하이브리드 오토마타 기반 실시간 시스템의 모니터링 기법

심재환○ 김진현 최진영
고려대학교 컴퓨터통신공학부
jhsim○@formal.korea.ac.kr jhkim@formal.korea.ac.kr choi@formal.korea.ac.kr

Monitoring Method for Real Time System based on Hybrid Automata

Jae-Hwan, Sim○ Jin-Hyun, Kim Jin-Young, Choi
Department of Computer and Communication Engineering, Korea University

요 약

컴퓨터 시스템의 디지털화가 진행됨에 따라 많은 시스템들의 거동이 이산적인 속성과 연속적인 속성이 혼재되어 있다. 이러한 시스템의 특성을 정확히 반영하기 위해서는 모델링 단계부터 이산적인 속성과 연속적인 속성을 반영할 수 있는 하이브리드 오토마타 기반의 모델링이 요구 되어 진다. 대표적으로 자동차, 항공 우주, 원자력 등 제어 시스템 분야를 예로 들 수 있다. 이런 분야는 특히 안정성 및 실시간성이 매우 중요하여 명세로부터 구현에 이르기까지 검증과 확인의 단계가 매우 중요하다. 또한 개발 단계에서 뿐만이 아니라 시스템의 운영 단계에서도 지속적인 모니터링이 필요하다 본 논문에서는 실시간 시스템의 운영 중에 하이브리드 오토마타 기반의 명세와 구현의 일치성 여부를 확인 할 수 있는 모니터링 기법에 관해 연구하였다.

1. 서 론

컴퓨터 하드웨어 소프트웨어의 기술의 발전으로 다양한 분야의 아날로그 시스템을 디지털 시스템으로 대체가 이루어지고 있다. 기존의 아날로그 시스템이 디지털화 되어 가면서 많은 시스템들의 거동이 이산적인 속성과 연속적인 속성이 혼재된 특징을 보인다. 이러한 현상은 특히 제어 분야에서 많이 찾아 볼 수 있는데, 자동차 엔진제어, 제동장치제어 및 항공기 분야 등이 그 예라고 할 수 있다. 이러한 분야는 기존의 연속상태 모델링 방식 혹은 이산상태 모델링 방식만으로는 정확한 거동을 표현하기 어렵다. 이러한 이유로 세계적으로 하이브리드 시스템 또는 하이브리드 모델링이라는 분야로 활발한 연구가 진행 중이다. 특히 제어 분야를 포함한 하이브리드 모델링의 대상이 되는 분야의 시스템들은 실시간성의 보장과 안정성의 보장이 매우 중요하다는 특성을 가지고 있다. 따라서 하이브리드 시스템의 경우 검증 및 확인 방법에 대한 많은 연구들이 진행되고 있다. 수학적인 모델링을 통해 시스템을 개발한 경우 그림 1에서와 같이 크게 세 단계의 검증(Verification) 및 확인(Validation) 절차를 거치게 된다. 첫 단계에서 명세(Specification)가 원하는 속성(Property)를 만족하는지 검증(Verification)을 하게 된다. 이때, 모델체킹 혹은 정리증명 등의 방법이 이용되어 질 수 있다. 다음 단계에서 명세와 구현(Implementation)이 일치하는지 일치성(Conformance)을 확인하게 된다. 이러한 방법들은 일치성관계(Conformance Relation)[1]을 이용한다. 마지막으로 시스템의 구현이 원하는 속성을 만족하는지를 시험하게 된다. [1]에서 LTS(Labelled Transition System) 기반의 Conformance Relation이 정의되어 일반적인 이산시스템의 일치성 확인에 사용되어져 왔다. 이후 실시간 시스템의 일치성 확인을 위해 Timed Automata 기반의 Conformance Relation[2][3]이 이용되었고, 최근 연속상태와 이산 상태가 혼재된 하이브리드 시스템의 일치성 확인을 위한 연구들이 진행되어 오고 있다. 기존의 연구들은 Conformance Relation을 태스팅을 위해 이용해 왔지만[4][5], 본 논문에서는 하이브리드 시스템의 일

