동적 심볼릭 수행을 이용한 경계 테스팅 연구

구근회*, 최석원**, 최진영***
*고려대학교 컴퓨터·전파통신공학과
**고려대학교 컴퓨터학과
***고려대학교 정보보호대학원
e-mail:khkoo@formal.korea.ac.kr

Boundary Testing research using Dynamic Symbolic Execution

Keun-Hoi Koo*, Seok-Won Choi**, Jin-Young Choi*
*Dept of Computer and Communications, Korea University
**Dept of Computer Science, Korea University
***Graduate School of Information Security, Korea University

요 약

전통적인 테스팅은 테스트케이스를 만드는데 많은 시간과 비용이 들기 때문에 시기적절한 출시를 해야 하는 휴대전화, TV와 같은 제품을 테스팅하는데 많은 어려움을 겪고 있다. 이러한 문제를 해결하려는 노력으로 관련 학계 및 산업체에서는 동적 심볼릭 수행을 이용한 커버리지가 높은 테스트케이스 자동생성 연구가 진행 중이다. 특히, Microsoft Research에서 만든 동적 심볼릭 수행도구인 PEX는 C#언어로 작성된 웹 또는 윈도우 프로그램의 테스트케이스를 자동 생성한다. 그러나 PEX의 사용자들로부터 테스트케이스가 부족하다는 피드백을 받았고, 그 결과로 경계 값 테스트케이스를 추가하는 연구를 진행하여 경계 값 테스트케이스를 추가하지 않았을 때 보다 더 많은 오류를 찾았다. 본 논문에서는 소프트웨어 오류가 있을 경우 엄청난 재산, 인명 피해가 발생하는 임베디드 소프트웨어 분야에서 자주 사용하는 언어인 C언어를 지원하는 동적 심볼릭 수행 도구 CREST를 수정하여 경계 값 테스트케이스를 생성하는 연구를 진행한다.

1. 서론

현대 사회에서 소프트웨어가 다양한 분야에서 많이 사용됨에 따라 소프트웨어 오류로 인한 피해도 점점 더 증가하고 있다. 특히, 국방, 항공, 자동차, 의료, 철도, 휴대전화, TV 등 임베디드 소프트웨어에 오류가 있을 경우 엄청난 금전, 인명 피해가 발생하는데 이러한 오류 중 상당 부분은 테스팅으로 찾을 수 있다.[1] 그러나 전통적인 테스팅은 테스트케이스를 만드는데 많은 시간과 비용이 들기 때문에 시기적절한 출시를 해야 하는 휴대전화, TV와 같은 제품을 테스팅하는데 많은 어려움을 겪고 있다. 이러한 문제를 해결하려는 노력으로 관련 학계 및 산업체에서는 동적 심볼릭 수행[3]을 이용한 커버리지가 높은 테스트케이스 자동생성 연구가 진행 중이다.[12] 특히, Microsoft Research에서 만든 동적 심볼릭 수행도구인 PEX[9]는 C#언어로 작성된 웹 또는 윈도우 프로그램의 테스트케이스를 자동 생성한다. 그러나 PEX의 사용자들로부터 테스트케이스가 부족하다는 피드백을 받았고, 그 결과로 경계 값 테스트케이스를 추가하는 연구를 진행하여 경계 값 테스트케이스를 추가하지 않았을 때 보다 더 많은 오류를 찾았다.[10~11] 본 논문에서는 소프트웨어 오류가 있을 경우 엄청난 재산, 인명 피해가 발생하는 임베디드 소프트웨어 분야에서 자주 사용하는 언어인 C언어를 지원하는 동적 심볼릭 수행 도구 CREST[8]를 수정하여 경계 값 테스트케이스를 생성하는 연구를 진행한다.

