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

김요엘, 최윤자

kimyoel2305@gmail.com, yuchoi76@knu.ac.kr

연구 개요

외부 라이브러리 모듈과 상호작용하는 소프트웨어의 검증/테스팅:

- 외부 모듈을 직접 가동하면서 테스트 → 검증 비용의 증가
- 외부 모듈을 단순하게 시뮬레이션 하는 스텝 (Stub)

스텝 자동 생성 프레임워크의 현주소:

- 뼈대만 제공할 뿐, 동작 코드는 사용자가 직접 작성
- Automock*: 몇몇 입력 케이스만 처리 가능한 스텝

목표: PBE (Programming-By-Example)을 통해, 정확하고 다양한 행동을 가진 스텝 코드 자동 생성

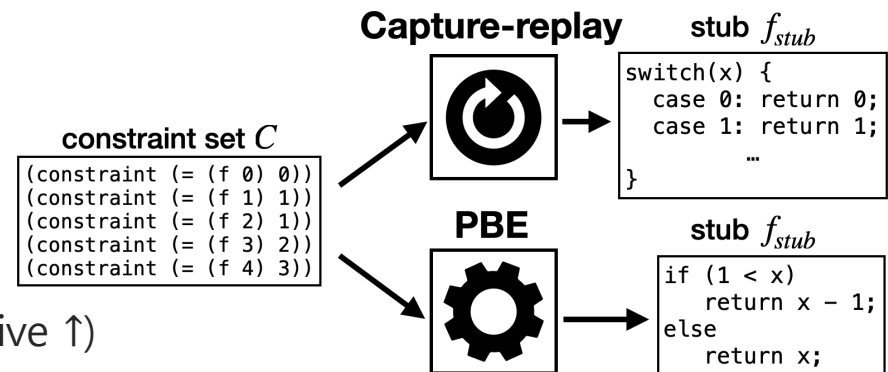
제약 조건 (입출력 예제)의 점증적 추출: 이전 스텝의 정확도와 출력 범위를 다음 스텝의 피드백으로

*Automock: [Automated synthesis of a mock environment for test case generation](#)

관련 연구

스텝 자동 생성:

- Capture-replay: 제한된 입력만 처리 가능 (false negative ↑)
- MOKA*: 최초로 PBE 방식 제안, 그러나 아이디어 수준



PBE, I/O기반 프로그램 합성:

- 입출력 예제를 제약 조건 (C)으로 받아 이를 만족하는 프로그램을 찾는 기법
- 외부 함수의 소스 코드가 비공개라도 바이너리만 존재하면 입출력 예제 수집 가능

SyGuS (Syntax-Guided Synthesis):

- 사용 가능한 연산자 및 상수를 명시하여 더 효율적으로 프로그램을 찾아낼 수 있는 기법
- 단, 외부 함수의 소스 코드가 비공개이므로 본 연구에서는 고정된 문법 제약 사용

[*A Framework for Automated Test Mocking of Mobile Apps](#)

제약 조건 추출의 필요성

랜덤하게 제약 조건을 추가하는 경우:

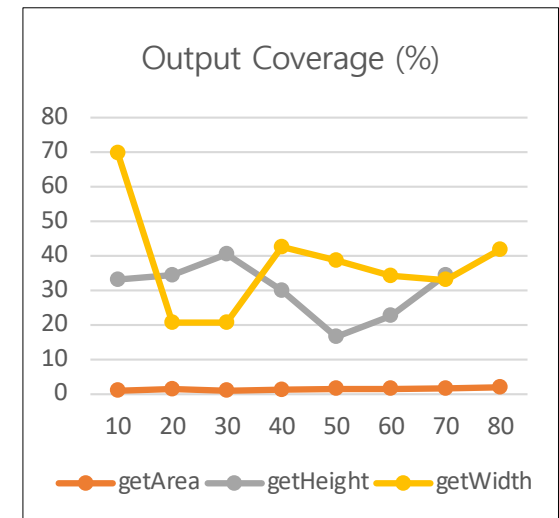
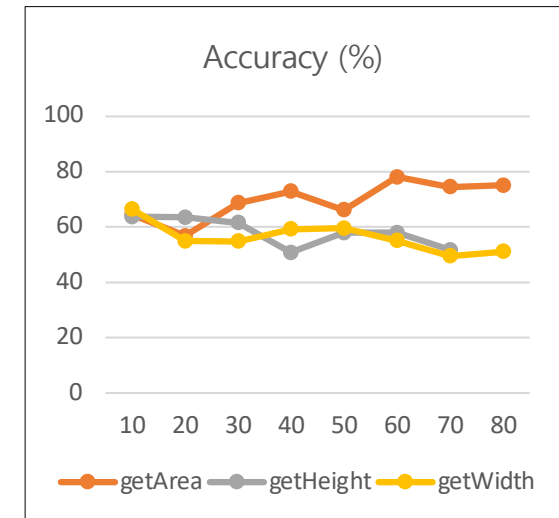
1. 제한 시간 내에 합성되지 않거나,
2. 합성된 스텝이 편향된 출력 범위를 가질 수 있음

