SIGPL 2022 겨울학교

이재호

2022.02.12–14 @제주

제주도에서 개최된 지난 SIGPL 2022 겨울학교에서, 여러 초청 강연과 발표를 경청하며 국내 다른 연구실에서 어떤 연구들을 진행하고 있는지 동향을 살펴볼 수 있었다. 이들 중 몇 주제를 골라서 요약해보았다.

1 들어가며

학부생 인턴으로 ROPAS에 들어온지 한 달이 채 지나기도 전에 SIGPL 2022 겨울학교에 참가하게 되었다. 연구실에 출근한지 며칠 후, 제주도에서 열리는 학회에 참석한다는 소식을 듣고 여러 감정이 교차했다. 사실 복학 준비를 위해 서울에 올라오기 전, 최근에 부모님과 제주도로 가족 여행을 다녀왔었다. 그때는 제주시에서 머물렀는데, 이번 겨울학교는 서귀포시에 위치한 곳에서 열려 색다른 경험이 될 것 같았다. 제주도에는 여러 번 방문했지만 서귀포시에는 기억을 더듬어봐도 들렀던 기억이 없다.

그림 1: SIGPL 2022 겨울학교. 맨 왼쪽이 필자이다.

SIGPL 홈페이지[1]를 살펴보니 SIGPL 계절학교의 역사가 30년도 더 되었다는 사실을 알고 적잖이 놀랐다. 비록 세월이 흐르면서 겨울학교의 성격이 조금씩 변화한 것 같지만, 이런 흐름의 일부가 된다는 생각에 가슴이 벅차기도 하고 신나기도 하였다. 아직 학부 3학년이기도 하고—입학한지 벌써 4년이 흘렀지만—, 연구실에 합류한 것도 가장 늦은 막내로서 배울 것이 많은 기회라고 생각했다.

2 기억에 남는 발표

2.1 초청 강연

코딩 과제물의 평가, 현실과 문제 (부산대 조환규)

부산대 조환규 교수님께서, 각종 대회와 과제로 제출된 학생들의 수많은 코딩 과제물을 평가하면서 겪으신 여러 경험을 매우 흥미롭게 들었다. 필자가 직접 학부 수업을 수강할 때 제출하였던 비효율적인 방식의 코딩 과제물—전기·정보공학부에 국한된 내용일 수도 있다—이 떠오르기도 하였고, 프로그래밍 과외를 여러 차례 진행하면서 겪은 기억도 생각났다. 또한 교수님의 재치있는 입담 덕분에 처음부터 끝까지 집중하면서 재밌게 발표를 들을 수 있었다.

사실 본 발표에서 제시된 유사도 산출 방식들은 모두 표면적인 코드의 모습만을 보고 판단하는 것이었다. 10여 년 전에 ROPAS에서 MeCC[2]와 같이 의미를 추적하여 표절을 찾아내는 방법이 연구된 것을 생각하면, 방법론적으로 특별히 새로운 것은 없었다.

다만 교수님께서 수많은 학생들의 과제를 평가하고 교육하며 얻은 노하우가 값졌다고 생각한다. 예컨대 한정된 자원에서 증폭되는 생산성이라던지, 서사를 가지고 발전되는 프로그래밍 과제물(USACO의 문제들처럼?)이라던지 말이다. 교수님께서 제안하신 것과 같이, 코딩 컨벤션에 따르는 정도나 표면적인 정돈성, 나아가 논리 구조에 다른 '우아함'을 측정하는 것도 재미있는 탐구가 될 것 같다.

본 강연의 내용은 신기하게도 다음 날 진행된 스타트업 엘리스의 프로그래밍 교육 플랫폼에 대한 강연으로 여러모로 이어지는 점들이 있었다.

