# ESTEREL 임베디드 소프트웨어를 위한 모델 기반 테스트 기법 연구

양진석° 김진현 심재환 김창진 최진영
고려대학교 컴퓨터 이론 및 정형기법 연구실
{jsyang°, jhkim, jhsim, cjkim, choi}@formal.korea.ac.kr

# A Study of the Model-based Testing for Embedded Software in ESTEREL

Jin-Seok YANG° Jin-Hyun Kim, Jae-Hwan Sim, chang-jin Kim, Jin-Young Choi
Computer Theory and Formal methods Lab. in Korea University

요 약

본 논문에서는 임베디드 시스템의 제어를 위해 동기화 언어인 에스테렐로 개발된 소프트웨어가 소프트웨어 요구사항을 만족하는지 확인하기 위한 Model-based Test(MBT) 프레임 워크를 제안한다. 제안된 프레임 워크를 기반으로 작성한 프로토 타입의 테스트 도구는 요구사항 모델을 참조하여 On-the-fly 방법을 통해 테스트 케이스를 랜덤하게 자동으로 생성하여 소프트웨어에 대하 테스트를 자동으로 수행한다. 간단한 case study로 레고 마인드스톰 로봇 제어 소프트웨어에 MBT를 수행하고, 그 결과를 확인한다.

## 1. 서 론

에스테렐(Esterel)과 루스터(Lustre)는 1980년대 반응형 시스템(Reactive System)을 디자인하기 위해 프랑스에서 개발되어진 대표적인 동기화 언어들(Synchronize Language)이다. 두 언어 모두 동기화 언어가 가지는 기본적인 시간적 의미와 동기화에 대한 가설을 바탕으로 입력에 대한 행위(behavior)와 그 출력(반응)을 쉽게 디자인 할 수 있다. 이런 특징은 오늘날 하드웨어와 소프트웨어의 개발이 동시에 이루어지는 임베디드 시스템의 개발과정에서 외부 환경 또는 하드웨어에 대한 입/출력에 대한 인터페이스를 단순한 신호(signal)과 센서(sensor)의 선언으로 쉽게 구현 할 수 있으므로, 임베디드 시스템의 제어 소프트웨어 개발에 유용하게 사용될 수 있다. 실제로 에스테렐은 프랑스 Dàssult사에서 제작한 라팔 전투기의 소프트웨어 개발에 사용되었으며, 루스터는 SCADE를 통해 에어프랑스 여객기의 소프트웨어를 제작하는 곳에 사용되었다 [1]. 뿐만 아니라, 소프트웨어 구현을 바로 C를 이용하여 구현하는 것이 아니라 이와 같은 정형 명세 언어를 이용하여 구현함에 따라 모델 체킹(Model Checking)이나 시뮬레이션을 통한 소프트웨어 검사(validation)이후 C코드를 자동으로 생성시킬 수 있기 때문에 직접 사람 손으로 개발된 C코드보다 좀 더 높은 신뢰성을 가질 수 있다.

## 2. 연구동기 및 관련연구

본 논문에서 대상이 되는 에스테렐 언어는 소프트웨어 개발 V 모델의 디자인 단계에서 시스템 요구 사항을 반영하는 모델을 작성 할 수 있고, 지원도구(XEVE 또는 XES)를 사용하여 특성(Property)에 대해 모델을 검증(Verification) 할 수 있다. 그 뿐만 아니라, 디자인 단계에서 작성된 모델에 호스트 언어(C 언어)의 추가적인 사용을 통해 행위를 좀 더 세부적으로 명세 할 수 있으며 에스테렐 컴파일러를 통해 에스테렐 코드로부터 C 코드를 자동으로 생성 하여 대상 시스템에 탑재 할 수 있다. 하지만, XEVE나 XES모두 모델에 대한 검증과 검사를 위한 도구여서 호스트언어가 함께 포함되어진 소프트웨어에 대한 검사를 수행하지 못한다. XEVE(Model Checker)의 경우 에스테렐에 호스트 언어가 함께 사용된 경우 호스트 언어가 사용된 부분에 대해서는 검증을 시행 할 수 없으며 XES(시뮬레이터)는 외부 입력에 대한 내부 변수 값 및 출력 신호의 변화를 잘 표현해 주지만, 매 단계마다 입력을 일일이 수동으로 설정해야 하므로 다양한 입력을 가지는 에스테렐 프로그램을 시뮬레이션 하기에는 불편함 점이 적지 않게 존재한다. 그러므로 위와 같은 사항들을 보완하기 위해 에스테렐로 구현된 소프트웨어가 요구사항을 만족하는지 검사하기 위한 자동 테스트 도구가 필요하다. 루스터로 개발된 소프트웨어의 경우, 이미 자동으로 테스트 케이스를 생성하고 테스트하는 도구로 Lutess와 Luress[2]가 개발되어 있으며, 본 논문에서는 이들 연구에서 제안된 도구 프레임워크를 참조하여 에스테렐에 적용할 수 있는 Model-based Test(MBT) 프레임워크를 제안한다. 그리고 간단한 case study로 레고 마인드스톰 로봇을 제어하는 간단한 제어 소프트웨어에 대해 MBT를 수행 하고 그 결과를 확인한다.

