ESTEREL을 이용한 RTOS Scheduler의 검증 및 구현

양진석[0] 김진현 심재환 이수영 최진영
고려대학교 정형기법 연구실
{ jsyang[0], jhkim, jhsim, sylee, choi }@formal.korea.ac.kr

Verification and implementation of RTOS Scheduler with ESTEREL

Jinseok Yang[0] Jinhyun Kim Jaehwan Sim Suyoung Lee
Korea Univ. Threory and Formal Methods Lab.

요 약

오늘날 RTOS가 운영되는 시스템이 고안정성을 요구 할수록 임베디드 소프트웨어인 RTOS의 중요성은 날이 갈수록 증대하고 있다. 검증된 RTOS의 개발을 목표로 본 논문에서는 RTOS의 태스크 스케쥴러를 정형 명세하고, 정형명세된 스케쥴러가 가져야 하는 몇가지의 검증특성들을 모델체킹 기법을 통해서 검증하였다. 또한 에스테럴로 정형 명세된 스케쥴러에서 자동 생성된 C 소스를 사용하여 직접 태스크를 스케쥴해 주는 간단한 RTOS를 제작함으로서 검증된 RTOS를 제작 할 수 있는 방법을 제시 하고자 한다.

1. 서 론

RTOS(Real-Time OS)[1]는 오늘날 대표적인 임베디드 소프트웨어로서, 그 중요성이 사회적, 기술적인 측면에서 점차 부각되어 가고 있다. uCOS와 같은 대표적인 RTOS가 실제 상용 임베디드 시스템에 사용이 되고 있지만, 원자로제어와 같은 고안정성 시스템(Safty Critical System)의 경우에는 시스템 자체가 가지는 안정성 때문에 그곳에 사용되어야 하는 RTOS는 반드시 정형 검증이 되어야 한다.

이 논문에서는 정형 검증된 RTOS 개발을 위해 RTOS의 핵심이라고 할 수 있는 태스크 스케쥴러를 정형 명세하고 검증한다. 그리고 uCOS와 같은 마이크로 커널을 가지는 RTOS에 검증된 스케쥴러를 사용하여 실제 구현한 것을 보여준다. 검증된 스케쥴러를 가지는 RTOS 개발을 위해 자연어로 기술된 요구사항(Requirement)과 특성(Property)을 프랑스 인리아(Inria)에서 개발된 정형 명세 언어인 에스테럴(Esterel)[2]을 이용한다. 그리고 역시 인리아에서 개발된 모델 체커 XEVE를 이용하여 에스테럴로 명세된 스케쥴러가 제시된 검증 특성에 대하여 만족을 하는지 검증(Verification)을 한다.

이 논문의 구성은 다음과 같이 되어 있다. 2장과 3장에서 자연어로 명세된 시스템과 관련된 스케쥴러의 요구사항, 기능 그리고 적용할 검증 특성을 살펴본다. 4장에서는 정형 명세 언어인 에스테럴을 이용하여 자연어를 어떻게 명세했는지 살펴본다. 마지막으로 XEVE를 통한 검증 결과를 확인과 실제 구현예를 살펴본 후, 결론과 향후 연구 방향에 대해 논한 후 마치도록 하겠다.

2. 시스템 자연어 명세

이 논문에서는 한정된 태스크, 짧은 Time Tick을 가지면서 고 안정성을 요하는 특정 임베디드 시스템에 사용되는 RTOS의 태스크 스케쥴러를 검증 대상으로 삼는다. 스케쥴러 요구사항 명세는 기능 설명과 요구사항으로 나누어져 있다.

2.1 기능설명

스케쥴러는 Time Tick이 발생할 때만 스케쥴을 한다. 스케쥴러는 우선순위(Priority) 알고리즘을 이용하여 한정된 수의 태스크들을 스케쥴한다. 스케쥴의 대상이 되는 태스크는 Idle 태스크를 포함하여 모두 4개 이며 각 태스크의 특성은 [표1]과 같다.