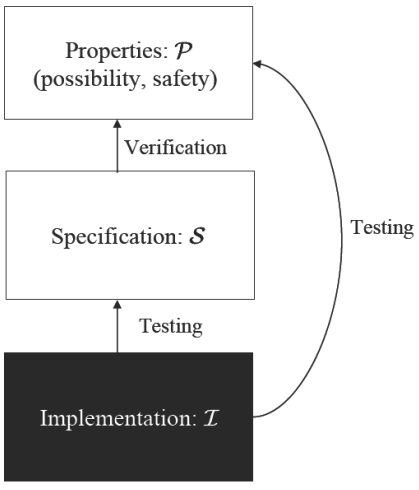

그림 1 시스템의 검증 및 확인 기법

치성 검사를 위한 hioco relation을 태스팅이 아닌 모니터링에 이용하여 하이브리드 오토마타(Hybrid Automata)[6]로 명세 된 시스템의 태스크를 모니터하기 위한 연구를 진행하였다. 그 이유는 기존의 연구들이 아래와 같은 제약사항을 가지고 있기 때문이다.

▪ 대부분의 실시간 시스템은 일회적인 작동을 하고 종료 되는 것이 아니라 계속적으로 환경과 반응을 하며 수행되는 반응형 시스템이다. 따라서 일정 수 이상의 테스팅만을 가지고 그 시스템의 구현이 명세와 일치하는지 확인하기 어렵다.

▪ 많은 연구들이 실시간 시스템을 시스템 전체적인 관점에서 시간 제약을 분석한다. 그래서 커널과 태스크 사이의 상호작용에 대한 Conformance는 분석할 수 없다.

위와 같은 이유로 본 논문에서는 일치성검사(Conformance Testing)를 실시간 태스크의 모니터링에 접목 시키려 한다. 앞으로 제시될 내용은 다음과 같다. 2장과 3장에서는 배경지식으로 하이브리드 오토마타와 Conformance Relation에 대해 언급 될 것이다. 4장에서는 제시 하려는 실시간 태스크의 모니터 구성에 대해서 언급하고, 5장에서 결론을 짓겠다.

2. Hybrid Automata

Hybrid System은 이산적인 속성와 연속적인 속성을 동시에 가지고 있는 시스템이다. Micro Processor를 가지고 연료 주입량을 조절하는 자동차 엔진이 좋은 예이다. 이러한 특성을 표현하기 위해 제안된 정형화 모델이 Hybrid Automata 이다. 본 논문에서 실시간 시스템의 명세와 테스팅에 사용될 Hybrid Automata HA는 다음과 같다.

Definition 2.1 (HA) Hybrid Automata HA는 다음과 같은 구성 요소로 이루어진다.

- variables : $X=\{x_1,x_2\cdots,x_n\}$는 연속적인 실변수의 집합이다. 여기서 n은 HA의 차원이다. 이 변수들은 시스템의 연속적인 상태변화를 위해 이용되어진다. 변수집합 $\dot{X}=\{\dot{x_1},\dot{x_2}\cdots,\dot{x_n}\}$은 변수들의 미분형태를 표현하기 위해 사용되어진다.
- Control Graph : HA는 방향성을 가진 multigraph (V,E)의 형태로 표현되어진다. $v\in V$인 Vertex는 제어 모드라고 부르고, $e\in E$인 edge는 제어전환이라고 부른다.
- Initial, invariant, flow condition : 세 개의 함수 $init$, inv, $flow$는 각각의 $v\in V$에 연결된 함수로서, $init(v)$는 v에서의 연속변수 $x\in X$를 초기화 시키는 함수이고, $inv(v)$는 v에서의 연속변수 $x\in X$가 지닐 수 있는 범위를 서술한다. $flow(v)$는 $X\cup\dot{X}$의 형태로 서술되며, 시스템의 동적인 변화를 표현한다. $flow(v)$은 일반적인 미분방정식의 형태로 표현된다.
- Jump Condition : 함수 $jump$는 제어전환 $e\in E$에 연결된 함수로서, $jump(e)$는 Control Graph에서 제어전환을 위한 조건으로 이용된다.
- Event : $event: E\rightarrow\Sigma$는 edge labeling 함수로서 제어전환을 야기하는 이벤트를 표현하는데 사용 된다. 이때, Σ는 이벤트의 집합이다.

