DOI: 10.3745/KIPSTD.2009.16D.6.911

SAT를 이용한 MC/DC 블랙박스 테스트 케이스 자동 생성

정 인 상†

요 약

항공 소프트웨어가 FAA(미국연방항공청)에 승인받기 위해서는 DO-178B 표준에 따라야 한다. DO-178B에서는 안전 필수 소프트웨어의 단위 테스트가 MC/DC 기준을 만족하기를 요구하고 있다. MC/DC에 따른 테스트는 안전성과 관련된 오류를 효과적으로 찾을 수 있는 수단으로 알려져 있지만 MC/DC를 만족하는 테스트 케이스를 생성하는 작업이 용이하지 않는 것도 사실이다. 이 논문에서는 MD-SAT이라고 명명한 SAT(SATisfiability) 기술을 사용하여 MC/DC를 만족하는 테스트 케이스를 자동으로 생성하는 도구에 대해 소개한다. 이 도구는 결정표 기반 테스트, 인과 그래핑 및 상태 전이 테스트 방법을 구현한 도구에서 보다 다양한 테스트 케이스 생성을 위해 사용될 수 있다.

키워드 : 프로그램 테스팅, 자동 테스트 데이터 생성, MC/DC, SAT

Automated Black-Box Test Case Generation for MC/DC with SAT

Chung, In-Sang†

ABSTRACT

Airbone software must comply the DO-178B standard in order to be certified by the FAA. The standard requires the unit testing of safety-critical software to meet the coverage criterion called MC/DC(Modified Condition/Decision Coverage). Although MC/DC is known to be effective in finding errors related to safety, it is also true that generating test cases which satisfy the MC/DC criterion is not easy. This paper presents a tool named MD-SAT which generates MC/DC test cases with SAT(SATisfiability) technology. It can be employed for generating diverse test cases in tools implementing various testing techniques including decision table based test, cause-effect graphing, and state-based test.

Keywords : Program Testing, Automated Test Data Generation, MC/DC, SAT

1. 서 론

소프트웨어는 항공기의 첨단 기능의 자동화를 위해 필수적일 뿐만 아니라 항공 교통 관제를 위해 통신, 항행, 감시 등과 같은 여러 분야에서 중추적인 역할을 한다. 소프트웨어 기반 시스템의 기능 및 복잡도가 증가함에 따라 이러한 시스템들이 안전성 요구사항을 포함한 자신의 요구사항을 만족하는지를 검증하는 것도 매우 복잡하게 되었으며 많은 비용을 초래하게 되었다. 오작동이 발생했을 때 인명 손실과 같은 재난을 가져오는 레벨-A로 분류되는 소프트웨어는 완전한 테스트를 요구하지만 오늘날 소프트웨어의 복잡도나 크기는 완전한 테스트를 불가능하게 한다[1].

DO-178B는 미국연방항공청(FAA)의 승인을 받기위해 항공 소프트웨어 개발자들에 의해 사용되는 주요 표준이다[1]. 이 표준에서는 소프트웨어 생명주기 행위 및 설계시 고려할 점등에 대해 기술하고 있으며 레벨 A 소프트웨어에 대해서는 MC/DC(Modified Condition/Decision Coverage) 테스트 기준을 만족하기를 요구한다. MC/DC 기준은 명세나 원시 코드의 술어에 있는 이진식(boolean expression)에 바탕을 둔 테스트 기준이다.

만약 n개의 변수가 주어진다면 2^{2^n}개의 서로 다른 이진 함수를 만들 수 있다. 이들을 서로 구분하기 위해서는 2^n개의 테스트 케이스가 요구된다. 그러나 이러한 형태의 테스트는 만약 n이 어느 정도 클 때 테스트 케이스 개수가 기하급수적으로 증가하므로 비용측면에서 현실적으로 수행할 수 없다. 현실적으로는 모든 가능한 테스트 케이스들의 부분 집합만을 선정하여 효과적으로 결함을 발견하기를 기대할 수밖에 없다. MC/DC도 이러한 방법 중의 하나이다.

1999년 항공 소프트웨어 업계에 대한 설문 조사에 의하면 응답자의 75%가 DO-178B의 MC/DC 요구사항을 충족시키

※ 본 연구는 2009년도 한성대학교 교내연구비 지원과제임.
† 정 회 원 : 한성대학교 컴퓨터공학과 교수
논문접수 : 2009년 9월 15일
수 정 일 : 1차 2009년 11월 23일
심사완료 : 2009년 11월 23일

는 것이 어렵다고 대답하였으며 74%는 비용이 엄두를 내지 못할 만큼 엄청나다고 주장하였다[2]. 반면에 Dupuy와 Leveson이 HETE-2 위성 소프트웨어에 수행한 연구 결과에 따르면 상대적으로 비용이 많이 든다 할지라도 MC/DC 기준에 따른 테스트는 안전성과 관련된 오류들을 찾아낼 수 있다고 한다[3]. 또한, Kapoor와 Bowen이 수행한 연구에서도 DC(Decision Coverage), FPC(Full Predicate Coverage) 등과 비교하여 조건이 많이 있는 경우에는 결함 식별 능력에서 가장 우수하다는 실험 결과가 있다[4]. 국내에서는 이와 같은 MC/DC의 효용성을 고려하여 원자력 발전소의 소프트웨어에 MC/DC를 필수 요건으로 규정하는 작업이 현재 논의되고 있는 중이다[5].

이진식에 관련된 변수가 많아질수록 MC/DC에 따른 비용을 줄이기 위해서는 자동화가 필수적이다. 이 논문에서는 주어진 이진식으로부터 MC/DC를 만족하는 테스트 케이스들을 자동 생성하는 MD-SAT이라는 도구를 소개한다. MD-SAT은 SAT(SATisfiability) 기술에 바탕을 두고 있다. SAT는 어떠한 이진식이 있을 때 그 식을 참이 되도록 하는 모델(model)이 존재하는지를 검사하는 문제를 말한다. 여기에서 모델이란 주어진 논리식을 참이 되게 하는 변수들의 값의 조합이다. 현재 zschaff[6], BerkMin[7], GRASP[8] 등과 같은 SAT 도구들이 개발되어 사용되고 있으며 현재의 MD-SAT은 SAT4J[9]를 사용하고 있다.

대표적인 블랙박스 테스트 방법으로 결정표 기반 테스트, 인과 그래핑(cause-effect graphing) 및 상태 전이 테스트(state-transition test)등이 있다[10]. 이러한 테스트 방법들은 기본적으로 이진식을 직접 또는 간접적으로 다룬다. 따라서 이러한 테스트 방법들을 실현한 도구들의 백엔드(back-end)에서 MD-SAT를 사용하여 테스트 케이스를 자동으로 생성하는데 활용할 수 있다. 실제 본 연구의 일환으로 MD-SAT를 테스트 케이스 생성 모듈로 활용한 인과 그래핑 테스트 도구를 개발하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 MC/DC에 대해 기술하고 SAT의 간략한 소개 및 SAT를 기반으로 하는 기존의 테스트 방법에 대해 기술한다. 3장에서는 MC/DC를 만족하는 테스트 케이스를 생성하기 위해 MD-SAT이 SAT 기술을 활용하는 방법에 대해 기술한다. 또한 기존의 테스트 방법들과 MD-SAT를 결합하는 방안에 대해서도 기술한다. 4장에서는 TCASII(항공기 충돌 회피 시스템)의 명세로부터 추출한 20개의 이진식에 대해 실험한 결과에 대해 기술한다. 마지막으로 결론 및 향후 연구에 관해 기술한다.

