POPL/VMCAI 2012

Philadelphia

2012. 1. 22 - 2012. 1. 27

서울대학교 프로그래밍 연구실
오학주, 이원찬, 이우석, 조성근, 이승중, 김진영, 윤용호, 이영석, 최준원, 강지훈

2012년 2월

차례

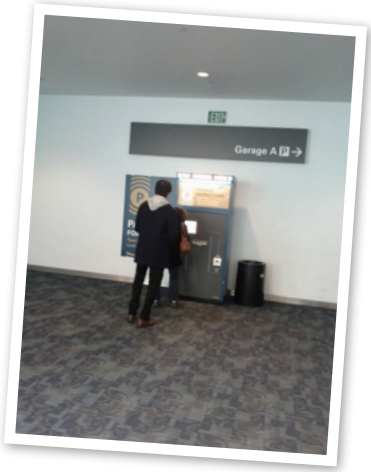
SAN FRANCISCO, BERKELEY

비행기 결항으로 받은 예기치 못한 선물

결항

강지훈

나는 성근 선배, 원찬 선배, 우석 선배와 함께 다른 사람들보다 하루 먼저 인천국제공항을 출발, San Francisco를 경유하여 Philadelphia로 갈 예정이었다. 그러나 공교롭게도 Philadelphia에 폭설이 내려 비행기가 결항되고 말았다. 어차피 Philadelphia에서 하루 묵을 것을 San francisco에서 하루 묵는다고 생각하며 좋게 생각하기로 했다. Philadelphia에 예약했던 호텔도 하루 취소하고 이제 어떻게 해야하나 생각하며 이 곳에서 하룻밤 묵을 호텔을 잡고 어디를 관광할지 생각해보기로 했다.



그러던 찰나 원찬 선배가 UC Berkeley에서 박사과정을 하고 있는 원태 선배에게 연락해보자고 했다. 우연히 우리가 환승하려고 했던 San Francisco가 Berkeley 근처였기에 가능한 일이었다. 원태 선배는 흔쾌히 공항으로 우리를 데리러 오겠다고 해주었다. 마침 근처에 있던 희종 선배와도 만날 수 있었다. 우리는 현지인들의 도움으로 앞으로의 약 열흘간의 일정중 가장 풍족한 생활을 San Francisco에서 보낼 수 있었다.

사진은 공항에서 본 Automata 전시회(?)와 공항에서 주차 비용을 지불하는 원태 선배의 뒷모습.

Mama's

강지훈

호텔을 잡기도 전에 원태 선배가 꼭 데려가야 할 곳이 있다고 했다. 그냥 제대로 음식을 만들어 파는 곳이라고 소개했다. 이름도 그냥 Mama's, 한국에 있었다면 '엄마의 손맛' 정도일 것 같은 약간 구린 이름에, 좁디 좁은 매장을 보고 평범한 밥집 아니겠는가 싶었다. 그러다가 주문하려고 기다릴 때부터 살살 느낌이 왔다.

'제대로 만들어 파는 집이구나.'

주문하면서 볼 수 있었던 음식 만드는 요리사들의 손놀림이 수려했다. 그에 반해 음식 이름들은 'Club Sandwich', 'Burger'와 같이 간단했는데 오히려 그런 간단한 이름들이 믿음을 주었다. 난 사실 메뉴도 잘 읽지 못해서 메뉴를 잘 둘러보지도 못하고 가장 이름을 읽기 쉬웠던 Club Sandwich를 주문했다. 약 $10 였다.

이럴수가!

놀라운 맛이었다. 그저 그런 재료에 향만 적당히 내고 야채는 그저 모양을 낼 구실로만 들어있는 그저 그런 샌드위치와는 전혀 다른 맛이었다. 각 재료들은 자신들을 뽐내지 않고 묵묵히 각자의 맛을 내고 있었지만, 맛의 조화로운 조합은 각 재료의 맛을 하나도 지우지 않고 온전히 간직하고 있었다. 야채는 그저 신선했고 빵은 바삭했고 고기는 부드러우면서도 씹는 맛이 느껴졌다. 글로는 그저 '일품이었다' 라고밖에 하지 못하는 것이 안타까울 정도의 맛이었다.

