



FSE 2022

싱가포르

이준희
소프트웨어 분석 연구실
고려대학교

22.11.13 - 22.11.18

1 개요

FSE'22 학회에 참석하기 위해 싱가포르에 다녀왔다. 11월 13일 일요일 (에도) 해서 18일 금요일 저녁에 떠났는데, 13일 (부) 17일 (까지) 내내 아팠다. 월요일에는 너4 아파서 학회 참석을 아예 » 하고, 약을 9고 토요일 오(와) 수요일 학회 (도만) 참석하고 나(지는 쪽) 에 있었다. 그(도) 학회를 (p) 금은 (가) 할 수 있었고, 마지막 금요일에 (A) U가 (나) 아(서) 싱가포르도 (p) 금 돌아다녀서 어떻게든 학회 참석 (A) 기를 남길 수 있어서 다행이다.

2 FSE 학회 참석

이(학) FSE 학회는 T 로나가 마(4) 리되는 단계(그동안 온(인)으로 진행했던 FSE'20, FSE'21까지 포함해서 (두) 오프(인)으로 같이 진행했다. 그(도)에도 각 ()마다 질(8) 시간을 넉넉하게 (U) (8) 할 수 있을만() 여유로웠는데, 아마 () 의 중국 연구자들이 T 로나 (E) 때(8)에 (두) 참석하지 » 해서 많이 여유로웠던 게 아닐까 싶다. 각 연구 ()마다 질(8) 시간이 길어서 ()들의 다양한 의견과 질(8)을 들어() 수 있어서 () 있는데, 이(학) 에는 특히 ()마다 부 () 인 질(8)을 많았던 것 같다. 예를 들() t, 특 () 기() 연구를 언급하() 서 이 연구가 이 연구와 4() 것이 다른건지, 9() 은 더 쉬운 기술로 해결이 가능한 () 가 아닌지 등의 질(8)이 많았다. 이(학) 에 특() 이했던 () 은 한국인 연구자분들이 굉장히 많이 () 었다. 아마 오() 만() 에 오프() 인으로 진행된 학회인데다, 싱가포르가 한국에서 꽤 가까운 곳인 () , 그리고 FSE'22 연구 () 만() 아니() 20년, 21년 연구도 포함된 때(8)에 그() 던() 게 아닐까 싶다.

2.1 분석 연구들

2.1.1 Past-Sensitive Pointer Analysis for Symbolic Execution

이 () 는 프로그() 분석 세션에서 들었던 () 인데, () 때는 (p) 감() 흥을 느끼지 » 했지만 논() 을 다시 () 아() 니() 꽤 () 재 () 는 연구였다. 이 연구는 () 에 드() 나() 듯() 이 기() 8 실행의 성능을 높이기 위해 포인() O 분석을 () 안한 연구이다. 나와 성준() 이도 MemFix() 부() O 해서 NPEX() 까지 위한 분석기를 디자인했고, 구 () 하() t() 서 () 예() 이() 분석 () 8 를 오류 수 () 에 특() T 된 휴리스틱으로 해결했() 기 때(8)에 이() 결() 논() 8으로 쓰고 싶은 갈() 증() 이 있었다. 분석기를 T 인으로 논() 8을 쓰기 위해서는 논() 8의 프레 () L 이() 셴 ()) 식을 많이 () 꺾() 야 해서 쉽게 쓰지 » 했는데, 이 논() 8이 그 () 8() 고() 가 될 것 같다. 이 논() 8에서는 특 () A() i() 에서만 () 용() 할 수 있는 분석 휴리스틱을 () 안했고, 이 기술이 유용함을 () 8() 이기 위해 다양한 기() 8 실행 시나리오를 () A() 하고 각 시나리오에서 유용함을 매우 구 () 으로 () 8() 여() 졌다. 이 논() 8에서 () 레() 를 든 기() 8 실행 시나리오는 다음 3가지이다.

- « () ^ () 째 () 레() 는 chopped symbolic execution이다. 이는 () 용() 자가 관심있는 부분을 중 () 으로 분석하는 기() 8 실행으로, 관련 없는 T 드() 를 포인() O 분석을 () μ() 해 알아내고 이를 기() 8 실행 단계에서 건너() 8() 여 () D() 용() 을 줄이는 게 핵심이다. 이때 포인() O 분석이 () U() 할수록, 관련 없는 T 드() 를 더 많이 건너() 8() 수 있다.

