제22회 한국정보처리학회 추계학술발표대회 논문집 제11권 제2호 (2004. 11)

임베디드 소프트웨어를 위한 코드 기반 모델 체킹 도구의 요구사항*

이태훈, 권기현
경기대학교 정보과학부

e-mail:{taehoon,khkwon}@kyonggi.ac.kr

Requirements for Code-Based Model Checking for Embedded Software

Taehoon Lee, Gihwon Kwon
Department of Computer Science, Kyonggi University

요 약

테스팅이 오류의 존재를 증명할 수 있는데 반해서, 정형 검증 기술은 시스템에 오류가 존재하지 않음을 증명할 수 있다. 모델 체킹은 이런 정형 검증 기술 중의 하나이다. 최근에 모델 체킹을 이용하여 코드를 자동으로 검증하려는 연구들이 많다. 하지만 이런 연구는 일반적인 환경에서의 검사만을 할 수 있다. 반면에 임베디드 소프트웨어는 실시간성, 외부 환경, 다중 스레드 등의 다양한 특성이 존재한다. 따라서 임베디드 소프트웨어와 같이 안전한 소프트웨어 시스템을 필요로 하는 환경을 위한 모델 체킹을 수행하기는 힘들다. 본 논문에서는 임베디드 소프트웨어에 대한 모델체킹 도구가 검증할 수 있어야 하는 실시간 시스템의 검증, 외부 환경에 대한 고려, 다중 스레드 시스템의 검증 등을 설명하고, 기존 도구들이 얼마나 만족하고 있는지 조사해본다.

1. 서론

1990년대 이후로 컴퓨터는 어디에서든 접할 수 있게 되었다. 임베디드 시스템 프로그래밍에 따르면 "1998년도에 팔린 99%이상의 마이크로프로세서가 임베디드 시스템에서 사용됐다." 라고 한다.[1] 이처럼 임베디드 시스템의 사용과 중요성은 계속 증가하고 있다. 무엇보다 몇몇 경우 임베디드 시스템은 안전 필수 시스템이다. 이런 시스템은 가장 신뢰성 있는 시스템이 되어야 한다. 그러나 기존의 테스팅과 같은 방법은 시스템에 오류가 없다는 것을 보장하지 못한다.

신뢰성 있는 시스템을 개발하기 위해서는 사람의 많은 노력이 필요하게 된다. 또한 사람은 소프트웨어를 디자인, 코딩, 테스팅을 반복적으로 개선하게 된다. 일반적인 질문은 이런 신뢰성 있는 시스템을 어떻게 쉽게 개발할 수 있을까 이다. 이를 위해 많은 방법이 제안되어졌다. 이런 방법 중에 하나는 정형검증 기법이다.

정형검증 기법은 오류가 없음을 증명한다. 반대로 테스팅은 오류의 존재를 증명한다. 모델체킹[2]은 정형검증 기법 중에 하나이다. 주어진 모델 M 과 시제 논리식 Φ를 받아서 자동으로 모델이 시제 속성을 만족하는지 검사한다. 만일 만족하지 않는다면 모델체킹은 왜 만족하지 않는지에 대해서 반례를 생성하게 된다.

모델체킹은 IBM[3], Microsoft[4], INTEL[5] 과 같은 업체에서도 많이 사용되고 있고, 이런 회사들은 자체 모델 체킹 도구를 가지고 있다. 이런 모델 체킹의 사용처는 점점 더 증가하고 있다.

현재 많은 사람들이 모델 체킹을 소프트웨어 검증에 적용시키려고 노력하고 있다. UML 모델과 같은 것과 일반 적으로 사용되는 언어에 대한 검증이다. 그러나 일반적으로 하드웨어 시스템의 검증에 비해서 소프트웨어 모델체킹은 수행하기 어렵다. 대부분의 소프트웨어 시스템은 상태공간으로 표현하면 매우 복잡하게 된다. 또한 많은 소프트웨어 시스템은 무한 상태를 가지고 있고, 이런 시스템은 모델 체킹

* 본 연구는 과학기술부 목적기초연구 (R05-2004-000-10612-0) 지원으로 수행되었음.

도구로 검증을 수행할 수 없다. 따라서 무한 상태 공간을 유한 상태공간으로 만들어주는 기술이 필요하게 된다.

일반적인 프로그램 코드를 검증하기 위한 많은 도구와 기술들이 연구되어 왔다. 그중 대표적인 연구는 Bandera[6], SLAM[4], MAGIC[7] 등이 있다. Bandera 는 캔사스 대학에서 개발 중인 자바 검증 도구이다. SLAM 은 마이크로 소프트에서 개발 중인 C 검증 도구이다. MAGIC는 CMU에서 개발 중인 C 언어 검증 도구이다. 이렇게 일반적인 언어를 개발하기 위한 여러 도구들이 나와 있지만, 일반적인 환경에서의 검증만을 지원한다. 이런 접근 방법은 때때로 임베디드 소프트웨어를 검증하는데 알맞지 않을 수도 있다. 따라서 본 논문에서는 임베디드 소프트웨어를 위한 모델체커가 가지고 있어야할 요구사항들을 설명하고 기존 도구들이 어느 정도 이런 요구사항을 만족하고 있는지 살펴본다.

