Conformance Monitor를 이용한 실시간 시스템의 모니터링

심재환, 김진현, 최진영
고려대학교 컴퓨터학과
e-mail:{jhsim, jhkim, choi}@formal.korea.ac.kr

Monitoring of Real Time System using Conformance Monitor

Jae-Hwan Sim, Jin-Hyun Kim, Jin-Young Choi
Dept. of Computer Science and Engineering, Korea University

요 약

실시간 시스템은 높은 수준의 안정성을 요구하는 시스템이다. 실시간 시스템에서 오류는 잘못된 응답 뿐만 아니라 시간적으로 늦은 응답도 오류를 발생 시킬 수 있다. 따라서 실시간 시스템의 안정성을 보장해 주기 위해서 실시간 태스크의 시간 제약에 대한 모니터링 및 오류발생 시에 오류를 처리해 줄 수 있는 기법이 필요하다. 본 논문은 Timed Conformance Monitor를 통해서 실시간 태스크가 시간의 제약을 만족하는지를 분석하고 또한 분석 결과에 따라 오류를 처리할 수 있는 Fault Handler를 추가하여 실시간 시스템에 대한 Fault Tolerance를 보장해 줄 수 있는 기법을 제시한다.

1. 서 론

실시간 시스템은 현재 항공, 로봇, 원자력, 철도 등의 분야 등의 안전과 밀접한 관계를 가지고 있는 분야에 적용 되고 있고, 오류가 발생할 시에 큰 인적 물질적 피해가 생길 수 있다. 실시간 시스템은 엄격한 시간적 제약을 가지는 시스템을 말한다. 실시간 시스템에서의 오류는 크게 두 가지로 구분 될 수 있다. 시스템이 입력에 대해 잘못된 응답을 하는 경우와 올바른 응답을 하더라도 시간적 제약을 맞추지 못한 경우 이다. 따라서 실시간 시스템의 태스크는 기능적인 부분에 대한 분석 뿐 만이 아니라 시간적 제약에 대한 분석은 매우 중요하다. 현재 실시간 시스템의 시간적인 부분에 대한 많은 연구가 진행되고 있다. 예를 들어 Worst Case Execution Time(WCET)를 분석하여 실시간 태스크의 스케줄링을 분석하는 것, Timed Conformance Relation을 이용하여 실시간 시스템을 Conformance Testing하는 것이 그 예이다. 그러나 이 같은 연구들은 몇 가지의 제약사항을 가지고 있다. 첫째로 대부분의 실시간 시스템은 일회적인 작동을 하고 종료 되는 것이 아니라 지속적으로 환경과 반응을 하며 수행되는 반응형 시스템이기 때문에 일정 수의 테스트만을 가지고 그 시스템이 시간적인 제약을 만족하는 지를 보장해 주기 어렵다. 둘째로, 많은 연구들이 실시간 시스템 전체적인 관점에서 시간 제약을 분석하기 때문에 커널과 태스크 사이의 상호작용에 대한 Conformance는 분석할 수 없다. 셋째로, 시스템이 수행 중에 있을 때 발생하는 예기치 않은 오류를 검출하기 어렵고 또한 오류의 발생시 Fault Tolerance를 보장해 줄 수 없다. 따라서 설계 단계에서 시간을 표현할 수 있는 Timed Automata를 이용해서 실시간 태스크를 설계하고, 이 설계를 모니터에 적용시켜 시간적 제약에 따른 Conformance Relation을 분석하여 오류가 있다면 이를 처리해 줄 수 있는 기법이 필요하다. 그래서 본 논문에서는 실시간 시스템의 오류를 찾고 또한 Fault Tolerance를 보장해 줄 수 있는 Timed Conformance Monitor를 제시한다.

본 논문의 구성은 다음과 같다. 배경지식으로 2장과 3장에서는 각각 Timed Automata와

Conformance Relation에 대해 설명할 것이고, 4장에서는 본 논문의 목적인 Conformance Monitor를 제시할 것이다. 그리고 5장에서는 Conformance Monitor를 이용하여 실제적인 실험을 한 결과를 보여주고, 6장에서 결론과 향후 연구 방향에 대해 언급하겠다.

2. Timed Automata

실시간 태스크의 설계와 모니터에 이용되는 Timed Automata는 $A=(L,l_0,X,Act,E)$로 정의된다. 이와 같이 Timed Automata A는 5개의 쌍으로 이루어지며, 여기서 L은 location의 집합이고, $l_0 \in L$는 초기의 location이고, X는 시간 변수의 집합이고, Act는 입력(input)과 출력(output)의 동작(Action)의 집합이다. 그리고 E는 $E \subseteq L \times G(X) \times Act \times L$인데, Edge들의 집합이다. 여기에서 $G(X)$는 location 간의 전이를 위한 시간적 조건이다. $G(X)$는 x # k의 형태로 표현 되는데, # $\in \{\leq, \geq, =, <, >\}$이다. 그리고 입력 동작의 집합 Act_{in}과 출력 동작의 집합 Act_{out}과 시간에 의한 동작 Act_t에 대해서 $Act = Act_{in} \cup Act_{out} \cup Act_t$이다.

