KIPS Tr. Software and Data Eng.
Vol.6, No.8 pp.385~390 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2017.6.8.385

Model-Based Automatic Test Data Generation Method Using Custom Parser and SMT Solver

Ki-Wook Shin† · Dong-Jin Lim††

ABSTRACT

Because of the ever-increasing software complexity, model-based development techniques are becoming an essential technique in software development. However, even if model-based techniques are used, the test case generation for complex software is still a challenge to solve. In this paper, we propose a method to generate automatic test cases based on UML model using custom parser and SMT solver. By proposed technique, a test case can be generated even though the model is described in a platform independent language such as action language, or in a platform dependent language. In addition, a concolic execution technique is applied to efficiently generate test cases in the model. In this paper, we present a case study on the power window switch model of Hyundai Santa Fe through the proposed test case generation technique.

Keywords : Test Case Generation, Model-Based Development, Custom Parser, SMT Solver, UML

커스텀 파서와 SMT 솔버를 활용한 모델 기반 테스트 데이터 생성 기법

신 기 욱† · 임 동 진††

요 약

지속적으로 증가하는 소프트웨어 복잡성으로 인해, 모델 기반 개발 기법은 소프트웨어 개발에 있어 거의 필수적인 기법이 되고 있다. 그러나, 모델 기반 기법을 활용한다 하더라도 복잡한 소프트웨어를 위한 테스트 케이스 생성은 여전히 풀어야 할 숙제이다. 본 논문에서는, 커스텀 파서와 SMT 솔버를 이용해 UML 모델 기반에서 자동 테스트 데이터를 생성하는 기법을 제안한다. 제안된 기법을 이용하면, 모델이 액션 언어(action language)와 같은 플랫폼 독립적인 언어로 구현되어 있거나, 플랫폼 종속적인 언어로 기술되어 있더라도 테스트 입력을 생성할 수 있다. 또한, 모델에서 테스트 케이스를 효율적으로 생성하기 위해 콘콜릭 수행 기법을 적용하였다. 본 논문에서는, 제안된 테스트 데이터 생성 기법을 통해 현대 산타페의 파워윈도우 스위치 모델에 활용된 사례를 기술한다.

키워드 : 테스트 케이스 생성, 모델 기반 개발, 커스텀 파서, SMT 솔버, UML

1. 서 론

소프트웨어 복잡성의 증가와 함께 모델 기반 개발의 필요성은 나날이 증가하고 있다. 모델 기반 개발 기법은 기존보다 훨씬 복잡해진 소프트웨어를 모델을 통해 구현하여 시뮬레이션과 같은 기능들을 통해 소프트웨어의 전문적인 분석이 가능하게 한다. 이러한 모델 기반 개발 기법 중 모델 기반 테스트에서는, 플랫폼 독립적인 모델에서 추상 테스트 케이스를 생성한다. 그리고, 이를 수행 시점에서 실제 수행 가능한 테스트 케이스로 인터프리트 하여 각 테스트 환경에 맞는 테스트를 수행한다. 이는 보편적인 모델 기반 테스트의 형태로서 정립되었지만, 한편으로는 테스트를 수행하기 위한 테스트 수행기를 비롯한 테스트 환경 전반적으로 추상 테스트 케이스를 인터프리트 할 수 있는 기능을 지원하는 도구를 반드시 사용해야 한다는 것을 의미한다. 이러한 점은 자동차 산업의 ISO 26262와 같이, 테스트 도구 또한 일정 수준의 안전 규격을 만족시켜야 하는 환경에 적용시키기 어렵다.

이를 위해 본 논문에서는 인터프리트 기능을 지원하는 테스트 수행기를 사용하지 않고, 모델에서 자동 생성한 테스트 케이스를 테스트 환경에 맞는 실제 테스트 케이스로 변환하는 방법을 제안한다. 메타 데이터 교환을 위한 XMI (XML Metadata Interchange)를 제외한, 플랫폼 독립적인 모델에서

※ 본 논문은 중소기업청 월드클래스300 R&D 사업 지원에 의하여 연구된 결과임.
† 준 회 원 : 한양대학교 전자시스템공학과 박사과정
†† 비 회 원 : 한양대학교 전자공학부 교수
Manuscript Received : December 9, 2016
First Revision : January 24, 2017
Second Revision : March 13, 2017
Third Revision : May 10, 2017
Accepted : May 15, 2017
* Corresponding Author : Dong-Jin Lim(limdj@hanyang.ac.kr)

다룰 수 있는 액션 언어(action language) 및 플랫폼 종속적인 모델에서 사용되는 코드와 같은 특정 언어를 분석할 수 있도록, 커스텀 파서 생성기를 활용한다. 이러한 방식은 IT와 같이 새로운 모델 기반 개발 기법과 도구가 쉽게 적용될 수 있는 분야와 달리, 자동차, 우주항공, 방위산업 등 안전 표준이 더 중요시되는 분야에 적용하기에 유리하다.