다른 선배들이 주문했던 몽테크리스토, 오믈렛, 버거 모두 Club Sandwich와 마찬가지로 훌륭했다. 버거의 고기는 마치 스테이크의 고기와 같았고 오믈렛은 씹지 않고 물고만 있어도 입안에서 고체가 액체로 변하는 기적을 체험할 수 있었고 몽테크리스토는 한국에서 먹었던 몽테크리스토는 도대체 어떻게 만든 것인지 의심케 하는 맛이었다. 미국 음식이 맛없다 맛없다 하지만, 역시 음식은 본고장에서 먹어야 한다고 생각했다.

San Francisco에 올 일이 있다면 Washington Square의 Mama's에 가길 강력 추천한다.

Ghirardelli 초콜릿 공장

조성근

원태를 따라 맛있는 밥을 먹고 후식을 위해 길라델리라는 초콜릿 가게로 향했다. 여기는 입장할 때부터 500원 두 개 크기만한 초콜릿을 쥐어 준다. 우리가 주문한 것 중 인상에 남는 것이 핫초코였다. 정말 이름 그대로 뜨거운 초코였다. 너무나도 진해서 식고나면 당장이라도 딱딱하게 굳을 것만 같았다. 오랜 비행으로 매우 피곤한 상태였는데 생전 처음 보는 달콤함에 잠이 확 깨었다. 너무 달아 '딸꿈함'이라 표현하고 싶다.

모두들 피곤한 상태에서 초콜릿까지 먹고 숙소로 향하는 차 안에서 깊은 잠에 빠져 있었다. 창밖으로는 비가 내리고 있었고, 안개 속으로 사라져 가는 아주 크고 멋있는 다리(Golden

Gate Bridge)가 눈에 들어왔다. 모두들 자느라 이 멋있는 광경을 놓치는 것이 조금 안타까웠다.

UC Berkeley

강지훈

Clinton 대통령이 San Francisco에 방문할 때마다 들린다는 유명한 음식점에 가려고 했으나, 개점시간에 갔음에도 불구하고 이미 예약이 되어서 식사하지 못했다. 그 아쉬움때문인지 카메라의 남은 음식점의 사진이 더 신비롭게 느껴졌다. 대신 매일 한 종류의 피자만 파는 곳에서 피자를 사 먹었다. 이 역시 훌륭한 맛이었다.

UC Berkeley 앞에서 커피를 마시면서 선배들이 연구에 관한 이야기를 하는 것을 들었다. 아직 공부가 부족해 무슨 이야기를 나누는지 다 알아듣지 못했지만, 연구에 대한 선배들의 열정을 엿볼 수 있는 시간이었다. 요즘 어떤 연구가 유행이고 유행의 이유는 무엇인지, 구체적으로는 Software model checking 하는 사람들은 어떤 문제제기를 가지고 있으며 문제에 어떻게 접근하고 있는지, 분석기 만드는 그룹이 할 수 있는 일이 무엇인지 이야기했다. 시간만 허락한다면 밤을 세워가며 이야기할 수 있었을 것 같았다.

떠나며

강지훈

비행기 시간이 되어 Philadelphia로 떠날 때가 되었다. 우연히 지하철에서 학주 선배와 승중 선배를 만났다. 선물같기도 하고 꿈같기도 했던 San Francisco와 Berkeley를 뒤로하고 이제 학회에 집중해야겠다고 생각했다.

이 자리를 빌어 다시 한 번 San Francisco와 Berkeley에서 우리를 환영해 주었던 희종 선배와 원태 선배에게 감사의 말씀을 전하고 싶다.

VMCAI 2012

오학주

VMCAI는 POPL에 비해서는 아담하지만 검증과 분석에 관한 한해동안의 연구결과가 발표되는 주요 학회이다.

VMCAI는 6-70명 규모의 사람들이 모여서 검증(Verification), 모델검증(Model Checking), 요약해석(Abstract Interpretation)에 대한 연구결과를 교류하는 장이다. 학회의 취지는, 이 세분야가 사실은 비슷한 목적을 가지고 있으므로 서로 교류하고 영향을 주자는 것이다. 하지만, 그 취지와는 달리, 각 분야를 융합하고 시너지를 내보려는 시도보다는 세 분야의 논문이 한데 어우러지는 느낌이 들기도 한다. 특히, 요약해석보다는 모델검증이나 검증에 치우치는 느낌이다. 요약해석 관련해서는 거의 유일하게 우리랩의 우석이가 논문을 한편 발표하였다.

