양상논리를 포함한 BML 스펙 확장

김선태*, 김제민*, 박준석*, 유원희*
*인하대학교 컴퓨터정보공학과
e-mail:kst_1205@nate.com

The Extension of BML Specification including Modal Logic

Seon-Tae Kim*, Je-Min Kim*, Joon-Seok Park*, Won-Hee Yoo*
*Dept of Computer and Information Engineering, Inha University

요 약

검증에 사용하는 명세는 대부분 1차 술어 논리(First Order Logic)로 이루어져 있다. 1차 술어 논리가 자연언어 대부분을 표현하지만 표현하지 못하는 부분도 존재한다. 이를 해결하기위해 양상논리(Modal Logic)를 추가한 명세방법이 존재하지만 간접적인 방법으로만 존재할 뿐 이다. 본 논문에서는, 양상논리를 이용한 명세의 직접적인 표현을 위해 BML(Bytecode Modeling Language)을 확장한다. 이를 통해, 명세정보 표현의 정확성을 향상시킨다.

1. 서론

프로그램 검증은 프로그램 수행 전에 프로그램이 안전하게 실행하는지 여부를 검사하는 기술로서 그 중요도가 높아지고 있다. 특히, 핵 발전소, 항공 우주 공학과 같은 최신 분야의 프로그램 수행 환경에서 중요한 것은 프로그램이 오류 없이 동작하는 것이다. 따라서 프로그램 검증은 매우 중요하다.

특히, 객체지향 언어로서 효율적인 개발환경과 재사용성, 안전한 실행 등의 장점으로 다양한 분야에서 사용되고 있는 자바 언어의 검증에 대한 연구가 다양한 방법으로 진행되고 있다. 자바 소스 코드에 명세를 추가하여 검증을 수행하는 JML(Java Modeling Language)[1]과 JML을 이용하여 논리식을 생성하고 생성된 논리식을 Key 증명기를 통해 검증을 수행하는 Key[2], 자바 바이트 코드에 명세정보를 추가한 BML(Bytecode Modeling Language)[3]을 통해 생성된 BoogiePL[4]을 Z3[5]와 같은 SMT Solver(Satisfiability Modulo Theories Solver)를 이용하여 검증을 수행하는 방법 등 많은 방법이 존재한다.

검증은 명세정보를 기반으로 수행한다. 프로그래머가 입력한 전조건, 후조건, 루프불변자와 같은 명세정보를 통해서 프로그램이 오류를 검출한다.

검증에 사용되는 명세의 대부분은 1차 술어 논리(First Order Logic)[6]로 이루어진다. 1차 술어 논리가 자연언어의 대부분을 표현할 수 있지만 표현하기 어려운 부분도 존재한다. 이를 해결하기 위해 양상논리(Modal Logic)[7]를 포함한 선형시간논리(Linear Temporal Logic)[8]를 입력으로 사용하는 JAG(Java Annotation Generator)[9] 도구를 이용하여 양상논리정보를 JML에 추가한 방법이 고안되었다. 이 방법은 양상논리를 1차 술어 논리로 변환하는 방법으로 선형시간논리를 생성하고 생성된 선형시간논리에 대응되는 프로퍼티(Property)를 통해 명세를 생성하여 JML 문법에 맞게 배치해준다.

하지만 JAG를 이용하여 JML로 변환하는 방법은 양상논리를 직접적으로 표현해주지 못한다. 이는 양상논리를 1차 술어 논리로 변환 시 의미가 올바르게 변환되지 못할 가능성이 존재한다는 것을 의미한다. 따라서 양상논리를 직접적으로 표현해 주는 방법이 필요하다.

JML은 1차 술어 논리를 표현하기 때문에 양상논리를 직접적으로 표현하기 힘들다. 반면에 BML은 명세정보를 추가한 자바 바이트 코드의 확장 형태이고, 명세정보는 BML 스펙을 따르며, 이진 코드로 표현되어 있기 때문에 BML 스펙에 양상논리를 추가하여 양상논리를 직접적으로 표현할 수 있다.

