임베디드 시스템을 위한 빈도 기반 모델 검증 기법

이성훈, 이동현, 인호*
고려대학교 정보통신대학
e-mail : shun2k@korea.ac.kr, tellmehenry@korea.ac.kr, hoh_in@korea.ac.kr

Frequency Based Model Checking for Embedded System

Sung-Hoon Lee, Dong-Hyun Lee, Hoh Peter In
Dept of Information and Communication, Korea University

요 약

Model Checking 기법은 시스템을 검증하고 반례를 제시해 주는 검증 방법으로 최근에 여러 분야에서 쓰이고 있다. 하지만 임베디드 시스템과 같이 그 검증에 있어서 시간, 자원적인 제한을 가지고 있는 분야에서는 검증할 항목을 임의로 선택해서 하는 경우가 대부분이다. 따라서 이 논문에서는 검증에 있어서 우선시 해야 할 기능들을 효율적으로 선정하는 빈도 기반 모델 검증 기법을 제안하고자 한다.

1. 서론

Model Checking[1] 은 시스템의 안전성, 설계상 오류 여부를 기능 중심으로 모델 내의 완전한 패스 검색을 통하여 검증하는 기법이다. 최근에는 하드웨어 검증을 위해 개발되었던 이 기술을 모델 변환을 통하여 소프트웨어의 검증에까지 그 영역을 넓혀오고 있다.

그러나 Model Checking 은 시스템이나 응용 소프트웨어의 모든 가능성을 논리적으로 따라가기 때문에 많은 Computing 자원을 요구하는 것이 일반적이다. 그리고 Time to Market 이 중요한 현대 시장에서 제품의 모든 기능을 검증하고 하는 데에는 많은 시간과 자원이 요구된다. 따라서 현실적으로나 경제적으로나 그 효율성이 떨어지므로 충분한 기능 검증이 이루어지지 않고 제품이 출시되는 경우가 많다.

이에 대한 해결책으로 할당된 시간과 자원을 가장 효율적으로 사용할 수 있고 모델 검증에 적용 가능한 검증할 기능 간의 우선 순위를 결정할 수 있는 방법이 필요하다.

본 논문에서는 임베디드 시스템에서 한정된 자원과 한정된 시간을 가지고 검증을 하기 위하여 UML[2] 기반 모델 분석을 통하여 검증 기능 간의 우선 순위를 매길 수 있는 Frequency Based Model Checking 을 제안한다.

2. 배경지식

기존의 모델 검증 방법과 UML 에 대해 간략히 소개한다. 먼저 기본이 되는 Model Checking 을 간략히 설명하고, Model Checking 에 쓰이는 Kripke 구조와 CTL 로직에 대해서 설명하고 장점 및 한계점을 다룬다. 마지막으로 UML 에 대한 전반적인 내용을 다룬다.

2.1 Model Checking

모델 검증 기법은 우선 시스템을 Model 화 하고 검증하고자 하는 조건을 Formula 로 나타낸 후 그것을 검증한다. 일반적으로 Model 은 kripke structure 로 표현하고 Formula 는 CTL(Computation Tree Logic)을 사용한다.

Kripke Structure 는 총 4 개의 Tuple 로 구성된다. 유한한 상태공간의 집합인 S 과 모델 내 속성들의 집합인 AP, 집합에 속하는 초기 상태공간의 집합 S0 과 각 상태공간 간의 관계인 R, 그리고 각 상태공간에서 참인 AP 의 집합 L 로 구성된다. 이를 언어로 정형 명세로는 **M = (S, S0, R, L) over AP** 로 표현할 수 있다.

CTL 은 Path quantifiers 와 temporal operators 로 구성되어 있는 formula 로써 reactive 시스템에서 일련의 states 간의 transition 을 묘사하는 정형기법인 Temporal logic 의 한 종류이다. 기본적인 Path quantifiers 와 temporal operators 는 다음과 같다;

- Path quantifiers
  - ✓ A : 모든 Computation Path 의 존재
  - ✓ E : 어떤 Computation Path 의 존재
- Temporal Operator
  - ✓ X : Path 의 2 번째 state 에 속성이 존재
  - ✓ F : Path 의 어느 state 에 속성이 존재
  - ✓ G : Path 의 모든 state 에 속성이 존재
  - ✓ U : 2 개의 속성에 대한 연산자
    - ■ Path 의 state 에서 2 번째 속성이 나오기 전의 모든 state 에 1 번째 속성 존재

2.2 Unified Modeling Language

UML 은 2003 년 The Rational Edge 에서 처음 소개되어 현재 2.0 버전까지 나와있다. UML 은 기존에 여러 가지 방법으로 제시되어 온 컴퓨터 어플리케이션 모델링 방법을 통합하여 표준화 한 언어로 프로그래밍

언어에 독립적이라는 특성을 지니고 있다.