2. 관련 연구

2.1 MC/DC

MC/DC 테스트 기준을 이해하기 위해서는 조건(condition)과 결정(decision)에 대해 명확하게 우선 구분할 필요가 있다:

- 조건(condition): AND나 OR와 같은 논리 연산자를 포함하지 않는 이진 식
- 결정(decision): 단일 조건으로 구성되거나 여러 조건들이 논리 연산자들을 사용하여 결합된 이진식

예를 들면 "A • B+ C"는 조건 A, B, C로 구성된 결정이다. 이 논문에서는 '•'가 AND를 '+'가 OR 연산자를 나타낸다. 이와 같은 용어의 정의를 사용하여 MC/DC를 정의하면 다음과 같다:

- MC/DC: 결정에 있는 모든 조건들이 최소한 한번은 모든 가능한 진리 값을 가져야 하고 각 조건은 독립적으로 결정의 진리 값에 영향을 미친다는 것을 보여야 한다. 한 조건이 결정 값에 독립적으로 영향을 미치는 것을 보이기 위해서는 대상이 되는 조건을 제외한 다른 조건들의 값을 고정하면서 해당 조건 값을 달리하여 결정의 값을 변화시키는 것을 보인다[11].

예를 들어 이진 식 "A • B+C"에서 이진 변수 A에 대한 MC/DC 테스트 케이스는 A: TRUE, B: TRUE, C: FALSE와 A: FALSE, B: TRUE, C: FALSE이다. 그 이유는 B와 C의 값은 TRUE와 FALSE로 고정되어 있으면서 이진 식 "A • B+C" 값을 TRUE로 만들지만 두 번째 테스트 케이스는 FALSE로 만든다. 즉 A의 값만을 달리하여 전체 결정의 값을 변화시키기 때문이다.

MC/DC의 정의로부터 알 수 있듯이 이진 식을 구성하는 조건 또는 변수의 개수가 많아지면 많아질수록 MC/DC를 만족하는 테스트 케이스들을 계산하는 것은 매우 어렵게 되는 것은 명확하다.

현실적으로는 테스트 케이스를 랜덤 또는 명세 정보를 바탕으로 생성하여 이들이 주어진 테스트 기준(이 경우에는 MC/DC)을 만족하는 지를 평가하는 방식을 취한다. 다행스럽게도 기존에 개발된 테스트 케이스들이 주어진 이진 식에 대해 MC/DC에서 요구하는 기준을 만족하는지를 평가하는 절차는 개발되었다. 따라서 MC/DC 평가 절차는 리그레션 테스트 등을 포함하여 화이트박스 테스트와 같이 실제 프로그램 코드를 대상으로 테스트하는 경우에는 유용하게 사용될 수 있다. 이 평가 절차에 관해서는 [2]에 자세하게 기술되어 있다.

MC/DC는 실제로는 이진 식이 나타나는 어떤 경우에도 적용할 수 있다. 즉, 결정표, 인과 그래핑 및 상태 전이도와 같이 이진 식을 추출할 수 있는 어떤 경우라도 테스트 케이스를 생성하는 전략의 하나로 사용될 수 있다. 이 논문의 주요 목적은 이와 같은 다양한 명세 정보를 기반으로 하는 블랙박스 테스트 방법들과 결합하여 MC/DC 테스트 케이스를 생성하는 데 사용할 수 있는 도구를 개발하는 것이다.

2.2 SAT과 프로그램 테스트

SAT은 어떤 이진식이 있을 때 그 이진식을 참이 되게 하는 값의 조합이 존재하는지를 검사하는 문제이며 참이 되

게 하는 값들의 조합을 모델(model)이라고 한다. 만약 주어진 이진식에 참이 되게 하는 모델이 존재하면 그 이진식은 "satisfiable"이라고 하며 존재하지 않으면 "unsatisfiable"이라고 한다. 대부분의 SAT 해결기들은 특별한 표현식이 없는 이진식 대신 대부분 CNF(Conjunctive Normal Form)을 입력 형식으로 많이 사용한다. 만약 $X=C_1 \cdot C_2 \cdot C_3 \cdot \cdots \cdot C_n$ (i.e., $C_i=(a_{i1}+a_{i2}+\cdots)$)와 같은 CNF가 있을 때 C_i를 절(clause)라고하고 절 내의 a_{ij}를 리터럴(literal)이라고 한다.

CNF를 이용하는 이유는 간단히 설명할 수 있다. 예를 들어 a · (!b+c+d)라는 CNF가 있을 때 변수 b, c, d 대신에 a에 참을 할당한다. 그 이유는 CNF에서는 각 절이 AND 연산자로 연결되어 있어 하나의 절이라도 거짓이 된다면 전체가 거짓이 되기 때문이다. 위 식에서 a와 같이 단위 절(unit clause)이라 불리는 하나의 리터럴로만 구성된 절이 있을 때 이 절을 참이 우선 되게 하는 방식을 쓰고 있다. 물론 단위 절이 없는 경우에도 여러 휴리스틱을 이용하여 리터럴에 값을 할당한다.

만약 어떤 리터럴에 참 값이 할당 되었다면 그 리터럴이 속한 절을 지울 수 있다. 그 이유는 리터럴은 절 내에서 OR 연산자로 연결되었기 때문에 그 리터럴이 속한 절은 참이 되기 때문이다. 반면에 리터럴에 거짓이 할당 되었다면 해당 리터럴만 지우면 된다. 이를 BCP(boolean constraints propagation)라고 하며 여러 SAT 알고리즘들의 기본을 이루고 있다.

실제 SAT를 이용하여 프로그램 테스트 데이터를 자동으로 생성하기 위한 연구가 진행되었다[12-13]. 이 연구들에서는 SAT 기술을 이용하여 자동 테스트 데이터 생성을 어렵게 하는 "모양 문제(shape problem)"와 "플래그 문제(flag problem)"를 다루었다.

모양 문제는 프로그램이 포인터를 사용하는 경우에 테스트 데이터는 리스트, 트리 또는 그래프와 같은 2차원적인 자료구조로 표현된다[14-15]. 따라서 포인터를 고려한 테스트 데이터 자동생성 방법은 특정 프로그램 경로나 특정 프로그램 블록을 실행할 수 있는 입력 자료 구조 모양을 결정하는 것이 주요 목적이라 할 수 있다. 이와 같이 입력 자료 구조의 모양을 식별하는 문제를 모양 문제라고 한다.

플래그 문제는 프로그램이 플래그 변수를 사용하는 경우에 발생한다. 만약 플래그 변수를 갖는 프로그램에서 원하는 테스트 데이터를 찾는 문제는 매우 큰 입력 공간에서 특정 값을 찾는 문제로 변한다. 따라서 진화 알고리즘 등을 이용한 기존의 테스트 데이터 생성 알고리즘이 랜덤 테스트와 동일해져 원하는 테스트 데이터를 식별하는 것이 매우 어렵게 된다[16].

이 두 문제를 다루기 위해 [12-13]에서는 입력 프로그램을 Alloy로 변환하는 방식을 취하였다. 그 이유는 Alloy가 기본적으로 SAT에 기반을 둔 제약식 해결도구(solver)이기 때문이다[17]. Alloy는 Alloy 명세를 입력으로 받아 우선 명제 논리식(propositional logic formula)으로 변환한 후에 이를 CNF로 바꾸는 과정을 거친다. 또한 현재 Alloy 분석도구는 기본적으로 여러 SAT 해결 방법들이 제공되기 때문에 이를 별다른 노력 없이 이용할 수 있다는 이점이 있기 때문이다.

그러나 이 방법은 기본적으로 프로그램을 대상으로 하는 화이트박스 테스트 방법이다. 수행하고자하는 프로그램상의 한 지점을 주고 이를 수행할 수 있는 입력 값을 찾는 것이 주요 목표이다. 따라서 결정표 기반 테스트, 인과 그래핑, 상태 전이 테스트와 같은 블랙박스 테스트 방법의 테스트 케이스 생성 전략에 직접적으로 활용될 수 없다.