본 논문의 구성은 다음과 같다 2장에서는 기존의 모델 체킹 도구에 대해서 설명한다. 3장에서는 임베디드 시스템을 위한 코드기반 모델체킹 도구가 가져야할 요구사항을 설명한다. 4장에서는 기존의 도구가 얼마나 우리의 요구사항을 만족하는지 살펴본다. 그리고 5장에서 결론을 맺는다.

2. 코드기반 모델체킹 도구

2.1 Bandera

Bandera 는 캔사스 대학에서 개발 중인 자바 검증 도구이다. Bandera 의 기본 목적은 자바 소스코드를 입력받아, 검증에 알맞은 유한 상태 모델을 추출해서 SMV 혹은 SPIN 과 같은 기존 모델 체킹 도구에서 실제 검증을 수행하는 것이다.

이를 위해서 Bandera 는 자바 소스코드를 입력 받은 후 Jimple 이라는 매개 언어로 변경한다. 이는 다시 여러 종류의 모델 체킹 도구의 입력언어 형태로 변경이 된다. Bandera 의 추상화는 Jimple 코드 상에서 수행되고, 기본적으로 제공되는 추상화 기법은 자료 추상화 이다.

Bandera에서 속성은 BSL 이라는 형식으로 속성을 기술한다. 일반적인 프로그래머들이 CTL 혹은 LTL 과 같은 시제 논리식에 익숙하지 않기 때문에 사용자가 원하는 속성을 작성하기가 힘들게 된다. 따라서 Bandera 에서는 많이 사용되는 속성을 패턴 화하여 사용자들이 패턴에 알맞게 입력을 하면, 도구에서 자동적으로 시제 논리식으로 변환을 시켜준다.

Bandera 에서는 SPIN, NuSMV 와 같은 기존의 모델체킹 도구를 이용하여 모델 체킹을 수행한다. 만일 검증을 수행했을 때 반례가 생성이 되었다면, Bnadera 에서는 반례를 Jimple 수준으로 해석을 하고, 다시 자바 수준으로 해석을 해서 어떻게 오류가 발생하였는지 알려주게 된다. 이를 통해서 사용자가 좀더 쉽게 에러가 어디서 일어났는지 알 수 있게 된다.

2.2 SLAM

SLAM 은 마이크로 소프트에서 개발 중인 C 언어 모델 체킹 도구이다. 총 3가지의 툴로 구성 되어있다. C2BP[5]는 C 언어를 술어 추상화를 이용해서 이진 프로그램(Boolean Program)이라는 추상화된 프로그램으로 변환을 수행한다. 이진 프로그램은 C 프로그램의 제어구조를 가지고 있지만 술어 에 대응되는 이진 변수만을 가지고 있는 프로그램이다. 원래 프로그램보다 좀더 많은 행위를 가지고 있고 따라서 이진 프로그램에서 어떤 경로를 찾아낸다고 하더라도, 원래 프로그램에서도 가능한 경로인지 검사를 해야 한다. Bebop은 이진 프로그램에 대해서 모델 체킹을 수행한다. 프로그램에서 만족해야할 속성은 SLIC[12] 라는 C언어 형태의 명세로서 속성을 명세한다. 이진 프로그램에 대해서 Bebop[13] 이 검사를 하게 되면 참 혹은 반례를 되돌려준다. 반례가 생성되었을 경우 NEWTON[14] 은 반례가 실제 가능한 반례인지 검사를 한다. 검사를 해서 반례가 실제 불가능한 반례라면 그 경로로 도달하지 않도록 해주는 술어를 추가 시켜준다. 그리고 추가된 술어를 이용해서 C2BP를 수행하게 된다.

SLIC에서 가능한 속성명세는 assert 의 도달가능성을 기본적으로 하고 있다. 그리고 일반적인 시제논리에 대한 검사는 불가능하다.