어플리케이션 개발 시, 이를 쉽게 이해할 수 있도록 도와주는 여러 유형의 다이어그램을 가진다. 그 중 가장 유용한 표준 다이어그램을 소개한다.

- Use Case Diagram : 기능 단위의 설명
- Class Diagram : Entity 들의 관계 설명
- Sequence Diagram : 특정 UseCase 의 흐름 설명
- State Chart Diagram : 클래스 내의 다양한 상태의 모델링 및 이동 설명
- Activity Diagram : 활동을 처리하는 동안 클래스 객체들간의 제어 흐름 설명
- Component Diagram : 시스템 속 컴포넌트들의 관계 설명
- Deployment Diagram : 하드웨어 환경에 시스템의 물리적 전개 설명

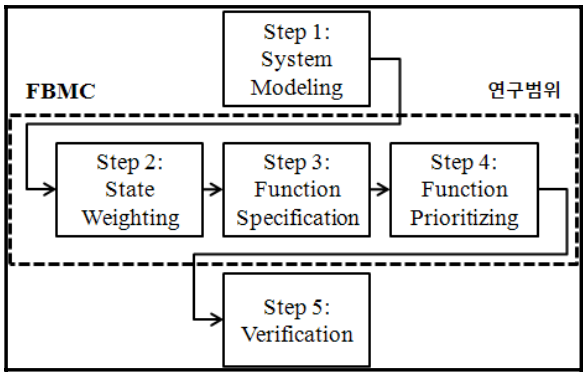
이외에도 Collaboration Diagram, Object Diagram 등이 있다.

3. 빈도 기반 모델 검증

빈도를 기반으로 하는 모델 검증 방법을 제시한다.

FBIMC 는 기능에 대한 우선순위를 시스템 모델의 구조적인 상관관계에서 추정하여 매김으로써 주어진 시간 내에 좀 더 체계적이고, 효율적인 검증이 가능하다.

3.1 Overview



(그림 1) FBMC Process

빈도 기반 모델 검증 (FBMC)은 다음과 같은 단계를 거친다;

Step1. System Modeling: 검증하고자 하는 시스템을 모델링 한다.

Step2. State Weighting: 모델 상태간의 연결 빈도에 따라서 가중치를 매긴다.

Step3. Function Specification: 검증하고자 하는 기능을 상세화 한다.

Step4. Function Prioritizing: 상세화된 기능의 속성을 포함한 상태 공간의 가중치의 합을 기반으로 우선 순위를 설정한다.

Step5. Verification: 기능들을 우선 순위 검증한다.

3.2 Step 1: System Modeling

먼저 시스템을 모델링하는 단계이다. 검증의 대상이 되는 시스템을 Kripke Structure 로 표현한다. Kripke Structure 로 표현된 시스템은 기본 적으로 상태 공간과 상태 공간 간의 변화, 각 상태에서 정의되는 속성, 그리고 시스템 전반의 모든 속성의 종류들로 구성된다.

3.3 Step 2: State Weighting

다음 단계에서는 전체 시스템에서 얼마나 많이 거쳐가게 될 것이며, 그 거쳐가는 빈도 비율에 따라 그 상태 공간의 중요도를 결정하게 된다.

각 상태공간의 연결상태를 살펴보자. 상태 공간으로 들어오는 방향으로 연결되어 있는 간선과 다른 상태 공간으로 나가는 방향으로 연결되어 있는 간선을 살펴본다. 모든 간선의 개수를 더하게 되면, 상태 공간의 전환 가능한 모든 가지 수가 나오게 되고, 그 중요도는 모든 간선의 개수를 더한 수가 된다. 예를 들어 들어오는 간선의 수가 2 이고, 나가는 간선의 수가 3 이라 하면, 그 상태공간의 중요도는 5 가 된다. 여기서 또 생각해야 할 것은 바로 self-loop 인데, 이 경우 상태의 변화가 일어나는 것이 아니기 때문에 self-loop 의 경우에는 중요도를 정하는데 영향을 미치지 않도록 한다.