한편 동적 환경에서 테스트 케이스를 생성하기 위한 기법으로 실제 수행 기법 또는 심볼릭 수행 기법이 있다. 이 중 실제 수행 기법은 테스트 입력 값을 랜덤 또는 특정 규칙에 따라 생성하나, 빠른 시간 안에 충분한 테스트 입력을 생성하기에는 부적절하다. 심볼릭 수행 기법은 수행 경로에 대해 소스 코드를 분석하여 심볼릭 테이블을 작성하고, 이를 기반으로 입력 값을 산출하기 때문에 단순히 입력 값을 무작위로 생성하여 코드 전체를 수행해보는 것보다 상대적으로 성능이 뛰어나다[1]. 최근에는, 이러한 두 기법의 장점이 결합된 콘콜릭 수행 기법을 이용한 자동 테스트 케이스 생성 기법이 널리 활용되고 있다.

본 논문에서는 커스텀 파서와 SMT (Satisfiability Modulo Theories) 솔버가 결합된 콘콜릭 수행 기반의 테스트 케이스 생성 구조를 통해, UML (Unified Modeling Language) 모델에서 생성한 클래스 다이어그램 및 상태 선도를 기반으로 자동 테스트 케이스 생성 기법을 제안한다. 이를 위해, 먼저 XMI로 변환된 모델 정보에서 테스트 케이스 생성에 필요한 각종 정보를 XPath (XML path language)를 이용해 불러온다. 그리고 나서, 상태 선도 및 함수에 포함된 액션 언어 및 구현 코드 등을 대표적인 커스텀 파서 생성기인 ANTLR를 이용해 파싱한다. 파싱된 텍스트는 SMT 솔버인 Yices 기반에서 자동 테스트 케이스 생성에 활용된다.

2. 관련 연구

모델 기반에서 테스트 케이스를 생성하려는 연구는 주로 UML과 같은 표준 언어 또는 Simulink와 같은 특정 도구를 활용한 모델에서 이루어졌다. 이러한 연구들은 주로 모델 내에 구현된 활동 다이어그램, 상태 선도 등을 분석하여 테스트 케이스를 생성한다[2-6]. 대표적인 상용 모델 기반 테스트 케이스 생성 도구로서 IBM 사의 Rhapsody ATG (Automatic Test Generator) 와 Simulink Design Verifier 가 있지만, ATG 는 C++ 언어를 제외한 다른 언어는 테스트 케이스를 생성하지 못하거나, Simulink Design Verifier는 일부 블록에 대해서는 테스트 케이스를 생성하지 못하는 등의 한계가 있다.

UML 모델 기반 관련 연구 중에서는 여러 가지 행위 다이어그램 중 상태 선도를 이용해 테스트 케이스를 생성하려는 시도가 많이 이루어져 왔다. Samuel, et al.의 연구에서는 UML 상태 선도에서 탐색 기법 및 판정 변환을 이용해 기존의 연구보다 적은 수의 테스트 케이스로 경로 기반 커버리지를 달성하는 생성 기법을 제안하였다[6]. 그러나 AVM (alternating variable method) 적용의 한계로 인해, 전역 최적해를 찾지 못하여 커버리지를 100% 만족하지 못했다. Gulia는 상태 선도에서 경로 기반의 테스트 케이스를 부모 해로 선정하고 유전 알고리즘을 이용해 이로부터 새로운 자식 해를 생성하는 방법을 제안하였다[7]. 그러나 타 알고리즘과의 성능 비교가 부족한 문제가 있다. 또한, 이러한 연구들은 주로 실제 산업 분야에 직접 적용되지 않은 한계가 있었다.

코드 기반에서 테스트 케이스를 생성하려는 노력 또한 지속적으로 이루어져 왔다. 주로 CREST, KLEE 등의 콘콜릭 기반 테스트 케이스 생성기를 활용한 다양한 연구[8, 9]들이 있다. 그러나 이러한 코드 기반의 테스트 케이스 생성 도구는 특정 언어에서만 테스트 케이스를 생성할 수 있는 단점이 있다.

이보다 더 나아가서, 요구사항 기반에서 테스트 케이스를 생성하려는 노력 또한 이루어지고 있다. 이러한 요구사항은 유즈케이스, 유즈케이스 다이어그램 등을 활용해 테스트 케이스를 생성하므로[10] 가장 고객 요구사항과 가까운 테스트 시나리오 작성이 가능한 장점이 있으나, 실제로 검증 도구 상에서 바로 실행 가능한 테스트 케이스를 만드는 데에는 한계가 있다. 이러한 환경에서 테스트 케이스를 만드는 데에도 커스텀 파서 생성기를 활용할 수 있다.

