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 밤에 도착해서 18일 금요일 저녁에 떠났는데, 13일 밤부터 17일 저녁까지 내내 아팠다. 월요일에는 너무 아파서 학회 참석을 아예 못하고, 약을 먹고 화요일 오후와 수요일 학회 정도만 참석하고 나머지는 쭉 방에 있었다. 그래도 학회를 조금은 참가할 수 있었고, 마지막 금요일에 상태가 나아져서 싱가포르도 조금 돌아다녀서 어떻게든 학회 참석 후기를 남길 수 있어서 다행이다.

2 FSE 학회 참석

이번 FSE 학회는 코로나가 마무리되는 단계라 그동안 온라인으로 진행했던 FSE'20, FSE'21까지 포함해서 모두 오프라인으로 같이 진행했다. 그럼에도 각 발표마다 질문시간을 넉넉하게 확보할 수 있을만큼 여유로웠는데, 아마 본토의 중국 연구자들이 코로나 정책 때문에 모두 참석하지 못해서 많이 여유로웠던 게 아닐까 싶다. 각 연구 발표마다 질문 시간이 길어서 사람들의 다양한 의견과 질문을 들어볼 수 있어서 좋았는데, 이번에는 특히 발표마다 부정적인 질문을 많았던 것 같다. 예를 들면, 특정 기존 연구를 언급하면서 이 연구가 이 연구와 무엇이 다른건지, 혹은 더 쉬운 기술로 해결이 가능한 문제가 아닌지 등의 질문이 많았다. 또 이번에 특이했던 점은 한국인 연구자분들이 굉장히 많이 보였다. 아마 오랜만에 오프라인으로 진행된 학회인데다, 싱가포르가 한국에서 꽤 가까운 곳인 점, 그리고 FSE'22 연구 뿐만 아니라 20년, 21년 연구도 포함된 점 때문에 그랬던 게 아닐까 싶다.

2.1 분석 연구들

2.1.1 Past-Sensitive Pointer Analysis for Symbolic Execution

이 발표는 프로그램 분석 세션에서 들었던 발표인데, 발표 때는 큰 감흥을 느끼지 못했지만 논문을 다시 찾아보니 꽤 재밌는 연구였다. 이 연구는 제목에 드러나듯이 기호 실행의 성능을 높이기 위해 포인터 분석을 제안한 연구이다. 나와 성준이도 MemFix부터 해서 NPEX까지 위한 분석기를 디자인했고, 구현하면서 여러 정적 분석 문제를 오류 수정에 특화된 휴리스틱으로 해결했기 때문에 이걸 논문으로 쓰고 싶은 갈증이 있었다. 분석기를 메인으로 논문을 쓰기 위해서는 논문의 프레젠테이션 방식을 많이 바꿔야 해서 쉽게 쓰지 못했는데, 이 논문이 그 참고가 될 것 같다. 이 논문에서는 특정 상황에서만 사용할 수 있는 분석 휴리스틱을 제안했고, 이 기술이 유용함을 보이기 위해 다양한 기호 실행 시나리오를 상정하고 각 시나리오에서 유용함을 매우 구체적으로 보여줬다. 이 논문에서 사례를 든 기호 실행 시나리오는 다음 3가지이다.

- 첫번째 사례는 chopped symbolic execution이다. 이는 사용자가 관심있는 부분을 중점으로 분석하는 기호 실행으로, 관련 없는 코드를 포인터 분석을 통해 알아내고 이를 기호 실행 단계에서 건너뛰어 비용을 줄이는 게 핵심이다. 이때 포인터 분석이 정확할수록, 관련 없는 코드를 더 많이 건너뛸 수 있다.

- 두번째로, symbolic pointer resolution은 기호 실행 중 포인터가 가리키는 객체가 여러개일 경우, 각 케이스 별로 기호 실행 스테이트를 나눠 분석하는 것을 말한다. 이때 스테이트 개수가 증가하면 비용이 증가하게 되는데, 가리킬 수 없는 객체를 포인터 분석으로 미리 판별하면 이 또한 비용을 줄일 수 있다.

- 마지막으로, write integrity testing은 비정상적인 메모리 접근을 통해 변수가 가리킬 수 없는 영역을 가리킬 수 있는지를 테스팅 하는건데, 포인터 분석을 통해 정상적으로 가리킬 수 있는 영역을 알아낸 후 이 이외의 객체를 가리킬 수 있는지를 판별하는 것이다. 이 때도 마찬가지로 포인터 분석이 정확해지면, 테스팅이 발견할 수 있는 오류가 더 많아진다.

