반복 횟수를 제한한 경로 탐색을 통한 Concolic 테스팅의 커버리지 달성 향상

Bounded Search Strategies for Concolic Testing to Achieve Higher Test Coverage Fast

요 약

본 논문은 Concolic 테스팅에서 한 실행 내에서 동일 분기의 반복 횟수가 특정 값을 넘지 않도록 제한하는 방향으로 경로 탐색을 전개하는 반복 횟수 제한 탐색 전략을 제안하고, 이를 실제 테스트 생성에 적용하여 분기 커버리지 달성을 평가한 결과를 소개한다. 제안하는 탐색전략은, 반복문 또는 재귀함수의 반복 횟수를 늘리는 방향으로의 탐색에 앞서, 반복 횟수가 제한된 상황 아래에서 새로운 실행 경로를 우선적으로 탐색함으로써, Concolic 테스팅이 제한된 시간 내에 다양한 실행 상황을 방문하도록 유도한다. 본 연구에서는 제안한 탐색 전략을 기존에 개발된 3종의 탐색 전략을 기반으로 적용하여 CREST 도구에 구현하였다. 이를 실제 C 프로그램인 Grep과 Vim을 대상으로 실험한 결과, 반복 횟수 제한 탐색기법이 기존의 탐색전략보다 대체로 높은 분기 커버리지를, 보다 짧은 시간 내에 달성함을 확인할 수 있었다.

1. 서론

Concolic 테스팅[1]은 (1) 검증대상 프로그램에 실제 입력 값을 주어 실행시킨 후 실행 상태를 관찰하여 해당 실행 경로의 조건을 프로그램 입력에 대한 기호식(symbolic path formula)으로 추출하고, (2) 추출한 기호식을 변형하여 새로운 경로에 대한 기호식을 도출한 후, (3) SMT 해법기를 이용해 새로운 경로 기호식을 만족하는 프로그램 입력 값을 구하는 과정을 반복함으로써 새로운 경로를 지속적으로 탐색하는 테스트 입력을 생성하는 기법이다[1]. Concolic 테스팅은 외부 라이브러리 호출, 비트 단위 연산 등 난해한 요소가 포함된 실행 경로에 대해서도 정교한 분석이 가능하여, 임베디드 소프트웨어와 같은 복잡한 실제 개발 산업 현장에서 소프트웨어 결함 자동 검출을 위해 이용되고 있다[2, 3, 4].

Concolic 테스팅의 탐색 전략(search strategy)이란 특정 시점에 추출한 실제 경로 조건식으로부터 도출 가능한 여러 개의 경로 기호식들 중 하나를 선택하는 방식(과정 2)을 뜻한다. 테스팅을 통해 효과적인 오류 검출을 위해서는 제한된 시간 내에 가능한 넓은 코드 영역에 도달하는 테스트 입력이 필요하므로[5], 제한된 시간 내에 높은 코드 커버리지 달성을 목표로 한 여러 Concolic 테스팅 탐색 전략이 휴리스틱 형태로 개발되어 왔다[6, 7]. 이러한 탐색 전략 휴리스틱은 특정한 가정을 만족하는 테스팅 상황(예: 검증 대상 프로그램의 구조적 특징, 심볼릭 테스트 케이스 구성, 테스팅 시간)에서 우수한 성능을 보이는 반면, 해당 휴리스틱의 가정에 맞지 않는 상황에 대해서는, 경로 탐색이 국지적인 실행 영역에 고착되어 Concolic 테스팅의 효용성에 제약이 있을 수 있다.

본 논문은 Concolic 테스팅의 탐색이 반복문 혹은 재귀함수로 인해 국지적 영역에 고착되지 않도록 전개하기 위하여 반복 횟수 제한을 갖는 Concolic 테스팅 탐색 전략의 사용을 제안한다. 반복 횟수 제한을 갖는 Concolic 테스팅 탐색 전략은 CFG[5]와 같은 기존의 Concolic 테스팅 탐색 전략을 기반으로 하되, Concolic 테스팅에서 한 실행 내에서 한 분기문이 반복되어 실행될 수 있는 최대 반복 횟수를 설정한 후, 기반 탐색 전략이 새로운 경로식을 도출하는 시점에서 각 분기문이 해당 한계치를 추가로 넘어서지 않는 범위 내에서 경로 탐색을 전개하도록 제한한다.

