고, '2.85초에 대한 시간 허용 오차는 0.15초이다' 라고 자연어로 덧붙이는 방법을 사용하였다. 비결정적인 시간 간격을 수학적 함수를 사용해서 나타내는 한계에 의해, 시간 함수를 정형적으로 정의하지 못하고 반정형적으로 정의한 방식이다. 명세를 완전한 정형식으로 기술하지 못하면, 명세가 포함하는 비정형적인(informal) 부분들 때문에 정형적으로 분석하고 자동 검증하는 것이 어려워지게 된다. 따라서, 정형적 분석이나 자동 검증을 하려고 한다면 명세의 정형적 기술은 필수적이다.

정형 검증의 어려움에 덧붙여 월성 SRS의 단점은 타이머 함수를 사용해 시간 제약을 명세한 것이 다소 복잡하고 이해하기 어렵다는 것이다. 위의 예에서 PDLDly의 요구사항이 PFS에서는 몇줄의 간단한 자연어 명세로 기술되지만, 함수 형태의 SRS로 나타내기 위해서는 두개의 함수와 세개의 타이머 함수가 사용된 것을 볼 수 있다. 또한 표 2의 f_PDLDly처럼 타이머 함수들을 포함한 테이블들은 대체적으로 직관적으로 이해하기 힘든 면을 가지고 있다.

이렇게 실시간 요구사항을 기술하는데 있어서 월성 SRS 명세가 갖는 한계와 단점을 본 논문에서는 시간 요구사항을 타임드 오토마타로 표현함으로써 해결하고자 한다.

3. 실시간 모델체커를 이용한 명세 및 검증

3.1 타임드 오토마타를 이용한 명세

타임드 오토마타(Timed Automata)[5]는 오토마타에 시간 제약을 추가한 것인데, 이 시간 제약들은 실수값을 가질 수 있는 여러 개의 클럭들을 사용하여 표현된다. 타임드 오토마타의 예가 그림 1에 보여진다.

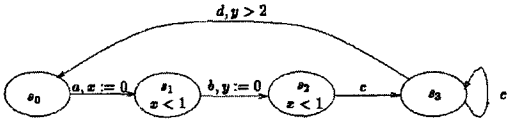

그림 1: 타임드 오토마타

여기서 x와 y는 클럭(clock)변수로서, 처음에는 모두 0에서 시작하며 같은 속도로 진행해 나간다. s_0상태에서 a라는 이벤트가 발생하면 x클럭을 0으로 리셋(reset)시킨 후에, s_1상태로 전이가 일어나게 된다. s_1 상태에서는 클럭 x의 값이 1보다 작아야 한다는 불변식이 만족되어야 하고, 이 조건이 만족되지 않으면 s_1상태를 벗어나게 된다. 이와 같이 클럭 변수들을 사용해서 오토마타에 시간 제약을 추가한 것이 타임드 오토마타이다.

본 논문에서는 UPPAAL 도구에서 지원하는 *System Editor*를 통해 타임드 오토마타로 월성 시스템의 일부를 명세하였다. UPPAAL에서는 타임드 오토마타에서 클럭 변수들 뿐만 아니라 데이타 변수들도 제약식에 사용될 수 있도록 확장된 타임드 오토마타를 모델로 삼는다[4]. 2장에서 기술하였던 PDLDly 부분의 PFS를 보고, 확장된 타임드 오토마타로 명세한 예는 그림 2와 같다.

네개의 센서를 통해 입력받은 압력값(m_PHTD)이 비정상이거나, 파워 (f_FaveC)가 임계값(k_FaveCPDL) 을 넘으면 PDLDly는 Normal에서 Waiting 상태로 가게 된다. z1이란 클럭을 리셋(reset) 시킨 후에, 27에서 31사이의 시간이 지난 후, 파워(f_FaveC)의 값이 여전히 비정상적이면 f_PDLDly의 값은 k_InDlyTrip 이 된다. 원래의 명세에서는 시간 제약이 [2.7,3.0]와 같이 실수로 표현되지만, 여기서는 모두 정수로 표현하였다. 실시간 모델 체킹을 가능하게 하기 위해 타임드 오토마타와 모델 체킹의 검증 속성에서 정수만을 허용하기 때문이다. 따라서, 월성 예제에서의 2.7초를 27로, 3.0초를 30으로 표현하였다. k_InDlyTrip은 Delayed Trip이 발생한 경우를 나타내는 상수이다.

이 시스템은 DlyTrip상태에서 9에서 11의 시간(0.9초에서 1.1초 사이의 시간 간격)이 경과된 후에 Pending상태에서 압력이나 파워

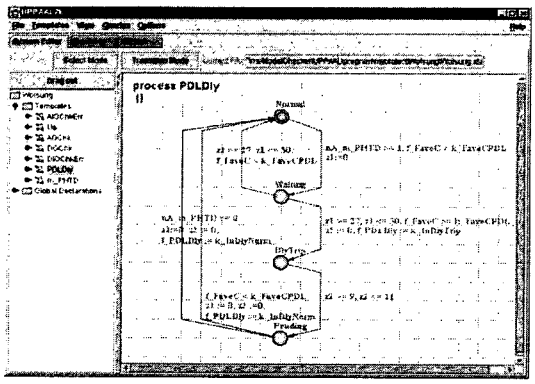

그림 2: 타임드 오토마타로 명세한 PDLDly 부분

를 다시 체크해서, 정상으로 돌아왔다면 Normal상태로 가게 된다. 이 때, f_PDLDly의 값은 정상을 나타내는 k_InDlyNorm으로 부여된다.