[표1] 태스크의 특성

태스크	설명	공유자원	특성
TASK1 (이하 T1)	외부 인트럽트에 의해 깨어나는 태스크	사용	Sporadic
TASK2 (이하 T2)	일반 태스크, T1와 같은 공유자원 사용	사용	Periodic
TASK3 (이하 T3)	일반태스크	미사용	Periodic
IDLE	휴면 태스크	미사용	-

T1은 외부 인터럽트에 의해서 대기상태에서 깨어나는 일을 처리하는 태스크이며, 가장 높은 우선순위를 가진다. T2와 같은 공유자원을 사용하여 T2가 Writing한 자료를 읽어온다. 그리고 T1의 경우 스케쥴러에 의해서 선택이 되면 무조건 1 tick안에 종료 된다고 가정한다. T2와 T3는 주어진 시간에 따라 주기적으로 일을 처리하는 태스크이다. 각 태스크들의 우선순위는 T1 > T2 > T3 순이다.

2.2 요구사항

가. T2와 T3는 시스템이 시작 할 때 RDY상태에 있어야 한다.
나. 주기적으로 작업을 수행하는 T2와 T3는 인터럽트가 무한하게 들어오지 않는다는 가정 하에, RDY상태에서의 대기 시

간 동안 작업을 완료해야 한다.

다. RDY상태에서의 대기시간동안 작업을 완료하지 못한 태스크는 WAIT상태로 상태를 변경해야 한다.

라. WAIT상태에서의 대기시간이 끝난 T2와 T3는 RDY상태로 상태를 변경해야 한다.

마. T1은 외부 인터럽트에 의해서만 WAIT상태에서 RDY상태로 상태를 변경해야 한다.

바. T1과 T2는 자원을 공유하며, 자원 동기화 문제로 T2가 자원을 사용하고 있는 경우(T2의 작업이 종료 안된 상태)라면, T1이 자원을 선점 할 수 없도록 스케쥴링 해야 한다.

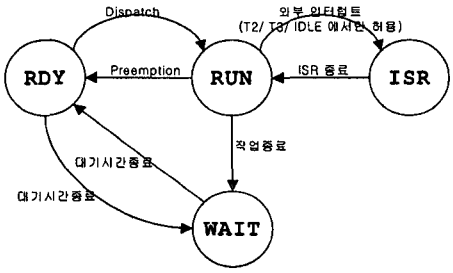

그림 1 태스크 상태도

3. 검증특성의 자연어 명세

이 논문에서는 2장에서 명세된 요구사항을 기초로 하여 아래와 같은 두 가지 검증 특성(Property)을 설정하고, 에스테렐로 명세된 스케쥴러의 검증에 이용한다.

검증특성1)
"자원 동기화에 따른 스케쥴링을 해야 한다."
T2가 자원을 선점하고 1 time tick안에 주어진 작업을 완료하지 못한 상태에서 외부 인터럽트가 발생했다면, 다음 tick에서 스케쥴러는 T1과 T2에 대한 스케쥴링을 해야한다. 스케쥴러는 우선순위 정책에 따라 T1을 선택해야 하나, 자원 동기화의 문제로 T2가 작업을 완전히 마칠 수 있도록 T2를 선택해야 하는 것을 의미한다.

검증특성2)
"인터럽트를 배제한 상태에서, 주기적 태스크 T2와 T3는 스케쥴러빌리티를 가져야 한다."
주기적 태스크 T2와 T3의 경우 외부 인터럽트가 없는 상태에서 주어진 RDY상태에서의 대기시간동안 태스크의 작업을 마칠 수 있음을 의미한다.