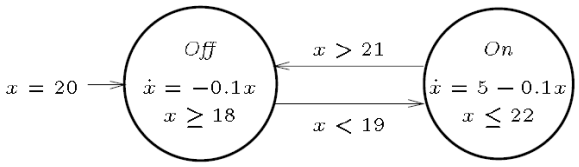

그림 2 Thermostat

위의 그림 1은 단순한 Hybrid Automata의 예제로서 Thermostat을 모델링하고 있다. 변수 x는 온도를 표현하고 On과 Off 두 개의 제어 모드를 가지고 있다. 제어 모드 Off에서 Heater가 꺼지면, $flow(Off)$인 $\dot{x}=-0.1x$에 따라 온도가 감소하며 연속상태의 전이가 일어나다가 $inv(Off)$인 $x\geq 18$과 jump condition $x<19$에 의해 제어 모드 On으로 이산상태의 전이가 일어나게 된다. 제어 모드 On에서도 이와 유사한 방식으로 연속상태 전이와 이산상태 전이가 일어나게 된다.

Definition 2.2 (하이브리드 오토마타의 상태) 하이브리드 오토마타의 상태의 집합은 $S\subseteq V\times\mathbb{R}^n$ 이고, 상태 s는 $s=(v,x)\in S$이다. 즉, 하이브리드 오토마타의 상태는 이산상태를 대표하는 제어 모드인 $v\in V$와 연속상태를 대표하는 연속 변수의 값으로 $x\in X$ 이루어져 있다.

Definition 2.3 (상태 전이) 하이브리드 오토마타의 상태 전이는 이산상태 전이와 연속상태 전이로 나뉘어 진다.

▪ 이산상태 전이 : 각각의 이벤트 $\sigma\in\Sigma$에 대해서 이산상태 전이는 $(v,x)\xrightarrow{\sigma}(v',x')$이다. 이 때, 다음을 만족하는 제어 전환 $e\in E$가 존재해야 한다.. e의 출발점은 v이고, e의 도착점은 v'이다. 또한 $event(e)=\sigma$이다.

▪ 연속상태 전이 : 각각의 실수 $\delta\in\mathbb{R}_{\geq 0}$에 대해서 연속상태 전이는 $(v,x)\xrightarrow{\delta}(v,x')$이다. 이 때 다음을 만족하는 미분가능함수 $f:[0,\delta]\rightarrow\mathbb{R}^n$이 존재해야 한다. $f(0)=x$이

고, $f(\delta)=x'$이다. 즉, 제어 모드의 변화 없이 연속상태 변수 x의 값만이 변화하는 것이 연속 상태 전이이다.
이 후에서 상태 $s\in S$에서 ρ에 의해 전이 가능한 상태 $s'\in S$가 존재 할 때, 즉, $s\xrightarrow{\rho}s'$인 s'이 존재 할 때, $s\xrightarrow{\rho}$로 표기 하겠다.

3. Conformance Relation

Hybrid input output Conformance Relation(hioco)는 임의의 상태의 변화 이후에 구현(Implementation)에서 발생하는 출력이 같은 상태 변화 이후에 설계 명세(Specification)에서 나올 수 있는 출력에 포함되는지의 관계를 말한다. hioco를 정의하기 위해 몇 가지 필요한 함수와 표기법을 정의하겠다.
하이브리드 오토마타 A에 대해서 S_A는 모든 상태의 집합이고, s_0^A는 초기 상태이고, $s_0^A=(v_0,x_0)$이다. $\mathbb{R}_{\geq 0}$과 유한한 이벤트의 집합 Σ가주어 졌을 때, Sequence의 집합 $(\Sigma\cup\mathbb{R})^*$를 $RT(Act)$라고 하자. $Act'\subseteq Act$인 Act'과 $\rho\in RT(Act)$가 $P_{Act'}(\rho)$를 Act'으로의 ρ의 정사영 함수라고 정의하자. 즉, $P_{Act'}(\rho)$ 함수는 전체 Sequence ρ 중 관측 가능한 Act'에 관한 Sequence만을 뽑아낼 수 있는 함수 이다. 이 때,A의 관측가능한 경로를 $Trace(A)$라 하면 ,

$Trace(A)=\left\{P_{Act}(\rho)|\rho\in RT(Act)\wedge s_0^A\xrightarrow{\rho}\right\}$ 이다.