제안하는 탐색전략은 반복 제한적 모델 검증[8]이 착안한 바와 같이, 반복문 또는 재귀함수의 반복 횟수를 늘리는 방향으로의 탐색에 앞서, 반복 횟수를 늘리지 않는 실행 경로를 우선적으로 탐색함으로써, Concolic 테스팅이 제한된 시간 내에 다양한 실행 상황을 방문하도록 유도한다. 이때 검증대상 프로그램에 따라 효과적인 Concolic 테스팅이 가능한 최대 반복 횟수가 다를 수 있으므로, 제안하는 탐색 전략에서는 테스팅 진행에 따른 커버리지 달성을 관찰하여 점진적으로 최대 반복 횟수를 증가하도록 구상하였다.

본 논문에서는 제안한 탐색 전략을 기존에 개발된 3종의 탐색 전략에 복합적으로 적용하도록 구현한 방법과(2장), 실제 C 프로그램인 Grep과 Vim을 대상으로 생성한 테스트 입력 값의 분기 커버리지 달성을 측정한 결과를 소개한다(3장). 실험한 결과, 반복 횟수를 제한한 탐색기법이 기존의 탐색기법보다 각각 더 높은 분기 커버리지를, 보다 짧은 테스팅 시간 내에 달성함을 확인할 수 있었다.

2. 최대 반복 횟수를 제한하는 경로 탐색 전략

알고리즘1은 기본적인 Concolic 테스팅 알고리즘의 구조[6]를 바탕으로, 본 논문이 제안하는 Concolic 탐색 전략을 기술한 알고리즘이다. 기본적인 Concolic 테스팅 알고리즘을 탐색 전략 알고리즘[6]을 확장하여 표현한 것이다. 알고리즘1은 기존에 개발된 Random Negation, Uniform Random, 혹은 CFG 탐색 알고리즘을 기반 탐색 알고리즘(10행을 설명하며 상술)을 호출하여 신규 탐색 대상 경로를 선정한다.

Concolic 테스팅은 입력으로 검증대상 프로그램(P)과 초기 실행 결과 얻어진 실행 경로(p)를 입력 받는다. 이 때 실행 경로는 실행된 분기들의 순열$< p_0\, p_1, \ldots, p_n >$로 표현된다. 알고리즘1은 커버된 분기의 집합인 $covered$ 을 공집합으로 초기화하고(1행), 반복 횟수 제한을 위한 변수인 $bound$, $plateau, threshold$를 초기화한다(2행). $bound$는 특정 시점에서 제한하는 최대 반복 횟수를 뜻하며 초기값은 테스팅 상황에 따라 설정할 수 있다.

$plateau$ 는 추가로 커버되는 분기가 발생하지 않은 실행 횟수를 기록하는 변수이며, 동일한 $bound$ 조건에서 $threshold$는 $plateau$의 최대 허용 값을 나타낸다. $threshold$의 초기값은 테스팅 상황에 따라 정할 수 있다.

Input: target program P, initial execution path p

1 $covered \leftarrow \emptyset$
2 $bound \leftarrow b_{init}$; $plateau \leftarrow 0$; $threshold \leftarrow t_{init}$
3 **while** termination conditions are not met **do**
4 $covered \leftarrow covered \cup \{ b \mid b \in p \}$
5 $repeat \leftarrow \lambda b.0$
6 **for each** $b \in p$ **do**
7 $repeat[b] \leftarrow repeat[b] + 1$
8 **end for**
9 $candidates \leftarrow \{ b \in p \mid repeat[\bar{b}] \leq bound \}$
10 $i \leftarrow$ BaseSearchStrategy$(P, p, covered, candidates)$
11 **if** $\exists I$ that traverses P through $p_0, \ldots, p_{i-1}, \bar{p_i}$, **then**
12 $p \leftarrow \text{Run}(P, I)$;
13 **if** $\{b \mid b \in p\} \subseteq covered$ **then**
14 $plateau \leftarrow plateau + 1$
15 **if** $plateau = threshold$ **then**
16 $bound \leftarrow bound + 1$
17 $threshold \leftarrow 2 \times threshold$
18 **end if**
19 **else**
20 $plateau \leftarrow 0$
21 **end if**
22 **end if**
23 **end while**