실험:

- 제약 조건을 10개씩 랜덤하게 추가해가며 합성
- 각 합성마다 300초의 시간 제한 (그 외 조건은 후술할 실험과 동일)

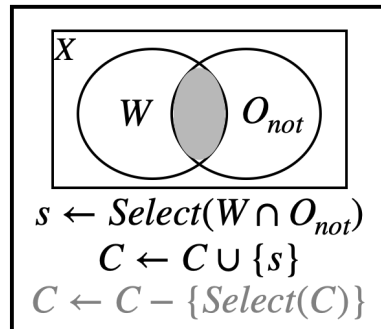
결과:

- 제약 조건 70~80개부터 timeout 발생
- getArea 함수의 경우 스텝의 출력 커버리지가 1~2%



점증적 제약 조건 추출 기법

Constraint Extraction

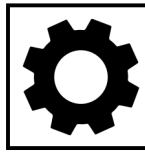


constraint set C

```

(constraint (= (f 0) 0))
(constraint (= (f 1) 1))
(constraint (= (f 2) 1))
(constraint (= (f 3) 2))
(constraint (= (f 4) 3))
    
```

PBE

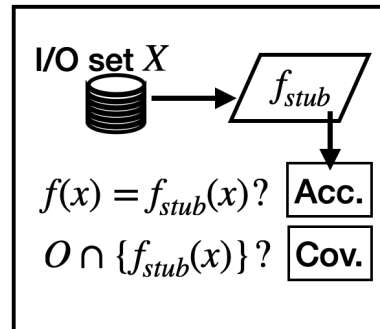


stub f_{stub}

```

if (1 < x)
  return x - 1;
else
  return x;
    
```

Stub Evaluation



f_{stub} 's wrong and not covered I/O

Constraint Extraction:

- W : 정확도를 계산하는 과정에서, f_{stub} 이 틀린 (잘못된 출력값을 반환하는) 입출력 예제 집합
- O_{not} : 출력 커버리지를 계산하는 과정에서, f_{stub} 이 커버하지 못하는 출력값을 가지는 예제 집합
- $O_{not} = \emptyset$ 이면서 정확도가 이전보다 낮아진 경우: local maximum을 방지하기 위해 임의로 제외

Stub Evaluation:

- 정확도 (Acc.): f_{stub} 이 원본 함수 f 대비 같은 출력값을 반환하는 정도
- 출력 커버리지 (Cov.): X 의 출력값 집합인 O 를 f_{stub} 이 커버하는 정도 (최우선사항)

실험 방법

- **실험 대상:** Object follower, Elevator controller, SV-Comp 2022의 여러 벤치마크에 정의되어 있는 함수 17개
- **PBE 도구:** SyGuS기반 DUET* 사용 (매 도구 실행 제한 시간 300초, 전체 합성 프로세스 1시간)
- **입출력 예제 수집:** Concolic/random execution을 통해 함수 당 최대 20,000개 수집 (합성 및 정확도 평가를 위해)
- **RQ:** 제안된 제약 조건 추출 및 합성 방식이 정확도 측면에서 수용가능한가?
→ 입출력 예제 집합의 기대 균형 정확도 (expected balanced accuracy)를 기준으로
 - ex. 예제 100개 중 90개는 true를, 10개는 false를 반환: $(0.9 \times 0.9) + (0.1 \times 0.1) = 81\%$

[*Combining Top-Down Propagation and Bottom-Up Enumeration for Inductive Program Synthesis](#)

실험 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

결과 분석

- **fisqrt**: 소스 코드가 비공개된 상태로 가정했기 때문에, 필요한 상수를 SyGuS 문법 스펙에 추가하지 못함. 해당 함수는 약 650개의 상수를 사용
- **getArea**: DUET이 지원하지 못하는 자료구조 (array 및 index 접근 연산)를 사용하기 때문
- **findPublicKey**: 비교적 분기 수 (18개)가 많으면서 동시에 출력 범위 (227개) 역시 넓음. 제안된 합성 방식은 최대한 많은 출력값을 커버하도록 설계되어 단점으로 작용
- **gcd_test**: 출력 커버리지가 85%에서 94%로 증가할 때 정확도가 44%에서 8%로 급격하게 감소하는 경우를 파악. 출력 커버리지를 최우선사항으로 설정했기 때문 (trade-off)

결론 및 향후 연구

결론:

- 소스 코드가 공개되지 않은 외부 라이브러리를 합리적으로 대체할 수 있는 스텝을 자동 생성할 수 있도록 점증적으로 제약 조건을 추출하는 방식을 제안
- 실험 대상 17개 함수 중 13개에서 높은 출력 커버리지 및 정확도를 달성하여 제안된 방식의 효율성을 입증하였음

향후 연구:

- 부족한 소스 코드 정보 (필요한 상수 및 분기 정보)로 인해 성능이 낮았던 케이스를 극복할 수 있는 방안을 연구할 계획