일반적으로 커스텀 파서 생성기는 주어진 문법 명세에 기초해서 어떠한 입력 언어에 대해 구문 분석할 수 있는 파서의 구현 코드를 생성할 수 있다. 나아가, 일반적인 C/C++, C#, Java와 같은 프로그래밍 언어 뿐만 아니라 어떠한 특정 규칙에 따라 명기된 언어 또한 그에 맞는 문맥을 정의하여 파싱이 가능하다. 이러한 파서 생성기는 도메인 특화 언어(Domain Specific Language)의 구문 분석 등에 활용될 수 있다. 이러한 구문 분석을 통해 요구사항으로부터 필요한 형태의 테스트 시나리오 또는 테스트 케이스로 변환하는 것이 가능하다.

3. 테스트 케이스 생성

실제 산업 분야, 특히 자동차 산업에서의 ISO26262와 같은 기능 안전 표준 하에서는 테스트 도구 또한 신뢰된 기관에서 인증을 받은 제품 사용이 요구된다. 그러나, 일반적인 모델 기반 테스트 기법에서는 테스트 수행 시점에 추상 테스트 케이스를 실제 테스트 케이스로 변환할 수 있는 인터프리터가 내장된 테스트 도구가 활용되는데, 대부분 오픈 소스 검증 도구에 해당하여 실질적으로 사용이 불가능하다.

본 논문에서는 이와 같이 검증 도구 또한 표준 준수가 요구되는 경우에도 기존 검증 환경의 교체 없이도 사용될 수 있도록 커스텀 파서와 SMT 솔버가 결합된 새로운 모델 기반 자동 테스트 생성 방식을 제안한다.

3.1 XMI 및 XPath를 이용한 모델 분석

먼저, 타 프로그램과 상호 교환을 위해 UML 모델을 XMI 포맷으로 변환한다. 변환된 모델 메타데이터 정보는 테스트 케이스 생성기를 통해 불러들여진다. 해당 정보는 XPath 기반의 모델 분석 도구에 의해 패키지, 클래스, 객체

등의 요소로 파싱되어, 테스트 케이스 생성을 위한 테스트 환경 구축, 연관 관계 정보 취득 등에 활용된다.

3.2 커스텀 파서를 이용한 액션 분석

모델 기반 개발 기법을 통해 먼저 플랫폼 독립적인 모델을 개발할 경우, ALF (Action language for Foundation UML)와 같은 액션 언어를 통해 구현하게 된다. 그리고 구체적인 플랫폼이 정해진 경우, 플랫폼 종속적인 모델을 구현하기 위해 C/C++과 같은 다양한 프로그래밍 언어를 활용해 모델을 구현하게 된다. 이러한 플랫폼 독립적인 모델 및 플랫폼 종속적인 모델을 전부 포괄하여 테스트 케이스를 생성하기 위해서는 액션 언어뿐만 아니라 다양한 프로그래밍 언어를 파싱할 수 있어야 한다.

이를 위해, 본 논문에서는 커스텀 파서 생성기를 이용하여 필요한 환경에 맞는 파서를 생성해, 플랫폼 종속적인 모델 및 플랫폼 독립적인 모델에 적용이 가능한 테스트 케이스 생성 구조를 제안한다[11]. 커스텀 파서 생성기는 크게 입력 문법 표현형에 따라 YACC나 EBNF을 분석할 수 있는 도구로 나눌 수 있으며, 세부적으로는 매우 다양한 파생 도구가 있으나, 본 논문에서는 상대적으로 출력 언어가 다양하며 IDE 환경을 제공하는 ANTLR v4를 사용했다.

이러한 커스텀 파서를 활용하면 플랫폼 종속적인 모델에 활용된 C/C++, Java와 같은 프로그래밍 언어를 비롯하여, 플랫폼 독립적인 모델에 활용되는 액션 언어와 같은 다양한 언어를 파싱하기 위한 커스텀 파서를 생성할 수 있다. 이를 통해 분석된 코드는 대표적인 SMT 솔버인 Yices를 통해 심볼릭 수행 기법에 활용된다.

3.3 SMT 솔버 기반 테스트 케이스 생성

주로 코드 기반에서 테스트 케이스 생성을 위해 SAT 솔버 또는 SMT 솔버가 대표적으로 활용된다. SAT 솔버는 표현형이 간단하여 상대적으로 SMT 솔버에 비해 처리 속도가 빠르지만 CNF (conjunctive normal form) 와 같은 특정 포맷으로 변환이 필요하다는 단점이 있다. 반면, SMT 솔버는 속도가 느리지만 비교문과 같은 조건문 내의 다양한 구문을 처리할 수 있다는 점에서 사용이 편리하다.