2.3 MAGIC

MAGIC은 CMU에서 개발 중인 모델 체킹 도구이다. MAGIC 의 특징 중 첫 번째는 검사할 속성에 있다. 기존의 검증 도구는 속성으로는 assert 문에 도달 가능한지와 간단한 안전성 속성에 대해서 위반하지 않는지에 대해서만 검증이 가능했다. 따라서 사용자가 바라는 복잡한 속성에 대해서는 검증이 힘들었다. 하지만 MAGIC 의 경우에는 속성을 LTS 로 기술하여 좀더 복잡한 속성에 대해서 쉽게 기술할 수 있도록 하였다. LTS로 속성을 기술하게 되면 기존의 시제 논리식에 비해 사용자가 쉽게 시스템이 만족해야하는 속성을 기술할 수 있게 된다. MAGIC 에서의 모델체킹 방법은 함수에 대한 LTS 모델을 생성하고, 명세로 사용되는 LTS 모델을 만족하는지 살펴본다.

두 번째는 조립성(Compositionality)으로서 MAGIC

은 각각의 함수마다 프로시저 추상화(PA)이라는 이름의 구조를 만든다. PA 안에는 함수에 대한 선언과 가드 조건과 그에 따른 LTS 에 대한 튜플이 있다. 이런 PA 는 두 가지 역할로서 사용이 가능하다. 하나는 함수의 행위를 설명하는 역할이고, 하나는 외부에서 함수 호출을 할 경우 사용하게 되는 경우이다. MAGIC은 함수 호출 그래프를 생성해서 각각의 어떤 함수를 검사할지에 따라 PA를 생성하고, 외부에서 호출하는 PA를 검증할 경우엔, 가정 PA로서 사용된다. 따라서 가정 PA를 먼저 만들어 놓았을 경우 그 함수를 호출하는 함수에 대한 검증도 가능하게 된다.

3. 임베디드 소프트웨어 모델체킹 도구 요구사항

3.1 환경에 대한 정의

일반적인 소프트웨어 시스템과는 다르게 임베디드 시스템은 외부 환경에서 받는 이벤트를 이용해서 행위를 수행한다. 따라서 외부 환경에 대한 정의를 통해서 전체 임베디드 소프트웨어의 동작을 검증할 필요가 있다. 외부 환경에서의 이벤트에 따라서 시스템의 제어 흐름이 변경될 수도 있기 때문에 외부 환경도 모델링을 하여 환경에 대한 행위를 설명할 수 있어야 한다. 또한 임베디드 시스템은 하드웨어와 소프트웨어로 구성되어 있다. 임베디드 소프트웨어를 검증하기 위해서는 임베디드 하드웨어에 대해서도 알고 있어야지 정확한 검증이 가능하게 된다. 따라서 하드웨어에 대한 모델링도 수행할 수 있어야 한다.

3.2 실시간성

실시간성이란 지정된 구문이 정해진 시간이내에 수행되야 함을 의미 한다. 많은 수의 임베디드 시스템은 실시간 시스템이다. 따라서 임베디드 시스템을 검증하기 위해서는 실시간 시스템을 검증할 수 있는 능력을 가지고 있어야 한다.

3.3 다중 스레딩

현재 많은 시스템은 다중 스레딩 환경에서 구현되어 있다. 임베디드 시스템 역시 똑같이 다중 스레딩 환경에서 구현되어 있다. 따라서 임베디드 시스템에서 검증을 수행하기 위해서는 이런 다중 스레딩으로 구성된 소스 코드에 대해서도 검증을 수행할 수 있어야 한다. 또한 대부분의 다중 스레딩 환경은 공유 메모리 구조를 이용하고 있다. 이런 공유 메모리를 기반으로해서 여러 프로세스 들이 동기화와 통신을 수행하게 된다. 이런 다중 프로세스의 동기화와 통신에 대한 소스코드 수준의 검증이 필요하게 된다.

3.4 다양한 속성에 대해 검증

다양한 형태의 임베디드 시스템이 존재하기 때문에 시스템이 올바로 동작하는지를 검사하기 위한 다양한 속성이 필요하다. 기본적인 안전성 속성과 궁극성 속성에 대해서 검사가 가능해야하고, 시스템의 행위가 원하는 대로 행동하는지도 검증해 줄 수 있어야 한다.

3.5 실제 코드에 적용 가능

실제 임베디드 시스템은 작은 크기에서부터 큰 규모의 시스템 까지 다양한 시스템이 있다. 시스템이 커질수록 더 많은 행위를 가지게 되고, 검증을 수행하기 어려워진다. 따라서 큰 시스템의 검증이 힘들게 된다. 모델체킹 도구는 실제 사용되는 코드와 같이 큰 규모의 코드를 검증 가능해야만 한다.