알고리즘1. 반복 횟수를 제한하는 Concolic 탐색 전략

Concolic 테스팅은 테스팅 시간 혹은 테스팅 횟수 등 사용자가 지정한 종료 조건이 만족될 때까지 테스트 입력 생성을 반복한다(3행). 새로운 테스트 입력 생성은 직전 실행의 실행 경로 중 일부 초기 부분(prefix)를 만족한 후, 그 다음 분기를 반대로 택하는 경로를 택하는 방식으로 진행된다(4~22행). 하나의 새로운 테스트 입력을 생성하기 위해서, 알고리즘 1은 우선 직전 실행(혹은 초기 실행)의 경로 정보를 바탕으로 $covered$을 갱신한 후(4행), 각 분기별로 직전 실행 경로에서 몇 회 방문 되었는 지를 $repeat$이라는 맵핑에 기록한다(6~9행).

제안하는 탐색 알고리즘은 지정한 최대 횟수 이상으로 반복이 확장되는 방향으로 탐색이 진행되지 않게 통제한다. 이를 위하여, 직전 실행 경로에 포함된 각 분기(b)의 반대 분기($\bar{b}$)가 반복된 횟수가 지정한 최대치 이내인 경우만을 탐색 대상 분기 집합인 $candidates$ 에 포함시킨다(9행). 그리고 $candidates$ 에 포함된 분기문에 한정하여 신규 탐색 경로를 선택하도록 수정된 기반 탐색 알고리즘을 이용하여 신규 탐색 경로를 구한다(10절).

기반 탐색 알고리즘은 직전 실행 경로를 바탕으로 신규 탐색 대상 경로 $<p_0\, p_1, \ldots, p_{i-1}, \bar{p_i}>$를 지정한다. 이 때 i는 직전 실행 경로가 해당하는 순열의 특정 인덱스이며 $\bar{p_i}$ 는 i번째 도달한 분기의 반대 분기이다. 기반 탐색 알고리즘은 검증대상 프로그램의 구조와 같은 정적 정보와 해당 시점의 커버리지와 같은 동적 정보를 활용하는 휴리스틱에 따라 신규 탐색 대상 경로를 지정한다. 본 논문이 제안하는 알고리즘에서는 기반 탐색 알고리즘이 분기의 집합인 $candidates$ 를 추가로 입력 받은 후, $\bar{p_i}$이 $candidates$에 있는 경우에 한하여 신규 탐색 대상 경로를 선정한다. 이로 인해, $repeat(\bar{p_i})$ 이 $bound$ 를 초과하는 경우는 신규 탐색 대상에서 배제되는데, 이는 $<p_0\, p_1, \ldots, p_{i-1}, \bar{p_i}>$를 도달하는 신규 실행을 생성할 경우 $\bar{p_i}$의 반복 횟수가 지정한 최대치를 초과하는 방향이 되는 것이 분명하기 때문이다. 기반 알고리즘으로 사용되는 CFG는 각 분기별로 탐색 우선순위를 정한 후, 우선 순위가 높은 순서에 따라 신규 탐색 대상 경로를 선정하는데, 본 알고리즘에서는 직전 실행 경로 중 $candidates$에 포함된 분기에 한정하여 우선순위가 가장 높은 경우를 택하도록 하였다. Random Negation과 Uniform Random 방식의 경우, 직전 실행 경로 중 $candidates$ 에 속한 분기만을 임의적 선택에서의 선택 대상으로 활용하고, $candidates$ 에 해당하지 않는 경우를 배제하도록 확장하여 사용하였다.