3. Conformance Relation

Input Output Conformance Relation(ioco)는 구현(Implementation)에서 발생하는 출력(Output)이 같은 시점에서 설계 명세(Specification)에서 나올 수 있는 출력에 포함되는지의 관계를 말한다. 그러나 ioco만으로는 실시간 시스템에서의 시간적 제약을 만족하는 지 여부를 확인할 수 없다. 그래서 시간적인 문제를 확인할 수 있도록 확장된 것이 Timed Input Output Conformance Relation(tioco)이다. tioco를 정형적으로 표현하면 다음과 같다.

- A_I : 구현의 Automata
- A_S : 설계 명세의 Automata
- $Seq(A)$: Timed Automata A의 모든 시퀀스의 집합
- $out(A,\rho)$: Automata A의 Sequence ρ 이후의 출력

위와 같이 정의 되어 있을 때, tioco는 다음과 같이 정의 된다.

$$A_I \text{ tioco } A_S \equiv \forall \rho \in Seq(A_S),\ out(A_I,\rho) \subseteq out(A_S,\rho)$$

위에 정의를 자연어로 표현하면, 설계 명세상의 임의의 시퀀스 ρ에 대해서 시퀀스 ρ이후의 구현에서 나오는 출력은 시퀀스 ρ이후에 설계 명세에서 나올 수 있는 출력에 포함 된다는 것이다. 즉, 구현물에서 나올 수 있는 모든 행위가 이미 설계에 명세된 것이라는 의미이다. 본 논문에서는 모니터가 설계와 구현 사이의 tioco를 분석하여 구현된 시스템이 오작동을 하는지를 해석하는 것이 목표이다.

4. Conformance Monitor

4.1 모니터의 구성

본 논문에서 실시간 시스템의 태스크들의 동작을 모니터 하기 위해서 2개의 모듈을 실시간 운영체제의 커널에 삽입하였다. 그 2개의 모듈은 아래와 같다.

- **Timed Conformance Monitor** : Timed 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.

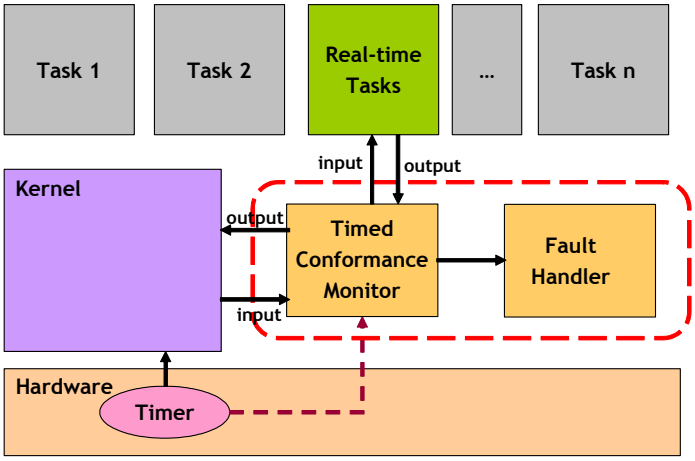

그림 1 Timed Conformance Monitor

그림 1은 Timed 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 입출력에서 시간적인 제약을 가지게 되는데 Timed Conformance Monitor는 커널과 실시간 태스크 간의 입출력을 Hooking하여 앞에서 언급된 실시간 태스크의 tioco를 분석하여 태스크가 태스크의 설계

명세대로 올바르게 동작하는지를 모니터한다. 그리고 만약 실시간 태스크가 설계 명세의 시간적 제약을 만족하지 못했을 경우에 Fault Handler가 상황에 맞는 처리를 해주게 된다. 이를 통해 실시간 태스크가 설계 명세대로 동작을 하는지를 분석할 수 있고, 오류의 발생시 Fault Handler를 통해 Fault Tolerance를 보장해 줄 수 있다. 이 두 개의 모듈이 태스크 수준이나 시스템 외부 모니터로 차가 된 것이 아니라, 커널 모듈로 추가가 되어서 생기는 장점은 다음과 같다. 첫째로, 위의 모듈을 태스크 수준으로 추가했을 경우 보다 CPU의 제어권을 쉽게 얻을 수 있어서 Monitoring과 Fault Handling이 용이하다. 둘째로, 외부에 모니터를 두어 시스템을 모니터 할 경우 모니터와 시스템간의 통신상의 지연 및 시스템에 가해지는 과부하로 인해 정확한 시간의 평가가 어렵지만, 커널 모듈로 추가 되었을 경우 이러한 문제가 해소 된다.