Synthesizing Efficient Controllers

Christian von Essen and Barbara Jobstmann

이영석

이 발표는 Marcov Decision Processes 를 응용하여서 효율적인 시스템을 만드는 연구내용 이었다. 이 발표의 앞뒤로 계속 Synthesizing 에 대한 얘기가 나왔다. Invited talk 에서 마이크로소프트의 엑셀의 예를 들면서 설명도 하였고, 굉장히 흥미로운 분야로 느껴졌다. 이 연구에서는 보상과 비용을 확률적인 환경에서 모델링 하여서 효율적인 시스템을 설계 하였고, 그것을 펌프의 예로 설명하였다. 이번 발표에서 몇 안되는 그나마 많이 이해한 발표 중에 하나이다. 중간 중간에 확실하게 이해되지 않은 부분이 있지만, 쉬운 예를 들어 설명을 해주어서, 처음 보는 사람들도 쉽게 이해 할 수 있고 어떠한 일을 하고자 하는지를 분명하게 전달해 주었던 발표라고 생각한다. 단지 아쉬웠던 부분은 모델 체킹에 대한 기본 배경 지식이 많이 부족하여서 얼마나 좋은 아이디어인지 대단한 것인지에 대한 판단을 할 수 없었던 것이 아쉽다.

Teaching Semantics with a Proof Assistant: Teach Proofs, not Logic!

Tobias Nipkow

강지훈

이 초청 강연은 Technische Universität München의 석사 1-2년차를 위한 의미론 과목에 증명 보조 도구인 Isabelle을 이용한 경험을 다뤘다. 저자인 Nipkow의 이름이 익숙해서 누구인지 생각해보다가, 저자의 홈페이지를 보고 "Term Rewriting and All That"의 저자임을 다시금 깨닫게 되었다. 지난 여름에 다시 쓰기rewriting 타입 이론을 접해보고, 이 방향으로 더 공부한다면 꼭 봐야할 교과서라고 들은 바가 있어 알게 된 책이었다.

저자는 이미 십여년전에 Winskel의 유명한 교과서 “Formal Semantics of Programming Languages”의 앞 부분을 Isabelle로 구현했다고 한다. 다만 효율을 생각해서 강의에 증명 보조 도구를 도입하지 않았는데, 얼마 전 역시 증명 보조 도구인 Coq을 이용해 의미론을 가르치는 Benjamin Pierce의 과목 Software Foundations를 보고 영감을 받아 강의를 재구성, 증명 보조 도구를 전면적으로 사용하기 시작했다고 한다. 이는 긍정적인 효과를 불러일으켰는데, 학생들의 성취도와 만족도가 올라갔고, 부정행위도 한 학기에 한 건밖에 없었다고 한다.

저자는 성공의 가장 큰 요인으로 증명 보조 도구가 바로바로 피드백을 주는 것을 꼽았다. 보통 이론 과목의 경우, 숙제를 제출하면 조교가 채점해서 1-2주 후에 돌려주는데, 이 1-2주간 이미 다른 진도를 나가게 되어 흥미가 떨어지게 된다. 증명 보조 도구를 이용하면 한 과정 과정마다 바로바로 결과를 알 수 있으므로 이런 문제를 많이 줄일 수 있고, 작은 돌도 두드려보며 건너게 되므로 얻어낸 결과에 대해 보다 확신을 얻을 수 있게 된다. 비록 많은 학생들이 너무 과목 부담이 크다고 불평하긴 했지만, 많은 것을 얻어갔다고 평한 점을 저자는 자랑스러워하는 것 같았다.

구체적으로 이 과목은 IMP라는 간단한 C와 비슷한 언어를 대상으로, 모양의 정의, 실행 의미 정의부터 간단한 프로그램 분석까지 Isabelle 위에서 구현한다. 한 학기에 다룰 수 있는 양이라고 상상도 할 수 없을 정도로 많은 분량을 그것도 증명 과정을 증명 보조 도구의 엄격한

검사를 통과하며 밟아갔다는 것을 미루어보아 독일 학생들은 공부를 굉장히 열심히 하는 것 같았다.

나도 매 방학마다 Coq을 배워볼까 했던 것이 생각났다. POPL 2008에서 선보였던 CoqIntro를 했던 기억, Software Foundations의 앞 부분을 잠깐 했던 기억을 되돌아봤다. 막히는 부분이 있을 때마다 더 열심히 고민하고 답을 찾으려 하기보다는 포기했던 적이 얼마나 많던가. 증명 보조 도구는 겉 핥기식으로 공부하게 놔두지 않고 꼭 진중하게 학습하도록 강제하는 면이 있는데, 나는 이런 면을 잘 활용하지 못했던 것 같다.