본 논문에서는 모델 구현을 위한 액션 언어에 포함되어 있는 비교문과 같은 구문을 다시 변환하는데 필요한 노력을 줄이기 위해 SMT 솔버를 활용하였다. 본 논문에서는, MiniSMT, Z3, Yices와 같은 다양한 SMT 솔버 중 상대적으로 좋은 성능을 보여준 Yices를 선정하였다[12]. Yices를 이용해 현재 입력 값에 대한 천이 가능 여부를 판단하고, 아직까지 도달하지 못한 목표 경로의 필요한 입력 값을 산출한다.

플랫폼 독립적인 모델을 구현하기 위해 ALF 와 fUML을 사용한 경우, 활동 다이어그램을 수행 가능하게 만들어줄 수 있는 오픈 소스 도구인 fUML Reference Implementation를 활용할 수 있다[13]. 그러나, 플랫폼 독립적인 모델뿐 만 아니라 플랫폼 종속적인 모델을 동시에 구현하였거나 fUML 기반으로 생성하지 않은 경우는 해당 모델을 실시간으로 동작시키기 어렵다.

이를 위해 Fig. 1과 같이 모델 내의 플랫폼 독립적인 모델 및 플랫폼 종속적인 모델에 관계없이 실시간으로 동작시키기 위한 별도의 테스트 프레임워크를 구현하였다. 이러한 프레임워크는 SUT를 검증하기 위해 외부 인터페이스를 기반으로 수행할 입력 인터페이스를 정의한다.

Fig. 2와 같이, 세부적인 테스트 입력 생성 절차로서 먼저 생성된 모델을 XMI를 이용해 테스트 입력 생성 프로그램에 불러들인다. 여기에서 대상으로 선정한 객체에 대해 프로퍼티 초기화를 진행한 후, 테스트 시뮬레이션 환경을 구성한다. 먼저 초기에는 임의의 입력 값을 사용하여 테스트를 한번 수행한 후, 해당 수행 정보를 통해 심볼릭 테이블을 작성한다. 그리고 나서 완료된 테스트 케이스가 새로운 상태 천이를 달성

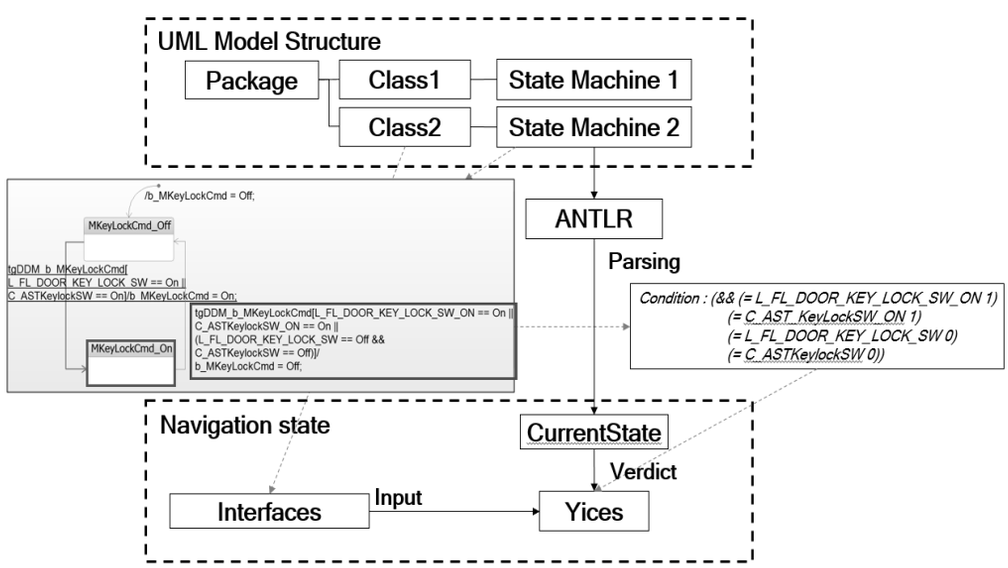

Fig. 1. A Model-Based Test Data Generation Process Using SMT Solver and Custom Parser