- 두 번째로, symbolic pointer resolution은 기8 실행 중 포인0가 가리0는 객´가 여러 개 일 경우, 각 이스 Å로 기8 실행 스L 이트를 나눠 분석하는 것을 말한다. 이때 스L 이트 개수가 증가하t D용이 증가하게 되는데, 가리0 수 없는 객´를 포인0 분석으로 0리 판Å하t 이 한 D용을 줄일 수 있다.
- 마지막으로, write integrity testing은 D Å 인 T´ 리 근을 µ해 Å수가 가리0 수 없는 영역을 가리0 수 있는지를 L스팅 하는건데, 포인0 분석을 µ해 Å 으로 가리0 수 있는 영역을 알아낸 Å 이 이외의 객´를 가리0 수 있는지를 판Å하는 것이다. 이 때도 마, 가지로 포인0 분석이 U해지t, L스팅이 견할 수 있는 오류가 더 많아진다.

자는 이 3가지 이스를 소개한 것 0만 아니, 구´인 예시로 어떻게 도움이 되는지도 0이고 각 응용 0례에 대한 실험도 진행하였다.

다만 \ 자´는 p급 한 느낌이었는데, 안한 기술로 4엇이 < 아지는 지0다 안한 포인0 분석이 4엇이고 이 결로 어떻게 기8 실행이 < 아지는 지 그 알고리즘을 위0로 설...하 려고 해서 그0 것 같다. \를 들0t 서는 그냥 많이 봤던 분석 휴리스틱 중 하나인데 왜 이결 특이한 이름을 붙여 안하는지, 이결 왜 h야 하는 지 Y각만 했던 것 같다. 요즘 오프|인 \ 들은 예 예 D해 U실히 \ 시간이 줄어들었는데, 어(피 기술 자´는 거기서 거기고 자세한 내용은 이해하기 힘들기 때8에 연구의 의의나 핵심 T세지에 집중해서 달해달|는 의0인 것 같다.

2.1.2 Domain-Independent Interprocedural Program Analysis using Block-Abstraction Memoization

이 연구는 쉽게 말하t 요즘 유행하는 summary 기 분석의 프레임위 를 시하는 논8이다. NPEX와 FL4APR 연구에서 개 한 분석기를 더 ´계 으로 만들 수 있지 않을까해서 들었던 \인데, 결론부0 말하t p 도움이 되지는 않을 것 같다. Summary 기 분석의 핵심 디자인 인 분석 결과를 파| 0OT 시0는) • 이나, summary를 함수 800지 에 맞게 구´ T 시0는) • 은 ´ 두 프레임위 의 입력이다. 게다가 \ 근에는 00팅 파워를 \용해서 서로 관련이 없는 함수들을 N렬´ 리로 ´르게 분석할 수 있도록 디자인하는 것도 중요한데, 이에 대한 지원 한 없었다.

다만, 이렇게 프레임위 를 시하는 논8이 어떻게 작성되는지는 0올 수 있었다. 이 연구는 (E한 대학교에서 교수가 8...이나 포함된 대 0의 CPAChecker|는 프로 트의 결과< 중 하나인데, 프레임위 가 연구되기 에 이0 아0 오´ 부0 수많은 ´ 델´ ä를 만들고 있었고 그 결과<들을 ...합해서 하나의 프레임위 로 만든 것으로 " ! 된다. 어떻게 0t 당연한 말이 지만, 프레임위 를 < 만들고 연구를 진행한 것이 아니| 예 연구를 0t 서 그 노하우를 프레임위 로 만들었다고 0 수 있다. 우리 오류 수 연구도 이´ 연구까지 하t 수 기를 4개째 구 하고 있는데, 매 연구마다 구 을 거의 0로하고 있다. 이´ 연구가 끝나t, 앞으로 연구들을 ´ 올 으로 하기 위해 오류 수 기술을 프레임위 T 할 수 없는지를 고0해0는 것도 < 올 것 같다.

2.2 Repair 연구들

오류 수 세션에서는 원석이 논8을 포함해서 5개가 \ 되었는데, 원석이 논8을 외하고는 " 두 답ì ñ을 ñ용한 오류 자동 수 기술이었다. 4개의 논8들은 거의 "A 답ì ñ " 델을 \ 용해서 B 오류를 수 해8았습니다" 도로 요약이 되는 게 아닐까 싶을 도로 특È이 없었다.

VulRepair VulRepair는 T5ì ñ " 델을 기 으로 학습을 해서 CWE 오류를 수 한 기술이다. 이 연구는 기t 답ì ñì ñ 기술에서 프리 트레이닝을 강T 시8고, T드 \ • 을 수 해서 성능을 올린 연구인데, 논8에 숫자 에 없고 예시가 하나도 없으니 이게 얼마나 대단한 기술인지 감이 잡히지 않았다. 어떻게 repair 연구에 동작 예시가 하나도 없을 수 있는지 잘 ñ 르겠다.