4. 에스테렐을 이용한 스케쥴러의 정형 명세

에스테렐을 이용하여 명세함으로서, 모델체킹을 통한 정형 검증이 가능할 뿐만 아니라, 자동으로 생성된 C 소스를 얻을 수 있다는 장점을 가질 수 있다. 하지만 에스테렐을 이용한 정형 검증을 하기 위해서는 순수 에스테렐(Pure Esterel) 코드로만 작성을 해야 하는 단점이 있는 관계로 [그림 2]처럼 명세단계에서 스케쥴과 관계된 태스크의 정보를 에스테렐 코드에 포함 시켜야 한다. 이로 인하여 스케쥴러가 처리 할 수 있는 태스크의 수가 한정되어 버리는 단점을 가지지만, 특정 목적에 사용되는 RTOS의 경우 처리 할 태스크의 수가 요구단계에서 이미 결정이 날 수 있으므로 이런 단점을 보완 할 수 있다.

```
T1_priority := 1 : integer,        % 우선순위
T1_shottime := 3 : integer,        % RDY 대기시간
T1_waittime := 0 : integer,        % WAIT 대기시간
T1_state := STOP : integer,        % TASK 상태
```

그림 2 에스테렐로 명세된 태스크의 TCB 정보

순수 에스테렐에서는 배열을 사용할 수 없기 때문에 태스크들을 비교하기 위해서는 비교문을 사용 할 수 밖에 없다. 태스크 간의 특성이 증가하고 스케쥴할 태스크의 숫자가 증가할수록 스케쥴러를 에스테렐로 명세할 때 그 길이가 길어 진다는 단점이 있다. [그림 3]은 검증에 사용된 스케쥴러의 태스크 선택 알고리즘을 에스테렐로 명세한 것이다.

```
if T1_state = RDY then
  if T2_state = RDY and T2_running = 0 then
    T_id := 1;
  elsif T2_state = RDY and T2_running = 1 then
        T_id := 2;
  else
    T_id := 1;
  end if;
elsif T2_state = RDY then
  T_id := 2;
elsif T3_state = RDY then
  T_id := 3;
else
  T_id := 4; % IDLE
end if;
```

그림 3 에스테렐로 명세된 태스크 선택 알고리즘

T1과 T2이 RDY상태에 있는 경우를 제외하고는, 우선순위 기반의 스케쥴 알고리즘을 이용하고 같은 우선순위를 가지는 태스크가 2개이상 없는 관계로 "if then else" 구문을 이용하여 간단하게 태스크를 선택할 수 있다.

5. 검증 특성에 대한 정형 명세

우선 3장에서 자연어로 언급한 검증특성을 시제논리를 이용하여 표현을 한다. 그리고 표현된 시제논리를 다시 에스테렐을 이용하여 명세한다.

검증특성1)
p : T2가 자원을 선점, q : T2가 종료되지 않음
r : 외부 인터럽트 신호가 있음, s : 스케쥴러가 T1을 선정

$f = G[p \rightarrow F(q \wedge r \wedge s \wedge tick)]$

```
var T2_isrunning := 0 :integer in
  loop
    present T2_RUNNING  then
        T2_isrunning := 1;
```

```
    end present;
  present INTERRUPT and not T2_WAIT then
     if ?TASK_ID = 1 then
        emit ERR_DEADLOCK;
     end if;
  end present;
 each tick;
end var;
```

검증특성2)

p : T2가 종료되지 않음, r : T2는 WAIT 상태

q : T2의 RDY상태 대기 시간이 종료됨

f = G[p ∧ q ∧ r ∧ tick]

```
loop
   present not T2_WAIT and T2_TIMEOVER then
         if ?T2_STATE = WAIT then
           emit ERR_SCHEDULABILITY;
         end if;
   end present;
each tick;
```

※ T2와 T3의 검증 특성에 대한 명세는 같음

6. 모델 체킹을 이용한 특성의 검증

검증특성 1)에 대해서 모델 체커인 XEVE를 사용하여 검증된 검증결과는 [그림 4]와 같다. 5장의 검증특성에서도 알 수 있듯이 자원 동기화 문제가 발생 했을 경우에 T1이 스케쥴링 되는 경우는 절대로 발생해서는 안된다. [그림 4]의 결과화면은 ERR_DEADLOCK이라는 출력신호에 대한 모델 체킹 결과 NEVER EMIT이라는 결과를 보여준다.