앞으로 하게 될 일들을 증명 보조 도구로 검증해야할 때 걸림돌이 되지 않도록 Coq을 배워야겠다고 마음먹었다. 구체적으로 지금 하고 있는 다단계 언어의 변환을 증명 보조 도구를 이용해서 증명해보고 싶다. 이번에는 Benjamin Pierce의 Software Foundations를 해서 중단하는 일 없이 끝을 봐야겠다.

Automating induction with an SMT solver

Rustan Leino

이원찬

SMT 풀이기를 사용해서 귀납법을 사용해야 하는 정리를 자동으로 증명한 논문이다. SMT 풀이기를 사용하여 어떤 명제 P가 있을 때 ¬P가 참이 될 수 있는 지를 검사함으로써 명제 P를 증명할 수 있다. 논문의 아이디어는 모든 x에 대해 어떤 명제 P(x)가 성립하는 지 증명할 때 ¬(∀x.P(x))가 참이 될 수 있는 지 물어보는 대신 ¬(∀x.(∀y<x.P(y)) ⇒ P(x))이 참이 될 수 있는지 물어보자는 것이다. 화살표 왼쪽에 귀납 가정인 ∀y<x.P(y)을 붙여서 SMT 풀이기에 넘겨주는 것이다. 단순히 ¬(∀x.P(x))가 참이 될 수 있는지만 물어볼 경우 SMT 풀이기가 알아서 귀납 가정을 붙여주지는 않는다.

논문의 아이디어가 동작하기 위해서는 두 가지가 필요하다. 하나는 SMT 풀이기가 "모든"(∀)을 지원해야 하고 다른 하나는 x에 대해 순서(<)가 정의 돼야 한다. 요즘 나오는 Z_3와 같은 SMT 풀이기 대부분은 "모든"을 지원하며 사용자가 정의한 자료구조에 대해서도 순서를 표현할 수 있다고 한다.

발표를 평하자면 개인적으로 VMCAI 최고의 발표가 아니었나 생각한다. 무엇보다도 두괄식 구성이었다. 시작하자마자 논문이 한 일과 그 방법을 두 마디로 설명했다. "SMT 풀이기로 귀납 증명을 자동으로 할 수 있습니다. 방법은 ∀x.P(x) 대신 (∀x.(∀y<x.P) ⇒ P)를 증명하게 하면 됩니다." 이 두 마디를 듣는 순간 논문의 내용이 명료하게 들어왔다. 짤막한 소개 다음에는 다양한 예제를 통한 시연으로 자칫 지루할 수 있는 발표를 흥미롭게 이끌었다. 종반에 가서는 세부적인 내용을 설명하여 앞의 발표로 관심이 생긴 청중의 마지막 궁금증을 해소해 주었다. 귀감이 되는 좋은 발표였다.

Software Verification with Liquid Types

Ranjit Jhala

최준원

이 발표가 있던 전날 같이 학회에 갔던 많은 선배들께서 이 발표가 꽤 여러차례 있어왔다고 말씀해 주셨다. 나는 한 번도 이 발표를 들어본 적이 없던지라 큰 기대를 가지고 들었는데 역시나 수려한 설명과 깔끔한 슬라이드에 깊은 감명을 받았고 이해도 잘 할 수 있었다. 이번 POPL 발표에서 저는 주로 타입을 변형하여 성과를 이루어내는 발표에 큰 재미를 느꼈는데 이 발표 또한 그런 종류였다.

발표의 첫 내용은 Refinement Type 에 대한 것이었다. 일반적인 타입의 경우 (변수):int 와 같이 해당하는 변수의 타입을 알려준다. Refinement Type 은 여기에 변수의 범위를 x ≤ (변수) ≤ y 와 같이 추가로 제시한다. 이와 같은 타입 시스템을 복잡한 자료구조나 High Order PL 에 도입하면 기존의 Solver 가 쉽게 풀어내지 못했던 불변값 관련 문제를 풀어낼 수 있다. Refinement Type 에서는 기존의 sub-type 개념을 Refinement Type 에 확장하여 {v=i} <: {o≤v≤a.len} (마치 부분집합 관계와 같은) 와 같은 관계를 만들어 내고 이를 풀어낸다.