"본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었 음" (IITP-2016-H8501161012000100 2)

2. 관련 연구

2.1 동적 심볼릭 수행

동적 심볼릭 수행(Dynamic symbolic execution, 또는 Concolic testing)[6~9] 은 테스트케이스를 생성하는 문제를 제약 만족 문제의 형태로 표현하고 Z3[5]와 같은 SMT solver(Constraints solver)[4]로 풀어

낸다. 동적 심볼릭 수행은 먼저 테스트 대상을 구체적인 입력 값으로 프로그램을 실행하고 심볼릭 수행을 통해 현재 실행한 경로의 경로 제약 식을 만든다. 현재 실행한 경로 제약 식을 기반으로 아직 실행하지 않은 새로운 경로를 표현하는 경로 제약 식을 만들고 풀어냄으로써 새로운 실행 경로를 수행하는 테스트케이스를 생성한다.[13]

2.2 증가한(Augmented) 동적 심볼릭 수행

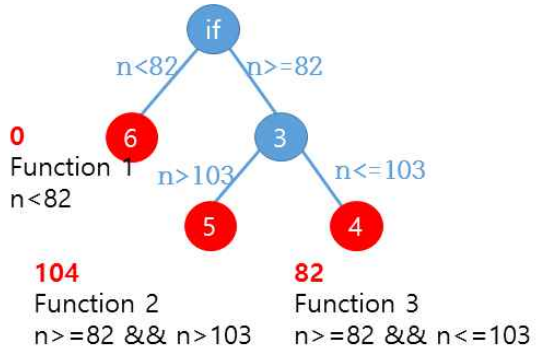
Microsoft Research에서 만든 동적 심볼릭 수행도구인 PEX는 C#언어로 작성된 웹 또는 윈도우 프로그램의 테스트케이스를 자동 생성한다. 그러나 PEX의 사용자들로부터 테스트케이스가 부족하다는 피드백을 받았다. 그 결과로 PEX에 경계 값에 해당하는 테스트케이스를 추가하는 연구를 진행하여 경계 값을 추가하지 않았을 때 보다 더 많은 소프트웨어 오류를 찾았다.

3. 실험 내용

경계 값에 해당하는 테스트케이스를 만들기 위해 C언어를 지원하고 오픈소스인 동적 심볼릭 수행 도구 CREST를 입력 도메인의 경계가 출력되도록 수정하여 사용한다.

```
void func(int n)
{
    if (n >= 82)
    {
        if (n <= 103)
            printf("Function 3\n");
        else
            printf("Function 2\n");
    }
    else
        printf("Function 1\n");
}
```

(그림 1) CREST로 테스트할 예제 프로그램



(그림2) 트리 구조로 나타낸 그림1의 예제 프로그램의 분기문

그림2의 3, 4, 5, 6은 프로그램의 Branch_id에 해당하며 4, 5, 6 단말 노드에서만 테스트케이스가 생성 가능하다.

3.1 수정된 CREST를 이용한 경계 출력

기존의 CREST는 입력 값으로 대상 프로그램을 실행한 결과인 그림3의 2)에 해당하는 "Function 1"만 출력하는데 수정된 CREST는 1)입력 값, 3)심볼릭 수행 경로, 4)심볼릭 경로 제약 식도 출력한다.

본 실험에서는 다음 입력 값을 구하기 위해서 제약 식을 풀 때 SMT solver로 Microsoft Research에서 만든 Z3를 사용한다.

```
[1. Initial values]
0

[2. Run Program]
Function 1

[3. Symbolic Execution Path : Branch_id]
[ 6 ]

[4. Symbolic Path Formula]
(<  (- x0 82 ) 0 )
```

(그림 3) 수정된 CREST 테스트 출력결과 (1)

그림3은 사용자가 지정한 0을 초기 값으로 해서 그림1의 예제 프로그램을 실행하여 "Function 1"을 출력한다. 그 다음으로 심볼릭 수행을 진행하여 그림2의 Branch_id의 6번 위치에 해당하는 심볼릭 경로 제약 식이 전위 표기법으로 출력한다. 이 식을 해석하면 n<82이다.

```
[1. Input Values]
82

[2. Run Program]
Function 3

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

[4. Symbolic Path Formula]
(>= (- x0 82 ) 0 )
(<= (- x0 103 ) 0 )
```

(그림 4) 수정된 CREST 테스트 출력결과 (2)

그림4는 이전 분기 경로인 n<82에서 not 연산을 해서 SMT solver로 풀면 82의 값을 얻게 되는데 다음 프로그램의 입력 값으로 사용한다. 이 값으로 프로그램을 수행하면 "Function 3"을 출력하고 심볼릭 수행을 진행하여 그림2의 Branch_id 3번을 지나

4번 위치에서 n>=82 && n<=103 심볼릭 경로 제약 식을 얻는다.

```
[1. Input Values]
104

[2. Run Program]
Function 2

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

[4. Symbolic Path Formula]
(>= (- x0 82 ) 0 )
(>  (- x0 103 ) 0 )
```