## 3. Model-based Test Framework

본 논문에서 제시한 MBT 프레임워크는 [그림 1]과 같이 크게 SUT(System Under Test), 테스트 환경, 그리고 신호 제어기(Signal controller)로 구성이 되어 있다. 테스트 환경은 일종의 테스트 드라이버(Test driver)로써 입력 데이터 생성기(Input data generator)와 요구사항 모델(Requirement model : 이하

모델)로 구성이 되어 있다. 모델은 소프트웨어에서 테스트하기를 원하는 요구사항을 행위 부분을 에스테렐 언어로 명세한 것이다. 입력 데이터 생성기는 모델을 근거로 소프트웨어 현재 상태에서 발생할 수 있는 입력 신호 또는 값을 임의로 선택하여 SUT 쪽으로 내보내는 역할을 수행하며 역시 에스테렐과 호스트 언어를 이용하여 명세 되어있다. 테스트 환경과 SUT사이의 입/출력 신호의 전달은 신호 제어기(Signal Controller)가 담당을 한다. 신호 제어기는 입/출력 신호의 전달 이외에 두 개의 에스테렐 프로그램(SUT와 테스트 환경)을 순서적으로 동작을 시키는 역할 및 부수적으로 필요로 하는 함수 등을 포함하고 있다. 두 개의 에스테렐 프로그램을 에스테렐에서 지원하는 병행 처리문을 사용하여 하나의 프로그램으로 구성하지 않고 중간에 신호 제어기를 두는 이유는 동기화언어의 특징인 인과관계(Casuality Relation)를 지양함으로써 입/출력 신호를 하나의 단계(Step)로 쉽게 처리하기 위해서 이다.

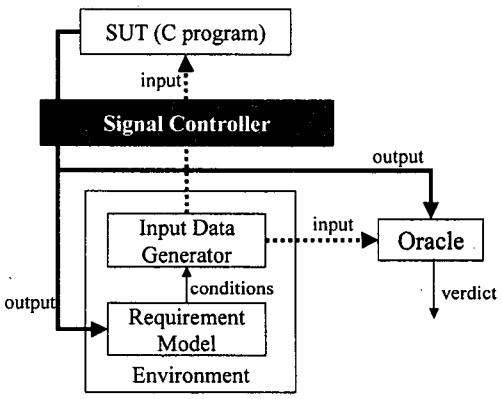


그림 1 Model-based Test Framework

본 논문에서 오라클(Oracle)은 실제적인 테스트 목적이며, 에스테렐을 이용하여 기본적으로 명제형식으로 되어 있다. SUT와 테스트 환경사이에 오고가는 입/출력은 매 단계마다 오라클로 보내지게 되며 오라클은 On-the-fly방식으로 매 단계 생성되는 입/출력이 명제에 위배되는지 않은지 판단하고, 위배가 되는 경우 즉시 테스트 케이스에 대해 실패(Fail)를 출력한다.

On-the-fly방식을 통해 테스트 케이스를 생성하기 위해 입력 데이터 생성기는 모델을 기반으로 현재상태에서 발생할 수 있는 조건에 해당하는 신호 또는 값을 임의로 선택하여 SUT로 전달한다. 현재는 임의 선택 방법만 가능하나 향후 오라클 참조를 통해 입력 데이터를 유도하는 방식을 추가할 예정이다.

