MC/DC 커버리지를 만족하는 콘콜릭 테스팅 연구

최석원*, 구근회**, 최진영***
*고려대학교 정보대학 컴퓨터학과
**고려대학교 정보대학 컴퓨터전파통신학과
***고려대학교 정보보호대학원
{swchoi*, khkoo**, choi***}@formal.korea.ac.kr

Concolic testing research to satisfy the MC/DC coverage

Seok-won Choi*, Keun-hoi Koo**, Jin-Young Choi***
*Dept of Computer Science and Engineering, Korea University
**Dept of Computer and Communication, Korea University
***Graduate school of Information Security, Korea University

요 약

콘콜릭 테스팅(Concolic testing)은 프로그램 실행을 실제 구체적인 입력 값(Concrete Values)을 입력하며, 동시에 심볼릭 실행(Symbolic execution)을 진행하는 테스팅 기법이다. 콘콜릭 테스팅은 모든 실행 가능 경로를 탐색하여, 테스트 케이스를 자동 생성한다. 그래서 높은 분기 커버리지를 나타내지만, 안전성을 목표로 하는 MC/DC 커버리지는 만족하지 못한다. 본 논문에서는 분기 커버리지와 MC/DC 커버리지를 만족하는 테스트 케이스를 자동생성하는 CREST 도구 개선을 제안한다.

1. 서론

임베디드 소프트웨어가 의료, 항공 등 다양한 분야에 적용되면서 이에 따른 안전성 문제가 많이 제시되고 있다. 그래서 임베디드 소프트웨어 테스팅에 대한 중요성은 커지고 있다. 이에 주로 사용되는 테스팅 기법인 콘콜릭 테스팅(Concolic testing)은 프로그램 실행을 실제 구체적인 입력 값(Concrete Values)을 입력하며, 동시에 심볼릭 실행(Symbolic execution)[6]을 진행하는 테스팅 기법이다. 콘콜릭 테스팅은 모든 실행 가능 경로를 탐색하여, 테스트 케이스를 자동 생성하는 장점이 있다. 하지만 안전성을 목표로 하는 MC/DC 커버리지는 만족하지 않는다. MC/DC 커버리지는 대표적으로 항공기에 관한 소프트웨어의 안전성을 인증하기 위해 사용되는 국제표준 DO-178B에서도 고레벨을 받기 위해 꼭 만족시켜야 한다.[4] 그래서 이와 관련된 연구들이 많이 진행되고 있다. [1]의 연구에서는 콘콜릭 테스팅 도구인 CREST에 XNCT라는 도구를 개발 및 적용하여 MC/DC 커버리지를 만족하게 도구를 수정하였고, [2]의 연구에서는 MC/DC 테스트 케이스를 진리표 형태로 생성해주는 도구를 연구하고 있다. 이와 달리 본 논문에서 제안하는 도구는 콘콜릭 테스팅을 통해 높은 분기 커버리지를 만족하는 테스트케이스 자동생성과 동시에 MC/DC 테스트케이스를 자동 생성하는 CREST 도구[3] 수정을 제안한다.

"본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음" (IITP-2016-H8501161012000l002)

2. 관련연구

2.1 콘콜릭 테스팅(Concolic testing)

콘콜릭 테스팅은 프로그램 실행을 실제 구체적인 입력 값을 입력하며, 동시에 심볼릭 실행을 진행하는 테스팅 기법이다. 심볼릭 실행에서 생성되는 심볼릭 제약식을 제약식 해독기인 SMT Solver[7]를 사용하여 실제 구체적인 입력값을 생성한다. 콘콜릭 테스팅의 장점은 실제 구체적인 테스트 케이스를 자동 생성해주며, 심볼릭 실행을 통하여 프로그램 내의 모든 실행 가능 경로를 탐색하기 때문에 높은 분기 커버리지를 보여준다. 전통적인 테스팅 방식에서는 프로그램의 요구사항에 맞는 테스트 입력 값을 생성하여 테스트 하였지만 콘콜릭 테스팅은 프로그램 내의 모든 실행 가능 경로를 탐색하여 테스터가 생성하기 힘든 테스트 케이스(Corner case)를 생성하기 때문에, 프로그램의 버그를 찾기에 용이하다.

2.1 MC/DC(Modified Condition/Decision Coverage) 커버리지

MC/DC 커버리지는 프로그램 내의 실행 가능 경로에서 각 개별 조건식이 다른 개별 조건식에 영향을 받지 않고, 전체 조건식의 결과에 독립적으로 영향을 주도록 한다. 이는 조건/결정 커버리지를 향상시킨 것으로 결정 커버리지, 조건/결정 커버리지보다 강력하다.[4] 다른 조건들의 변동이 없고 자신의 조건이 변경 되었을 때 결과 값에 영향을 미치는 경우 해당 상태를 MC/DC를 만족한다고 한다.[2] 특히 항공기에 관한 소프트웨어의 안전성을 인증 받기 위해 사용되는 국제 표준 DO-178B에서는 MC/DC 커버리지

를 100%만족 해야한다.