3. SAT를 기반으로 하는 MC/DC 테스트 케이스 생성

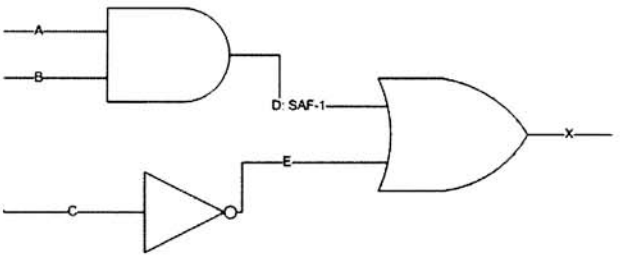
이 장에서는 SAT를 사용하여 MC/DC를 만족하는 테스트 케이스를 생성하는 방법에 대해 기술한다. 또한 제안된 방법을 다양한 블랙박스 테스트 방법과 결합하여 활용하는 것에 대해서도 기술한다.

3.1 이진 구분 함수(Binary Difference Function)

2.1절에서도 기술하였듯이 SAT는 주어진 이진식을 참이 되게 하는 모델이 존재하는가를 결정하는 문제이다. 또한 그러한 모델, 즉 이진 식을 참이 되게 하는 변수 대입(satisfying assignments)이 존재하지 않는지를 결정하는 문제도 중요한 만족성 문제이다. 만약, 이진식을 참이 되게 하는 변수 대입이 존재하지 않는지를 결정하기 위해서는 모든 가능한 경우에 대해 주어진 이진식이 거짓이 된다는 사실을 밝혀야만 한다.

기본적으로 SAT 문제는 NP-complete이다[18]. 그러나 현실적으로 많은 문제들은 많은 변수와 제약조건들이 관련되어 있다 할지라도 매우 빠른 속도로 답을 구하거나 답이 없음을 보일 수 있다. 또한 SAT 문제를 효과적으로 풀기 위하여 많은 기술적 진보가 있어 왔다. 이러한 기술은 기본적으로 이진 식을 만족하는 모델을 찾기 위해 탐색 공간의 크기를 줄이는 방법에 기반하고 있다. 대표적이면서 가장 기본적인 탐색 알고리즘은 DPLL(Davis-Putnam-Logeman-Loveland)이라 불리는 방법이며 현재의 많은 SAT 해결기들은 DPLL 알고리즘을 모순 분석(conflict analysis)나 절 학습(clause learning)과 같은 휴리스틱들을 결합하여 성능을 개선한 방법들에 기초하고 있다[6-8].

이 논문에서는 MC/DC 테스트 케이스 집합을 구하는 문제를 SAT 문제로 표현하기 위해 이진 구분 함수(Boolean Difference Function) 개념을 도입한다. 이진 구분 함수는 원래 조합 회로(combinational circuit)를 SAF(stuck-at fault)라 불리우는 결함을 검출할 목적으로 사용되었다[19]. SAF는 회로상의 임의의 한 라인에 항상 0 또는 1 신호만 제공하도록 원래의 회로를 변경시킨다. 회로 테스트의 목적은 원래의 회로와 SAF를 지닌 회로를 구분하는 입력 신호 값들을 식별하는 것이다. 이러한 입력 신호들을 테스트 패턴



(그림 1) 라인 D에 SAF-1 결함이 있는 회로

이라 한다. 만약 (그림 1)에 보여진 회로의 라인 D에 SAF-1 결함이 있다고 가정하자. SAF-1(또는 SAF-0)은 해당 라인에 1 (또는 0) 신호만 제공되는 결함을 나타낸다.

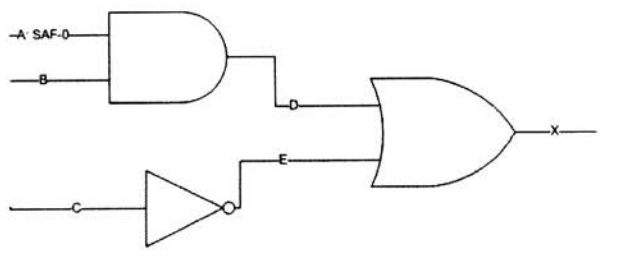
만약 D(A, B, C)=A·B라 할 때 (그림 1)의 원래의 회로는 X(A, B, C)=F(D(A, B, C), A, B, C)=D+$\overline{C}$로 나타낼 수 있다. 여기에서 $\overline{C}$는 C의 negation을 의미한다. 반면에 라인 D의 SAF-1 결함이 있는 회로는 X′(A, B, C)=F(1, A, B, C)로 모델링된다. 이 때 D(A, B, C)의 위치에 1이 있음을 주목할 필요가 있다. 따라서 테스트 패턴을 생성하기 위해서는 X(A, B, C)≠X′(A, B, C)가 성립되어야 하므로 다음 식을 만족하는 X, Y, Z 값들을 식별해야 한다: F(D(A, B, C), A, B, C)⊕F(1, A, B, C). 여기에서 '⊕'은 exclusive-or 연산을 나타낸다.

또한 D(A, B, C)가 1이 아니어야 F(D(A, B, C), A, B, C)와 F(1, A, B, C)가 다를 수 있기 때문에 위 식은 $\overline{D}$(A, B, C)·F(0, A, B, C)⊕F(1, A, B, C)로 표현된다. 여기에서 F(0, A, B, C)⊕F(1, A, B, C)를 이진 구분 함수라 한다.

그러나 위의 조합회로에서와 달리 MC/DC 테스트 케이스를 생성시키기 위해서는 단지 입력 값만 달리하여 결과 값에 차이가 발생하도록 할 필요가 있다. 이는 SAF 결함이 입력 라인에만 나타난다고 가정하여 모델링 할 수 있다. 예를 들어 (그림 1)의 X(A, B, C)에서 변수 A에 대한 MC/DC 테스트 케이스는 입력 라인 A에 SAF-0 결함이 있다고 간주하자 (그림 2 참조).

이 때 입력 라인 A의 SAF-0 결함이 있는 회로는 다음과 같은 절차를 거쳐 X′(A, B, C)=F(0, B, C)로 모델링된다:

$$\begin{aligned} X'(A, B, C) &= F(A(A, B, C), A, B, C) \\ &= F(A, A, B, C) \;//A(A, B, C)=A \\ &= F(A, B, C) \\ &= F(0, B, C) \end{aligned}$$



(그림 2) 라인 A에 SAF-0 결함이 있는 회로

X(A, B, C)≠X′(A, B, C)가 되기 위해서는 입력 라인 A에 1이 제공되고 그 결과 값이 0이 입력되는 경우와 달라야 한다. 즉 다음과 같은 이진 구분 함수를 만족하는 테스트 패턴을 생성 한다:

$$F(0, B, C)\oplus F(1, B, C)$$

위 식을 MC/DC 관점에서 해석하면 A의 값을 달리했을 때 전체 이진논리식의 값을 변화시키는 B와 C의 값을 구하는 문제로 볼 수 있다.

$$F(0, B, C)\oplus F(1, B, C)=\overline{C}\oplus(B+\overline{C})$$

위 식을 만족하는 값은 (A=1, B=1, C=1), (A=0, B=1, C=1)이며 이는 변수 A에 대해 MC/DC를 만족하는 테스트 케이스임을 알 수 있다. 따라서 일반적으로 변수 X_i에 대한 이진함수 $F(X_1, X_2, ..., X_n)$에 대한 테스트 케이스 T_p, T_q는 다음 조건 (1)-(3)들을 만족해야 한다:

(1) $F(T_p)\oplus F(T_q)$
(2) $\forall k\ (k=1..i-1,i+1..n) \mid T_p[k]=T_q[k]$
(3) $T_p[i]=T_q[i]$

여기에서 $T_x[i]$는 T_x의 i번째 변수의 값을 나타낸다.

