검증 반례 재연을 통한 모델 기반 커널 테스팅

김문주, 홍창기, 홍신

한국과학기술원 전산학과 Provable SW 연구실
대전 유성구 과학로 335
http://pswlab.kaist.ac.kr
moonzoo@cs.kaist.ac.kr
kornin83@kaist.ac.kr
hongshin@kaist.ac.kr

요약: 최근 내장 시스템의 용도에 따라 이에 적합한 운영체제 커널을 제작하고자 하는 요구가 증가되고 있다. 하지만 커널 개발은 코드의 복잡성, 유닛 테스팅의 어려움, 동시성으로 인한 동작 경우의 수 증가, 적합한 툴의 부재 등의 이유로 인해 아직도 개발 및 테스팅 비용이 큰 실정이다. 이러한 커널 개발 및 테스팅의 어려움을 극복하기 위해, 모델 기반의 커널 테스팅 (MOKERT) 프레임워크를 제안한다. 본 프레임워크는 모델 검증 기법과 테스팅 기법을 조합하여 운영체제 커널 내의 동시성 오류 검출을 지원한다. 본 연구에서는 검증하고자 하는 리눅스 2.6 커널의 파일시스템 구성요소에 MOKERT 프레임워크를 적용하여 이의 효율성을 증명하였다.

핵심어: 모델 검증 기법, 테스팅, 검증 반례 분석, 모델 추출

1. 서론

소프트웨어 테스팅은 보통 소프트웨어의 총 개발 기간 및 가용 자원의 50% 이상을 차지한다 [1]. 여러 소프트웨어 제품들 가운데 운영체제 커널은 개발하고 분석하기 가장 어려운 소프트웨어 중 하나라고 알려져 있으며, 그 이유는 다음과 같다.

- 코드의 복잡함
- 동시성으로 인한 동작 경우의 수 증가
- 유닛 (Unit) 테스팅의 어려움
- 적합한 툴의 부재

커널의 복잡성과 크기를 고려할 때, 커널을 구성요소 단위로 분석하는 것이 절실하지만, 위의 이유들로 인해 구성 요소 단위로 커널을 분석하는 것은 실질적으로 매우 어렵다. 하지만 내장 시스템 (embedded system)이 일상 생활 모든 영역에 널리 퍼짐에 따라 특정 목적을 위한 운영체제 (휴대폰 또는 센서 네트워크를 위한 운영체제) 개발의 필요성이 점차 증가하고 있다. 따라서 커널의 유닛 단위 분석을 지원하는 프레임워크를 개발하는 것이 매우 중요하다.

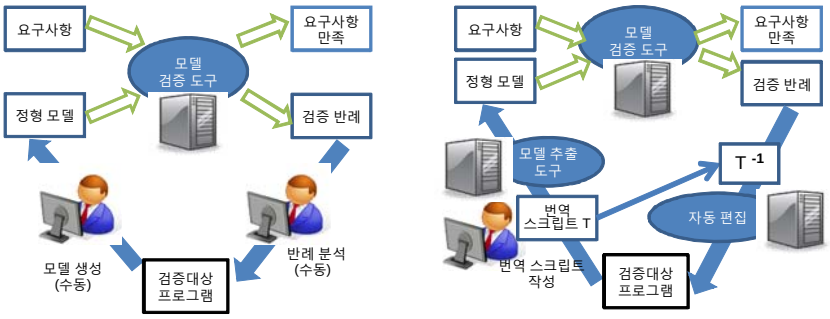
본 연구는 모델 추출 [2] 과 모델 검증 기법 [3][4]을 이용해 커널 상의 동시성 에러를 검출하는 모델 기반의 테스팅 프레임워크를 개발하는 것을 목적으로 한다. 앞서 언급한 이유와 같이 커널을 테스팅 하는 것은 매우 어렵기 때문에, 모델 검증 기법이 커널을 분석하는데 대안이 될 수 있다 [5]. 하지만 검증 대상 커널 프로그램과 이에 해당하는 추상화 모델 간의 차이로 인해 모델 검증 기법이 제시한 검증 반례 (counter example) 가 실제 프로그램 상의 에러인지, 아니면 잘못된 알람 (false alarm) 인지 여부를 개발자들이 판단하기 힘들기 때문에, 순수하게 모델 검증 기법 하나만으로는 커널을 검증하는데 문제가 있을 수 있다.

본 논문에서 제시하고 있는 모델 기반의 커널 테스팅 (MOKERT) 프레임워크는 모델 검증 과정에서 발견한 반례를 실제 커널 코드를 바탕으로 하는 실행에서 재연 (replay) 하는 기능을 제공한다. 이 기능을 통해 MOKERT 프레임워크는 커널 개발자들에게 모델 검증 기법과 테스팅의 장점만을 제공할 수 있다. 이 기능은 모델 검증에서 발견된 오류의 원인 분석, 결함 패치 (bug patch) 의 적합성 확인, 추상화 모델의 정밀화 등의 작업을 가능하게 한다. 위 프레임워크의 효율성은 리눅스 2.6 커널의 파일시스템 구성요소에 적용함을 통해 증명 되었다.