플랫폼 중심 프로그래밍 교육 운영 사례 (엘리스 김재원)

카이스트 스타트업에서 시작된 프로그래밍 교육 플랫폼 엘리스를 운영하면서 겪은 경험을 나눠주신 김재원 대표님의 강연도 매우 흥미롭게 들었다. 전 날 조환규 교수님의 발표와 같은 테마이지만, 교수자가 아닌 플랫폼 개발자의 관점에서 바라본 발표였다. 특히 미국 명문대의 프로그래밍 수업의 수강생들이 몇 백에서

[1] https://sigpl.or.kr/notice/

[2] https://dl.acm.org/doi/10.1145/1985793.1985835

천 단위에 이른다는 사실에 매우 놀랐는데, 우리 학교에서도 저런 규모의 교육이 가능할지, 기반은 닦여져 있을지 궁금하였다.

사업적인 측면에서는, 원래 인프라만 갖춘 방향으로 진행하려고 하였으나 (대기업의 요청으로) 컨텐츠를 자체 제작하였더니 매출이 엄청나게 늘었다는 사실이 재미있었다. 개발자적인 사고 방식을 가진 필자도 '재미있는 부분'인 인프라 구축에만 힘을 줬겠지만, 자체적으로 컨텐츠를 만들어서 공급하지 않는다면 돈이 덜 된다는 사실을 다시금 깨달았다. 어떻게 보면 뼈대보다는 내용이 중요하다는 것을 다시 한 번 입증해주는 사례인 것이다. 맞는 비유인지 확실하진 않지만, 기기를 만드는 삼성의 갤럭시보다 운영체제를 만드는 구글의 안드로이드가 더 우위를 점하는 것과 비슷한 경우 아닐까?

구현 측면에서는, 사용자의 키스트로크 하나 하나까지 기록하여 코드를 작성한 전 과정을 모두 돌려볼 수 있게 하였다는 점이 놀라웠다. 당연히 구현 방식 등에서는 완전히 다르겠지만 키스트로크를 기록한다는 점에서 asciinema가 떠올랐다. 이런 방식으로 부정 행위를 방지하거나, 학생들의 입력 패턴과 학습 성취도의 관계 파악 등을 위한 데이터 수집을 위해서 도움이 될 것이다. 다만 여담으로, vim 키바인딩(혹은 기타 에디터)에 의존적인 필자와 같은 부류의 사람들에게는 꽤나 불편한 제약일 것이다.

발표를 듣고 생각해보니, '싹수 분석'이 무르익는다면 적용할 수 있는 부분이 살짝 보였다. 사용자가 작성하고 있는 코드 조각이 '싹수'가 있는지 판별하여 작성자에게 힌트를 줄 수 있지 않을까? 농담이지만, 조환규 교수님이 말씀하신 '가망이 없으면 어두워지는 에디터'를 실현할 수 있는 한 가지 방법이다. 싹수 분석까지는 아니지만 POPL'19에서 발표된 타입을 가진 구멍이 있는 실시간 함수형 프로그래밍도 엘리스 프로그래밍 환경에 적용하면 유용할 듯 하다.

2.2 연구 발표

싹수 분석으로 프로그램 합성 보조하기 (서울대 윤용호)

윤용호 선배님의 발표는, '싹수 분석' 문제의 bit-vector 연산에서의 성과를 보여주었다. 위에서부터 언급한 '싹수 분석'이라고 함은, 입출력 제약을 만족하는 프로그램을 합성할 때 폭발적으로 증가하는 탐색 공간의 가지들을 가능한 일찍 제거하는 기술을 말한다. 아직 완성되지는 않은 미완성 프로그램이지만, 여기서 빈 칸에 어떤 내용을 채워넣더라도 가망이 없는 경우 싹수가 없다고 표현해 싹수 분석인 것이다.

필자가 이해한 바로는, 싹수 분석의 핵심은 통상적인 정방향 분석보다도 역방향 분석에 있다. 연산의 결과가 주어졌을 때 연산의 대상이 되는 값들이 가져야 하는 성질을 분석할 수 있어야 하기 때문이다. 간단한 예시로, bit-vector b_1과 b_2가 각각 `TTTTTTTT`, `TT10101T`의 형태이고, $b_1 \wedge b_2$의 결과는 `T100T01T`라 하자.