Refinement Type 의 설명이 끝난 후에 본격적으로 문제를 풀이하는 방법인 Automatic Refinement Inference 에 대하여 설명하였다. 전통적인 타입 시스템에서의 타입 추론이 확장된 타입에서는 문제를 푸는 도구로 변하는 것이 신기하였다. Refinement 에 해당하는 템플릿을 타입 시스템에 추가하고 문맥 의미를 통해 제한값을 계산하면서 Fixpoint 를 얻어낼 때까지 계산을 반복하는 과정이 요약 해석에서의 그것과 비슷하여 흥미를 느꼈다.

New Applications of Underapproximations in Static Analysis

Alex Aiken

김진영

Saturn을 만드신 스탠포드의 Alex Aiken의 초청 강연이었다. 발표 전체를 완전히 이해하지는 못하였지만 그래도 재미있게 들은 발표 중 하나였다.

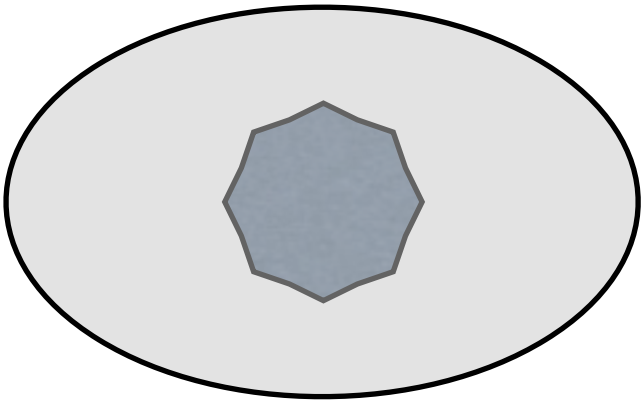
핵심은 요약해석에서 우리가 요약을 할 때 대부분의 경우 “실제 가능성을 모두 포섭하는 요약”을 하게 되는데, 이를 정 반대로 뒤집어서, “실제 가능성에 완전히 포함되는 요약”을 생각해 보자는 내용이었다. 이런 요약 도메인을 만들어서 그 안에서 이런저런 연산과 성질을 정의하고 살펴보았다.

그리고 이처럼 포섭하는 요약과 포함되는 요약을 함께 가지고 다니면 요약 도메인 아래에서 부정 연산을 붙이고서도 요약을 유지할 수 있게 된다는 점이 흥미로웠다. 여집합을 계산하면 집합의 포함관계가 뒤집히게 되므로 예전에 포섭하는 요약은 포함되는 요약이 되는 식이다.

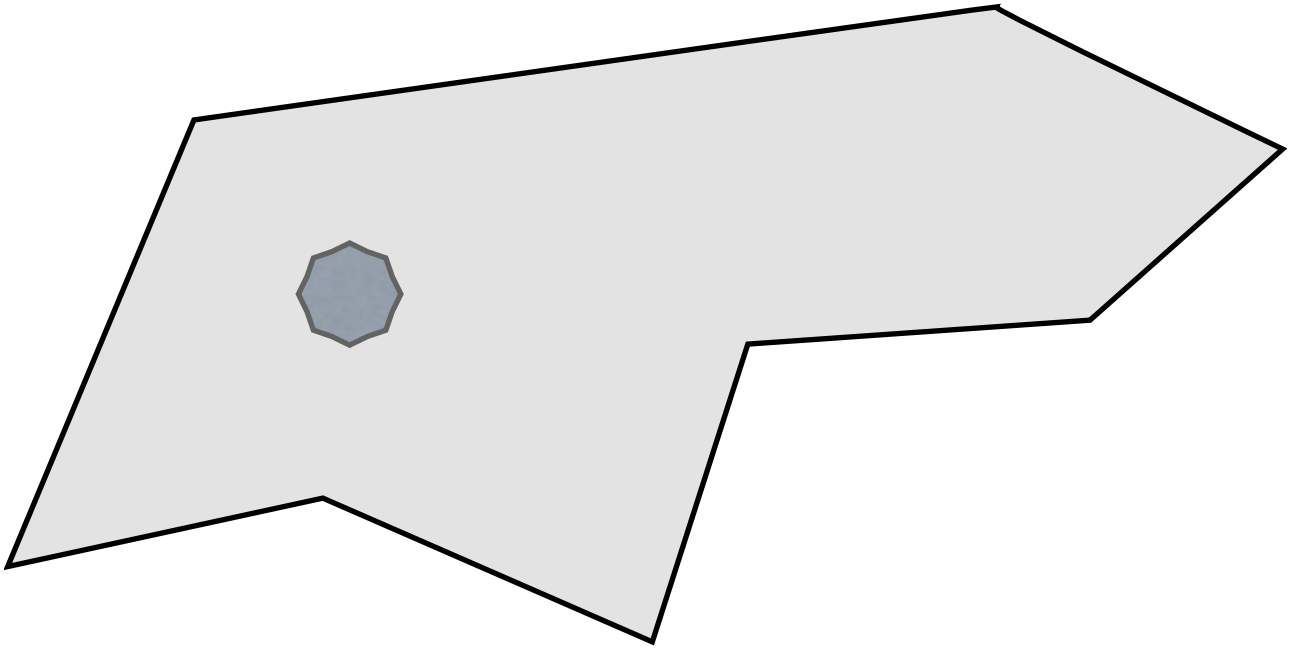
마치 로직에서 일종의 **may**와 **must**와 비슷한 것 같다는 생각도 들었고, 필요조건 충분조건을 따지는 내용과도 비슷하다는 생각도 들었다.