본 논문에서는 양상논리를 직접적으로 표현하기 위해 BML 스펙을 확장한다. 논문의 구성은 다음과 같다. 2장에서는 BML 확장에 필요한 배경지식에 대해 설명하고 3장에서는 BML 스펙 구조에 대해 설명한다. 4장에서는 확장된 BML 스펙을 설명하고 마지막 5장에서는 결론 및 향후 연구 방향에 대해 설명한다.

2. 배경지식

2-1. JML(Java Modeling Language)

자바 소스 코드에서 프로그래머가 전조건, 후조건, 루프 불변자 등의 명세정보를 주석형태로 명세가 가능한 언어이다. 명세정보는 프로그래머가 의도한 프로그램의 방향을 표현한다.

2-2. BML(Bytecode Modeling Language)

JML2BML[10] 프로그램을 통해 생성된 이진 코드로서 자바 바이트 코드에 명세정보를 BML 스펙에 맞게 확장한 언어이다. 검증에 필요한 전조건, 후조건, 루프불변자 등의 명세정보를 이진 코드 형태로 표현한다.

2-3. JML2BML

JML2BML은 JML입력으로 받아 BML을 생성해 주는 프로그램이다. 프로그래머가 입력한 명세정보를 BML 스펙에 맞게 변환해 주는 도구로서, 하이레벨로 표현된 명세 정보가 로우레벨 형태로 변환이 가능하다.

2-4. 선형시간논리(Linear Temporal Logic)

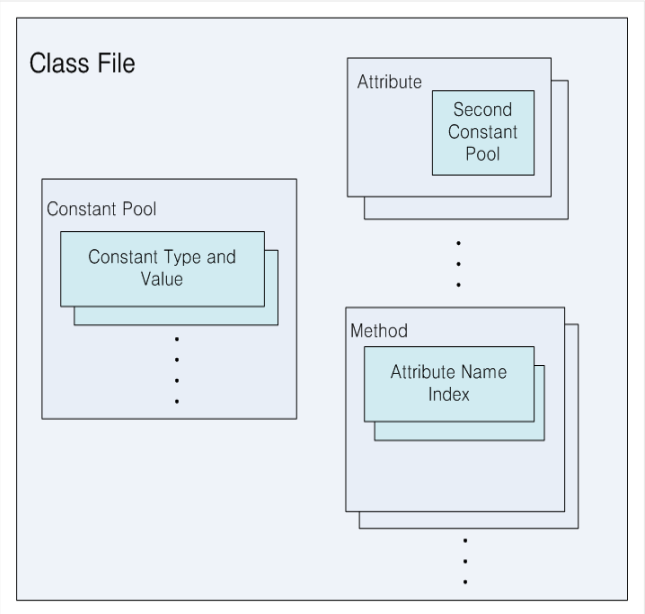
선형시간에서 양상논리의 연산자를 통해 표현한 논리식이다. 선형시간논리를 통해 프로그램 수행시간이 행해지는 연산을 양상논리 연산자로 표현할 수 있다.

3. BML 스펙

```
ClassFile {
        u4 magic;
        u2 minor_version;
        u2 major_version;
        u2 constant_pool_count;
        cp_info
        constant_pool[constant_pool_count-1];
        u2 access_flags;
        u2 this_class;
        u2 super_class;
        u2 interfaces_count;
        u2 interfaces[interfaces_count];
        u2 fields_count;
        field_info fields[fields_count];
        u2 methods_count;
        method_info methods[methods_count];
        u2 attributes_count;
        attribute_info
        attributes[attributes_count];
 }
```

(그림 1) BML 스펙 구조

BML은 자바 바이트 코드에서 명세정보를 추가하여 확장된 이진 코드이다. BML 스펙 구조를 나타낸 그림 1과 같이 JVM[11]에서 제공하는 클래스 파일 스펙과 동일하다. 자바 바이트 코드를 역 컴파일 할 때 사용되는 명령어인 Javap 명령어를 통해 BML을 역 컴파일 했을 때에도 자바 바이트 코드와 동일한 코드가 생성됨을 확인할 수 있다. 이는 클래스 정보, 메소드 정보 그리고 상수 타입과 변수 등의 정보들이 자바 바이트 코드와 동일하게 표현된다는 것을 의미한다.