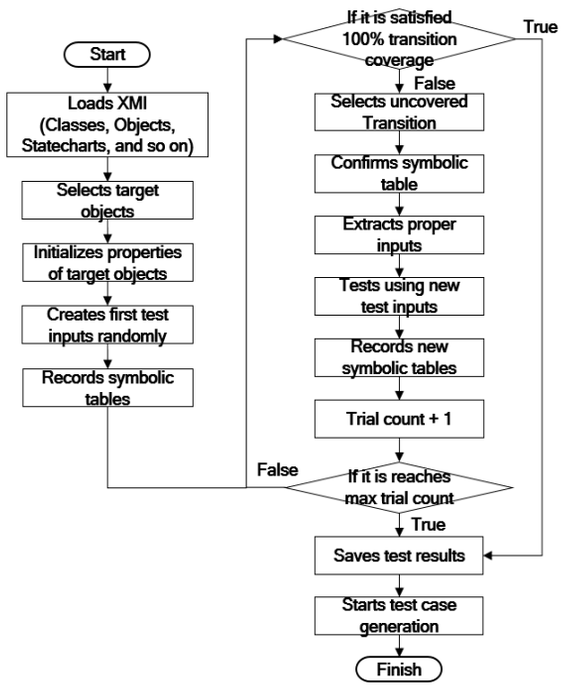

Fig. 2. A Test Input Generation Process on Concolic Execution

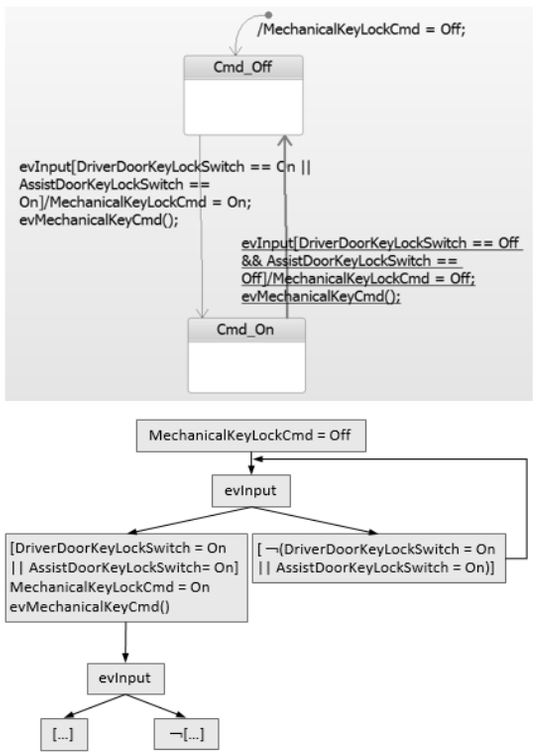

Fig. 3. A Statechart and a Symbolic Execution Tree of the MechanicalKeyInput Class

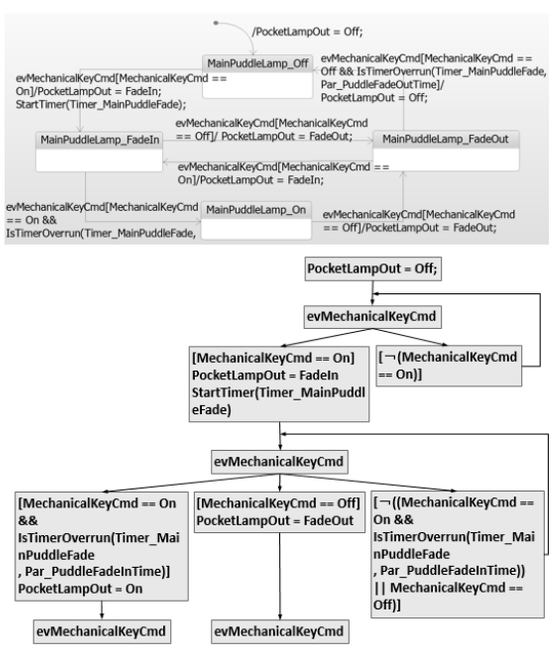

Fig. 4. A Statechart and a Symbolic Execution Tree of the PocketLampOutput Class

한다면 해당 테스트 케이스는 테스트 케이스 해집합에 추가된다. 만약, 테스트 케이스 해집합이 100% 천이 커버리지를 만족하지 못한다면 아직까지 만족되지 않은 상태 천이를 선택한다. 작성된 심볼릭 테이블로부터 이를 커버하기 위한 적절한 입력 값을 추출하고 이를 기반으로 테스트를 수행한다. 이러한 시도는 일정 횟수 이상 반복되고 나서 완료될 수 있다.

예를 들어, A 라는 모듈에서 B 모듈의 어떠한 특정 입력이 있어야 천이가 일어날 수 있는 경우 심볼릭 테이블이 활용될 수 있다. 일정 횟수 이상 천이에 실패할 경우, 수행을 통해 작성된 심볼릭 테이블로부터 해당 출력이 나오기 위한 B 모듈의 특정 입력 조건과 시퀀스를 찾아 이를 테스트 데이터로 활용한다. 만약 B 모듈이 먼저 수행되지 않아 참조할 수 있는 심볼릭 테이블이 아직 작성되지 않았을 경우에는, B 모듈 먼저 수행하도록 우선 순위를 부여한다.