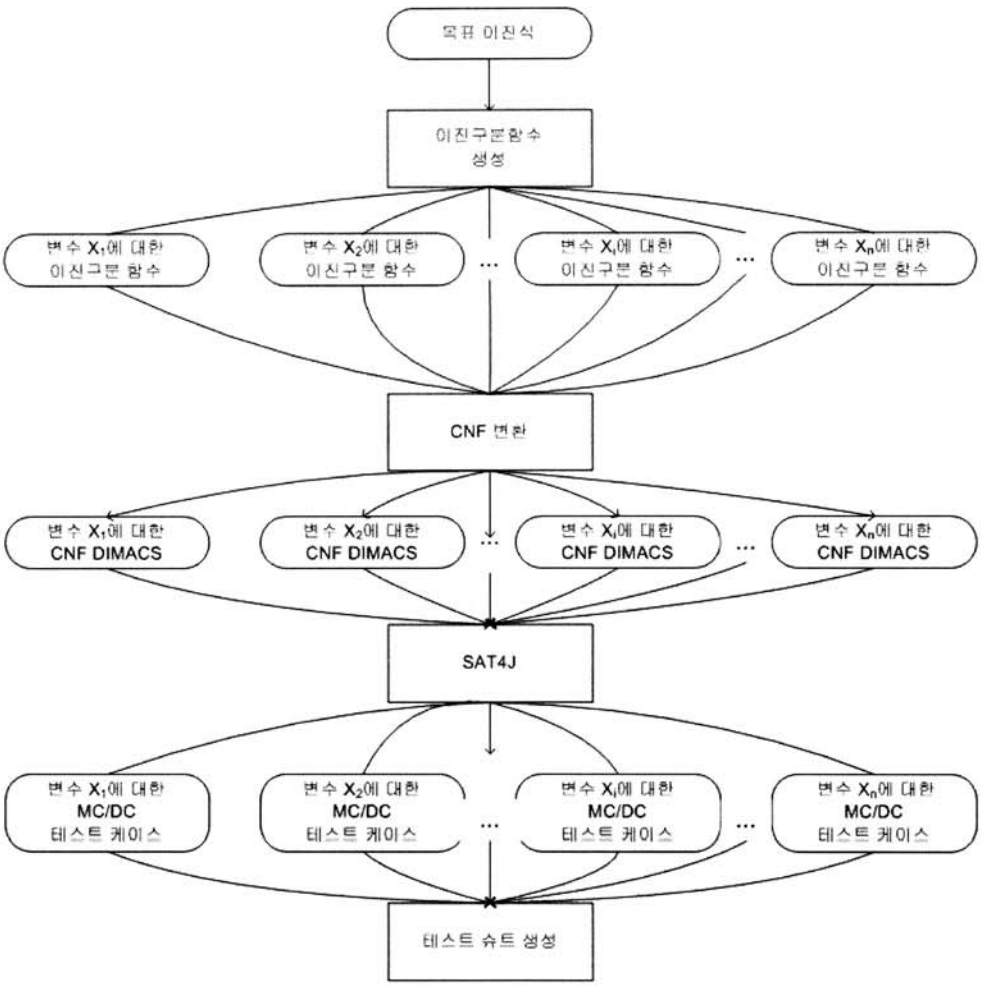
3.2 MC/DC 테스트 케이스 생성 도구: MD-SAT

본 연구에서 개발한 MD-SAT는 SAT 기술을 이용하여 MC/DC를 만족하는 테스트 케이스를 생성하는 도구이다. SAT 해결기로 SAT4J를 사용하였다. SAT4J는 SAT 기술을 JAVA 소프트웨어에서 쉽게 이용할 수 있는 매우 효율적인 라이브러리를 제공해준다. 따라서 SAT 기술의 세세한 면을 모르는 사용자일지라도 자신의 애플리케이션에 내장하여 목적에 맞게 사용할 수 있다.

(그림 3)은 MD-SAT의 전체적인 구조 및 작업 흐름을 보여준다. 그림에서 볼 수 있듯이 MD-SAT은 크게 4개의 처리 단계가 있다. 첫 번째 단계는 이진식을 3.1절에서 기술한 방식에 따라 이진식의 각 변수에 대해 이진 구분 함수를 생성한다. 그러나 SAT를 풀기 위한 대부분의 알고리즘들은 2.2절에서 언급하였듯이 특별한 형식을 지니지 않은 일반 표현식보다 CNF를 입력형식으로 받는다. 예를 들어 (그림 1)의 논리 회로를 CNF로 표현하면 다음과 같다:

$$(A+\overline{C})(B+\overline{C})$$

이와 같이 CNF로 표현된 이진식을 SAT4J의 입력으로 사용하기 위해서는 DIMACS 형식으로 변환하여야 한다. DIMACS는 여러 SAT 알고리즘들 및 휴리스틱들을 테스트하고 서로 비교하기 위해 고안된 표준 형식이다[20]. 위 CNF를 DIMACS 형식으로 표현하면 다음과 같다:



(그림 3) MD-SAT의 구조 및 작업 흐름

```
p cnf 3 2
1 -3 0
2 -3
```

제일 첫 줄은 파일 형식이 CNF라는 것이며 3은 입력 이진식에 있는 변수의 개수를 나타내고 2는 절의 개수를 나타낸다. CNF를 구성하는 각 절은 두 번째 줄부터 표현된다. DIMACS에서는 각 변수를 1부터 번호를 매기며 예에서 1은 A, 2는 B , 3은 C를 나타낸다. 음수는 변수의 논리 부정(negation)을 나타낸다. 따라서 예에서 -3은 $\overline{C}$을 나타낸다.. 두 째줄 마지막에 있는 0은 줄 구분을 나타내기 위해 사용한다. 즉, 1 -3 0은 A+$\overline{C}$ 절을 표현한 것이다. 마지막 줄은 줄 구분을 사용하지 않기 때문에 2 -3은 B+$\overline{C}$를 나타낸다.

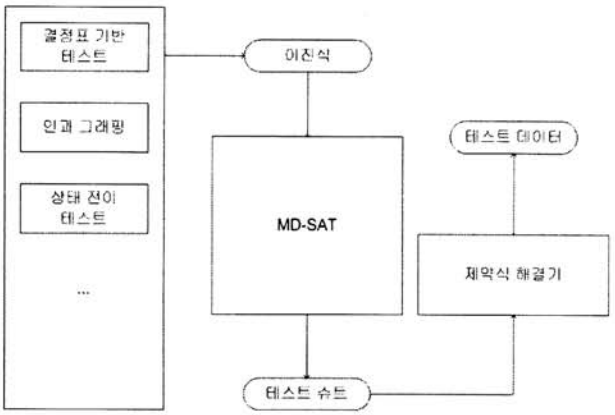
다음 단계는 각 변수에 대한 CNF DIMACS를 SAT4J에 입력으로 제공하여 해당 변수의 MC/DC 테스트 케이스를 구한다. 이와 같이 생성된 테스트 케이스들은 중복된 것도 있기 때문에 중복을 제거하여 전체 이진식의 MC/DC 테스트 슈트를 생성하는 단계가 MC/DC 테스트 케이스를 생성하는 최종 단계이다.

3.3 응용

MD-SAT은 결정표 기반 테스트, 인과 그래핑 및 상태 전이 테스트와 같이 이진식으로 결과를 생성할 수 있는 테스트 방법들의 테스트 케이스 생성 전략으로도 사용될 수 있다. 이 절에서는 결정표 기반 테스트와 인과 그래핑 및 상태 전이 테스트 각각의 방법과 연계하여 사용할 수 있는 방법을 기술한다 (그림 4 참조).

결정표(Decision Table) 기반 테스트는 명세의 형태를 결정표를 사용하여 기술한다. 결정표 자체가 일종의 이진식을 표현한 것으로 볼 수 있다. 예를 들어 <표 1>은 총 주문액과 컴퓨터의 주문 대수에 따라 할인율을 달리하는 시스템에 대한 결정표 명세이다.