4. 기존 도구의 요구사항 만족 여부

4.1 Bandera

Bandera는 일반적인 자바 언어 검증을 위해 개발된 도구이다. 하지만 이 도구의 경우 일반적인 자바 환경을 대상으로 개발 되었다. 따라서 외부 환경에 대한 고려는 되어 있지 않다. 또한 아직 실시간 시스템에 대한 검증도 역시 불가능하다. 하지만 자바에서 지원하는 다중 스레드를 검증할 수 있고, CTL 과 LTL 등의 여러 속성에 대해서 명세가 가능하다. 또한 일반적인 프로그래머가 CTL 혹은 LTL 형식의 속성명세가 어렵기 때문에 패턴을 이용해서 쉽게 검증가능하게 만들었다. 하지만 Bandera 의 경우 실제 프로그램을 검증할 수 있을 만큼 확장성이 크지 않다. 따라서 좀더 큰 시스템을 검증 할 수 있도록 확장이 필요하다.

4.2 SLAM

SLAM 은 Device Driver를 검증하기 위해 개발되었다. 검증방법은 오류 상태에 도달 가능한지를 검사한다. SLAM 에서는 외부의 환경은 Windows 시스템이다. 그 이외에 특별히 모델링은 불가능하다. 또한 윈도우용으로 개발되었기 때문에 실시간성에 대한 검증도 불가능하다. 또한 소개 논문에서 다중 스레딩에 대한 내용도 포함하고 있지 않다. SLAM 에서 검증할 수 있는 속성은 간단한 안전성 속성이다. 따라서 좀더 다양한 속성을 검사할 수 있어야 한다. SLAM 은 이미 실제 사용되고 있는 윈도우의 DDK 에 있는 소스코드에 대한 검증에 사용되었고

효율적으로 검증을 수행했다.

4.3 MAGIC

CMU에서 개발 중인 MAGIC은 속성을 LTS로 표현하는 것이 특징이다. 또한 현재 시스템이 사용하는 외부 시스템은 PA 라는 형식으로 표현하고 이들을 조립하는 방법으로 검증을 수행한다. 따라서 외부 라이브러리 혹은 환경에 대해서 PA 로 정의를 해주면 모델링과 검증이 가능하다. 하지만 MAGIC은 실시간 시스템에 대해서 검증은 불가능하다. 또한 아직까지 다중 스레딩을 지원하지 않는다. 반면 속성을 LTS 로 기술하기 때문에 다양한 속성에 대해서 기술이 가능하고 CTL 혹은 LTL 에 비해서 좀 더 쉽게 표현이 가능하다. 또한 비교적 큰 규모의 시스템에서도 적용이 가능하다.

5. 결론

지금까지 개발된 코드기반 모델 체킹 도구를 살펴보고, 임베디드 시스템을 위한 모델 체킹 도구가 가져야할 요구사항이 무엇인지 살펴보았다. 이를 통해서 기존의 도구들이 가지고 있는 특성을 분석할 수 있었고, 어떤 개선 사항들을 가지고 있는지 살펴볼 수 있었다.

하지만 본 논문에서 제시하는 요구사항을 모두 만족하고 있는 모델 체킹 도구는 없다. 따라서 임베디드 시스템을 위한 모델 체커의 요구사항을 만족할 수 있는 모델 체킹 도구의 개발이 필요하다.

참고문헌

[1] Scott Briggs, "Manage your Embedded Project", Embedded Systems Programming, pp 26-46, April 2000.

[2] E.M. Clark, O. Grumberg and D.A. Peled, Model Checking, The MIT Press, 1999.

[3] S. Ben-David, C. Eisner, D. Geist, Y. Wolfsthal "Model Checking at IBM" , Formal Verification and Testing Technologies in System Design, 2003

[4] T. Ball and S.K. Rajamani, "Automatically Validating Temporal Safety Properties of Interfaces," in Proceedings of SPIN 2001, pp. 103-122, 2001.

[5] http://www.intel.com/research/scl/index.htm

[6] J. Corbett, et.al., "Bandera: Extracting Finite-state Models from Java Source Code," in Proceedings of International Conference on Software Engineering, 2000.

[7] S. Chaki, E.M. Clarke, A. Groce, S. Jha and H. Veith, "Modular Verification of Software Components in C," IEEE Transactions on Software Engineering, Vol.30, No.6, pp.388-402, 2004.

[10] E.M. Clarke, D. Kroening, N. Sharygina and K. Yorav, "Predicate Abstraction of ANSI-C Programs using SAT," Formal Methods in System Design, 2004.

[11] S. Graf and H. Saidi, "Construction of Abstract State Graphs with PVS," in Proceedings of Computer Aided Verification, pp.72-83, 1997.

[12] T. Ball and S.K. Rajamani, "SLIC: A Specification Language for Interface Checking (of C)," Technical Report MSR-TR-2001-21, 2001.

[13] T. Ball and S.K. Rajamani, "Bebop: A Symbolic Model Checker for Boolean Programs," in Proceedings of SPIN 2000, pp.113-130, 2000.

[14] T. Ball and S.K. Rajamani, "Generating Abstract Explanations of Spurious Counterexamples in C Programs, Technical Report, MSR-TR-2002-09, 2002.