상기 언급한 테스트 데이터 생성 기법의 이해를 위한 예제로서, 파워윈도우 스위치 모듈에서 키 박스에 키를 넣었는지 여부에 따라 램프가 동작하는 기능을 검증하기 위한 테스트 데이터를 생성한다고 가정하자. 이러한 기능 구현을 위해 두 개의 클래스 MechanicalKeyInput, PocketLampOutput가 있고, 하기의 Fig. 3 및 Fig. 4와 같은 각 클래스의 상태 다이어그램에 의해 기능 동작이 구현된다.

MechanicalKeyInput 클래스는 키 입력 상태를 확인하여 커맨드 및 이벤트를 생성하는 역할을 수행하고, PocketLampOutput 클래스는 MechanicalKeyInput 클래스의 키 입력 상태에 따른 커맨드에 따라 램프를 출력하도록 이루어져 있다. 따라서, Fig. 4의 PocketLampOutput 클래스 동작은 Fig. 3의 MechanicalKeyInput 클래스의 상태 천이에 의해 영향을 받게 된다. 예를 들어, evInput 이벤트 발생과 함께 DriverDoorKeyLockSwitch가 On이 된 경우, MechanicalKeyLockCmd가 On으로 변경되어 PuddleLamp 출력도 Off인 상태에서 FadeIn 상태로 천이하게 된다. 이러한 연관 관계가 있는 동작 입력 조건을 찾기 위해, 콘콜릭 수행 방식을 활용한다.

Fig. 2의 알고리즘 흐름과 같이 먼저 초기화된 각 객체의 상태 선도로부터 심볼릭 수행 트리를 작성하고 나면, 해당 입력이 타 모듈로부터 입력받은 변수인지 외부 변수인지에 따라 현재 미달성된 전이를 체크한다. 타 모듈로부터 입력받은 변수일 경우 해당 모듈의 심볼릭 테이블을 참고하여 필요한 입력 값을 추출하고 이 값을 이용해 직접 수행하여 해당 입력 값이 커버되지 않은 천이를 만족시킬 수 있는지 확인하고 심볼릭 수행 트리를 업데이트하는 과정을 거치게 된다.

4. 적용 사례

본 논문에서는 이러한 적용 대상으로서, 현대자동차의 싼타페 DM에 장착되는 파워윈도우 스위치 모듈을 선정하였다. 소프트웨어 모델은 IBM 사의 Rational Rhapsody를 사용하여 UML 기반 모델로 작성되었으며, 시스템의 구조를 표현하기 위해 클래스 다이어그램 및 객체 다이어그램, 행위 다이어그램으로는 상태 선도가 주로 활용되었다.

파워 윈도우 스위치 모듈은 크게 파워윈도우 기능과 미러 폴딩, 램프 및 히터 기능 등을 포함하고 있다. 이러한 각 주요 기능들은 개별적인 상태 선도로서 구현되어, 각 상태 선도 내 모든 천이(transition) 을 1번 이상 지나가는 천이 커버리지를 달성할 수 있도록 테스트 케이스 생성에 활용한다.

먼저, 상태 선도를 기반으로 콘콜릭 수행 기법에 따라 입력 인터페이스 및 해당 입력 값을 무작위로 생성한다. 그리고 현재 상태에서 특정 이벤트가 발생하게 되면 이에 해당하는 상태로 천이하게 된다. 이러한 수행 정보들은 심볼릭 테이블 작성에 활용되어 다음 테스트 케이스 생성 시 미달성된 경로를 커버하기 위한 입력 값 생성에 사용된다. 진입 불가능한 경로가 발생하거나 지나치게 한 경로에서 오랜 시간을 반복하는 문제를 줄이기 위해 일정 제한 시간이 지나면 테스트 케이스 생성을 중단하도록 하였다.

그러나 경로 탐색 알고리즘에 대해 랜덤 방식을 활용하여 테스트 케이스를 생성할 경우, 충분한 커버리지를 만족시키는 데에 어려움이 따른다. 예를 들어, 파워 윈도우 클래스에서 상태 전이를 위해 도난 경보 클래스의 경보 상태 출력 값을 참조하고, 다시 경보 상태 출력 값은 특정 조건을 만족해야 하는 경우와 같이, 클래스가 상호적인 연관 관계에 있거나 복잡한 연관 관계를 가지면 주어진 시간 내에 목표 커버리지를 달성하지 못할 수 있다. 이러한 문제를 해결하기 위해 본 논문에서는 지역 최적화 탐색 알고리즘을 활용하였다.