제안한 알고리즘은 SMT해법기를 활용하여 선정된 신규 탐색 대상 경로의 경로조건식을 만족하는 프로그램 입력 값을 구한다(11행). 신규 탐색 대상 경로를 따르는 입력 값을 구한 경우, 이 입력 값으로 검증대상 프로그램을 실행한 후 실제 탐색 경로를 구한다(12행).

새롭게 실행한 경로 정보를 바탕으로 제안한 알고리즘은 $bound$ 값을 조정한다. 새롭게 실행한 경로에서 커버리지 증가가 없는 경우, $plateau$ 를 증가시킨다(13~14행). 이 때 $plateau$ 는 커버리지 증가가 없는 테스트 입력 생성의 연속 횟수를 나타낸다. 그리고 $plateau$가 $threshold$에 다다를 경우, $bound$ 를 증가시켜 검증대상 프로그램의 탐색 범위를 증가시키고(16행), 또한 $threshold$ 값을 2배로 늘리는 방식으로 그 이후 $bound$의 증가 조건을 조정한다.

3. 실험을 통한 평가

3.1 실험 설계

연구 질문. 제안한 Concolic 테스팅 탐색 전략으로 인한 효용성과 효과성 향상을 평가하기 위해서 다음의 두 가지 연구 질문에 대한 실험을 수행하였다:

- RQ1. 기반 Concolic 탐색 전략을 사용한 경우에 비하여, 최대 반복 횟수를 제한한 탐색 전략을 추가로 적용하였을 경우, 분기 커버리지의 달성이 얼마만큼 향상되는가?
- RQ2. 기반 Concolic 탐색 전략을 사용한 경우에 비하여, 최대 반복 횟수를 제한한 탐색 전략을 추가로 적용하였을 경우, 높은 분기 커버리지를 달성하는 데 소요되는 테스트 입력 값 생성 시간이 얼마만큼 단축되는가?

Concolic 테스팅 탐색 기법. 본 실험에서는 기반 탐색 알고리즘으로 기존 연구[6]에서 제안한 CFG, Random Negation, Uniform Random을 사용하였다. 그리고 각 기반 탐색 알고리즘에 본 논문에서 제안한 최대 반복 횟수 제한을 추가한 Bounded CFG, Bounded Random Negation, Bounded Uniform Random을 구현하여, 총 6개의 서로 다른 기법 간의 결과를 비교하였다. 본 실험은 Concolic 테스팅 도구인 CREST[9]를 SMT해법기로 Z3 4.3.5[9]를 사용하도록 수정한 도구를 이용하였다.

테스트 대상 프로그램. 본 실험에서는 기존 연구[6]의 실험에서 사용한 Grep 2,2, Vim 5.7을 테스트 대상으로 사용하였다[1]. Grep과 Vim 모두, 사용자 명령 및 파일 내용에 해당하는 입력을 심볼릭 입력 변수로 정하는 심볼릭 드라이버를 각각 정의하여 사용했다. 테스트 생성 시, Grep의 경우 50개, Vim은 10개의 심볼릭 입력 변수가 각각 사용되었다.

[1] [6]의 Replace도 실험했으나 유의미한 성능차이가 없어, 지면 관계상 생략했다.

Search Strategies		Grep	Vim
CFG	Bounded	**2292.0**	**8224.1**
	Base	2193.6	8127.4
Random Negation	Bounded	2035.6	**8306.5**
	Base	**2041.4**	8241.3
Uniform Random	Bounded	**1643.8**	**5368.4**
	Base	1637.0	4816.4

표1. 분기 커버리지 달성

Search Strategies		Grep	Vim
CFG	Bounded	**379.0**	**970.0**
	Base	845.4	1134.6
Random Negation	Bounded	140.5	**695.7**
	Base	**131.5**	894.1
Uniform Random	Bounded	**526.4**	**918.5**
	Base	658.9	N/A

표2. 최대 분기 커버리지의 95% 달성까지의 시간(초)

실험 수행 및 측정. 반복 횟수 제한 탐색전략의 사용에 따른 커버리지 달성 효과성(RQ1)과 효율성(RQ2)를 평가하기 위하여, 총 6개의 탐색 전략을 각각 Concolic 테스팅에 적용하여 Grep과 Vim에 대해 각각 1200초, 1800초 간 테스트 입력 생성을 수행했다. 테스팅 시간은 사전조사를 통해 시간 연장이 결과에 유의미한 차이를 보이지 않는 지점을 파악해 정했다. bound과 threshold 초기값(b_{init}, t_{init})은 각각 1, 5로 하였다.