Bit-wise and 연산자 $\wedge$의 특성에 따라 결과가 1이면 입력 값은 둘 다 1이어야 하고, 결과가 0인데도 불구하고 한 입력의 값이 1이면 다른 하나의 값은 무조건 0이어야 한다는 사실을 사용하면, b_1은 T10T0T1T, b_2는 T110101T의 형태로 더 구체화할 수 있다.

연산자 별로 정방향 및 역방향 분석을 맞춤 수행해야 하기 때문에 범용 프로그래밍 언어에서의 반복문 구조라던지 순서쌍 등의 형태를 가진 데이터에서 어떻게 적용할 것인지가 '싹수 분석'의 앞으로 발전 방향일 것이다.[3]

프로그램 자동탐색 기술을 이용한 동형암호 회로 최적화 (서울대 이동권)

사실 해당 발표는 이동권 선배님께서 최근에 필자를 포함하여 새로 합류한 인턴들을 위해 전체적인 조망을 해 주신 적이 있었기 때문에 내용이 생소하지는 않았다.

동형암호란, 정보 처리자 자신이 처리하고 있는 정보의 의미를 알 수 없게끔 암호화된 입력 상에서 각종 연산을 수행하여, 정보를 소유하고 있는 사용자만 복호화할 수 있는 형태의 연산 결과를 산출할 수 있게 하는 암호화 기술이다. 이러한 특징 덕분에 사용자는 민감한 개인 정보의 유출을 원천적으로 방지할 수 있다.

그러나 일상적인 프로그램을 동형암호로 처리된 정보를 다룰 수 있는 동형암호 프로그램의 형태로 변환하는 것은 매우 어렵다. 기존에는 전문 지식을 갖춘 동형암호 개발자가 일일이 저수준의 명령어로 코드를 작성하면서 여러 매개 변수들을 조절해야 했는데, 이는 많은 노력을 요할 뿐만이 아니라 그렇게 산출된 동형암호 프로그램이 효율적인 것도 아니었다. 그렇다고 해서 평문 프로그램을 동형암호 프로그램으로 자동 번역 및 최적화를 해주는 동형암호 컴파일러가 빠른 결과를 내놓는 것도 아니었다.

본 연구에서는 프로그램 합성과 e-graph를 통한 동일식 모두탐색 기술을 이용하여 기존의 동형암호 프로그램을 두 배 가량 빠르게 하는 동형암호 컴파일러 최적화 기법을 제시하였다. 주어진 동형암호 회로와 동일한 연산을 하면서 더 곱셈 깊이가 얕은 형태의 회로를 합성함으로써 인간은 생각하기 쉽지 않은 최적화 규칙들을 생성하고, 동일식 모두탐색 기법을 통해 최적화 순서 문제를 공략하였다.

JavaScript Static Analysis for Evolving Language Specifications (KAIST 박지혁)

본 발표는 스타트업이나 동아리 등에서 프론트엔드 앱 개발을 여러 차례 해 보았던 필자의 배경을 생각해 본다면 가장 와닿으면서도 신선하게 느껴졌다. 어떻게 보면 대충 만들어졌지만—JavaScript의 열렬한 지지자들은 쫓아올 수도 있는 발언이지만—너무나도 널리 퍼져서 정보화 시대의 근간이 되어 버린 언어인 JavaScript는 굉장히 자유로운 영혼을 가졌다. 때문에 다른 언어에서는 바로 컴파일 에러나

[3] 사실 이 문제가 교수님께 받아 고민하고 있는 문제이다.

런타임 에러를 일으킬 문제를 관대하게 넘겨준다. 예컨대 `round`에 수가 아니라 문자열을 넣어주는 행위처럼 말이다!

