스텁 자동 생성을 위한 입출력 예제 기반 점증적 제약 조건 추출 기법

김요엘, 최윤자
경북대학교 컴퓨터학부
kimyoel2305@gmail.com, yuchoi76@knu.ac.kr

An Approach of Incremental Constraint Extraction Based on I/O Examples for Automatic Stub Generation

Yoel Kim, Yunja Choi
School of Computer Science and Engineering, Kyungpook National University

요 약

본 논문은 PBE (Programming-By-Example) 기반 프로그램 합성 기법을 이용하여 다양한 입력에 대해 처리 가능하고 비교적 정확한 행동을 수행할 수 있는 스텁을 자동 생성하는 방안을 제안한다. 제안하는 방법은 매 합성된 스텁이 커버하지 못하는 출력값을 가진 입출력 예제를 점증적으로 추출하여 합성하는 방식으로, 실험에서 출력 범위를 최대한으로 높이면서도 비교적 높은 정확도를 가지는 스텁을 효율적으로 자동 생성하는 것이 가능함을 보인다.

1. 서 론

검증 대상 소프트웨어를 엄밀하게 검증하기 위해선 관련 외부 라이브러리와 상호작용할 수 있도록 하는 검증 환경 구성이 필수적이다. 그러나, 관련 외부 컴포넌트를 실제로 가동하면서 검증하기에는 검증 비용이 증가하는 문제가 발생한다. 검증 비용을 줄이기 위해 외부 라이브러리를 단순하게 시뮬레이션하는 스텁을 작성하여 대체하는 방법이 일반적이다 [1]. 그러나, 스텁의 동작 코드를 직접 작성해야 하거나, 스텁 자동 생성 프레임워크 [2]을 사용하더라도 몇몇 입력 케이스에만 처리할 수 있어 외부 라이브러리의 다양한 행동을 반영할 수 없는 문제가 존재한다.

본 논문은 C 프로그램을 대상으로 소스 코드가 공개되지 않은 함수를 PBE (Programming-By-Example) 기반 프로그램 합성 기법 [3]을 이용하여, 다양한 입력에 대해 비교적 정확하고 다양한 행동을 수행할 수 있는 스텁을 자동 생성하는 것을 목표로 한다. 따라서, 제약 조건을 추출할 때 출력 범위를 고려하여 넓은 출력 범위를 가지도록 하고, 적은 제약 조건 수부터 점증적으로 합성하여 합성의 난이도가 급격하게 증가하는 것을 방지하며, 그러면서도 스텁의 정확도를 증가시키는, 입출력 예제 기반 점증적 제약 조건 추출 기법을 제안한다.

2 점증적 제약 조건 추출 기법

그림 1에서 점증적 제약 조건 추출 프로세스를 대략적으로 보여주고 있다. 함수 f의 입출력 예제 집합 X가 입력으로 주어졌을 때, 가장 높은 출력 커버리지와 정확도를 가진 스텁 함수 f_{stub}을 반환한다.

제안하는 프로세스는 크게 제약 조건 추출 – PBE – 스텁 평가로 구성되며, 먼저 제약 조건 추출 단계에서 선택된 제약 조건 집합 C를 PBE 합성 도구에 전달하여 스텁 함수 f_{stub}을 반환한다. 합성된 스텁 f_{stub}은 스텁 평가 단계에서 주어진 입출력 예제 집합 X의 각 입력값을 대상으로 실행되어 정확도 (Acc.)와 출력 커버리지 (Cov.)를 계산한다. 정확도를 계산하는 과정에서 f_{stub}의 틀린 입출력 예제 집합 W를, 출력 커버리지를 계산하는 과정에서 f_{stub}이 커버하지 못하는 출력값을 가진 입출력 예제 집합인 O_{not}을 피드백 정보로써 제약 조건 추출 단계에 전달한다. 전달된 피드백 정보는 새로운 제약 조건을 추가할 때 사용되며, W와 O_{not}을 모두 만족하는 예제 1개를 선택하여 C에 추가하고 다시 합성한다.

이때, 다음과 같은 상황에 따라 C에서 제약 조건 하나를 제외하고 재합성을 시도할 수 있다: (1) PBE 합성 도구 실행의 제한 시간을 초과한 경우, 합성 난이도를 낮추기 위해 제외한다. (2) O_{not} = ∅인 상황에서 정확도가 이전보다 감소한 경우, 최적의 제약 조건 집합을 찾아내기 위해 제외한다.