저자는 이 3가지 케이스를 소개한 것 뿐만 아니라, 구체적인 예시로 어떻게 도움이 되는지도 보이고 각 응용 사례에 대한 실험도 진행하였다.

다만 발표 자체는 조금 밋밋한 느낌이었는데, 제안한 기술로 무엇이 좋아지는 지보다 제안한 포인터 분석이 무엇이고 이 걸로 어떻게 기호 실행이 좋아지는 지 그 알고리즘을 위주로 설명하려고 해서 그런 것 같다. 발표를 들으면서는 그냥 많이 봤던 분석 휴리스틱 중 하나인데 왜 이걸 특이한 이름을 붙여 제안하는지, 이걸 왜 써야 하는 지 생각만 했던 것 같다. 요즘 오프라인 발표들은 예전에 비해 확실히 발표 시간이 줄어들었는데, 어차피 기술 자체는 거기서 거기고 자세한 내용은 이해하기 힘들기 때문에 연구의 의의나 핵심 메세지에 집중해서 전달해달라는 의미인 것 같다.

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

이 연구는 쉽게 말하면 요즘 유행하는 summary 기반 분석의 프레임워크를 제시하는 논문이다. NPEX와 FL4APR 연구에서 개발한 분석기를 더 체계적으로 만들 수 있지 않을까해서 들었던 발표인데, 결론부터 말하면 큰 도움이 되지는 않을 것 같다. Summary 기반 분석의 핵심 디자인인 분석 결과를 파라미터화 시키는 방법이나, summary를 함수 호출지점에 맞게 구체화 시키는 방법은 모두 프레임워크의 입력이다. 게다가 최근에는 컴퓨팅 파워를 활용해서 서로 관련이 없는 함수들을 병렬처리로 빠르게 분석할 수 있도록 디자인하는 것도 중요한데, 이에 대한 지원 또한 없었다.

다만, 이렇게 프레임워크를 제시하는 논문이 어떻게 작성되는지는 배울 수 있었다. 이 연구는 뮌헨 대학교에서 교수가 8명이나 포함된 대형 랩의 CPAChecker라는 프로젝트의 결과물 중 하나인데, 프레임워크가 연구되기 전에 이미 아주 오래전부터 수많은 모델 체커를 만들고 있었고 그 결과물들을 종합해서 하나의 프레임워크로 만든 것으로 추측된다. 어떻게 보면 당연한 말이지만, 프레임워크를 먼저 만들고 연구를 진행한 것이 아니라 여러 연구를 쌓으면서 그 노하우를 프레임워크로 만들었다고 볼 수 있다. 우리 오류 수정 연구도 이번 연구까지 하면 수정기를 4개째 구현하고 있는데, 매 연구마다 구현을 거의 새로하고 있다. 이번 연구가 끝나면, 앞으로 연구들을 효율적으로 하기 위해 오류 수정 기술을 프레임워크화 할 수 없는지를 고민해보는 것도 좋을 것 같다.

2.2 Repair 연구들

오류 수정 세션에서는 원석이 논문을 포함해서 총 5개가 발표되었는데, 원석이 논문을 제외하고는 모두 딥러닝을 사용한 오류 자동 수정 기술이었다. 4개의 논문들은 거의 "A 딥러닝 모델을 활용해서 B 오류를 수정해보았습니다" 정도로 요약이 되는 게 아닐까 싶을 정도로 특색이 없었다.

VulRepair VulRepair는 T5라는 모델을 기반으로 학습을 해서 CWE 오류를 수정한 기술이다. 이 연구는 기존 딥러닝 기반 기술에서 프리 트레이닝을 강화시키고, 코드 표현법을 수정해서 성능을 올린 연구인데, 논문에 숫자밖에 없고 예시가 하나도 없으니 이게 얼마나 대단한 기술인지 감이 전혀 잡히지 않았다. 어떻게 repair 연구에 동작 예시가 하나도 없을 수 있는지 잘 모르겠다.