SRS에서 세 개의 타이머 함수와 두 개의 기능 함수로 복잡하게 표현되었던 것이 타임드 오토마타를 사용함으로 직관적이고 이해하기 쉽게 표현될 수 있으며, 시간제약을 정확히 표현할 수 있다.

이런 식으로 월성 SDS2시스템에서 시간에 관련된 여섯개의 부분을 타임드 오토마타로 명세하고, 시뮬레이션 시킨 결과가 그림 3과 같다.

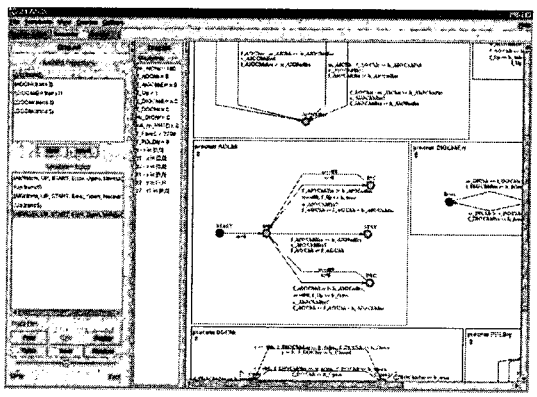

그림 3: 월성 시스템 일부의 시뮬레이션

3.2 실시간 모델체커를 이용한 검증

PDLDly에 대한 간단한 시간 요구사항을 실시간 모델체커를 통해 검증하였다. PDLDly를 명세한 타임드 오토마타에는 클럭 변수가 세 개 사용되었다. z,z1,z2 인데, z1은 압력과 파워가 둘다 비정상임을 체크한 후에 2.7에서 3.0초간 기다리는 조건을 명세하기 위한 클럭이고, z2는 delayed parameter trip이 일어난 후에, parameter trip D/O(Digital Output)을 1.0초간 유지해야 함을 명세하기 위해서 사용된 클럭이다. Normal상태를 지날 때, 0으로 reset된 z 클럭은 다시 Normal상태로 오기 전까지 계속 진행되는데, Pending 상태에 왔을 때에, z는 최소 $2.7 + 0.9 = 3.6$초 이상 되어야 한다. 이 속성을 다음과 같은 식으로 검증하였다.

$\forall\Box(PDLDly.Pending \Rightarrow z \geq 36)$

이 식은 UPPAAL의 syntax로 다음과 같이 표현된다.

```
A[ ] ( PDLDly.Pending imply z>=36 )
```

'PDLDly의 Pending상태에 있다면 항상 z는 36이상이다.'를 의미하는 시제 논리식이다. 월성 모델과 이 속성에 대해 모델 체킹을 하면, "Property is satisfied" 라는 메세지를 출력하면서 모델에서 위의 속성이 만족됨을 보여준다. 이 속성은 아주 간단한 것이지만, 시스템의 안전성을 검증하기 위한 더 많은 복잡하고도 유용한 속성들이 검증될 수 있다. 속성이 만족되지 않은 경우에 대해서는 UPPAAL이 제공하는 counter example까지의 시뮬레이션을 보여주는 기능을 사용하여 에러를 찾아내는데 도움을 얻을 수 있다.

4. 결론

본 논문에서는 월성 SDS2 시스템의 기존 명세에서, 실시간 요구사항을 이해하기 쉽고 정형적으로 표현하기 위해, 실시간 요구사항에 관한 부분을 타임드 오토마타로 모델링하는 방법을 제안하였다. 또한, 실제적인 시스템의 명세를 실시간 모델 체커 UPPAAL을 통해 모델링하고 검증함으로써 시스템이 요구사항을 만족함을 자동적으로 검증할 수 있도록 하였다. 간단한 실시간 제약을 나타내는 속성을 가지고, 실제 실시간 시스템에 대한 모델 체킹을 이용한 검증이 성공적으로 이루어짐을 보였다.

안전성이 중요한 실시간 시스템들에 대한 시간 제약을 명세하기 위해서 타임드 오토마타를 사용하여 모델하는 것은 이해하기 쉽고 정확한 명세를 가능하게 한다. 실시간 모델 체커 UPPAAL을 사용한 검증은 타임드 오토마타를 기반으로 한 시스템에 대해 시간 제약을 나타내는 속성들을 검증하는데 있어 자동화되고 효율적인 방법을 제공한다.

참고 문헌

[1] Gerald J. Holzmann, Design and Validation of Computer Protocols, Prentice Hall, 1991

[2] Kenneth L. McMillan, Symbolic Model Checking, Kluwer Academic Publisher, 1993

[3] R.Alur and C. Courcoubetis and D.L.Dill, Model-checking in dense real-time, Information and Computation 104(1), p.2-34, 1993, preliminary version appeared in Proc. 5th LNCS, 1990

[4] Johan Bengtsson, Kim G. Larsen, Fredrik Larsson, Paul Pettersson and Wang Yi, UPPAAL-a Tool Suite for Automatic Verification of Real-Time Systems, In Proceedings of the 4th DIMACS Workshop on Verification and Control of Hybrid Systems, New Brunswick, New Jersey, 22-24 October, 1995

[5] R.Alur. Timed Automata. To appear in NATO-AI 1998 Summer School on Verification of Digital and Hybrid Systems. A revised and shorter version appears in 11th International Conference on Computer-Aided Verification, LNCS 1633, pp.8-22, Springer-Verlag, 1999.