3. 실험 내용

본 논문에서 사용하는 콘콜릭 테스팅 도구인 CREST는 C프로그램을 대상으로 테스팅을 하는 오픈소스로 개발된 도구이다. CREST의 동작 방식은 1)테스트할 프로그램 내에서 심볼릭 변수를 결정한 후 2) crestc 도구를 사용하여 프로그램을 수정 후 컴파일을 한다. 3) 마지막으로 컴파일한 프로그램을 run_crest 도구를 사용하여 실행하게 된다. 본 논문에서는 CREST를 사용하여 생성되는 테스트 케이스와 심볼릭 실행을 통해 생성된 심볼릭 제약식을 바탕으로 MC/DC 테스트 케이스를 도출하여 MC/DC 커버리지를 비교한다.

3.1 콘콜릭 테스트케이스와 MC/DC테스트 케이스 비교

```
#include <crest.h>
#include <stdio.h>

int main(void) {
  int a, b, c, d;
  CREST_int(a);
  CREST_int(b);
  CREST_int(c);
  CREST_int(d);

  if (a == 10) {
    if (b == 19) {
      if (c == 7) {
        if (d == 4) {
          printf("GOAL₩n");
          }
      }
    }
  }
  return 0;
}
```

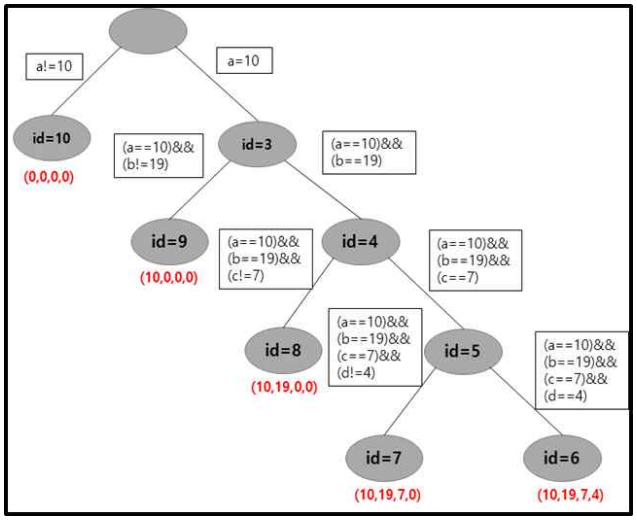
(그림 1) 예제 소스코드

먼저 MC/DC 테스트 케이스를 생성하기 위하여, CREST 도구를 수정하였다. 수정된 CREST는 그림2와 같이 1) 입력 값, 2) 출력 값, 3) 실행경로 4) 심볼릭 제약식을 출력하게 수정하였다. 수정된 CREST 도구를 사용하여 그림 1의 예제 코드를 실행하였을 때, 그림 3과 같은 프로그램 실행 경로를 트리 형태로 나타낼 수 있다.

```
[1. Input Values]
10 19 7 4

[2. Run Program]
GOAL

[3. Symbolic Execution Path : Branch_id]
[ 3 -> 4 -> 5 -> 6 ]

[4. Symbolic Path Formula]
(=  (- x0 10 ) 0 )
(=  (- x1 19 ) 0 )
(=  (- x2 7 ) 0 )
(=  (- x3 4 ) 0 )
```

(그림 2) 수정된 CREST 출력화면

먼저 초기 값을 0으로 설정하여 첫 번째 분기문을 만족하지 못하는 경로를 탐색한다. 그리고 이 때 생성된 심볼릭 값(a!=10)에 not 연선자를 추가하여 첫 번째 분기문을 만족하는 경로 (!(a!=10))를 탐색하게 된다. 이런 과정을 반복하고 더 이상 실행 가능한 경로가 없을 때까지 실행하여 최종적으로 모든 분기문을 만족하는 노드6에 도달할 수 있다. 이 때 생성된 테스트 케이스는 (0,0,0,0), (10,0,0,0), (10,19,0,0), (10,19,7,0), (10,19,7,4) 5가지가 생성된다.



(그림 3) 예제 코드의 실행경로

그리고 그림 2의 출력된 심볼릭 제약식을 통하여 MC/DC 테스트 케이스를 도출할 수 있다. 이 때 각각의 조건 a=10, b=19, c=7, d=4를 A,B,C,D로 추상화하여 그림 4와 같은 진리표를 도출할 수 있다. RESULT는 A&&B&&C&&D 에 대한 값이다. 진리표를 통하여 MC/DC를 만족하는 테스트 케이스는 1,2,3,5,9 이다.