DeepDev-PERF: A Deep Learning-Based Approach for Improving Software Performance 이 연구는 마찬가지로, 딥러닝을 활용해서 퍼포먼스 오류 수정에 활용한 연구다. 연구 개요에는 오류의 53%를 수정했다고 했는데, 이는 추천 패치 500개까지 봤을때의 결과이고 첫번째는 8.3%에 불과했다. 게다가 각 추천된 패치들은 단순히 프로그램을 입력으로 받아 출력한 결과는 아니고 모델 추론 이외에도 기존 오류 수정 기술처럼 복잡한 과정을 거쳐서 생성된다; 기존 오류 자동 수정 기술과 같이 regression 테스트와 failing 테스트 (퍼포먼스 기준)를 입력으로 받고, 메소드를 기준으로 결함 위치 추정을 수행을 해서 수정할 메소드를 정하고, 메소드 별로 패치 후보를 생성한 후, 그 중 컴파일이 가능하면서 regression 테스트를 통과한 패치만을 출력한다. 그러니까 정리하자면, 기존 오류 자동 수정 시스템에서 패치 코드 합성 부분만 딥러닝으로 교체해본 연구라고 할 수 있다.

Less Training, More Repairing Please: Revisiting Automated Program Repair via Zero-Shot Learning 이 연구 또한 CodeVert라는 모델을 사용해서 Defects4J의 오류를 수정해본 연구이다. 여기서 제로샷 러닝이라는 키워드가 제목에 들어가는 이유는 기존에 학습된 모델을 파인 튜닝 없이 바로 사용했기 때문이다. 성능은 기존 최신 기술과 큰 차이가 없고, 기존 기술에 비해 395개 오류 중 2개 정도 더 수정하는 정도이다. 성능 향상이 크지 않은 이유는 첫번째로 수정할 코드를 찾는 문제는 해결하지 않고 그냥 기존 기술을 사용했기 때문이고, 두번째로는 딥러닝을 이용한 코드 탐색이 생각보다 잘 동작하지 않기 때문이다. 우리가 조사한 바에 따르면, 395개 오류 중 160개 정도는 라인 하나만 수정하면 되는 비교적 간단한 오류임에도 이 기술은 해당 라인을 알려주고, 5000개의 패치를 나열했음에도 74개밖에 정확하게 수정하지 못했다. 게다가 51개 오류는 아예 테스트를 통과하는 패치조차 생성하지 못했다. 패치 코드 합성기를 잘 디자인하는 것이 꽤 노가다의 영역이기 때문이 이 부분만이라도 딥러닝으로 대체할 수 있을까 했는데, 아직은 대체가 되지 않는 영역인 것 같다.

3 싱가포르

마지막 금요일에 몸이 조금 나아져서 싱가포르를 조금 둘러볼 수 있었다. 그동안 전혀 돌아다니지 못한 것이 한이 되었던지, 저녁에 공항에 갈 때까지 약 13키로를 걸어다니면서 구경했다. 둘러본 곳은 가든스 바이 더 베이, 싱가포르 플라이어 (관람차), 술탄 모스크 (이슬람 사원) 정도였는데, 하늘이 맑고 거리가 깨끗해서 걸으면서 구경하기 좋았다. 가든스 바이 더 베이는 잘 꾸며놓은 큰 공원으로, 걷기 좋고 나무들도 커서 구경하기 좋았다. 다만, 중심부의 시그니쳐 나무들은 모두 돈을 내야 관람을 할 수 있었는데 그 가격이 생각보다 비싸서 그냥 주변만 구경하였다. 그 후에는 싱가포르 플라이어에 탑승하였는데, 관람차 가격 자체는 5만원 정도로 꽤 비쌌지만 흔히 보는 애버랜드의 관람차와는 다르게 관람차 자체가 매우 크고, 한 차에 소수의 사람들만 탈 수 있어 쾌적하게 탑승할 수 있었다. 그 이후로는 시간이 얼마 없어 주변부를 구경하다가 술탄 모스크까지 걸어갔는데, 거리가 워낙 깨끗하고 건물이 이뻐서 지루하지 않게 걸었다.




4 마무리

마지막으로, 학회에 참석할 수 있게 해주신 교수님께 감사드린다. 이번 학회는 연구 발표 없이도 다녀올 수 있게 해주셨는데, 비록 아파서 제대로 참석하지는 못했지만, 지금 하고 있는 연구가

의미가 있는 연구임을 더 깨우쳐주는 계기가 되었다. 많은 사람들이 오류 자동 수정 기술에 관심을 갖고 있는 것을 다시 확인했고, 또, 많은 사람들이 단순히 머신러닝에 의존하여 성능 향상에 꾀하는 현재 상태에 불만을 갖고 있는 것도 인지할 수 있는 좋은 기회였다. 다음에는 지금 하고 있는 연구를 잘 마무리해서 학회에서 연구 발표를 할 수 있게 해서 오겠다.