(그림 5) 수정된 CREST 테스트 출력결과 (3)

그림5는 이전 분기 경로인 n<=103을 not 연산을 해서 n>=82 && n>103를 얻고 이 제약 식을 SMT solver로 풀면 104의 값을 얻는다. 이 값을 입력으로 하여 프로그램을 수행하면 “Function 2”를 출력하고 심볼릭 수행을 진행하여 그림2의 branch_id 3번을 지나 5번 위치에서 n>=82 && n>103 심볼릭 경로 제약 식을 출력한다. 더는 다른 경로로 향하는 제약 식을 얻을 수 없으므로 프로그램은 종료한다.

3.2 출력된 경계를 통한 경계 테스팅

(그림 6) CREST의 동등 분할에 해당하는 테스트케이스 생성

기존의 CREST는 그림1의 예제 프로그램을 수행하면 그림6처럼 동등 분할에 해당하는 테스트케이스 0, 82, 104를 자동 생성한다.

(그림 7) CREST의 경계 값에 해당하는 테스트케이스 생성

수정된 CREST로 그림1의 예제 프로그램을 수행하면 그림3~5처럼 n<82, n>103, 82<=n<=103 의 경로 제약 식을 출력한다. 이 제약 식을 이용하여 입력 도메인의 정확한 경계 값 82, 103을, 바로 아래 경계 값 81, 102를, 그리고 바로 위 경계 값 83, 104에 해당하는 경계 값 테스트케이스를 수동으로 생성해서 경계 테스팅에 사용할 수 있다.

4. 결론

본 연구에서는 입력 도메인의 경계가 출력되도록 수정된 CREST를 이용하여 동등 분할 테스트케이스보다 오류를 더 많이 찾는 경계 값에 해당하는 테스트케이스를 생성[2]하였다. 본 실험에서는 출력된 경계를 보고 수동으로 경계 값에 해당하는 테스트케이스를 생성하였는데, 향후 연구는 자동으로 경계 값에 해당하는 테스트케이스를 생성하고 경계 값 테스트케이스를 추가하지 않았을 때보다 추가 했을 때 더 많은 오류를 찾는지 비교 연구하는 것이다.

참고문헌

[1] National Institute of Standards and Technology, “The Economic Impacts of Inadequate Infrastructure for Software Testing,” Planning Report 02-3 May 2002.

[2] Reid, Stuart C. "An empirical analysis of equivalence partitioning, boundary value analysis and random testing." Software Metrics Symposium, 1997. Proceedings., Fourth International. IEEE, 1997.

[3] King, James C. "Symbolic execution and program testing." Communications of the ACM 19.7 (1976): 385-394.

[4] Barrett, Clark, Aaron Stump, and Cesare Tinelli. "The satisfiability modulo theories library (SMT-LIB). www." SMT-LIB. org 15 (2010): 18-52.

[5] 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.

[6] Godefroid, Patrice, Nils Klarlund, and Koushik Sen. "DART: directed automated random testing." ACM Sigplan Notices. Vol. 40. No. 6. ACM, 2005.

[7] Sen, Koushik, Darko Marinov, and Gul Agha. "CUTE: a concolic unit testing engine for C." ACM SIGSOFT Software Engineering Notes. Vol. 30. No. 5. ACM, 2005.

[8] Burnim, Jacob, and Koushik Sen. "Heuristics for Dynamic Test Generation (short paper)." 23rd IEEE/ACM International Conference on Automated Software Engineering, ASE. 2008.

[9] Tillmann, Nikolai, and Jonathan De Halleux. "Pex - white box test generation for. net." International conference on tests and proofs. Springer Berlin Heidelberg, 2008.

[10] Jamrozik, Konrad, et al. "Augmented dynamic symbolic execution." Automated Software Engineering (ASE), 2012 Proceedings of the 27th IEEE/ACM International Conference on. IEEE, 2012.
[11] Jamrozik, Konrad, et al. "Generating test suites with augmented dynamic symbolic execution." International Conference on Tests and Proofs. Springer Berlin Heidelberg, 2013.
[12] 김윤호, et al. "동적 심볼릭 수행을 응용한 테스트 케이스 자동 생성 도구 비교." (2010).
[13] 김윤호, and 김문주. "동적 심볼릭 수행과 유전 알고리즘을 사용한 테스트 생성 기법 비교." (2011).