MC/DC 테스트를 하기 위해서는 우선 각 입력 조건 및 행위들에 해당하는 변수들을 생성한다. 이 절차에 따라 <표 1>에서 5개의 입력 조건에 변수 C1~C5를 할당하였으며 3개의 행위에 대해서는 F1~F3 변수들을 생성하였다. 다음 단계에서 각 행위(또는 결과)에 대해 이진식을 도출한다.. 예를 들어 만약 5% 할인이 되는 경우를 MC/DC 테스트를 하기 위해서는 <표 1>에서 2번과 3번 규칙이 성립되는 경우이므



(그림 4) 블랙박스 테스트 방법과의 결합

〈표 1〉 결정 표 예: 1: 참 0: 거짓 X: Don't care

	규칙	1	2	3	4
입력 조건	주문액<500(C1)	1	1	0	0
	500 ≤ 주문액 < 1000(C2)	0	0	1	0
	주문액 ≥ 1000(C3)	0	0	0	1
	컴퓨터주문대수<10(C4)	1	0	X	X
	컴퓨터주문대수 ≥ 10(C5)	0	1	X	X
행위	할인 없음(F1)	YES	NO	NO	NO
	5% 할인(F2)	NO	YES	YES	NO
	10% 할인(F3)	NO	NO	NO	YES

로 이는 다음과 같은 이진식으로 나타난다: F2=C1$\overline{C2}$$\overline{C3}$$\overline{C4}$C5+$\overline{C1}$C2$\overline{C3}$. MD-SAT는 이 이진식을 입력으로 받아들여 3.2절에서 기술한 절차에 따라 MC/DC 기준을 만족하는 테스트 케이스를 생성한다.

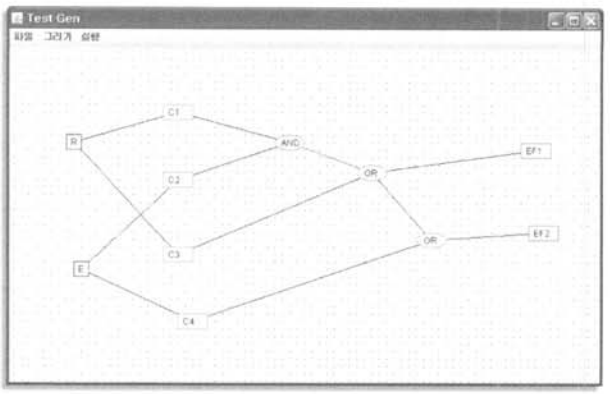
<표 2>는 입력 조건 (C1)에 대해서 MD-SAT가 생성한 MC/DC 테스트 케이스들을 보여주며 총 5개의 경우(테스트 케이스 쌍 1-2, 3-4, 5-6, 7-8, 9-10) 중에서 어느 하나를 선택하여도 C1에 대한 MC/DC 테스트 기준을 만족함을 알 수 있다. <표 2>의 테스트 케이스는 테스트하는 데 사용되는 입력 값이 만족되는 조건을 기술한 것이므로 실제 구체적인 입력 값은 lp_solve와 같은 제약식 해결기의 도움을 받아 계산되어야 한다.

이 연구에서 개발한 MD-SAT는 대표적인 블랙박스 테스트 방법인 인과 그래핑(cause-effect graphing)에도 사용될 수 있다. 인과 그래핑은 테스트 케이들을 개발할 목적으로 명세를 AND, OR, NOT과 같은 연산자를 사용하여 입력 조건에 따른 프로그램의 행위를 묘사한다. 따라서 아주 자연스럽게 명세가 이진식으로 모델링될 수 있다. (그림 5)는 저자가 개발한 인과 그래핑 도구[1]의 화면 스냅 샷을 보여준다.

(그림 5)의 인과 그래프는 두 개의 행위가 묘사되어 있다:

〈표 2〉 입력 조건 C1에 대한 MC/DC 테스트 케이스

테스트 케이스	C1	C2	C3	C4	C5	결과
1	T	F	T	T	T	T
2	F	F	T	T	T	F
3	T	T	F	T	F	F
4	F	T	F	T	F	T
5	T	T	F	F	T	F
6	F	T	F	F	T	T
7	T	F	F	F	T	T
8	F	F	F	F	T	F
9	T	T	F	F	F	F
10	F	T	F	F	F	T



(그림 5) 인과 그래핑 도구 화면 샷

EF1과 EF2. 인과 그래프로부터 MC/DC 테스트 케이스들을 생성하기 위해서는 각 행위를 묘사하는 이진 함수들을 식별해야 한다. 이 작업은 그래프 순회 알고리즘을 이용하여 매우 간단하게 수행할 수 있다. 예를 들면 행위 EF2에 해당하는 이진 함수는 다음과 같다:((C1C2)+C3)+C4. 그러나 이렇게 추출된 이진 함수는 EF2 행위를 완전하게 기술하지 않는다. 그 이유는 입력 조건들 간의 제약사항을 반영하지 않았기 때문이다. 예제 인과 그래프에서 입력 조건 C1과 C2는 "Require" 관계가 있으며 C2와 C4는 "Exclusive" 관계가 존재한다. "Require" 관계는 어느 한 입력 조건이 만족되면 다른 조건은 반드시 만족되어야 하는 상황을 기술한다. 이는 논리 연산자 "implication"으로 모델링할 수 있다. 또한 "Exclusive" 관계는 입력 조건들이 동시에 참이 될 수 없는 상황을 나타내므로 논리연산자 "exclusive-or" ⊕로 모델링된다.

따라서 이진식과 입력 제약조건들이 동시에 만족되어야 하므로 이들을 'AND'하면 원하는 최종 이진식이 생성된다. 따라서 예제 인과 그래프로부터 EF2에 대한 이진식은 다음과 같다:

$$(((C1C2)+C3)+C4)(C1=>C3)(C2\oplus C4)$$

이 이진식을 바탕으로 3.2절에 기술한 절차에 따라 MC/DC 기준을 만족하는 테스트 케이스를 구하면 된다.

상태 전이 테스트의 주요 관심사는 시스템이 입력 이벤트

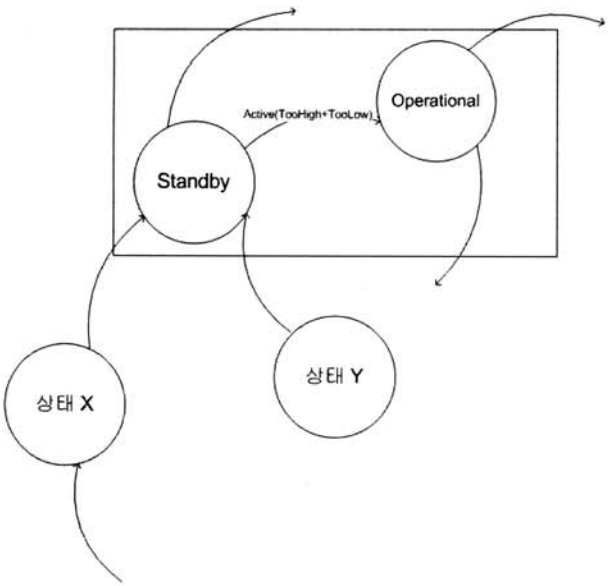
1) 이 논문의 주요 목적이 인과 그래핑 도구가 아니기 때문에 자세한 설명은 이 논문에서 기술하지 않는다.

에 대한 응답으로 정확한 행위가 발생하고 올바른 상태에 도달하는지를 검증하는 것이다. 이를 위하여 상태 전이 테스트는 시스템을 상태전이도로 모델링한 후 상태전이도로부터 체계적으로 테스트 케이스들을 선정한다. 이 논문에서는 상태간의 전이가 올바로 되는지를 테스트하기 위해 상태 전이로부터 MC/DC 테스트 케이스들을 추출하는 방법에 대해 기술한다.

