SIGPL 2024 겨울학교 Trip Report

이준협

February 6, 2024

1 감상

이번 2024년 SIGPL 겨울학교는 대전 KAIST에서 진행되었다. 겨울학교 기간 중 있었던 활동 중, 기억에 특히 남은 일들 몇 가지를 먼저 기록하겠다.

1.1 번개, 포스터 발표

SIGPL의 꽃, 번개 및 포스터 발표에서 현재 진행하고 있는 따로분석을 발표했다. 여느 때와 같이, 너무 이론적이라는 반응들이 많이 나왔다. 나 역시 동감하는 바이며, 어서 실제 따로분석의 예시 (ex. "Hybrid Top-down and Bottom-up Interprocedural Analysis"에 나오는 분석같은)를 내 이론으로 설명하고 싶다는 생각을 했다.

이러한 생각을 하게 된 이유는 오학주 교수님과 허기홍 교수님께서 내 포스터 발표에 와서 질문을 해 주셨기 때문이다. 둘의 공통적인 질문은 "지금 bottom-up analysis 방식과 다른 점이 무엇이냐"였다. 나는 "다른" 것이 아니라, "bottom-up analysis"나 "modular analysis"가 **무엇**인지에 대한 답을 찾고 있다고 답을 했다. 뜬 구름 잡는 이야기 같은 것은 나도 잘 알고 있으나, 어서 실제 분석 예시를 이 이론으로 설명해야겠다는 필요성을 강하게 느꼈다. SIGPL 중에 메모리가 있는 의미구조와 없는 의미구조 사이의 동등함을 보였으니, 이제야말로 분석의 세계로 발을 내딛어야겠다.

1.2 조별활동

이번 SIGPL에서는 특이하게 대학원생들의 친목을 도모하기 위한 조별활동 시간이 있었다. 첫째 날에는 조별로 조장의 인솔에 따라 대전의 맛집에서 저녁을 먹는 시간을 가졌으며, 둘째 날에는 PL 관련 골든벨을 통한 경품 추첨 시간이 있었다.

첫째 날에 우리 조에서는 닭도리탕을 먹었다. 닭도리탕은 가격에 비해 양을 상당히 많이 주었기 때문에 1인당 15,000원의 예산으로도 배가 터지도록 먹을 수 있었다. 무엇보다도 닭도리탕이 상당히 맛있었다. 우리 조에는 중국에서 온 학생과, 에티오피아에서 온 학생이 있었다. 한국말이 익숙치 않아 즐겁게 식사할 수 있을까 걱정이 되었으나, 연구에 대한 대화로 곧 왁자지껄했다.

생각해 보니, 첫째 날부터 맛있는 것을 참 많이 먹었다. 사흘 내내 케이터링으로 성심당 빵이 나왔는데, 대전의 명물이라는 말이 과장이 아니라고 느꼈다. 특히 샌드위치들이 (1일차: 참치, 2일차: 소세지) 참 맛있었다. 마지막 날에는 집에 가져가라고 잘 상하지 않는 디저트 류 빵들을 도시락 곽에

담아 가져갔는데, 샌드위치도 있었으면 좋았을걸 생각이 들었다. 그래도 점심을 먹고 기차를 타기 전에 성심당에 들려 모찌모찌 식빵을 사갔으니 만족스럽다.

Figure 1: 유일하게 찍을 생각을 한 사진인 성심당. 성심당이 곧 대전이니까 괜찮지 않을까?

조별활동 이야기를 이어서 하자면, 이틀째의 골든벨에서는 전날 있었던 번개 발표 슬라이드의 일부를 확대하여 누구 발표인지 맞추는 퀴즈가 있었다. 모든 사람의 이름이 나왔는데, 공교롭게도 내 슬라이드만 그대로 나오지 않고 개그를 섞어서 나왔다. "따로분석"이 "따로국밥" 같았는지, 국밥집 메뉴판을 편집하여 "국밥"을 "분석"으로 바꾸어 나온 것이다. 전혀 불쾌하지 않았고, 오히려 재밌었다. 발표가 인상에 남았다는 거니까, 나쁜 것이 아니지 않을까?

PL 관련 퀴즈에서는 다익스트라의 이름 (Edsger)을 맞추거나, Lisp의 개발자 (John McCarthy)를 맞추는 등 PL을 사랑하는 사람들을 위한 문제가 많이 나왔다. 나 역시 맞출 수 있는 문제가 몇 가지 나왔으나, 사회자와 자리가 너무 떨어져 있어 많이 맞추지 못했다. 아쉬웠다.