다른 상태 공간과 연결되어 있는 간선의 개수가 많을 수록 곧 그 상태 공간이 다른 상태 공간으로 변화할 가능성이 다른 상태 공간보다 높으며, 보다 많은 기능 패스에 포함될 가능성이 높으므로 보다 중요한 상태 공간이라 생각할 수 있다.

3.4 Step 3: Function Specification

그 다음 시스템에서 검증되어야 하는 모든 기능들을 상세화한다. 이는 모델에서 실제 접근 가능한 모든 패스들이 실제 시스템에서 의미를 지니는 기능이 아니므로 이에 우선 순위를 할당하여 검증한다는 것은 무의미 한 자원낭비라 할 수 있으므로 개발자가 의미 있는 기능을 상세화해 주어야 한다.

먼저 CTL 표현을 ¬(not), ∧(and)나 ∨(or)만으로 표현을 전환한다. 예를 들어, AG(start ->AF(heat))는 AG (¬start∨AF(heat))로 전환한다.

3.5 Step 4: Function Prioritizing

다음 단계에서 기능의 상세를 구성하는 속성을 가지고 있는 상태 공간의 Weight 의 합으로 그 기능들 간의 우선 순위가 결정된다.

상세에서 각 속성이 ∨(or)로 묶여져 있는 경우 각 속성을 가지는 상태공간들을 합친 상태공간이 그 목적 상태공간이 된고 이들의 weight 를 모두 더한 값이 그 상세의 Weight 가 된다. 만약 중간에 ∨가 아닌 ∧로 묶여 있다면 두 속성을 지닌 상태공간의 교집합이 그 목적 상태 공간이 될 것이다.

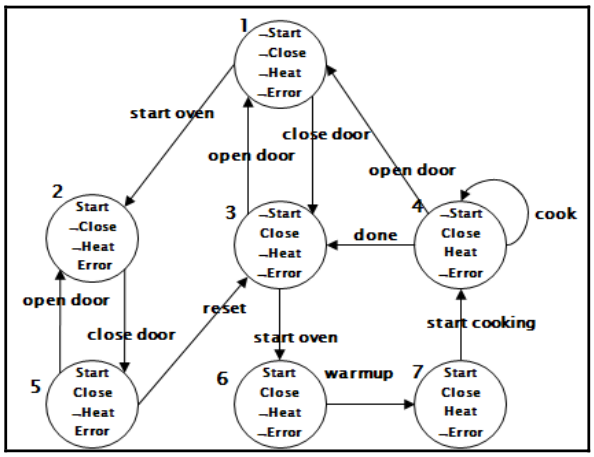
3.6 Step 5: Verification

다음과 같이 각 기능 상세간의 우선 순위가 정해지면, 해당 우선 순위에 따라 모델 검증의 순서를 재구성하여 실제 모델 검증에 들어간다. 모델의 구조 분석 결과에 기반한 검증 순서만 바꾼 것이기 때문에 기존의 모델 검증기를 써서 검증을 하면 된다. 이렇게 우선 순위에 따라 기능 검증 결과 값이 False 가 나올 가능성이 높은 상세를 우선시 하여 검증을 함으로써 보다 시스템의 결함을 빨리 발견할 수 있다.

4. 사례연구

Microwave Model[1]을 통해 FBMC 의 접근 방법을 살펴 본다. Microwave 는 일상 생활에서 가장 많이 쓰이는 제품 중 하나이며, 기본적인 행위는 복잡하지 않으면서 FBMC 를 보여주기에 충분한 수의 기능 명세를 지니고 있다.

4.1 Step 1: System Modeling



(그림 2) Microwave Model

위 모델은 각 상태 공간내에 참인 AP 와 거짓인 AP 가 표시되어 있다. 간선 위에 표시되어 있는 것은 Kripke structure 의 부분이 아니라 상태 전이를 일으키는 Microwave 의 행위를 표시한 것이다.