본 발표는 ECMA-262, 즉 ECMAScript 공식 정의 문서의 자동화된 검증 및 JS 엔진들의 구현 검증에 관한 것으로, 사실은 여러 연구의 집합체이다. 먼저 JISET (JavaScript IR-based Semantics Extraction Toolchain)을 통해서 영어 문장으로 구성된 ECMA-262로부터 기계화된 명세를 추출을 한다. 이렇게 추출된 명세를 사용해 JEST (JavaScript Engines and Specification Tester)로 V8, GraavlVM, QuickJS, moddable 등의 JS 구현 엔진을 검증한다. 이 때 소수의 엔진에만 문제가 있다면 이는 엔진 버그일 가능성이 높고, 반대로 소수의 엔진만 테스트를 통과한다면 이는 명세 자체의 오류일 가능성이 높다는 점에서 착안해 테스트를 진행한다. 이를 통해 44개의 엔진 버그와 27개의 명세 버그를 탐지하였다고 한다.

JSTAR (JavaScript Type Analysis for Specification)은 명세에 있는 타입 오류를 검증할 수 있도록 진행된 연구인데, 이를 통해 앞서 언급하였던 `Math.round` 표준 함수 명세에 있는 타입 오류를 찾아내기도 하였다. 무려 93개의 타입 오류를 탐지하였다고 한다.

마지막으로 제출 중에 있는 JSAVER (JavaScript Static Analyzer via ECMAScript Representations)는 ECMAScript의 명세를 입력으로 받아 JS 프로그램에 대한 정적 분석기를 유도하는 연구이다. 어떻게 보면 앞서 언급된 연구들의 결정체라고 부를 수 있을 것 같다. 기존의 정적 분석기는 표준이 매 해 최신화될 때마다 변형해야 해서 현 시점의 코드의 반도 덮지 못할 수준이 되었지만, 이를 통해 표준이 새로 만들어져도 정적 분석기를 유도해낼 수 있게 되었기 때문이다.

부산물로 '명세 실행기'도 잠시 보여주셨는데, 일반적인 JS 실행기와 다르게 코드가 실행될 때 명세의 어떤 부분에 따라서 엔진이 돌아가는지 모두 볼 수 있게 하는 장치였다. C++과 같은 (마찬가지로) 거대한 언어에 대해서도 가능할지 질문하였는데, C++은 ECMAScript와 다르게 명세가 듬성듬성 있다고 하여서 당장은 어려울 수 있다고 하였다. 그러나 분명 명세가 촘촘하게 보완된다면 분명 적용할 수 있을 것이라고 한다.

3 맺으며

겨울학교 둘째 날에는 '폭풍의 언덕'에서 산책을 하였는데, 날씨가 실로 폭풍의 언덕이라고 부를 수 있을 만한 것이었다. 물론 폭풍우가 몰아친 것은 아니었지만, 결코 화창한 날씨는 아니었다. 하지만 이런 날씨에 올레길을 걷는 것도 인생에 몇 번 없을 기회였기 때문에 풍경을 눈에 많이 담아 놓았다. 올레길을 가리키는 표식에 흡사 λ를 닮아서 '람다의 길'이라고 농담을 하며 담소를 나눈 기억이 있다 (그림 3).

이번 학회에서 또 하나 뜻 깊은 인연은, 고등학교 졸업 후 4년 만에 동창을 만났다는 것이다. 그 친구도 필자와 이름이 같은 재호인데, 학회 며칠 전에 메일링

그림 2: λ의 길

리스트에서 낯익은 GitHub 닉네임을 보고 연락하게 되었다. 고등학교 동창이 4년이 흘러 같은 분야에 흥미를 가졌다는 것이 굉장히 놀랍고 신기하였다. 앞으로도 관심사를 나누면서 교류할 수 있는 깊은 인연을 맺게 되었다.

다음에 열릴 여름학교에서도 국내 다른 연구자 분들의 연구도 배우고 새로운 인연을 만들어가면 의미 있을 것이다. 그 때는 필자도 더 실력을 키워 더 많은 연구 주제들을 소화하고 대화할 수 있는 역량을 갖추도록 부단히 노력해야겠다.