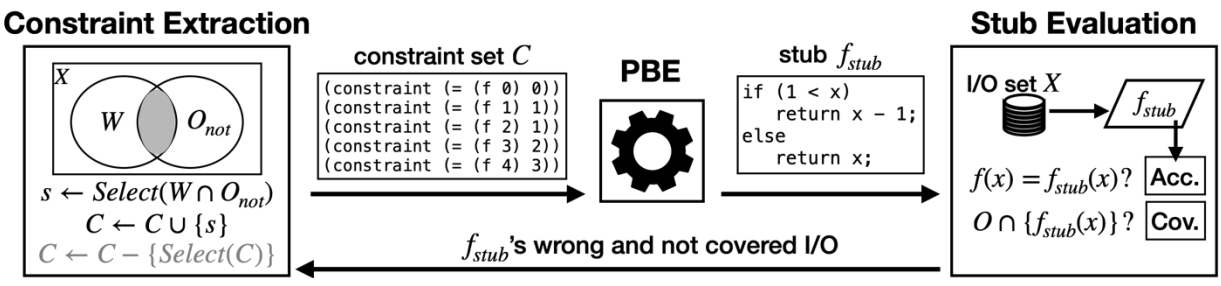

그림 1 점증적 제약 조건 추출 프로세스

표 1 각 합성 대상 함수 별 점증적 제약 조건 추출 방식을 적용한 스텁 생성 결과

Function	|P|	|B|	Ex. BA (%)	|O|	Origin Size	Synthesized Stub			
						Acc (%)	Cov (%)	Size	|C|
getWidth	42	0	37.3	251	3	78.19	98.8	66	90
getHeight	42	0	42.66	247	3	81.86	97.17	48	78
getEamilFrom	3	4	64.49	199	8	100	100	6	4
isEncrypted	3	4	64.51	113	8	100	100	6	4
isReadable_role_Encrypt	3	6	67.36	2	18	99.76	100	10	9
isReadable	4	8	67.66	2	27	99.3	100	16	10
isButtonPressed	12	6	69.88	2	1,683	89.87	100	14	14
getbiggestrect	43	6	34.99	9	405	48.85	100	18	33
floorIsOrdered	14	32	74.79	2	2,587	86.22	100	6	10
fisqrt	1	4	6.04	23	67	16.64	47.83	22	11
fibonacci	1	4	93.82	18	225	100	100	32	19
check	8	10	92.23	2	11	95.38	100	6	12
is_master_triggered	2	4	99.98	2	5	100	100	6	6
is_transmit1_triggered	2	4	99.98	2	5	100	100	6	5
getArea	42	0	72.4	700	11	71.32	25.43	52	36
findPublicKey	14	18	70.19	227	29	44.96	87.22	88	73
gcd_test	2	6	40.2	102	14	8.58	99.02	24	13

3. 실험

본 논문에서 제안한 제약 조건 추출 기법을 평가하기 위해, (1) 근처의 표적 물체를 따라 이동하는 자동차 로봇인 Object Follower [4], (2) 엘리베이터 컨트롤러의 시뮬레이터 [5], (3) SV-Comp 2022 [6]의 벤치마크 일부를 선정하였다. 본 실험을 위한 PBE 합성 도구는 DUET [3]을 사용하였고, 매 합성 도구 실행의 제한 시간은 300초, 전체 프로세스의 제한 시간은 1시간으로 설정하였다. 스텁 합성 및 정확도 평가를 위해 각 함수에서 입출력 예제를 최대 20,000개까지 수집하였다. 제안된 합성 방식이 정확도 측면에서 수용 가능한 수준인지 판단하기 위해, 그 기준을 기대 균형 정확도보다 합성된 스텁이 더 높은 정확도를 가진 경우로 정의하였다. 테스트 집합 X에 대해 각 출력값 o의 분포 비율을 X(o)라 정의할 때, 기대 균형 정확도는 $\Sigma(X(o))^2$로 정의할 수 있다.

3.1 실험 결과

표 1은 실험 예제에서 정의된 각 합성 대상 함수들에 대해 본 논문에서 제안한 점증적 합성 방식을 적용하여 재사용 가능한 스텁을 자동 생성했을 때의 결과를 보여준다. 맨 왼쪽 열부터 각 함수의 이름, 파라미터 수, 코드 내의 분기 수, 수집된 입출력 예제 집합의 기대 균형 정확도, 예제 집합의 서로 다른 출력값의 수 (출력 범위), 각 함수의 코드 사이즈 (조건, 할당, 반환 문장 개수의 합)을 나타내고, 그다음 열부터는 합성된 스텁의 정확도, 출력 커버리지, 코드 사이즈, 이때 사용된 제약 조건 (입출력 예제) 수를 나타낸다.

각 행은 합성된 스텁의 정확도 (Acc) 대비 기대 균형 정확도 (Ex. BA)와의 차이가 큰 순서로 정렬되었으며, getArea 함수부터 합성된 스텁의 정확도가 더 낮은 경우이다. 전체적으로, 합성 대상 함수 17개 중 14개에서 기대 균형 정확도보다 더 높은 정확도를 가지는 스텁을 합성하여 제안된 점증적 합성 프로세스가 정확도 측면에서 수용 가능하다고 평가할 수 있다. 또한, 정확도가 높은 14개 함수 중 fisqrt 함수를 제외하면 대체적으로 높은 출력 커버리지를 가지고 있어 본 연구에서 제안한 제약 조건 추출 기법을 통해 효율적으로 스텁을 자동 생성할 수 있다고 평가할 수 있다.

4. 결론 및 향후 연구

본 논문에서는 PBE 기반 프로그램 합성 기법을 이용해 소스 코드가 공개되지 않은 외부 라이브러리를 합리적으로 대체할 수 있는 스텁을 자동 생성할 수 있도록 점증적으로 제약 조건을 추출하는 방식을 제안하였다. 실험 결과 대체적으로 기대 균형 정확도보다 더 높은 정확도를 보였고, 또한 높은 출력 커버리지를 보여 제안된 방식의 효율성을 입증할 수 있었다. 향후 연구로는, 소스 코드 정보를 사용하는 스텁 생성 방식을 마련하고 이를 본 연구의 방식과 비교할 계획이다.

Acknowledgement

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2021R1A5A1021944).

참고 문헌

[1]. R. Osherove, “The art of unit testing: with examples in c,” page 50, 2013.

[2]. N. Tillmann and W. Schulte, “Mock-object generation with behavior,” in *21st IEEE/ACM International Conference on Aotomated Software Engineering*, pp.365-368, 2006.

[3]. W. Lee, "Combining the top-down propagation and bottom-up enumeration for inductive program synthesis," *Proceedings of the ACM on Programming Languages,* 5(POPL), pp.1-28 2021.

[4]. “Object follower,” https://github.com/addud/object-follower/

[5]. “Elevator controller,” https://github.com/Sandbergo/elevator-controller/

[6]. “SV-Comp 2022,” https://sv-comp.sosy-lab.org/2022/