## 4. Case Study

실험에서, [그림 2]에서와 같이 레고 마인드스톰(Mindstorms)으로 제작된 간단한 로봇을 제어하는 소프트웨어를 테스트 한다. 로봇의 기본적인 요구사항은 다음과 같다. 로봇은 두 개의 감광센서를 가지고 있으며, 매 1 tick마다 센서를 작동한다. 물체가 감지된 경우 센서 값은 30 이하이다. 로봇은 초기 상태는

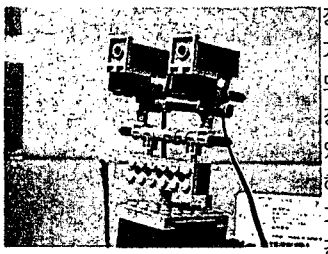

그림 2 레고 마인드스톰 로봇

정지 상태이다. 물체가 두 개의 센서 모두에서 감지가 되거나 즉, 로봇이 물체를 정면에서 똑바로 응시하고 있거나 물체를 감지하지 못하면 정지상태가 된다. 물체가 좌우로 움직임을 가지는 경우 하나의 센서에만 물체를 인식하게 되고 인식된 센서가 있는 방향으로 고개를 돌려서 따라가게 된다. 이러한 로봇의 움직임을 LTS(Label Transition System) 모델로 자세히 표현하면 다음 [그림 3]과 같다.

모델에서 알 수 있듯이 로봇의 동작은 STOP, LEFT, 그리고 RIGHT의 3가지 상태로 되어 있다. 그리고 각 상태간의 전이(Transition)을 위한 조건은 모두 4가지가 존재한다. 두 개의 센서 값으로부터 4개의 조건이 만들어 지며 LEFT 상태에서 RIGHT상태(그 반대의 경우도 같음)로 곧바로 전이를 할 수 있는 조건은 존재하지 않는다.

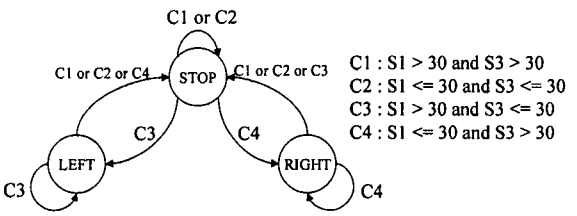


그림 3 마인드스톰 로봇의 모델과 전이조건

[그림 3] 모델을 통해 실제 에스테렐로 구현될 제어 소프트웨어에 는 최소 2개의 입력 신호에 대한 3개의 출력 신호가 존재해야 함을 알 수 있다. 하지만 실제 RCX에 탑재될 제어 소프트웨어를 에스테렐로 작성을 한 결과 실제 센서 및 모터 하드웨어를 제어하기 위한 모듈을 포함하여 모두 4개의 모듈로 구성이 되었으며, 모듈간의 내부 신호 및 모터 구동을 위한 신호 등 더 많은 신호를 사용하게 된다. 아래는 소프트웨어의 에스테렐로 작성된 코드의 일부이다.

```
% Light EYE Sensor
sensor          LIGHT_1_VALUE : integer;
% Left EYE Sensor
sensor          LIGHT_3_VALUE : integer;
% Output Siganls
output          MOVE_LEFT, MOVE_RIGHT, STOP;
% Motor
output          MOTOR_C_DIR(integer);
output          MOTOR_C_SPEED(integer);
...
% Program Part
signal      REYE_FOUND,
            LEYE_FOUND
in
...
```

```
module    LEYE_SENSOR_MOD :
sensor              LIGHT_1_VALUE : integer;
output              LEYE_FOUND;
loop
        if ?LIGHT_1_VALUE <= 30 then
                emit LEYE_FOUND;
        end if;
each tick;
end module
...
```

위와 같이 작성한 에스테렐 소스를 레고용 에스테렐 컴파일러를 통해 Hitachi CPU용 C소스코드를 생성하고 크로스 컴파일을 통해 RCX에 탑재를 하면 로봇은 물체를 감지하고 물체가 이동하는 방향으로 이동하는 것을 직접 실험해 볼 수 있다. 하지만 [그림 3]의 모델처럼 로봇이 왼쪽에서 오른쪽(또는 그 반대의 경우)로 이동을 하기 위해서는 반드시 정지 상태를 거쳤다가 오른쪽으로 이동을 하는지 정확히 측정을 할 수 없다. 왜냐하면 에스테렐로 명세 되어 있는 센서의 작동 주기가 1 tick인 경우, 실제 마인드스톰에서는 1/80초에 한번씩 센서를 작동시키기 때문에 모터의 급정지 상태가 아주 짧아서 실제 로봇의 움직임을 사람의 육안으로 미처 인식을 할 수 없기 때문이다. 본 논문에서는 제시한 MBT 프레임워크를 사용하여 위와 같은 요구사항을 소프트웨어가 만족하는지 테스트를 하였다. MBT 프레임워크를 이용함으로써 RCX를 위한 크로스 컴파일을 할 필요 없이 센서를 통해서 들어오는 값을 조절함으로써 테스트를 수행 할 수 있었다.