수업시간에 배운 정도로만 요약해석 프레임웍을 이해하고 있던 수준이어서 그런지 발표가 전반적으로 흥미롭게 다가왔다. 이해도 중반부까지는 어렵지 않게 차근차근 강의하듯 설명해주는 느낌이어서 좋았다. 다만 후반부에 경로 민감 분석의 경우와 재귀호출이 있는 경우까지 논의를 확장하면서 흐름을 좀 놓쳐서 따라가기 버거웠다.

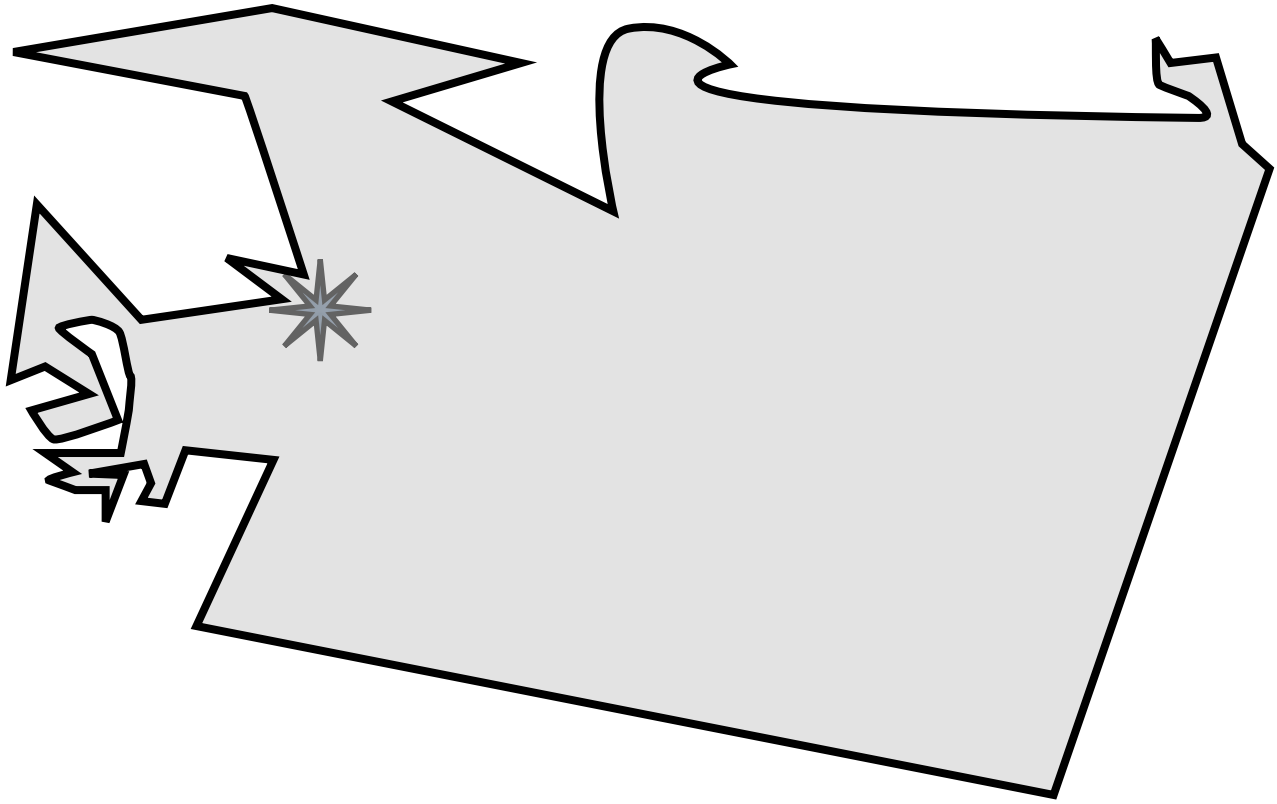
발표 주제와는 큰 상관이 없었지만 의외의 부분이 가장 기억에 남는다. 요약해석을 그림으로 나타낼 때 보통 이런 식으로 그리게 되는데,