기본적으로 상태 간의 전이는 술어(predicate)로 기술될 수 있다. 예를 들면 (그림 6)은 센서를 이용해서 주변의 온도가 너무 높거나 낮을 때 어떤 행위를 수행하는 시스템을 기술하는 상태 전이도의 일부분이다. 여기에서 "Stanby" 상태에서 "Operational" 상태로 전이가 되는 상황을 생각해보자. 이 전이가 정확하게 이루어지는 상황은 다음과 같은 이진식으로 기술되어 있음을 알 수 있다: Active·(TooHigh+TooLow). 여기에서 Active는 시스템의 작동여부를 나타내는 제어(이진) 변수이고 TooHigh, TooLow는 온도가 너무 높거나 낮음을 나타내는 제어 변수이다.

이 전이를 기술하는 술어를 MD-SAT 입력으로 제공하면 <표 3>에 보이는 MC/DC 테스트 케이스 집합이 생성된다.

테스트 케이스 (1-2), (3-7), (5-6)은 조건 Active에 대한 MC/DC 테스트 케이스 쌍들이고, (3-4)는 조건 TooHigh, (4-5)는 TooLow에 대한 MC/DC 테스트 케이스 쌍이다. 테스트 케이스 1만 정상적으로 전이가 이루어지는 것을 검증하는 것이고 나머지는 비정상적으로 전이가 이루어졌을 때의 상황을 테스트한다. 예를 들어, 테스트 케이스 4번을 보면 Active가 참이지만 온도가 너무 높지도 낮지도 않은 상태인데 "Operational" 상태로 전이가 되었다면 시스템이 잘못된 것이라고 판단할 수 있다. 이처럼 상태 전이를 나타내는 술어에 MC/DC 기준을 적용하여 매우 다양한 경우에 대해서 테스트를 수행할 수 있다.



(그림 6) 상태 전이도 예제

〈표 3〉 MC/DC 적용하여 생성된 테스트 케이스 집합

테스트 케이스	Active	TooHigh	TooLow	결과
1	T	T	T	T
2	F	T	T	F
3	T	T	F	T
4	T	F	F	F
5	T	F	T	T
6	F	F	T	F
7	F	T	F	F

4. 실험 결과

이 절에서는 이 논문에서 제안한 SAT 기반 MC/DC 테스트 케이스 생성 방법의 효용성을 보이기 위해 실험한 결과를 기술한다. 실험을 위해 항공기 충돌 회피 시스템의 명세 TCAS II의 20개의 이진식을 대상으로 한다 <표 4 참조>[21].

SAT가 효과적으로 MC/DC 테스트 케이스 생성에 적용된다는 것을 보이기 위해 이진식의 크기에 따른 테스트 케이스 생성 시간을 측정한다. 이진식의 크기를 다양하게 정의할 수 있지만 이 논문에서는 이진식을 구성하는 변수의 수와 연산자의 수의 합으로 정의하여 사용한다. 예를 들어 A(B+C)D(E+F+B)의 크기는 변수의 개수(중복을 포함)가 7

〈표 4〉 실험에 사용된 TCASII 이진 명세

	이진식(Boolean Expression)
1	!(ab)(d!e!f+!de!f+!d!e!f)(ac(d+e)h+a(d+e)!h+b(e+f))
2	(a((c+d+e)g+af+c(f+g+h+i))+(a+b)(c+d+e)i)!(ab)!(cd)!(ce)!(de)!(fg)!(fh)!(fi)!(gh)!(hi)
3	(a(d+e+de!(fghi+ghi)!(fglk+gik))+!(fghi+ghi)!(fglk+gik)(b+cm+f))(abc+abcabc)
4	a(!b+!c)d+e
5	a(!b+!c+bc!(!fgh!i+!ghi)!(!fglk+!g!ik))+f
6	(!ab+a!b)!(cd)(f!g!h+!fg!h+!f!g!h)!(jk)((ac+bd)e(f+(i(gj+hk))))
7	(!ab+a!b)!(cd)!(gh)!(jk)((ac+bd)e(!i+!g!k+!j(!h+!k)))
8	(!ab+a!b)!(cd)!(gh)((ac+bd)e(fg+!fh))
9	!(cd)(!ef!g!a(bc+!bd))
10	a!b!cd!ef(g+!g(h+i))!(jk+!jl+m)
11	a!b!c!((f(g+!g(h+i)))+f(g+!g(h+i))!d!e)!(jk+!jl!m)
12	a!b!c(f(g+!g(h+i))(!e!n+d)+!n)(jk+!jl!m)
13	a+b+c+!c!def!g!h+i(j+k)!l
14	ac(d+e)h+a(d+e)!h+b(e+f)
15	a((c+d+e)g+af+c(f+g+h+i))+(a+b)(c+d+e)i
16	a(!d+!e+de!(!fgh!i+!ghi)!(!fglk+!g!ik))+!(!fgh!i+!ghi)!(!fglk+!g!ik)(b+c!m+f)
17	(ac+bd)e(f+(i(gj+hk)))
18	(ac+bd)e(!i+!g!k+!j(!h+!k))
19	(ac+bd)e(fg+!fh)
20	!ef!g!a(bc+!bd)

이고 AND 연산자가 3개, OR 연산자가 3개이므로 7+6=13이다.

테스트 케이스 생성 시간은 두 가지 방식으로 측정하였다. 첫 번째는 이진식이 주어졌을 때 이를 각 변수에 대한 이진 구분 함수를 생성하여 CNF로 변환한 후 SAT4J가 처리하는 데까지 걸리는 총시간을 구하였다. 이를 <표 5>에서 시간1로 나타내었다. 두 번째는 순수하게 SAT4J가 처리하는 시간만을 고려하였다. 즉, 시간1에서 CNF로 변환하는데 걸리는 시간은 고려하지 않았다. 이 시간을 <표 5>에서 시간2로 나타내었다.

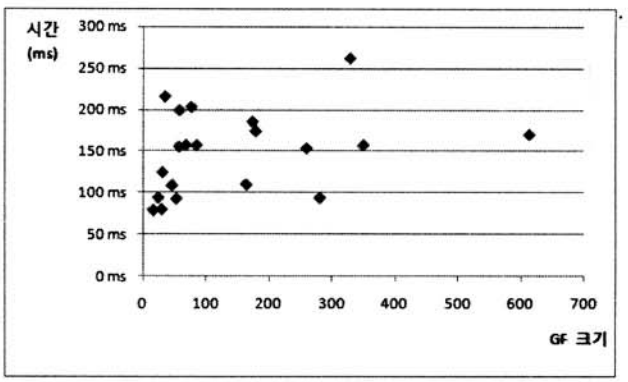
이와 같이 정의된 이진식의 크기와 테스트 케이스 생성 시간의 관계는 세 가지 방식으로 이루어진다. 우선 원래의 이진식(GF로 표시)과 시간의 관계가 측정하고 CNF로 변환된 후에 변환된 CNF 크기와 시간과의 관계를 측정해 보았다. 또한 변수의 개수(이 경우에는 중복을 허용하지 않음)만을 사용하여 테스트 케이스의 생성 시간과의 관계를 알아보았다. <표 5>는 이렇게 수행된 실험결과를 표로 보인 것이며 (그림 7)은 GF 크기. CNF 크기, 변수의 개수와 시간1과 관계를 그래프로 나타낸 것이다.