TESTCASE	A	B	C	D	RESULT
1	T	T	T	T	T
2	T	T	T	F	F
3	T	T	F	T	F
4	T	T	F	F	F
5	T	F	T	T	F
6	T	F	T	F	F
7	T	F	F	T	F
8	T	F	F	F	F
9	F	T	T	T	F
10	F	T	T	F	F
11	F	T	F	T	F
12	F	T	F	F	F
13	F	F	T	T	F
14	F	F	T	F	F
15	F	F	F	T	F
16	F	F	F	F	F

(그림 4) MC/DC 테스트 케이스

진리표를 통하여 생성된 MC/DC 테스트 케이스 5가지를 만족해야 MC/DC 커버리지를 100% 만족한다고 한다. 하지만 그림 5와 같이 콘콜릭 테스팅에서 생성된 테스트 케이스와 MC/DC 테스트 케이스를 비교했을 때 콘콜릭 테스팅은 MC/DC 커버리지를 40% 만족하는 것을 확인할 수 있었다.

▶Concolic Test suite	▶MC/DC Test suite
1. a=0, b=0, c=0, d=0	1. a=10, b=19, c=7, d=4
2. a=10, b=0, c=0, d=0	2. a=10, b=19, c=7, d=0
3. a=10, b=19, c=0, d=0	3. a=10, b=19, c=0, d=4
4. a=10, b=19, c=7, d=0	4. a=10, b=0, c=7, d=4
5. a=10, b=19, c=7, d=4	5. a=0, b=19, c=7, d=4
MC/DC coverage : 40%	MC/DC coverage : 100%

(그림 5) 콘콜릭 테스트 케이스와 MC/DC 테스트 케이스 비교

4. 결론

본 논문에서 콘콜릭 테스팅을 통해 생성된 테스트 케이스와 MC/DC 테스트 케이스를 비교하여 콘콜릭 테스팅에서는 MC/DC 커버러지리를 만족시키지 못하는 것을 확인하였다. 콘콜릭 테스팅이 높은 분기 커버리지를 달성하고, MC/DC 커버리지도 만족한다면 보다 안전성이 높은 소프트웨어 개발에 도움을 줄 수 있다. 향후 연구에는 수정한 CREST 도구를 개선시키는 방향으로 할 것이다. 먼저 심볼릭 제약식을 통하여, 그림4와 같은 진리표를 생성하고, 생성된 진리표에서 MC/DC 테스트 케이스를 제약식 해독기를 사용하여 생성하게 수정한다. 따라서 수정된 도구에서 예상 출력화면은 그림 6과 같이 콘콜릭 테스팅을 통해 생성된 테스트 케이스와 MC/DC 커버리지를 만족하는 테스트 케이스를 출력하게 한다.

▶Concolic test suite + MC/DC Test suite
1. a=0, b=0, c=0, d=0
2. a=10, b=0, c=0, d=0
3. a=10, b=19, c=0, d=0
4. a=10, b=19, c=7, d=0
5. a=10, b=19, c=7, d=4
6. a=10, b=19, c=0, d=4
7. a=10, b=0, c=7, d=4
8. a=0, b=19, c=7, d=4

MC/DC coverage 100%

(그림 6) 도구 개선시 예상되는 테스트 케이스

참고문헌

[1] Godboley, Sangharatna, et al. "Enhanced modified condition/decision coverage using exclusive-nor code transformer." Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), 2013 International Multi-Conference on. IEEE, 2013.

[2] Haque, Ariful, Irman Khalil, and Kamal Z. Zamli. "An Automated Tool for MC/DC Test Data Generation." 2014 IEEE Symp. Comput. Informatics, Kota Kinabalu, Sabah, Malaysia. 2014.

[3] CREST Tool : https://github.com/jburnim/crest

[4] Hayhurst, Kelly J., et al. "A practical tutorial on modified condition/decision coverage." (2001).

[5] Kannavara, Raghudeep, et al. "Challenges and opportunities with concolic testing." Proceedings of the 2015 IEEE National Aerospace Conference/Ohio Innovation Summit & IEEE Symposium on Monitoring & Surveillance Research (NAECON-OIS 2015). 2015.

[6] Paqué, Daniel. "From Symbolic Execution to Concolic Testing." Logikseminar Wintersemester 2014/15 - AG Meyer; TU Kaiserslautern

[7] De Moura, Leonardo, and Nikolaj Bjørner. "Z3: An efficient SMT solver." International conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer Berlin Heidelberg, 2008.

[8] Godboley, Sangharatna, et al. "An improved distributed concolic testing approach." Software: Practice and Experience (2016).