어떻게 양심이 있지 이런 그림을 그릴 수 있느냐고 했다. 최소한,



이 정도로는 그려야 하고, 사실 현실에서는



이런 일도 빈번하지 않냐는 이야기를 하셨다. 재미있었고, 공감했다.

Alternating Control Flow Reconstruction

Johannes Kinder and Dmitry Kravchenko

이승중

이 발표는 바이너리 코드의 분석에서 제어흐름을 보다 정확하게 만드는 것에 대한 내용이다. 일반적으로 요약분석을 하다 보면, 레지스터의 값이 모든 값을 가리키게 되어 프로그램 포인트 전체로 뛰게 되는 문제점이 발생한다. 프로그램의 가능한 모든 포인트로 실행흐름이 퍼지게 되면 분석이 부정확해지는 문제가 생긴다.

이 발표는 그 문제점을 해결하고자 프로그램의 제어흐름을 안전하게 요약하지 않았다. 부정확해지는 프로그램 포인트에서는 랜덤 입력을 갖고 프로그램을 실행시켜 가능한 값 중 하나를 만들어낸다. 나머지 흐름들은 동적 테스트 생성기법을 통해서 만들어낸다.
실험결과만 놓고 보았을 때 대체적으로 좋은 성능을 보여주었다. 가능한 한 값만을 얻어내는 것과 프로그램 실행흐름을 포괄하는 방법보다 도달범위가 넓었다. 또한 안전한 요약방식은 큰 프로그램에 대해서 분석이 불가능한데 반해, 이 방식은 비교적 짧은 시간에 분석이 가능했다.

실제 프로그램을 제대로 분석하기 위해서는 그 프로그램의 상황에 맞게 대처하는 것이 중요하다는 생각이 들었다. 예를 들면 C++ 소스를 컴파일 해서 얻어진 바이너리의 경우에는 가상

함수 호출이 많이 일어나서 실행흐름이 여러 군데로 퍼질 수가 있는데, 여기서 이 방법을 사용할 경우 프로그램의 도달범위를 많이 높일 수 있었던 것 같다.

강지훈

기계 언어로 작성된 프로그램의 제어흐름도CFG를 재구성하기가 힘든 문제를 해결하고자 하는 논문이다.

기계 언어는 상태에 따라 다른 곳으로 제어를 이동시킬 수 있기 때문에 (indirect jump), CFG를 잘 구성하기가 매우 힘들다. 예를 들어

```
0: x = Top
4: jump x
```

와 같은 코드 조각을 실행시킬 경우, 4에서 x의 값이 무엇일지 전혀 알 수 없기 때문에 (분석 결과가 아무런 정보를 주지 못하기 때문에) jump 명령어를 안전하게 분석하려면 이 명령어가 아무 곳으로나 뛴다고 가정해야 한다. 따라서 안전하게 재구성한 CFG는 보통 너무 정확도가 낮아 쓸모가 없다. 해를 끼치는 프로그램 (malware)를 잘 분석하기 위해서는 정확도가 높은 CFG를 재구성하는 것이 필수적이기 때문에 이 문제는 어렵고도 중요한 문제로 잘 알려져왔다.
이 논문은 안전함을 포기하는 대신, 번갈아가며 (alternatively) 올려서 어림짐작 (over-approximation) 하기도 하고 내려서 어림짐작 (under-approximation) 하기도 하며 CFG를 재구성하여 정확도를 매우 높이 올렸다.
실험 결과도 인상적인데, 손으로 만들어본 assembly 프로그램과 실제로 산업에 쓰이는 작은 C 프로그램의 경우 평균적으로 50%의 실행 경로를 찾아냈고 정확도는 거의 100%에 육박했다.
우리 연구실에서 진행중인 binary 프로그램 분석에 어떤 식으로든 적용할 수 있으면 좋겠다.
이 논문의 최대 약점은 실행 의미를 안전하게 요약하지 않는다는데 있는데, 이를 해결하면서도 원래 논문의 강점이었던 부분을 잘 흡수할 수만 있다면 가능할 것이라고 생각했다.

