1. Introduction.

소프트웨어의 크기와 복잡도가 증가함에 따라 automatic program repair 연구의 필요성이 커졌고 실제 많은 연구들이 이행되고 있다. Apr의 시간적인 그리고 비용적인 비용을 줄이는 것이며 개발자의 개입을 최소화 하고 작업의 전반적인 자동화를 통해 이루려 한다. 여러 기반 기술들이 제안되고 적용 되고 있으며, 그 중 다수가 search-based 기술로서 generate-and validate라고도 불린다.

이 기술들은 소스코드의 의심되는 위치를 먼저 식별한 이후 미리 정의되어 있던 mutation operators를 적용시키며 패치 후보군을 찾고 이 패치 후보군이 주어진 테스트 케이스를 모두 통과하는지 여부를 통해 최종 패치를 찾는다. 실제 Search-based APR 기술들은 다양한 종류와 많은 수의 버그를 고치는 데에 효과를 나타내고 있다.

하지만 이러한 효과에도 불구 하고 큰 한계점들이 있다. 첫 번째는 옳은 패치가 항상 서치스페이스 안에 없다는 것 두 번째는 옳은 패치를 찾기 위해 무작위로 프로그램을 변이 시키는 것으로 인한 서치 스페이스를 크게 늘려야 한다는 것이다.

이러한 문제를 해결하기 위해 새로운 APR기반의 기술들의 필요성이 생겼고 한 가지의 방법으로 수정 패턴을 기반으로 하는 concrete patches를 생성하는 기술이다. 이 접근법은 개발자에 의해 작성된 패치의 정보를 활용하는 것인데 오픈소스 프로젝트에서 수집된 패치들로부터 나타나는 공통적인 수정 패턴을 추출하여 이 고정된 패턴을 버그 수정에 적용시키는 것이다. PAR에서 수정 패턴의 실용성을 보여주었고 ( 수정 패턴을 기반으로 하는 기술) 들은 다양한 수정 패턴과 방식으로 apr의 효율을 높이고 있다.

또 다른 한가지는 기계학습을 통한 버그 수정이다. 신경망 기계 번역은 가장 인기 있는 딥러닝 접근법이며 신경망 구조를 활용해 주어진 입력 시퀀스를 특정 시퀀스로 생성하는 접근법이다. NMT는 인코더를 통해 트레이닝 데이터로부터 얻어지는 입력값을 표현하고 디코더를 통해 목표로 하는 시퀀스를 표현하고자 한다. 이 NMT는 언어 번역 분야에서 주로 사용되고 있으나 최근 APR 분야에서 적용되어 좋은 효율을 보여주고 있다.

이러한 다른 방식의 다양한 연구들이 제안되고 있지만, 각각의 연구들에 대한 실증적인 분석이 부족하다. 적용된 기반의 기술들이 실제 패치에 대해 미치는 영향 또는 어떤 요인을 적용시켰을 때 발전되는 양상에 대한 실제 예시에 관한 실증적인 분석이 필요하다. 다른 방식의 5개의 연구에 대한 비교를 통해 적용된 방식이 미치는 영향과 또 성능 향상을 위해 적용되는 추가적인 기술들에 대한 구체적인 예시 분석을 통해 해당 기술들이 기여한 바와 미래에 발전될 방향에 대해 다루려 한다.

5개의 기술 Jgenprog,Cardumen,TBar,SequenceR,CoConut의 실제 성능 측정 및 분석을 위해 APR기술에서 가장 범용적으로 사용되고 있는 Defects4j benchmark의 최신 버전을 동일하게 활용 하였으며 각각의 기술들에 대한 fix process에 대한 공평한 결과 비교 및 fault localization에 대한 잡음을 없애기 위해 완벽한 폴트 위치 식별을 제공한다.

요약하여 해당 논문의 기여는 다음과 같다.

1. 기존의 APR기술과 최근의 효율 좋은 기술의 적용에 대한 영향력에 대한 실증적인 분석
2. 해당 기술들의 환경 조건들을 동일하게 하여 공평한 퍼포먼스 비교
3. 특정 상황의 버그타입에 유리한 기술이 있는지에 대한 분석
4. Background

Generate-and-validate 기술은 다음의 단계로 구성되어 있다. 버그 위치 식별, 후보 패치 생성, 패치 검증. 버그 위치 식별 단계에서는 폴트가 발생할 수 있는 의심가는 스테이트 먼트의 순위를 생성하기 위해 passed and failed test case의 커버리지 정보를 활용한다. 버그가 있는 것으로 의심되는 statement를 식별한 이후 각각의 techniques들은 설계한 모델에 맞추어 후보 패치를 생성하게 된다. 생성된 후보 패치는 최종적으로 패치 검증 단계에서 테스트 수트에 있는 테스트 케이스들을 모두 통과하는지 여부를 판단하여 해당 후보 패치를 검증한다. 이 모든 과정을 통과 하였다 하더라도 패치가 test case에 overfitting 되었을 수도 있기 때문에 개발자가 최종적으로 패치가 옳게 되었는지 검증하는 단계를 거친다. 이 과정에서 Test suite의 TC들을 모두 통과한 패치를 plausible patch, 최종적으로 사람에게 옳다고 확인된 패치는 Correct patch 라고 한다.

Template-based techniques (pattern-based techniques)

기존의 search-based techniques은 ingredient pool에 올바른 패치가 존재 하지 않는 문제와 옳은 패치를 하기 위해 무작위로 서치하게 되어 서치 스페이스가 방대하게 커지는 문제를 가지고 있었다. 이를 해결하기 위해 template-based 기술이 제안되어 연구되고 있다.

이 approach는 human-written patches의 지식을 활용하는 것인데, 먼저 오픈 소스프로젝트의