$A\ after\ \sigma$는 ρ 이후 도달할 수 있는 모든 상태이다. 따라서 $A\ after\ \sigma=\left\{s\in S_A|\exists\,\rho\in RT(Act).s_0^A\xrightarrow{\rho}s\wedge P_{Act}(\rho)=\sigma\right\}$이다. 연속적인 상태전이에 의한 출력을 ξ라 하자. 상태 $s\in S$가 주어졌을 때, $out(s)$는 상태 s에서 관측 가능한 출력의 집합이다. 이것은,

$$out(s)=\begin{cases}\left\{a\in Act_{out}|s\xrightarrow{a}\right\}\cup\{\xi\} & \cdots\ \text{case1}\\ \left\{a\in \mathrm{Act_{out}}|s\xrightarrow{a}\right\} & \cdots\ \text{case2}\end{cases}$$ 이다. 이 때,

case 1은 연속적인 상태전이가 발생한 경우이고, case 2는 연속적인 상태전이가 발생하지 않은 경우이다.
그리고 $out(S)=\bigcup_{s\in S}out(s)$ 이다.

A_I는 구현, A_S는 명세라고 할 때, 최종적으로 Hybrid input output Conformance Relation(hioco)는 다음과 같다.

A_I hioco $A_S \equiv \forall\sigma\in Trace(A_S),$
$out(A_I\ after\ \sigma)\subseteq out(A_S\ after\ \sigma)$

4. 모니터의 구성

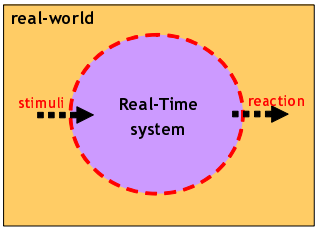

그림 3(a)

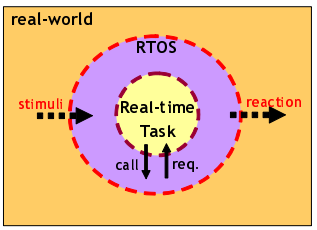

그림 3(b)

실시간 시스템을 바라보는 관점은 그림3의 두 그림과 같다. 그림3(a)는 실시간 시스템의 전체 하나의 시스템으로 보고 전체 시스템과 외부와의 반응과 응답에 대해서 실시간 시스템을 분석할 수 있다. 하지만, 이런 경우 실시간 시스템 내부에서 커널과 실시간 태스크와의 입출력 관계를 상세히 분석하기가 어렵다. 본 논문에서는 실시간 시스템을 그림3(b)와 같이 보고 커널과 태스크 간의 입출력에 대한 모니터를 하는 데에 초점을 맞추었다.
이러한 관점에서 커널에 2개의 모듈이 추가 되었고, 2개의 모듈은 다음과 같다.

- **Hybrid Conformance Monitor:** Hybrid Conformance Monitor는 태스크와 커널의 입출력을 Hooking하여 실시간 태스크가 설계 명세와 Conformance Relation을 만족하는지를 모니터하는 모듈이다.
- **Fault Handler:** Fault Handler는 실시간 태스크가 설계 명세와 Conformance Relation이 만족하지 않을 때, 오류를 처리해주는 모듈이다.

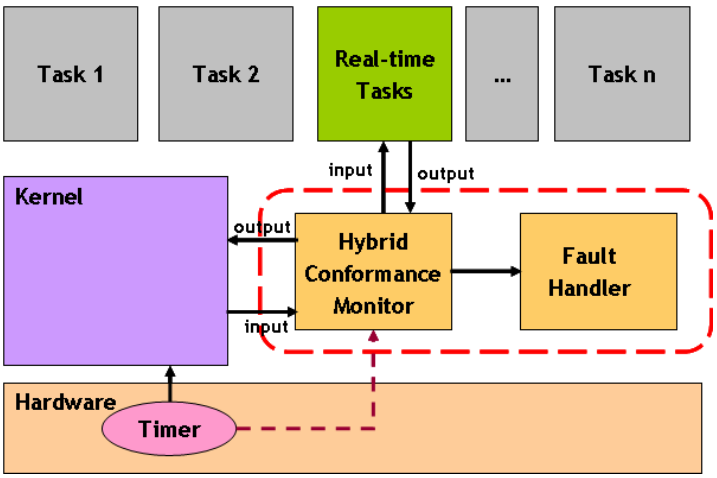

그림 4 Hybrid Conformance 모니터의 구조

그림 2는 Hybrid Conformance Monitor의 개략적인 구조이다. 시간의 제약을 가진 실시간 태스크는 커널을 통해서 외부 환경에 반응을 하게 된다. 외부의 자극을 하드웨어와 커널을 통해서 태스크에게 입력을 주고 처리