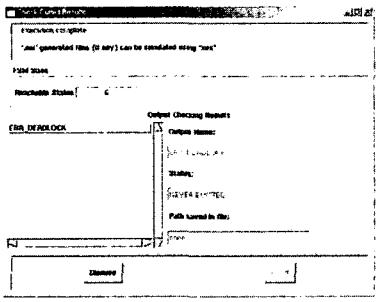

그림 4 검증특성1)에 대한 검증결과

검증특성 2)에 대해서도 같은 방법으로 살펴본 결과 추력 신호 ERR_SCHEDULABILITY에 대한 모델 체킹 결과가 NEVER EMIT이 나오는 것을 볼 수 있다.

7. 검증된 스케쥴러의 사용

지금까지의 과정에서 우리는 스케쥴러를 정형 명세 언어로 명세하고, 명세된 스케쥴러가 2가지 검증 특성에 대해서 만족하는 것을 알 수 있었다.

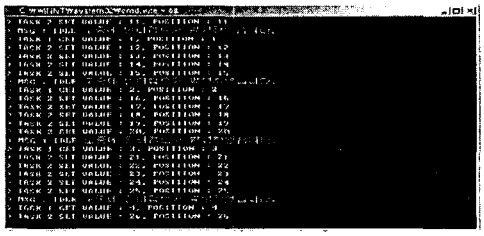

그림 5 검증된 스케쥴러를 사용한 RTOS 동작

검증이 된 에스테럴 명세는 자동으로 C 소스를 생성해 준다. 이렇게 자동으로 생성된 C 소스를 이용하여 자원동기화와 Time Tick마다 스케쥴링을 해주는 간단한 RTOS를 제작 하였다. 자동으로 생성된 C 소스는 함수형식으로 구성되어 있기 때문에 스케쥴러에 대한 인위적인 코드의 수정이나 추가/삭제는 이루어지지 않았다.

8. 결론 및 향후 연구 과제

본 논문에서는 고 안정성 임베디드 시스템에 사용되는 임베디드 소프트웨어 RTOS의 스케쥴러 부분에 대한 설계, 정형명세 언어인 에스테럴을 이용한 요구사항 명세, 그리고 모델 체킹 기법을 사용하여 명세된 요구사항이 올바르게 명세되었는지 대하여 검증을 하였다. 그리고 추가적으로 검증된 에스테럴 명세로부터 자동으로 생성되는 C 코드를 사용하여 간단한 RTOS를 만들어 보았다.

에스테럴을 사용하여 RTOS의 스케쥴러를 명세해 본 결과 스케쥴과 관련된 모든 정보를 에스테럴이 가져야 하는 관계로 요구사항이 변경됨에 따라 에스테럴 코드를 직접 수정해야 하는 불편함과 RTOS개발에 범용성이 떨어지는 사실을 알 수 있었다. 하지만 에스테럴를 이용하여 스케쥴러를 명세하고 RTOS를 제작해본 결과 아래와 같은 장점이 있다. 첫째, C 소스코드의 자동 생성 기능에 의해 생성된 C 소스를 수정 없이 바로 사용함으로서, 코딩 단계에서 발생하는 오류를 최소화 할 수 있다. 둘째, 자동 생성된 C 소스코드로 구현을 했음에도 불구하고 24K 정도의 사이즈로 RTOS를 구현 할 수 있었다.

본 논문에서는 RTOS의 기본 기능인 스케쥴러만을 구현하였다. 향후 RTOS에 반드시 필요로 하는 부분에 대하여 에스테럴로 명세하고 구현하는 작업을 통해 검증된 RTOS를 제작 할 수 있도록 계속 연구를 진행해 나갈 것이다.

9. 참조문헌

[1] Jean. J. Labrosse, "MicroC/OS-II The Real-Time Kernel 2/E". CMP 1992

[2] Gerard Berry, "The Esterel v5 Language Primer Version v5_91", INRIA, 2000