2. 모델 검증 도구 SPIN 과 모델 추출 도구 Modex

2.1 모델 검증 도구 SPIN

모델 검증 도구 SPIN [4] 에서는 Promela 라는 모델 명세 언어를 사용한다. SPIN 은 Promela 모델을 입력으로 받아, 모든 가능한 실행 시나리오를 분석하고, 오류 발견 시 구체적인 검증 반례를 제시한다.



(a) 사용자가 제공하는 모델을 이용하는 전통적인 모델 기반 테스팅 방법 (b) 모델 추출 도구를 이용하는 (반)자동 모델 기반 테스팅 방법

그림 1 모델 기반 테스팅

2.2 모델 추출 도구 Modex

Modex [2] 는 C 프로그램과 번역 스크립트 (translation script) 를 입력으로 받아, C 프로그램을 Promela 언어로 자동 번역하는 모델 추출 도구이다. 생성된 Promela 모델에는 번역 대상 C 프로그램의 몇 번째 줄에서 번역이 이루어 진 것인지에 대한 대응 정보가 주석의 형태로 포함되어 있다.

3. 모델 기반 커널 테스팅 (MOKERT) 프레임워크

모델 기반 커널 테스팅 (MOKERT) 프레임워크는 모델 검증 기법의 결과인 검증 반례를 실제 검증 대상 프로그램에 적용함으로써, 커널 코드에서의 동시성 결함 테스팅의 어려움을 해소하는 것을 목표로 한다.

전통적인 모델 기반 테스팅 방식 [6] 은 그림 1(a)에 표현되어 있는 것과 같이, 정형 모델을 사용자가 수작업(manually)으로 생성하고, 반례의 분석도 사용자가 수작업으로 수행한다. 이러한 과정은 많은 시간과 노력이 소모된다. MOKERT 프레임워크에서의 모델 기반 테스팅 방식은 그림 1(b)에 표현되어 있다. MOKERT 프레임워크에서는 Modex 를 사용하여 검증 대상 C 프로그램으로부터 (반)자동으로 정형 모델을 추출한다. 이 추출 과정에서는 번역 스크립트 T 가 사용된다. 번역 스크립트 T 에는 검증 대상 C 프로그램의 특정 구문이 Promela 의 어떠한 구문으로 번역되어야 하는지 명세 되어 있다. 이러한 T 의 역(reverse)인 T^{-1} 은 번역된 Promela 모델의 특정 구문이 검증 대상 C 프로그램의 어떠한 구문을 번역한 것인지에 대한 정보를 나타내며, 따라서 T^{-1} 을 이용하면 주어진 반례가 C 프로그램에서 어떠한 구문의 실행을 표현하는 것인지 이해할 수 있다. 그리고 이 정보를 바탕으로, 검증 대상 프로그램이 반례에서 표현된 실행과 같은 형태의 실행이 되도록 편집(instrumentation)이 가능하다. 이렇게 편집된 프로그램을 실행함으로써, 실제 커널 프로그램에서 반례의 재연(replay)가 가능하다.

4. 사례 연구 : Proc 파일 시스템의 proc_readdir()과 remove_proc_entry() 간의 자원 경쟁 (data race)

Proc 파일 시스템의 proc_readdir() 함수와 remove_proc_entry() 함수 간의 자원 경쟁 오류가 발생하여, 삭제한 파일이 삭제되지 않은 것처럼 출력되는 오류를 MOKERT 프레임워크를 통해 확인할 수 있었다.

5. 결 론

본 연구는 운영체제 커널의 동시성 결함을 효과적으로 분석하기 위한 MOKERT 프레임워크를 제안하고 있다. 이 프레임워크는 추상화 모델링 작업을 통해서 운영체제 커널의 각 구성요소를 독립적으로 검증하는 것을 돕는다. 또한 검증 반례 재연 기능을 통하여 모델 검증 결과를 실제 커널 코드에 적용할 수 있다. 따라서 MOKERT 프레임워크는 동시성을 가지는 커널의 동작을 모델 검증을 통해 면밀히 검사함으로써 커널 코드 분석의 신뢰성을 향상시킨다. 우리는 리눅스 2.6 커널을 대상으로 하는 사례연구를 통해 이 프레임워크의 실효성을 확인하였다.

참고문헌

[1] R. S. Pressman: “Software Engineering: A Practitioner’s Approach 6th ed”, McGrawHill, 2005

[2] G. J. Holzmann, R. Joshi: “Model-driven software verification”, Spin Workshop, April 2004

[3] E. M. Clarke, O. Grumberg, D. A. Peled: “Model Checking”, MIT Press, January 2000

[4] G. J. Holzmann: “The Spin Model Checker”, Wiley, New York, 2003

[5] J. Yang, P. Twohey, D. Engler, M. Musuvathi: “Using model checking to find serious file system errors”, Operating System Design and Implementation, 2004

[6] M. Utting, B. Legeard: “Practical model-based testing: a tools approach”, Morgan Kaufmann, 2007