4.2 모니터의 실행

실시간 태스크는 Timed Automata를 이용하여 설계하게 된다. 이 설계를 바탕으로 실시간 태스크를 구현하게 된다. 그리고 Timed Automata로 설계된 설계 명세는 Timed Conformance Monitor의 자료구조로 들어가서 Timed Conformance Monitor가 실시간 태스크와 설계 명세간의 Conformance Relation을 분석하는데 이용이 된다. 중요한 사항은 모니터내의 설계 명세는 모든 Location에서의 정보를 관리하는 것이 아니라, 현재 Location의 입출력 정보만 갖도록 설계 명세가 전이할 수 있도록 설계하였다. 그 이유는 모든 Location에서의 I/O정보를 다 보관한다면, 상태폭발을 일으킬 수 있는데, 이 때 모니터의 매우 많은 메모리 자원을 필요로 하게 된다. 이를 막기 위해서 현재 Location의 정보만을 가지고 있도록 설계를 하였다.

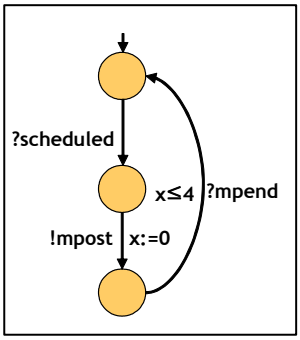

그림 2(a) 설계 명세

```
void TaskA()
{
    while(1)
    {
        ...
        A=...
        kMboxPost(mbox1, A);
    }
}
```

그림 2(b) 구현

그림 2(a)는 구현하고자 하는 실시간 태스크의 설계 명세이다. 이 태스크는 커널로부터 스케줄링 받아 메일박스에 메시지를 보낸 뒤 4 tick안에 다른 태스크가 메시지를 받아가는 태스크를 Timed Automata로 명세한 태스크이다. 그림 2(b)는 이 명세대로 실제 태스크를 구현한 것이다. 구현에서 이 태스크는 A라는 메시지를 내보내기만 하면 된다.

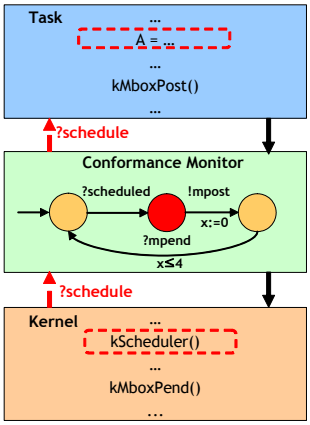

그림 3(a)

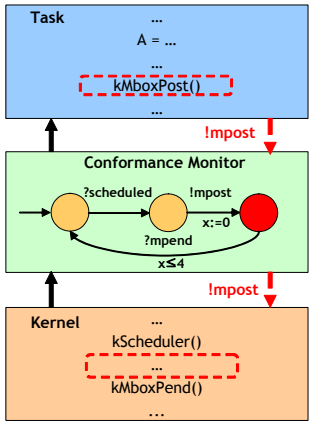

그림 3(b)

그림 4로 실제 모니터가 어떻게 동작을 하는지 설명하겠다. 앞의 Timed Automata로 모델링된 설계 명세는 모니터로 변환된다. 명세대로 스케줄러가 태스크를 스케줄링 해주게 되면, 모니터가 이 정보를 받아서 모니터 내부에 있는 설계 명세를 초기 location에서 다음 location으로 전이를 시켜준다. 그리고 태스크가 수행된 후 메일박스로 메시지를 전해준다. 이 때 모니터는 이 함수를 Hooking하여 태스크로 부터의 출력이 설계 명세 상에서 현재 위치하고 있는 location에서 나올 수 있는 출력인지를 판단한다. 만약 잘못된 출력 이었다면 Fault Handler를 통해 오류상황에 대한 처리를 해주고, 정상적인 시점에 나온 출력이라면, 설계 명세의 location을 해당하는 다음 location으로 전이 시켜 주고 커널에게 메시지를 전달한다. 이런 과정이 반복되면서, 태스크의 출력이 설계 명세상의 현재 location으로부터 나올 수 있는 출력인지를 판단하는 것이 Timed Conformance Monitor이다. 이와 같이 앞선 그림 2(a)의 설계 명세에서는 태스크와 커널간의 시간의 제약이 표현되었지만, 구현상의 태스크에는 따로 시간적 제약을 표현해 주기 어렵다. 그러나 Timed Conformance Monitor를 통해 커널과 태스크간의 입출력에 대한 시간적 분석이 가능하고, 오류상황에 대처하기 쉬워 시스템의 Fault Tolerance를 보장해 줄 수 있다. 특히 선점형 커널에서 태스크 내부 루틴 간의 시간 제약을 분석하기에 용이하고, 데드락

상황 시 해제가 용이하다. 모니터를 실행하기 위한 알고리즘은 다음과 같다.