우선 이 결과의 의미를 기술하기 전에 MC/DC 테스트 케이스를 찾아내기 위한 탐색 공간의 크기를 계산해보자. 이진식의 탐색 공간의 크기는 다음과 같이 계산된다. 만약 n개의 변수를 지닌 이진식으로부터 MC/DC 테스트 케이스를 생성하기 위해서는 각 변수에 대해 모든 가능한 진리 값들(i..e., 2^n)로 부터 2개의 테스트 케이스를 뽑아야 하기 때문에 주어진 이진식 전체 탐색 공간의 크기는 $n \times {}_{2^n}C_2$가 된다. 예를 들면, <표 4>에 사용된 20개 이진식들의 평균 변수의 개수는 약 10개 이다. 이 경우 탐색 공간의 크기를 계산하면 약 10^7이 되기 때문에 원하는 테스트 케이스를 식별하는 것은 매우 힘들다는 것을 알 수 있다.

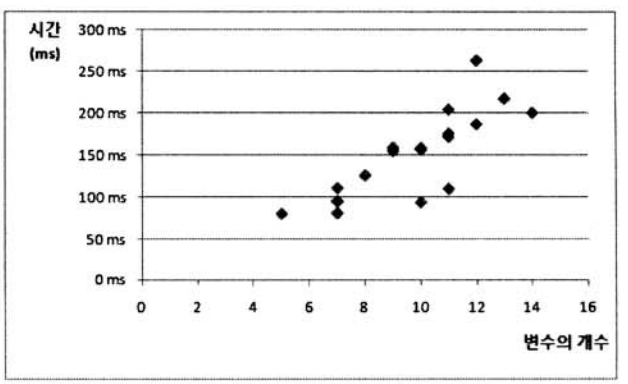
〈표 5〉 테스트 케이스의 생성 시간

이진식	GF 크기	CNF 크기	변수의 개수	시간1(ms)	시간2(ms)
1	54	280	7	94	47
2	80	349	9	158	46
3	95	613	11	171	47
4	11	15	5	79	31
5	49	67	9	158	31
6	66	178	11	175	47
7	52	84	10	158	47
8	38	52	10	93	47
9	24	29	7	80	31
10	35	34	13	217	31
11	49	76	11	204	31
12	41	57	14	200	31
13	30	173	12	186	47
14	24	163	7	110	47
15	35	259	9	154	31
16	92	328	12	263	48
17	21	45	11	109	32
18	27	56	10	156	31
19	18	30	8	125	32
20	19	23	7	94	32

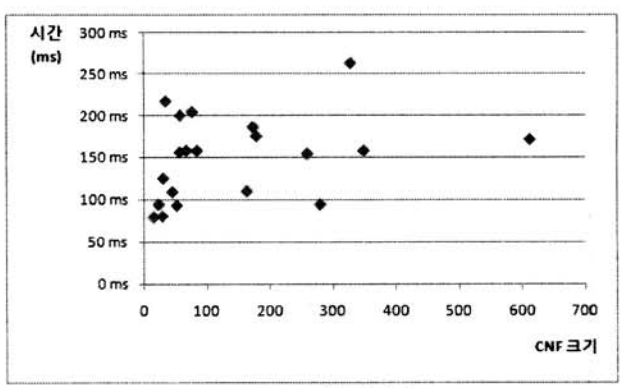
우선 시간1 관점에서 결과를 분석하자. <표 5>에서 쉽게 유추할 수 있는 사실은 평균 테스트 케이스 생성시간은 149ms이고 최대 생성 시간이 263ms이다. 이는 주어진 이진식에 대한 탐색 공간의 크기를 고려할 때 MC/DC 테스트 케이스 생성이 매우 짧은 시간 내에 이루어진다는 점이다. (그림 7)-(b)에서 볼 수 있듯이 GF 크기와 CNF 크기에 따른 테스트 케이스 생성 시간은 그다지 큰 차이를 보이지 않



(a) GF 크기와의 관계



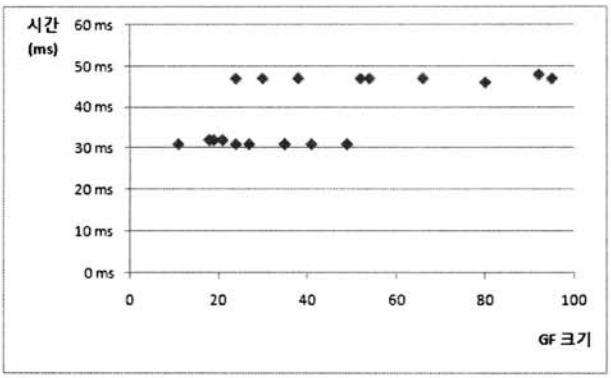
(c) 변수 개수와의 관계



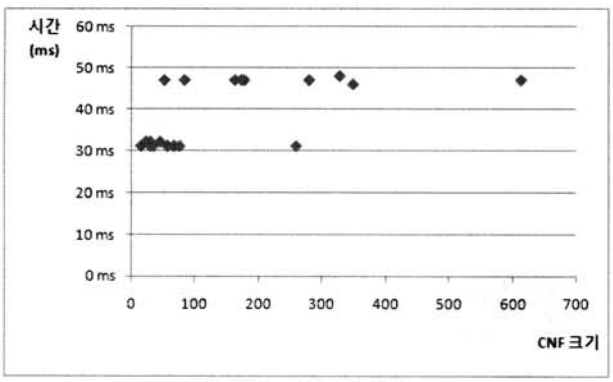
(b) CNF 크기와의 관계

(그림 7) 이진식 크기와 테스트 케이스 생성시간(시간1)과의 관계

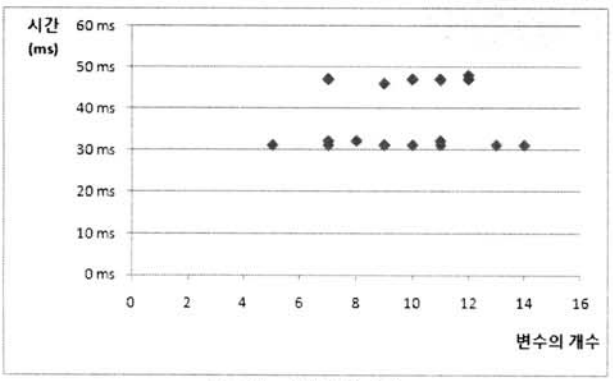
는다. 실제로 GF 크기와 시간과의 상관계수는 약 0.6으로 같다. 이 사실은 원래의 예제에 사용된 이진식의 복잡도가 정규화 된 식에도 반영이 된다는 사실을 의미한다. 실제로 GF 크기와 CNF 크기의 상관계수는 약 0.81로 상당히 강한 상관관계를 갖고 있다. (그림 7)-(c)에서 볼 수 있듯이 변수의 개수와 테스트 케이스 생성 시간은 상당히 선형적으로 밀접한 상관관계를 가지고 있음을 알 수 있다. 변수의 개수와 테스트 케이스 생성 시간의 상관계수는 약 0.80이다. 이는 GF 크기와 CNF 크기보다도 변수의 개수가 더욱 테스트 케이스 생성 시간에 영향을 준다는 사실을 의미한다. 즉, 변수의 개수와 MC/DC 테스트 케이스 생성시간이 (선형적으로) 비례한다. 이는 MC/DC 테스트 케이스는 각 변수에 대해 이진 구분 함수를 생성하기 때문에 변수의 개수가 많아질수록 시간이 많이 소요된다.



(a) GF 크기와의 관계



(b) CNF 크기와의 관계



(c) 변수 개수와의 관계

(그림 8) 이진식 크기와 테스트 케이스 생성시간(시간2)과의 관계

(그리 8)은 시간2의 관점에서 이진식 크기와의 관계를 그래프로 표현한 것이다. 그래프에서 볼 수 있듯이 이진식의 크기에 생성 시간이 선형적인 관계가 그리 강하지 않다는 것을 알 수 있다. 시간2는 실제 SAT 해결기에서 소요되는 시간만을 측정한 것이기 때문에 예제로 사용된 이진식들의 복잡도는 SAT 해결기의 처리 성능에 그다지 큰 영향을 미치지 않음을 알 수 있다. 평균 처리 시간은 약 38ms로 아주 빠른 시간 내에 주어진 이진식에 대한 MC/DC 테스트 케이스들을 생성함을 알 수 있다.