(그림 2) BML의 이진파일 구조

BML과 자바 바이트 코드의 차이점은 그림 2의 BML의 이진파일 구조를 통해 확인할 수 있다. 그림 2에 나타난 것과 같이 BML에는 JVM과 달리 속성 정보 안에 2차 상수풀이 존재한다. 2차 상수풀에는 1차 상수풀에 표현되지 않는 이름값과 인덱스 정보들이 표현된다. 2차 상수풀에 표현된 정보를 통해서 명세정보가 정의된 BML 스펙에 접근하여 명세정보를 추출할 수 있다. 명세정보의 추출은 표 1과 같은 단계를 거쳐 이루어진다.

<표 1> BML 명세정보 추출 과정

1.	클래스 파일 안의 메소드 속성 정보 검사
2.	메소드 속성 정보안에 있는 속성이름과 인덱스 정보 추출
3.	2차 상수풀에 정의되어 있는 이름과 값 정보를 추출
4.	BML스펙과 매칭하여 일치하는 정보를 반환

표 1의 단계를 통해 반환된 정보는 그림 3과 같은 형태로 표현된다. 그림 3은 다양하게 표현되는 BML 스펙 중 전조건의 정보를 나타낸다. 속성이름의 주소와 길이 등의 기본정보를 표현하며, 전조건에 대한 명세정보는 formula_info를 통해 표현한다.

```
AssertTable_attribute {
        u2 attribute_name_index;
        u4 attribute_length;
        u2 asserts_count;
        {
                u2 point_pc;
                u2 index;
                formula_info assertion;
        } asserts[asserts_count];
}
```

(그림 3) org.bmlspecs.AssertTable 레코드 정보

4. 확장된 BML 스펙

3장에서 설명한 BML 스펙을 통해 BML이 표현하는 명세정보는 표 1의 단계를 거쳐 formula_info를 통해 표현됨을 알 수 있다. formula_info는 표 2에서 보는 것과 같이 논리식과 비 논리식으로 나누어 BNF 형태로 표현된다. 각각의 세부 정보가 1 바이트 이진 코드로 대응하여 표현됨을 알 수 있다.

<표 2> formula_info 내용

Logical	
TRUE	= 0x00;
FALSE	= 0x01;
AND	= 0x02;
OR	= 0x03;
.....	
Non Logical	
PLUS	= 0x20;
MINUS	= 0x21;
MULT	= 0x22;
.....	

따라서 formula_info에서 표현하는 세부정보 외에 새로운 정보를 추가하여 표현하기 위해서는 기존의 formula_info에서 사용하지 않는 이진코드를 사용하여 표현해야 하고, 1 바이트 이진코드 형태를 유지해야 한다. 표 3은 양상논리를 추가한 formula_info이다. 양상논리에서 사용되는 연산자가 모두 추가되었다. 이를 통해 양상논리를 BML에서 표현할 수 있다.

양상논리로 표현된 명세정보를 반환하기 위해서는 그림 3의 전조건의 BML 스펙과 같은 형태로 양상논리를 정의해야한다. 기본 형식은 그림 3과 같이 속성이름의 주소, 길이 등의 기본정보를 나타내고, 양상논리를 추가한 formula_info를 통해 명세정보를 표현한다. 그림 4는 양상논리에 관한 BML 스펙을 나타낸다.