Integer 변수의 입력 범위를 1~100과 같이 지나치게 작게 제한하는 경우에는 지역 최적화 알고리즘과 같은 기법이 별로 효과가 없을 수 있다[14]. 이를 위해 본 논문에서는 파워윈도우 스위치 모델의 특성을 고려하여 Integer 변수 입력 범위를 0~65535로 설정하였다. 이보다 더 큰 입력 범위를 사용한 경우 천이 커버리지에는 특별한 차이가 없으면서 생성 시간이 오래 걸리는 문제점이 발생하였다.

Fig. 5에서 파워윈도우 스위치 모듈의 중 퍼들 램프 기능에 대해 일반적인 랜덤 알고리즘을 사용한 경우와 지역 최적화 알고리즘을 사용했을 경우를 10회 반복 시의 평균 결과로 비교하였다. 해당 실험 결과에서는, 테스트 입력 생성 시 일반적인 랜덤 방식을 활용하면 테스트 케이스가 비교적 빨리 커버리지를 달성하지만 덜 익은 수렴을 하는 현상이 발생했다. 반면, 지역 최적화 알고리즘을 활용한 경우, 랜덤 탐색 방식보다 더 많은 코드 커버리지를 달성할 수 있다.

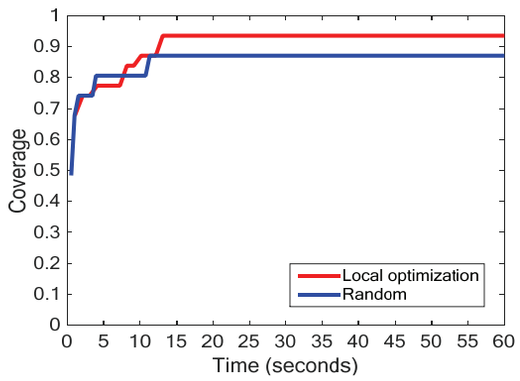

Fig. 5. A Comparison of Generation Time and Transition Coverage Between Random Navigation and Local Optimization

이를 전체 파워윈도우 스위치 모듈을 대상으로 각 기능별 최대 제한시간 5분 이내에서 테스트를 생성하였다.

Table 1. Transition Coverage Results for Each Feature of Power Window Switch Module

Module	No. of state diagram	No. of states	No. of transition	Coverage (Random)	Coverage (Optimization)
CourtesyLamp	2	6	13	92.30%	100%
PuddleLamp	8	26	31	90.32%	100%
PowerWindow	14	44	83	46.67%	100%
MirrorOperation	14	51	183	75%	94.44%

Table 1에서와 같이, 상태선도가 많고 복잡한 기능에 대해서는 랜덤 방식보다 지역 최적화 기법을 적용했을 때 더 유용한 결과를 얻을 수 있다. 특히 파워윈도우의 경우, 단위 함수 간에 연관 관계가 있는 변수가 많아 커버되지 않은 경로를 만족시킬 수 있는 경로를 우선 탐색하도록 지역 최적화 방식을 적용하여 커버리지를 향상시켰다. 100% 달성되지 않은 미러 기능은 입력 파라미터로서, 다른 단위 모듈로부터 미러 폴딩 명령 변수의 값을 입력받아 배열에 순차적으로 입력된다. 그리고 나서, 기존 값과 현재 값의 변화를 비교하여 동작하는 천이 조건이 주어진 시간 내에서는 생성하지 못하였다.

5. 결 론

일반적인 모델 기반 테스팅을 위해 활용되는 추상 테스트 케이스는 테스트 수행 시점에 이러한 추상 테스트 케이스를 인터프리트 할 수 있는 기능을 필요로 한다. 그러나, 이러한 인터프리트 기능을 가지고 있는 테스트 수행기는 자동차 산업과 같은 분야에서 사용되기 위한 표준에 부합하지 못해

실제 활용되기에는 어려움이 있었다.

이러한 문제를 해결하기 위해 본 논문에서는, 산업 분야에서도 활용할 수 있도록 추상 테스트 케이스 대신 커스텀 파서와 SMT 솔버를 활용해 테스트 환경에 맞춰 바로 테스트 케이스를 생성하는 구조를 제안하였다. 또한, 모델 기반 개발 기법 적용 시 플랫폼 종속적인 모델 및 플랫폼 독립적인 모델 두 가지가 동시에 혼재되어 있는 환경에서도 테스트 케이스를 생성할 수 있도록 SMT 솔버 활용 전에 커스텀 파서 생성기를 통해 다양한 모델 구현 언어에도 대응할 수 있는 방식을 고안하였다. 또한, 해당 기법을 자동차 전장 산업 분야의 파워윈도우 스위치 모듈에 지역 최적화 기법을 이용한 테스트 케이스 생성을 통해, 실제 적용 사례를 제시하였다.