마지막으로는 PL 관련 단어를 ChatGPT에게 주고 GPT가 생성한 이미지만을 보고 어떤 단어인지 맞추는 문제들이 나왔다. 이 코너가 가장 신박했는데, 이미지가 처음에 너무 어렵게 나와서 맞추기 어려웠다. 나중에 가면서 프롬프트를 잘 주는 방법을 깨우친 것 같기도 한데, 그 때는 이미 늦었던 거 같기는 하다. 어찌 되었든, 재미있는 활동들을 준비해 준 카이스트 학우들에게 감사함을 느낀 하루였다.

2 강연

이번 SIGPL에서는 기존에 많이 보지 못한 연구자 분들의 강연을 들을 수 있도록 편성을 하였다고 한다. 간단하게 이번에 들은 강연들의 내용을 여기에 정리해보겠다.

딥 전이학습 기반 프로그램 자동 수정: 산업 적용 사례를 중심으로 (김미수 (전남대)) 먼저 원래 하시던 연구인 "정보검색 기반 버그 추적 자동화" (IRBL: Information retrieval-based bug localization)를 설명하셨다. 여기서 "정보검색 기반"이라는 것은, 버그 리포트를 받아서 그 내용을 검색하여 결함 위치를 특정할 수 있다는 뜻이다. 또한, "추적 자동화"라는 것은 검색의 결과인 결함 위치를 자동으로 추적할 수 있다는 뜻이다. 많은 버그 리포트를 분석하고, 거기에서 결함 위치에 중요한 부분을 추출하는 연구를 하셨다고 할 수 있다.

현재는 “자동 프로그램 수정” (Automated Program Repair) 연구를 하고 계신다고 하셨다. 특히, 패치 생성 기법을 통한 자동 프로그램 수정을 하고 계신다고 한다. 코드 패치 틀을 만들고, 그 틀에 맞춘 수정을 가하는 방식이다. 이 때, 코드 패치 틀을 생성하는 방법으로 최근에 딥러닝과 딥전이학습을 사용하셨다고 했다. 딥러닝을 이용한 방식은, 이미 있는 패치에 대한 훈련을 하여 패치 생성 모델을 만드는 방식이다. 이 방식에서 중요한 점은, “잡음 토큰”이 있어서 이러한 토큰을 전처리하는 것이 성능 향상에 핵심적이라는 것이다. 딥전이학습을 이용한 방식은, 이미 있는 모델에 패치 생성능력 장착하는 방식이다. 이 방법의 장점은, 이미 존재하는 딥러닝 파이프라인 안에 통합할 수 있다는 것이다.

실제 버그를 많이 보고 어떤 기술이 개발자들에게 가장 피부로 다가오는 결과를 줄 수 있는지 고민한 흔적이 보이는 좋은 연구라고 생각했다. 실제 사용에 중요한 점으로 (1) 기존 파이프라인에 적응성 및 유연성 (2) 정확도 보장 을 꼽았는데, 두 부분 모두 실제 정적분석기를 짜본 입장에서 매우 공감되는 어려움들이었다.

임베디드 시스템을 위한 컴파일러 디자인 (허선영 (경희대)) 원래 하시던 연구는 자동 프로그램 분산화 (Automatic Program Partitioning)라고 하셨다. 이는 분산시스템, 특히 여러 다른 기기들이 협동해야하는 분산시스템을 만들 때 사용하는 컴파일 기법이다. 간단히 설명하자면, 물건 중심 프로그래밍 방식으로 프로그래밍하여, 각 기기별 class가 저절로 잘 협동하도록 (method call→remote call) 컴파일을 해주는 기술이다.

현재는 임베디드 시스템을 위한 컴파일러를 만들고 계신다고 한다. 임베디드 시스템이란, 정보처리 시스템이 더 포괄적인 일을 하는 제품의 일부로 포함되는 경우로, 전력과 메모리 제약이 크다. 그런데, 요즘에는 머신러닝 모델을 이러한 저전력 환경에서 실행하고 싶어한다. 이러한 제약 하에서 잘 프로그래밍 하기 위해 Tiny ML framework라는 것들이 만들어지고 있다. 특히, TensorFlow Lite라는 도구는 동적 메모리 사용 최적화, 부동 소수점 정확도 낮추기 (quantization), 실행 파일 크기 압축 (필요한 연산자 정의만 가져옴)등을 실행해준다고 한다. 이러한 접근의 제약으로써 개발과정이 라이브러리에 매우 강하게 의존적이라는 점, 메모리 최적화를 위한 설정값을 일일이 사람이 정해줘야 한다는 점, 그리고 메모리 효율을 위한 조건문 때문에 성능이 떨어진다는 점이 있다.