탐색 전략에 따른 커버리지 달성 효용성(RQ1)을 평가하기 위해, 최종적으로 생성된 테스트 입력 값이 커버한 분기의 개수를 측정하였다. 또한, 효율성(질문2)을 비교하기 위해, 각 기반 테스팅 별로 최종적으로 커버한 최대 분기 개수의 95%를 달성하기까지 소요된 시간의 평균을 측정했다. 예를 들어, Grep에 대한 Bounded CFG와 CFG 간의 효율성 비교를 위해, 둘 중 최종적으로 더 많은 분기를 커버한 Bounded CFG의 결과인 2292.0의 95% 지점, 즉 2175.5를 달성되는 데 Bounded CFG와 CFG가 각각 소요한 테스팅 시간을 구했다. 본 실험에서 사용한 기반 탐색 전략에는 난수 발생 요소가 있으므로 동일한 실험을 10회 반복한 후, 평균 결과를 구했다.

3.2 실험 결과

커버리지 달성 효과성(RQ1). 표1은 총 6개의 테스팅 기법이 최종적으로 커버한 분기 개수를 나타낸다. 이 때 Bounded는 본 논문이 제안한 반복 횟수 제한 탐색 전략을 적용한 경우(2, 4,6행)이며, Base는 적용하지 않은 경우(3,5,7행)를 각각 나타낸다. 표1의 두 번째 열은 Grep, 세번째 열은 Vim 결과에 해당한다.

표1의 결과에 따르면, 대체로 반복 횟수 제한 탐색 전략을 적용한 경우(Bounded)가 그렇지 않은 경우(Base)보다 더 많은 분기 커버리지를 달성하였다. Vim의 경우, 모든 경우에서 반복 횟수 제한 탐색 기법이 기반 탐색 기법보다 최대 11.5% 높은 커버리지를 달성했다(Bounded Uniform Random과 Uniform Random 비교). Grep의 경우, Bounded CFG와 Bounded Uniform Random의 경우, 기반 탐색 전략 대비 4.5%, 0.4%가 더 많은 분기를 커버하였다. 반면, Bounded Random Negation은 Random Negation보다 -0.3% 낮은 분기를 보였다.

커버리지 달성 효율성(RQ2). 표2는 최대 분기 커버리지의 95%를 달성하기까지 걸린 테스트 생성 시간이다. 표2의 결과에 따르면, 대체로 반복 횟수 제한 탐색 전략을 적용한 경우(Bounded)가 그렇지 않은 경우(Base)보다 더 짧은 시간 내에 95% 지점을 달성하였다. Vim을 대상으로 한 실험에서는 모든 경우에서, 반복 횟수 제한을 적용한 탐색 전략(Bounded)가 더 적은 테스팅 시간에 소요되었다. Vim에 대한 Uniform Random의 경우, 모든 실행에서 최대 Coverage의 95%를 달성하지 못했다(N/A). Grep의 경우, Bounded CFG와 Bounded Uniform Random에 대해서는 더 짧은 시간이