결과의 출력은 커널을 통해 외부 환경으로 전해지게 된다. 이 때 실시간 태스크는 커널과의 입출력간의 속성을 가지게 되는데 Hybrid Conformance Monitor는 커널과 실시간 태스크간의 입출력을 Hooking하여 앞에서 언급된 실시간 태스크의 hioco를 분석하여 태스크가 태스크의 설계 명세대로 올바르게 동작하는지를 모니터한다. 그리고 만약 실시간 태스크가 설계 명세와 일치성에 부합하지 않는 행위를 하게 되었을 경우에 Fault Handler가 상황에 맞는 처리를 해주게 된다. 실시간 시스템은 Hybrid Conformance Monitor를 통해서 실시간 태스크의 설계 명세의 거동특성을 만족하였는지 분석 할 수 있고, Fault Handler를 통해 Fault Tolerance를 보장해 줄 수 있다. Hybrid Conformance Monitor와 Fault Handler가 태스크 수준이나 시스템 외부 모니터로 추가 된 것이 아니라, 커널 모듈로 추가가 되어서 생기는 장점은 다음과 같다.

- 태스크 수준으로 추가했을 경우 CPU의 제어권을 쉽게 얻을 수 없어 모니터와 Fault Handle을 하기 어렵다.
- 외부에 모니터를 두어 시스템을 모니터 할 경우 모니터와 시스템간의 통신상의 지연 때문에 정확한 시간 등의 평가가 어렵다.

3. 결론 및 향후 연구 과제

실시간 제어 시스템은 시스템이 안정성과 실시간성이 매우 중요한 시스템이다. 그리고 실시간 제어 시스템의 대부분의 안적필수 시스템에 적용되고 있다. 따라서 시스템이 시간적 제약을 만족하지 못하면, 재산, 인명에 크나큰 피해를 안길 수 있다. 그래서 실시간 시스템에서는 구현이 명세단계에서 검증된 모델과 일치하는지의 여부가 매우 중요하다. 따라서 명세가 구현과 일치하는 지에 대한 많은 연구들에서 진행되어 왔다. 그러나 대부분의 연구가 테스팅라는 방법으로 이루어져 왔고, 이런 방법으로는 무한히 동작하여야 하는 실시간 반응형 시스템의 구현이 명세를 만족하는지 보장해 줄 수 없다. 그래서 본 논문에서는 실시간 시스템을 커널수준에서 모니터하여 분석하고 문제가 발생하게 되면 Fault Handler를 통해 적절한 처리를 해 줄 수 있도록 하였다. 이 기법을 통하여 커널과 태스크간의 상호작용에서 실시간성 및 안정성을 체크할 수 있어서 단지 태스크가 정상적인 출력을 내보내는지에 관한 것뿐만 태스크 내부루틴의 입출력에 대한 시간적인 평가가 가능하게 되었다. 또한 이런 기법은 Fault Tolerant시스템에 적용이 가능하고, 태스크가 데드락에 걸렸을 때 이를 해제할 수 있다. 이를 통해 안전성이 높은 실시간 시스템을 개발 할 수 있게 될 것이다.

하지만, 현재 연구 단계까지에서 실시간 제어 시스템에 대한 실제적인 적용을 해보지 못했다. 향 후 연구에서는 실제 의미 있는 실시간 제어 시스템에 본 논문의 방법론을 적용해 보고, 도구 제작에까지 연구를 진행할 예정이다.

4. 참고 문헌

[1] J. Tretmans and E. Brinksma. TorX: Automated Model Based Testing. In A. Hartman and K. Dussa-Ziegler, editors, *Proceedings of the 1st European Conference on Model-Driven Software Engineering*, 2003.

[2] Moez Krichen, Stavros Tripakis, "Black-box Conformance Testing for Real-Time Systems", In *SPIN'04 Workshop on Model Checking Software*, 2004

[3] Marius Mikucionis, Kim G. Larsen, Brian Nielsen, "Online On-the-Fly Testing of Real-time Systems", http://www.brics.dk, 2003

[4] UPPAAL Homepage
http://www.uppaal.com

[5] T- UPPAAL Homepage
http://www.cs.aau.dk/~marius/tuppaal

[6] T. Henzinger, "The theory of hybrid automata", *Proceedings of the 11th Annual Symposium on Logic in Computer Science, pp.278-292, IEEE Computer Society Press, 1996*