이러한 방식들은 기존의 코드 기반 테스트 케이스 생성기가 기본적으로 구현 언어에 제약이 되는 한계를 넘어, 모델 기반 개발 기법이 적용된 개발 환경에서도 유연한 테스트 케이스 생성기의 개발이 가능하도록 한다.

References

[1] C. Cadar, P. Godefroid, S. Khurshid, C. S. Păsăreanu, K. Sen, N. Tillmann, and W. Visser, "Symbolic execution for software testing in practice: preliminary assessment," *Proceedings of the 33rd International Conference on Software Engineering*, ACM, pp.1066-1071, 2011.

[2] C. Mingsong, Q. Xiaokang, and L. Xuandong, "Automatic test case generation for UML activity diagrams," *Proceedings of the 2006 International Workshop on Automation of Software Test*, ACM, Shanghai, China, pp.2-8, 2006.

[3] M. Prasanna, K. R. Chandran, and K. Thiruvenkadam, "Automatic Test Case Generation for UML Collaboration Diagrams," *IETE Journal of Research*, Vol.57, No.1, pp.77, 2011.

[4] P. Samuel, R. Mall, and P. Kanth, "Automatic test case generation from UML communication diagrams," *Information and Software Technology*, Vol.49, No.2, pp.158-171, 2007.

[5] M. Sarma, D. Kundu, and R. Mall, "Automatic Test Case Generation from UML Sequence Diagram," *Advanced Computing and Communications, 2007. ADCOM 2007. International Conference on*, pp.60-67, 2007.

[6] P. Samuel, R. Mall, and A. K. Bothra, "Automatic test case generation using unified modeling language (UML) state diagrams," *IET Software,* Vol.2, No.2, pp.79-93, 2008.

[7] P. Gulia and R. S. Chillar, "A new approach to generate and optimize test cases for UML state diagram using genetic algorithm: http://doi.acm.org/10.1145/180921.2180933." *ACM SIGSOFT Software Engineering Notes*, Vol.37, No.3, pp.1-5, 2012.

[8] Y. J. Kim, M. Z. Kim, Y. H. Kim, and U. J. Jung, "Comparison of Search Strategies of KLEE Concolic Testing Tool," *Journal of KIISE: Computing Practices and Letters*, Vol.18, No.4, pp.321-325, 2012.

[9] K. Yunho, K. Moonzoo, and J. Yoonkyu, "CREST-BV: An Improved Concolic Testing Technique Supporting Bitwise Operations for Embedded Software," *Journal of KIISE: Software and Applications*, Vol.40, No.2, pp.90-98, 2013.

[10] S. J. Park, "A Method for Managing Traceability of Business Application Services through Automatic Generation of Analysis Model," Ph.D. dissertation, Dept. of Computer Science, Sogang University, 2008.

[11] K. W. Shin, J. H. Lee, S. S. Kim, J. U. Park, and D. J. Lim, "Automatic Test Case Generation Based on the UML Model of the Automotive-Embedded Software using a Custom Parser and SMT Solver," *JSAE 2016 Annual Congress (Spring)*, pp.1198-1202, 2016.

[12] C. Barrett, M. Deters, L. de Moura, A. Oliveras, and A. Stump, "6 Years of SMT-COMP," *Journal of Automated Reasoning*, Vol.50, Issue 3, pp.243-277, 2013.

[13] Zoltán Micskei, R.-A.K. Benedek Horváth, Oszkár Semeráth, András Vörös, Dániel Varró, "On Open Source Tools for Behavioral Modeling and Analysis with fUML and Alf," *1st Workshop on Open Source Software for Model Driven Engineering '2014,* Valencia, Spain, pp.31-41, 2014.

[14] S. Ali, M. Zohaib Iqbal, A. Arcuri, and L. C. Briand, "Generating Test Data from OCL Constraints with Search Techniques," *IEEE Transactions on Software Engineering*, Vol.39, No.10, pp.1376-1402, 2013.

신 기 욱

e-mail : zenixion@hanyang.ac.kr

2011년 한양대학교 전자정보시스템공학과 (학사)

2013년 한양대학교 전자시스템공학과(석사)

2013년~현 재 한양대학교 전자시스템공학과 박사과정

관심분야 : 화이트 박스 기반 테스트 자동 생성, 소프트웨어 테스트 기법, 모델 기반 소프트웨어 개발

임 동 진

e-mail : limdj@hanyang.ac.kr

1979년 서울대학교 전자공학과(학사)

1981년 서울대학교 전자공학과(석사)

1982년~1983년 금성(現, LG전자) 연구원

1988년 University of Iowa, U.S.A. Electrical and Computer Engineering (Ph.D.)

1988년~1991년 포항산업과학연구원(RIST) 연구원

1991년~현 재 한양대학교 전자공학부 교수

관심분야 : 제어 시스템, 임베디드 소프트웨어, Unified Modeling Language (UML)