4.2 Step 2: State Weighting

위의 모델에서 1 번 상태공간을 살펴보자. 1 번 상태 공간은 3 번과 4 번 상태 공간으로부터 들어오는 방향의 간선이 연결되어 있고, 2 번과 3 번 상태 공간으로 나가는 방향의 간선이 연결되어 있다. 따라서 상태 공간 1 의 경우 다른 상태 공간으로의 전환 가능한 수가 모두 4 가지이므로 그 중요도는 4 가 된다. 마찬가지로 3 번 상태공간의 경우, 1 번과 6 번 상태 공간으로 나가는 방향으로 간선을 가지고 있고, 1 번 4 번 5 번 상태 공간으로부터 들어오는 방향으로 간선을 가지고 있다. 따라서 3 번 상태 공간의 경우 그 중요도는 5 가 된다. 모든 상태공간의 weight 는 다음의 표와 같다.

State	Weight
1	2 (in) + 2 (out) = 4
2	2 (in) + 1 (out) = 3
3	3 (in) + 2 (out) = 5
4	1 (in) + 2 (out) = 5
5	1 (in) + 2 (out) = 3
6	1 (in) + 1 (out) = 2
7	1 (in) + 1 (out) = 2

<표 1> 상태공간의 Weight

4.3 Step 3: Function Specification

위의 Microwave Model 에서 검증할 기능을 상세화한 CTL 표현을 살펴보면, 다음과 같다.

- **AG(start ->AF(heat))**
- **AG(close->AF(start))**
- **AG(error->AF(-start))**

위의 세 CTL 문장은 다음의 CTL 문장과 각각 동치를 이룬다;

- **AG (¬start ∨ AF(heat))**
- **AG (¬close ∨ AF(start))**
- **AG (¬error ∨ AF(-start))**

4.4 Step 4: Function Prioritizing

첫 번째 상세를 살펴 보면 ¬start 라는 속성이 heat 라는 속성과 ∨(or)로 묶여져 있다. 이 경우 ¬start 의 속성을 가지는 상태공간인 1,3,4 과 heat 의 속성을 지닌 4,7 상태공간을 합친 상태공간인 1,3,4,7 이 검증의 대상이 되기 때문에 이들 상태 공간의 weight 를 모두 더한 16 이 AG (¬start ∨ AF(heat))의 weight 가 된다. 같은 방법으로 나머지 2 개 상세의 weight 와 우선 순위는 다음의 표와 같다.

Prior.	Function	TS	Weight
2	AG(¬start ∨ AF(heat))	1,3,4,7	16
3	AG(¬close ∨ AF(start))	1,2,5,6,7	14
1	AG(¬error ∨ AF(-start))	1,3,4	18

<표 2> Function Priority

4.5 Step 5: Verification

기존에 나와있는 모델 검증기를 이용하여 위의 우선순위에 따라 AG(¬error ∨ AF(-start))를 먼저 검증하고, 그 다음으로 AG(¬start ∨ AF(heat))과 AG(¬close ∨ AF(start))를 순서대로 검증한다.

5. 결론

본 논문에서 제시한 빈도 기반 모델 검증 기법은 하나의 제품에 대하여 검증이 필요한 기능들 중에서 모델 상에서 차지하는 역할의 중요도에 따라서 검증 순서를 정하고 있다. 이런 기법을 통해서 정해진 테스트 시간을 좀 더 중요한 기능을 검증하는 데 사용함으로써, 검증의 효율성을 높일 수 있다. 추가적인 연구 방향은 우선, 모델링 상에서의 기능의 중요도가 제품 상에서의 중요도와 일치하는 지에 대한 검증의

수행이 필요하다. 그 제품의 실 사용자가 생각하는 기능의 중요도와 제품의 테스터 입장에서 생각하는 기능의 중요도가 다르다. 따라서 제품의 결함을 발견하는 데에 있어서 어느 측면에서 검증이 진행되어야 좀 더 나은 검증 결과를 이끌어 낼 수 있는지에 대한 연구가 필요하다.

Acknowledment

이 논문(저서)은 2007 년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2007-331-D00360)

참고문헌

[1] Model Checking (E.M.Clarke, O.Grumberg, D.A.Peled), MIT Press (ISBN 0262032708)
[2] Unified Modeling Language (www.ibm.com/developerworks/kr/library/769.html)
[3] Risk based testing - How to choose what to test more and less (Hans Schaefer, Software Test Consulting, Norway)