DeepDev-PERF: A Deep Learning-Based Approach for Improving Software Performance 이 연구는 마, 가지로, 답ì ñ을 \ 용해서 퍼포<스 오류 수 에 \ 용한 연구다. 연구 개요에는 오류의 53%를 수 했다고 했는데, 이는 " œ 패X 500개까지 봤을때의 결과이고 « ^ 짜는 8.3%에 불과했다. 게다가 각 " œ된 패X들은 단순히 프로그램" 을 입력으로 아 œ력한 결과는 아니고 " 델 " 론 이외에도 기t 오류 수 기술 ù Ò잡한 과 을 거8서 Ý성된다; 기t 오류 자동 수 기술과 같이 regression L 스트와 failing L 스트 (퍼포<스 기준)를 입력으로 고, T소드를 기준으로 결합 위X " 을 수행을 해서 수 할 T소드를 하고, T소드 Ä로 패X Ä8를 Ý성한 Ä, 그 중 Ò파일이 가능하8서 regression L 스트를 ì과한 패X만을 œ력한다. 그ì ñ까 리하88, 기t 오류 자동 수 시스\에서 패X T드 합성 부분만 답ì ñ으로 교´ 해8 연구ì 고 할 수 있다.

Less Training, More Repairing Please: Revisiting Automated Program Repair via Zero-Shot Learning 이 연구 한 CodeVertì ñ " 델을 ñ용해서 Defects4J의 오류를 수 해8 연구이다. 여기서 로÷ ñì ñì ñ ñ 워드가 ©에 들어가는 이유는 기t 에 학습된 " 델을 파인 튜닝 없이 로 ñ용했기 때8이다. 성능은 기t \ 신 기술과 p (ñ가 없고, 기t 기술에 D해 395개 오류 중 2개 도 더 수 하는 도이다. 성능 향Ä이ì 지 않은 이유는 « ^ 짜로 수 할 T드를 >는 8 ñ 해결하지 않고 그냥 기t 기술을 ñ용했기 때8이고, 두^ 짜로는 답ì ñ을 이용한 T드 ðÈ이 Ý각8다 잘 동작하지 않기 때8이다. 우리가 p ñ한 에 따88, 395개 오류 중 160개 도는 ñ인 하나만 수 하8 되는 D교 간단한 오류임에도 이 기술은 해당 ñ인을 알려8고, 5000개의 패X를 나열했음에도 74개 에 U하게 수 하지 » 했다. 게다가 51개 오류는 아예 L 스트를 ì과하는 패Xp (Ý성하지 » 했다. 패X T드 합성기를 잘 디자인하는 것이 꽤 노가다의 영역이기 때8이 이 부분만ì 도 답ì ñ으로 대´ 할 수 있을까 했는데, 아직은 대´ 가 되지 않는 영역인 것 같다.

3 싱가포르

마지막 금요일에, 이 p금 나아8서 싱가포르를 p금 들ì ü 수 있었다. 그동안 돌아다니지 » 한 것이 한이 되었던지, 념에 공항에 갈 때까지 약 13km를 걸어다니t 서 구경했다. 들ì ø 곳은 가든스 이 더 이, 싱가포르 플ì 이어 (관CE(), 술Ä `` 시 (이슬CE ~원) 도였는데, 하늘이 맑고 거리가 깨끗해서 걸으t 서 구경하기 < 았다. 가든스 이 더 이는 잘 꾸p놓은 p 공원으로, 걷기 < 고 나4들도 ä서 구경하기 < 았다. 다만, 중심부의 시그니D 나4들은 `` 두 돈을 내야 관CE을 할 수 있었는데 그 가격이 Ý각ô다 D싸서 그냥 üÄ만 구경하였다. 그 Ä에는 싱가포르 플ì 이어에 Ñ승하였는데, 관CE(가격 자´ 는 5만원 도로 꽤 D 지만 흔히 ô는 에, œ드의 관CE(와는 다르게 관CE(자´ 가 매우 l 고, 한 (에 소수의 ~CE들만 È 수 있어 CE 하게 Ñ승할 수 있었다. 그 이Ä로는 시간이 얼마 없어 üÄ부를 구경하다가 술Ä `` 시 까지 걸어갔는데, 거리가 워낙 깨끗하고 건<이 이D서 지루하지 않게 걸었다.



4 마무리

마지막으로, 학CE에 8석할 수 있게 해ü신 교수님께 감~드린다. 이~ 학CE는 연구 \ 없이도 다녀올 수 있게 해ü셨는데, D록 아파서 대로 8석하지는 » 했지만, 지금 하고 있는 연구가

의미가 있는 연구임을 더 깨우치는 계기가 되었다. 많은 학생들이 오류 자동 수정 기술에 관심을 갖고 있는 것을 다시 확인했고, , 많은 학생들이 단순히 자신에게 의지하여 성능 향상에 피하는 재앙에 불만을 갖고 있는 것도 인지할 수 있는 좋은 기회였다. 다음에는 지금 하고 있는 연구를 잘 마무리해서 학회에서 연구 결과를 할 수 있게 해서 오겠다.