<표 3> 양상논리를 추가한 formula_info 내용

.....	
Next	= 0xA0;
Strong until	= 0xA1;
Weak until	= 0xA2;
Always	= 0xA3;
Eventually	= 0xA4;
.....	

```
ModalTable_attribute {
        u2 attribute_name_index;
        u4 attribute_length;
        u2 modal_count;
        {
                u2 point_pc;
                u2 index;
                formula_info modal;
        } modals[modal_count];
}
```

(그림 4) 양상논리의 BML 스펙

기존 BML 스펙과 formula_info에 양상논리를 추가하여 생성된, 확장된 BML은 표 1의 단계를 통해 양상논리로 표현된 명세정보를 반환할 수 있다.

5. 결론 및 향후연구과제

본 논문에서는 BML에 양상논리를 추가하여 BML을 확장하였다. 이를 이용하여 1차 술어 논리 뿐 아니라, 양상논리로 표현된 명세정보를 반환할 수 있다. 표현할 수 있는 범위가 넓은 양상논리를 명세에 추가함으로써, 1차 술어 논리에서 표현하지 못하는 명세정보를 정확하게 표현할 수 있다. 명세의 정확한 표현은 검증의 정확성을 향상시킨다.

향후 연구과제로 확장된 BML을 통해 검증을 수행한다. 검증을 위해서는 검증에 적합한 중간 표현 언어가 필요하다. BML을 입력으로 받아 생성하는 중간 표현 언어중에 BIRS(Bytecode Intermediate Representation Specification) [12]가 존재한다. 확장된 BML에 맞추어 BIRS를 확장하는 것이 필요하다. 확장된 BML에 적합한 중간 표현 언어 생성 후, SMT 해결기를 이용하여 검증을 수행한다.

논문 사사(acknowledgement)

이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (No. 2011-0026495)

참고문헌

[1] Gary T. Leavens. 2007. Tutorial on JML, the java modeling language. In Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering (ASE '07). ACM, New York, NY, USA, pp. 573-573.

[2] Ahrendt, W., et al., "KeY: a formal method for object-oriented systems," In Proceedings of the 9th IFIP WG 6.1 international conference on Formal methods for open object-based distributed systems (FMOODS'07), Springer-Verlag, Berlin, Heidelberg, 2007, pp. 32-43.

[3] Lilian Burdy, Marieke Huisman and Mariela Pavlova, "Preliminary Design of BML: A Behavioral Interface Specification Language for Java bytecode," In Fundamental Approaches to Software Engineering (FASE 2007), pages 215-229. Volume 4422 of Lecture Notes in Computer Science, Springer-Verlag, 2007.

[4] Robert DeLine, K.Rustan and M. Leino, "BoogiePL : A Typed procedural language for checking object-oriented program," Technical Report MSR-TR-2005-70, 27 May 2005.

[5] De Moura, L. and Bjørner, N., "Z3: an efficient SMT solver," Proceedings of the Theory and practice of software, 14th international conference on Tools and algorithms for the construction and analysis of systems, March 29-April 06, 2008.

[6] http://en.wikipedia.org/wiki/First-order_logic/.

[7] http://en.wikipedia.org/wiki/Modal_logic/.

[8] http://en.wikipedia.org/wiki/Linear_temporal_logic/.

[9] A. Giorgetti and J. Groslambert. JAG: JML Annotation Generation for Verifying Temporal Properties. In FASE, LNCS, pages 373 - 376. Springer, 2006.

[10] JML2BML, http://mobius.ucd.ie/repos/.

[11] Java Virtual Machine Specification, http://java.sun.com/docs/books/jvms/second_edition/html/ClassFile.doc.html#1169.

[12] 김선태, 김제민, 박준석, 유원희, "자바 프로그램 검증을 위한 스택 리스 중간 표현 언어 생성기 구현", 한국정보기술학회, 2011년 9월 30일.