Monitor Execution Algorithm

Initially Z:= {< l_0, 0>}.
update location:
  while (Z≠Ø) do below :
    if recv(act) ∈ kernel_ouput(Z) then
      send act to implementation
      Z:=update_location(Z, act)
      set time variable
    else if recv(act) ∈ imp_ouput(Z) then
      send act to kernel
      call check conformance
      Z:=update_location(Z, act)
      set time variable
    else if recv(act) == tick then
      decrement time variable
check conformance:
  if recv(act) ∉ imp_output(Z) then
    call Fault Handler
    return fail
  else if Z=Ø then
    call Fault Handler
    return fail
  else continue

5. 구현 및 실험

본 논문에서는 제어 장비인 PLC의 오동작 여부를 확인하기 위해 Conformance Monitor를 구현하였다. PLC는 TMS320C32 DSP 프로세서를 사용하고 Target RTOS로는 uCOS-II를 사용하였다.

```
void OSSched (void)
{
    INT8U y;

    OS_ENTER_CRITICAL();
    if ((OSLockNesting | OSIntNesting) == 0) {    /* 
        y             = OSUnMapTbl[OSRdyGrp];     /* 
        OSPrioHighRdy = (INT8U)((y << 3) + OSUnMapTbl
        if (OSPrioHighRdy != OSPrioCur) {          /*
            OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy
            OSCtxSwCtr++;                          /* 
                    I
      #if    OS_TIMED_MONITOR_EN
        TMonitor(SYS_SCHED, OSPrioCur, 0);
      #endif

            OS_TASK_SW();                          /* 
        }
    }
    OS_EXIT_CRITICAL();
}
```

그림 4 Conformance Monitor가 삽입된 코드

위의 그림 4는 uCOS-II의 커널 코드에 모니터가 삽입된 것이다. 이와 같은 방법으로 각각의 커널 함수에 모니터가 삽입되어 수행되는 태스크의 동작이 올바른지를 확인한다. 모니터 대상이 되는 PLC는 총 6개의 시스템 태스크가 동작을 하는데, 이 중 shell 태스크는 시스템의 진단하고 문제가 없을 경우에 정상적인 동작을 표현하도록 LED를 깜빡이게 되어있다. 모니터와 함께 PLC를 작동 시키고 42일 경과후 LED가 동작을 하지 않는 경우가 생겼다. 이를 확인하기 위해 태스크의 동작이 설계 명세에 위배 될 경우 그 때의 레지스터 값을 알려 주게끔 Fault Handler를 작성을 하였더니, shell 태스크 내의 loop에서 잘못된 위치의 스택 포인터를 참조하는 것을 발견 할 수 있었다. 다음은 Conformance Monitor 안에서 Stack Pointer를 읽어주는 루틴이다.

```
Show_SP   .MACRO
          PUSH   R0
          LDI    SP, R0
          SUBI   1, R0
          PUSH   R0
          Call   _outhex32
          SUBI   1, SP
          POP    R0
          .ENDM
```

6. 결론 및 향후 연구 방향

실시간 시스템은 시간 제약에 대해 엄격한 시스템이다. 또한 대부분의 실시간 시스템이 안전필수 시스템인 경우가 많다. 따라서 실시간 시스템은 오류가 생기게 되면 인명, 재산에 대한 크나 큰 피해를 안길 수 있다. 그리고 오류는 시스템의 기능적인 문제 뿐 만이 아니라 시간적 오류에서도 기인할 수 있다. 그래서 실시간 시스템에서 설계시의 시간적 제약이 구현에서 잘 반영되었는지에 대한 많은 연구가 진행되어왔다. 하지만, 대부분의 연구가 테스트라는 방법으로 이루어져왔고 이런 방법으로는 무한히 동작하여야 하는 실시간 반응형 시스템이 시간적인 제약을 만족하는 지에 대한 보장은 해줄 수 없다. 본 논문에서는 이러한 점을 해결하기 위해 실시간 시스템에 대한 모니터링 기법을 제시 하였다. 이를 통해 태스크와 커널간의 상호작용에 대한 시간적 제약을 체크 할 수 있어서 단지 태스크가 제시간에 스케줄링 되는지 뿐만 아니라 태스크 내부루틴의 입출력에 대한 시간적인 평가가 가능하게 되었다. 또한 이런 기법은 Fault Tolerant가 필수적인 시스템에 적용이 가능하여 높은 안정성을 요구하는 실시간 시스템을 개발 할 수 있다. 하지만 아직은 설계 명세로부터 모니터를 구현하는 과정이 자동이 아니고, 또한 플랫폼에 의존적이다. 향 후 연구 방향은 이를 해결하는 것이다.

참고문헌

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

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