요구사항에 따라, 좌/우로의 움직임의 방향이 변경 될 때 로봇의 상태가 정지 상태를 거치는지에 대한 여부를 확인하기 위해 오라클을 다음과 같이 에스테렐을 이용하여 명세하였다.

```
present MOVE_LEFT then
    present pre(MOVE_RIGHT) then
        emit VERDICT(0);
    end present;
end present;
```

MOVE_RIGHT라는 신호가 현재 단계에서 SUT의 출력으로 나왔다면, 바로 이전 단계에서 MOVE_LEFT라는 신호가 SUT의 출력으로 나와서는 안 된다는 일종의 Safety Property이다. 만약 On-the-Fly 방식으로 생성되는 테스트 케이스에 MOVE_LEFT가 출력으로 나온 단계 바로 아래 단계에 MOVE_RIGHT가 출력으로 나오면 이 테스트 케이스는 실패(Fail)로 처리되고 에스테렐로 구현된 소프트웨어가 잘 못 되었음을 알 수 있다.

에스테렐로 작성된 로봇 제어 소프트웨어를 검사하기 위해서 본 논문에서 제안한 MBT 프레임워크를 이용하여 임의의 테스트 데이터(입력 신호값)를 생성하였다. [그림 4]의 왼쪽편의 그림은 12단계 만에 실패 결과를 생성한 테스트 케이스 가운데 하나이다. 11단계에서 오른쪽으로 이동하고 있던 로봇이 12단계에서 바로 왼쪽으로 이동을 하고 있는 것을 테스트케이스를 알 수 있고, 명세해 놓은 오라클을 위배하는 것을 볼 수 있다. 즉, MBT 테스트 결과를 통해 에스테렐로 작성된 소프트웨어가 요구사항을 만족하지 않음을 알 수 있다.

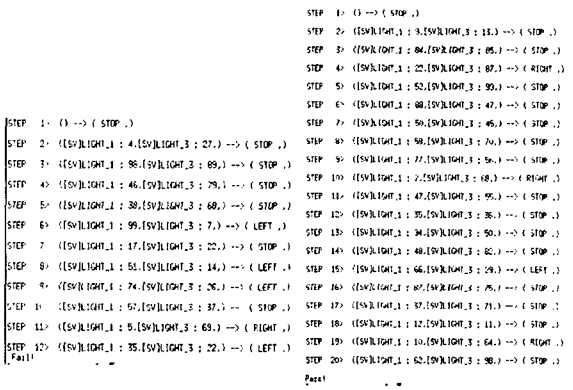

그림 4 테스트 결과

[그림 4]의 오른편의 그림은 소프트웨어를 수정한 후, 다시 테스트를 수행한 한 결과이다. 총 20단계까지 진행할 수 있도록 설정하였으며, 20단계까지 진행하면 오른쪽에서 왼쪽으로(반대의 경우 역시)바로 이동하는 경우 없이 항상 STOP 상태가 되는 것을 확인 할 수 있었다.

## 4. 결론 및 향후과제

본 논문에서는 에스테렐로 개발한 임베디드 소프트웨어를 모델 기반으로 테스트하기 위한 프레임워크의 원형(Prototype)을 제시하고, 레고 마인드스톰 로봇을 제어하는데 사용하는 소프트웨어를 MBT 프레임워크를 이용하여 테스트를 해 보았다.

에스테렐로 소프트웨어를 작성함으로써 에스테렐 컴파일러가 자동으로 생성함으로 생성된 코드의 신뢰성을 높일 수 있다. 또한 소프트웨어를 테스트하기 위한 테스트 환경과 오라클 모두 에스테렐 언어를 이용함으로서 대상 환경과의 외부환경과 하드웨어 인터페이스를 쉽게 표현하고 입/출력 결과에 대해 쉽게 확인 할 수 있는 장점을 가지고 있다.

본 논문에서 제시한 MBT 프레임워크의 원형에서는 핵심이라 할 수 있는 입력 데이터 생성기에서 생성하는 입력 신호를 임의로 결정하는 관계로 주어진 오라클을 반영하여 시스템을 움직임을 유도 할 수 없으므로 자동으로 생성되는 입력 값을 오라클을 반영하여 유도할 수 있거나, 모델의 모든 상태를 방문할 수 있도록 입력 값 생성기를 개선할 필요가 있다. 이와 함께 에스테렐을 모르는 사람도 쉽게 테스터를 자동으로 구성할 수 있는 환경을 개발을 해야 할 필요가 있다.

## 5. 참고자료

[1] 강인혜, 양진석, 초보자를 위한 에스테렐 프로그래밍, 홍릉과학 출판사, 2005

[2] Axel Belinfante, Lars Frantzen, Christian Schallhart, Tools for Test Case Generation, 2004