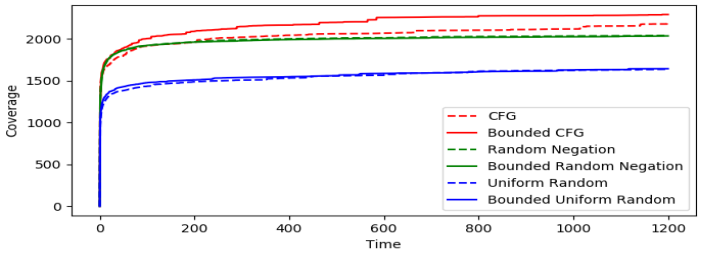

그림1. Grep의 시간-분기 커버리지 그래프

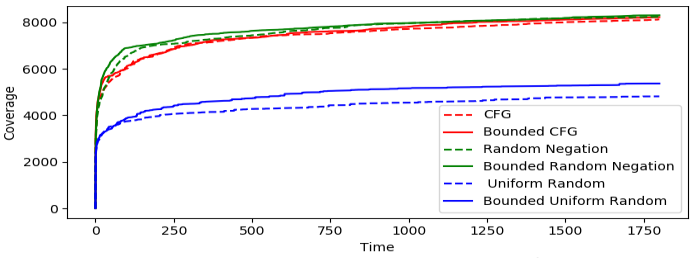

그림2. Vim의 시간-분기 커버리지 그래프

소요되었다. 반면, Bounded Random Negation은 Negation 보다 평균 19초가 더 소요되었다.

그림 1과 그림 2는 시간에 따른 Grep과 Vim의 분기 커버리지를 나타낸 그래프이다. 최대 분기 반복 횟수를 제한한 탐색 전략(Bounded)의 결과는 실선으로, 비교 대상인 기반 탐색 기법(Base)은 점선으로 표현하였다. 대체로 반복 횟수를 제한한 탐색 기법이 이에 대응하는 기반 탐색 비법보다 더 높은 커버리지를 빨리 달성함을 볼 수 있다. 테스팅 과정 중 bound는 초기값 1에서 출발하여 Vim의 경우 종료 시점에 9~15, Grep은 12~16까지 점차적으로 증가함을 관찰할 수 있었다.

4. 결론 및 향후 연구

본 논문에서는 Concolic 테스팅에서 한 실행 내에서 동일 분기의 반복 횟수가 특정 값을 넘지 않도록 제한하는 방향으로 경로 탐색을 전개하는 반복 횟수 제한 탐색전략을 제안하였다. 기존의 CFG, Random Negation, Uniform Random 탐색전략에 제안한 탐색 기법을 적용한 결과, 분기 커버리지 달성의 효과성과 효율성이 향상되는 것을 Vim과 Grep을 통해 실험적으로 확인할 수 있었다. 향후에는 각 반복문에 따라 적절한 반복 횟수를 개별적으로 제한하는 방법을 연구하는 한편, 보다 다양한 기반 탐색 알고리즘(예: DFS, CGS[6])에 대해 제안한 반복 제한 방식을 적용한 결과를 실험적으로 평가하는 연구를 수행할 계획이다.

참조문헌

[1] K. Sen, D. Marinov, G. Agha, CUTE: a concolic unit testing engine for C, ESEC/FSE, 2005

[2] Y. Kim, Y. Kim, T. Kim, G. Lee, Y. Jang, M. Kim, Automated Unit Testing of Large Industrial Embedded Software Using Concolic Testing, ASE, 2013

[3] Y. Park, S. Hong, M. Kim, D. Lee, J. Cho, Systematic Testing of Reactive Software with Non-deterministic Events: A Case Study on LG Electric Oven, ICSE, 2015

[4] Y. Kim, D. Lee, J. Baek, M. Kim, Concolic Testing for High Test Coverage and Reduced Human Effort in Automotive Industry, ICSE, 2019

[5] James H. Andrews, Lionel C. Briand, Yvan Labiche, Akbar Siami Namin, Using Mutation Analysis for Assessing and Comparing Testing Coverage Criteria, TSE, 2006.

[6] J. Burnim, K. Sen, Heuristics for Scalable Dynamic Test Generation, Technical Report, 2008.

[7] H. Seo, S. Kim, How We Get There: A Context-Guided Search Strategy in Concolic Testing, FSE 2014.

[8] A. Biere, A. Cimatti, E. Clarke, Y. Zhu, Symbolic Model Checking without BDDs, TACAS, 1999.

[9] CREST, https://github.com/jburnim/crest

[10] Z3, https://github.com/Z3Prover/z3