5. 결 론

이 논문에서는 MC/DC 테스트 케이스를 효과적으로 생성하기 위해 SAT 기술을 이용하는 테스트 데이터 생성 방법을 이용한 도구 MD-SAT에 대해 기술하였다. 이 논문에서 기술한 방법은 기존의 방법과는 다르다. 기존의 방법은 모양 문제나 플래그 문제와 같이 프로그램 코드를 대상으로 할 때 발생하는 문제 해결에 주안점을 두는 데 반해 이 논문에서 기술한 방법은 명세 정보에서 추출한 이진식을 바탕으로 MC/DC 테스트 케이스를 생성한다는 점이다. 따라서 결정표 기반 테스트, 인과 그래핑, 상태 전이 테스트와 같이 다양한 테스트 방법에서 테스트 케이스를 생성할 수 있는 도구로 사용될 수 있다. 또한, .jar로 개발되어 다른 테스트 도구에서도 쉽게 이용 가능하다. 가까운 시일에 이 도구를 공개할 계획을 가지고 있다.

SAT를 이용하여 이진식으로 부터 테스트 케이스를 생성하는 방법은 MC/DC를 만족하는 테스트 케이스 생성에만 머무를 이유가 없다. MC/DC 방법이외에도 이진식으로 부터 테스트 케이스를 생성하는 여러 기준 및 방법에 대해 제안되어 왔으며 여전히 활발한 연구가 진행 중이다[22]. 따라서 향후의 연구로 이진식에 바탕을 둔 여러 테스트 방법에 대해 SAT 기술이 얼마나 효과적으로 적용될 수 있는지 평가할 필요가 있으며 이들을 자동화 도구로 구현할 필요가 있다.

또한 프로그램 코드를 대상으로 MC/DC 기준을 만족하는 테스트 케이스 생성 방법에 대한 연구도 필요하다. 이를 위하여 프로그램 코드에 논리의 제어를 위해 'IF' 문이나 'WHILE' 문의 조건식에 나타나는 이진식의 정보를 바탕으로 테스트 케이스를 자동 생성하여야 한다. 그러나 이 경우에 다음 두 문제를 고려해야 한다. 첫 번째 문제는 우선 대상이 되는 조건식까지 도달할 수 있는 입력 값을 생성하는 데 단순한 프로그램에서는 별 문제가 되지 않을 수 있지만 복잡한 제어 흐름을 갖는 프로그램에서는 매우 어려운 일이다. 두 번째로 내부 변수(internal variable) 문제가 있다. 테스트 케이스의 입력 값은 결국 입력 변수의 값들로 표현되어야 한다. 그러나 프로그램 코드는 계산 및 계산 결과의

저장 등 여러 목적을 위해 입력 변수들 뿐만이 아니고 내부 변수들을 사용한다. 이러한 내부 변수들의 값은 입력 변수 값들의 관계를 나타내는 함수로 표현되며 이 함수를 식별하여야 원하는 테스트 케이스를 생성할 수 있다. 그러나 내부 변수와 입력 변수간의 관계는 매우 복잡할 수 있기 때문에 관계를 정확하게 파악하여 테스트 케이스를 생성하는 작업은 매우 어렵다.

참 고 문 헌

[1] RTCA/DO-178B, "Software Considerations in Airborne Systems and Equipment Certification," RTCA, Washington D.C., 1972.

[2] K. J. Hayhurst, D. D. Veerhusen, J. J. Chilenski, and L. K. Rierson, "A Practical Approach to Modified Condition/ Decision Coverage," *Proc. the 20th Digital Avionics Systems Conf.(DASC)*, pp. 1B2/1-1B2/10, FL., USA, Oct., 2001.

[3] A. Dupuy and N. Leveson, "An Empirical Evaluation of the MC/DC Coverage Criterion on the HETE-2 Satellite Software," *Proc. the 19th Digital Avionics Systems Conf.(DASC)*, Phil., USA., RTCA, Washington D.C., Oct., 2000.

[4] K. Kapoor and J. Bowen, "Experimental Evaluation of the Variation in Effectiveness for DC, FPC and MC/DC Test Criteria," *Proc. the 2003 International Symp. on Empirical Software Eng.(ISESE)*, pp.185-194, 2003.

[5] 김형태, "소프트웨어 테스트 커버리지 요건 및 규제방향", *NUPIC 2009 제1회 원전계측제어 심포지움*, 충무 마리나리조트, Nov., 2009.

[6] M. W. Moskewicz, Y. Zhao, L. Zhang, and Malik, "Chaff: Engineering an Efficient SAT solver," *In Proc. 38th Design Automation Conference(DAC)*, pp.530-535, 2001.

[7] E. Goldgerg and Y. Nivikov, "BerkMin: A Fast and Robust SAT solver," *In DATE*, pp.142-149, 2002.

[8] J. P. Marques-Silva and K. A. Sakallah, "GRASP: A Search Algorithm for Propositional Satisfiability," *IEEE Trans.. on Computers*, Vol.48, pp.506-521, 1999.

[9] http://www.sat4j.org.

[10] A. P. Mathur, 'Foundations of Software Testing,' 1st ED., Addison-Wesley, 2008.

[11] J. J. Chilenski, "An Investigation of Three Forms of the Modified Condition Decision Coverage(MCDC) Criterion," Technical Report DOT/FAA/AR-01/18, , US Depart. of Transportation, Washington, DC., 2001.

[12] 정인상., "SAT에 기반한 포인터가 있는 프로그램을 위한 목적지향 테스트 데이터 생성", 인터넷정보학회논문지, 제9권 2호, pp.89-105, 2008.

[13] 정인상., "SAT를 기반으로 하는 플래그 변수가 있는 프로그램 테스팅을 위한 테스트 데이터 자동 생성", 정보처리학회논문지, 제16-D권 제3호, pp.371-380, 2009.

[14] J. Edvardsson, "A Survey on Automatic Test Data Generation", In *Proc. the Second Conf. on Computer Science and Engineering*, pp.21-28, 1999.

[15] I.S. Chung and J. M. Bieman, "Generating Input Data Structures for Automated Program Testing", Software Testing, Verification and Reliability, Vol.19, pp.3-36, 2009.

[16] M. Harman, R. Hu, R. Hierons, A. Baresel, and M. Sthamer, "Improving Evolutionary Testing by Flag Removal", Information and Software Test Technology, Vol.43, No.14, pp.841-854, 2001.

[17] S. A. Cook, "The Complexity of Theorem-Proving Procedures", In. *Proc. 3rd ACM Symp. on Theory of Computing*, pp.151-158, 1971.

[18] D. Jackson, "Alloy: A light weight object modeling notation", Technical Report 797, MIT Lab for Computer Science, Feb., 2000.

[19] N. K. Jha and S. Gupta, 'Testing of Digital Systems', 1st ED., Cambridge University Press, 2003.

[20] ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/doc/satformat.dvi.

[21] E. Weyuker, T. Goradia, A. Singh, "Automatically Generating Test Data from a Boolean Specification", IEEE Trans. on Software Eng., Vol.20, No.5, pp.353-363, 1994..

[22] G. Kaminski, G. Williams, and P. Ammann, "Reconciling Perspectives of Software logic Testing", Software Testing, Verification and Reliability, Vol.18, pp.149-188, 2008.

정 인 상

e-mail : insang@hansung.ac.kr

1987년 서울대학교 컴퓨터공학과(학사)

1989년 한국과학기술원(KAIST) 전산학과 (석사)

1993년 한국과학기술원(KAIST) 전산학과 (박사)

1999년~현 재 한성대학교 컴퓨터공학과 교수

관심분야 : 소프트웨어 공학, 소프트웨어 테스팅