이런 문제점을 해결하기 위해, 이미 있는 모델을 디컴파일 하여 높은 수준에서 최적화를 진행하는 컴파일러를 개발하고 있다고 한다 (Edge Impulse). 이렇게 하면 좋은 점은, 라이브러리에 대한 의존도가 낮아진다는 점이다. 디컴파일된 코드는 다른 라이브러리에 의존하지 않기 때문이다. 또한, 동적 메모리 계획이 아닌, 정적 분석을 통한 최적화 (liveness analysis)를 실행하여 메모리 사용량과 실행 시간 두 마리의 토끼를 잡았다고 한다.

근사 컴퓨팅의 자동 적용을 위한 컴파일러 최적화 기법 (박영준 (연세대)) GPU를 사용하는 근사 컴퓨팅의 품질을 보존하며 성능을 최적화하는 내용의 발표였다. GPU의 구조는, 같은 일을 하는, 서로 의존하지 않는 많은 작업이, 동시적으로 돌아가는 구조를 가지고 있다. GPU의 구조를 가장 잘 이용할 수 있으려면, 내 프로그램 특성에 맞춰서 코드를 짜야 한다. 하지만 환경과 상호작용에 따라 프로그램 실행 특성이 많이 달라질 수 있기에, 동적으로 최적화를 할 필요가 있다.

근사컴퓨팅은, 최소한 만족 가능한 품질 내에, 최대한의 성능을 이끌어낼 수 있는 근사를 해야한다. 근사컴퓨팅을 위한 컴파일 최적화는 주로 패턴 기반으로, 최적화 패턴 중 품질이 괜찮은 것을 선택하는

방식으로 이루어진다. 하지만, 당연히도, 패턴을 많이 만들어 놓아야 최적화 기회가 많이 생긴다. 또한, 패턴 기반이기에 같은 의미를 가지지만 겉모습이 달라서 패턴 적용 못하는 경우도 있다. 더 나아가, 적용하는 패턴의 종류에 따라 성능에 미치는 영향이 예측 불가할 수 있다.

위 문제들을 해결하기 위해, 어셈블리 단계의 정보 흐름 그래프를 가지고 최적화 가능성을 파악하는 방법을 생각했다고 한다. 이렇게 하면 좋은 점은, 의미 기반으로 최적화를 진행할 수 있다는 것이다. 또한, 최적화마다 입력에 따라 품질에 영향을 많이 주고/안주고 차이가 있기에, 최대한 성능에 영향을 덜 주는 최적화부터 적용하는 식으로 성능을 향상시킨다고 했다. 간단하게 보이는 접근이었지만, 실제로 GPU 성능 향상에는 커다란 영향을 미친 것을 보았다.

LeakPair: 단일페이지 웹 어플리케이션내의 메모리 누수를 고치기 위한 선제적 디버깅 (김동선 (경북대)) 한 웹페이지 내에서 실시간으로 컨텐츠를 받아오며 렌더링을 하는 경우, 메모리 누수가 일어나기 쉽다. 컨텐츠를 받아올 때마다 메모리 할당이 되는데, 이를 수동으로 해제해주지 않는다면 브라우저의 GC가 제대로 처리를 못 해주기 때문이다. 이러한 프로그래머 실수를 패턴 기반으로 감지하고, 수정을 제안해주는 연구를 했다.

이 연구에서 주목되는 점은 실제 개발자들의 사고방식을 변수로 고려하여 연구를 했다는 점이다. 옳음을 검증하기 어려운 패치나 까다로운 문제는 개발자들이 모른 체 하는 경우가 많다는 점에 착안해, 제안하는 패치의 종류를 선정했다. 즉, 제안하는 패치들이 유용성을 검증하기 쉬우며, 패치를 적용했을 때 프로그램의 기능에 아무 영향을 주지 않는 것을 매우 중요하게 생각했다. 실제 자신의 연구가 널리 쓰이게 하려면 어떠한 고민들을 해야하는지 느낄 수 있었던 강연이었다.

다중 노드 환경에서 빅데이터 스트리밍 엔진의 성능 평가 방법 (양신형 (연세대)) 스트리밍 서비스의 성능을 측정하기 위해서는, 컴퓨터 사이의 시간을 정확하게 측정할 수 있어야 한다. 그러나, NTP와 같은 프로토콜로는 오차가 너무 커서 컴퓨터 사이의 시간이 음수가 나오는 등, 쓸 수 없는 데이터가 나온다고 한다. 또한, 정밀 타임 프로토콜 (Precision Time Protocol)과 같은 방법은 너무 비용이 높다고 한다. 어떻게 하면 네트워크 통신 만으로, 최대한 정밀하게 시간을 측정할 수 있을 지 고민한 연구였다.

실제 성능 측정할 때 이러한 문제가 있을 수 있겠구나 생각하게끔 한 연구였다. 특히, 스트리밍과 같이 일분 일초가 아닌, 마이크로초 단위로 성능을 측정해야 하는 경우, 뼈를 깎는 심정으로 정밀도를 높여야겠다는 느낌이 들었다. 좋은 연구란 어떤 문제이든, 해결하기 어려우며 여러 사람이 중요함을 인정한 문제에서 나온다는 것을 느낄 수 있었다.

소프트 오류로부터의 신뢰성 향상을 위한 비트 단위 정적 분석 기법 (고유선 (연세대)) 하드웨어는 언제든지 전기적 오류가 발생할 수 있는데, 이러한 오류에 민감하지 않는 바이너리를 컴파일러에서 생성할 수 있을지 고민한 연구였다. 분석 방법으로는, LLVM IR 단에서 각 변수가 가질 수 있는 값을 비트 단위로 상향식-하향식 요약해석으로 분석하는 방식이었다. 하향식으로는 각 비트가 가지게 될 값을, 상향식으로는 각 비트에 만약 전기적 오류가 발생했을 경우 프로그램 실행에 어떤 영향을 끼칠 수 있는지 요약하는 방식이었다.

이 연구에서 또 한 가지 주목할 점은 바로 실험방법이었다. 프로그램이 소프트 오류에 민감하지 않게 하기 위해서는 불가피하게 실행 시간이 길어질 수밖에 없는데, 그러면 발생할 수 있는 소프트

오류의 절대적 수가 늘어나기 때문에 이를 고려하여 실험을 해야한다는 것이었다. 이렇게 실험을 했더니, 이전 연구들 중에서는 오히려 신뢰도가 나빠진 프로그램이 발견되었다고 한다. 연구의 이론적 내용을 뒷받침해주기 위해서는 공정한 평가 방법 역시 필수적이라는 것을 느낄 수 있었다.

코드 속성 그래프의 부분 그래프 매칭을 통한 대규모 취약점 탐지 (위성일 (UNIST)) 현재 웹에서 일어날 수 있는 보안 취약점을 없애는 연구를 하고 계신다고 한다. 특히, 프로그램 분석을 통해 보안 취약점을 드러낼 수 있는 테스트 입력을 생성하는 연구를 하신다고 한다. 이번 발표에서는 코드 성질을 정적분석을 통해 그래프로 나타내고, 부분그래프 동일성 매칭을 통해 보안 취약점을 일으킬 수 있는 코드를 파악하는 연구를 소개한다고 한다.

코드 성질 그래프(Code Property Graph, CPG)라는 것은 AST+CFG+PDG(Program Dependency Graph)이다. 버그 패턴이 있는지 파악하기 위해 그래프 탐색을 하면, 그 패턴에 맞는 코드가 나오는 식으로 버그 탐지를 할 수 있다. 이 연구에서는 이미 알려진 취약점들을 바탕으로, CPG의 부분그래프가 그러한 취약점과 유사한지 탐지한다. 하지만, 부분그래프 동일성은 NP-complete 문제여서, 큰 프로그램에 대해 어렵다고 한다. 이를 해결하기 위해, CPG 가지치기(실제 취약점에 꼭 필요한 간선만 유지)를 했다고 한다. 또한, 생긴 것이 동등하지 않아도 의미는 같을 수 있기에, CPG를 추가적으로 요약하는 과정이 필요했다고 한다. 웹에서 돌아가는 PHP 코드를 검증했다고 했는데, 코드 성질 그래프라는 것이 어떻게 생성되는지 궁금했다.

3 총평

이번 SIGPL이 로파스에 들어오고 나서 네 번째로 참여하는 SIGPL이다. 매번 SIGPL에 올 때마다 주변 연구자들의 훌륭한 연구와 질문, 그리고 열의에 동기부여를 받는 것 같다. 나 역시 어서 좋은 연구를 해서, 작더라도 세상에 혹 하나쯤은 남기고 싶다는 생각이 든다.

이번 SIGPL의 모토는 “사이좋게 지내요”라고 강지훈 교수님이 말하셨다. 솔직히 처음에 조별활동 이야기를 들었을 때 그리 의욕이 나지는 않았으나, 막상 참여하고 나니 재미있었다. SIGPL에 참여한 모두가 프로그래밍 언어 분야에 애정과 열정을 가지고 있기 때문에, 앞으로도 계속 교류를 하며 서로의 연구를 응원하고 자극할 수 있으면 좋겠다는 생각을 하게 되었다.