Donut Domains: Efficient Non-Convex Domains for Abstract Interpretation

Khalil Ghorbal, Franjo Ivancic, Gogul Balakrishnan, Naoto Maeda and Aarti Gupta

이우석

VMCAI에서 내 논문과 더불어 거의 유일한 요약해석 논문이다. 새로운 요약 도메인에 대한 논문인데 이름에서 유추할 수 있듯 가운데가 뻥 뚫린 모양의 제약식을 표현할 수 있는 도메인이다.

이 도메인은 가능한 값들의 over approximation을 convex invariant 형 태로 표현하는 요약 객체와, 불가능한 값들의 under approximation을 마찬가지로 convex 불변식들로 표현하는 요약 객체의 합으로 이루어진 non-convex 요약 도메인이다. 이와 비슷한 것으로 여러 개의 인터벌을 표현할 수 있는 BOXES 도메인과 disequality를 표현할 수 있는 polyhedra 도메인 등이 있는데, 도넛 도메인은 변수간 관계도 추적할 수 있고 disequality 보다 더 일반적인 형태의 구멍(hole)도 고려할 수 있다는 점에서 더 일반적이다.

실제로 분석기 허위경보들을 살펴보다 보면, disequality 조건문들을 요약 도메인이 표현하지 못해 허위경보가 발생하는 경우들이 종종 보인다. 아 주 간단하게는 변수가 어떤 특정 값과 같게 되면 탈출이 일어나는 그런 반복문에서, 정지 조건이 부등호 < 를 써서 표현되어 있지 않아 widening이 정확도를 오염시키는 경우들이 이에 해당한다. 이 도메인은 그러한 경우들에 대해 정확도를 향상시키는데 사용될 수 있을 것이다.

좋은 내용임에도 불구하고 발표는 많이 알아듣지 못하였다. 나와 같이 거의 유일하게 요약해석에 관련한 논문이기도 하고 연구실 분석기 향상에 도움이 될 내용이었으나, 발표 내용이 매우 기술적이고 발표가 내 발표차례 직전이었기 때문에 긴장이 돼서 전혀 집중이 되지 않았다. 후에 차분히 논문을 살펴봐야겠다.

VMCAI를 정리하며

강지훈

VMCAI 발표가 끝나가고 연이어 시작할 POPL에 대한 기대감으로 부풀어 올랐다. 하지만 우리 연구실 사람들은 VMCAI가 끝날 때까지 마지막 발표에 대한 기대감으로 긴장의 끈을 놓을 수가 없었다. 마침내 VMCAI의 마지막 발표자인 우석 선배가 마이크를 잡았다.

우리의 발표

VMCAI 발표, *POPL* 학생 발표

VMCAI 발표

이우석

작년 TEXAS Austin 에 서 열린 VMCAI, POPL'11에는 발표하는 논문 없이 그냥 참석만 하였는데, 이번엔 발표자로서 참석하게 되어 부담이 되었다.

VMCAI'12 맨 마지막 세션 마지막 발표에 "Sound Non-statistical Clustering of Static Analysis Alarms" 논문 을 발표하였다. 이 논문은 이원찬 연구원, 이광근 교수님과 함께 작성하였다. 이 논문은 내가 파수닷컴이라는 회사에서 인턴을 하는 동안 동기를 얻어 시작하게 된 연구의 결과물이다. 파수닷컴은 연구실에서 만든 정적 분석기 Sparrow를 상용화한 회사이다. 나는 이 회사에서 인턴을 하는 동안 고객사의 프로그램 분석 결과 나온 알람들을 살펴보는 일이 잦았고, 비슷한 알람들을 묶으면 사용자가 안전하게 더 적은 알람만을 볼 수 있겠다는 생각에 일을 시작하였다. 인턴이 끝나고 학교로 돌아와 원찬이 형과 교수님과 함께 논문으로 정리하여 제출하게 되었다. 우리는 모든 정적 분석에 대해 안전하게 알람들을 묶을 수 있는 이론적 틀을 제시하고, 실 제 프로그램들에 대해 절반의 알람을 사용자가 덜 검사할 수 있음을 실험적으로 보였다.

1. 자신의 경험에 대한 이야기와 함께 연구 동기를 설명하는 것은 좋다.

발표 초반부에 연구 동기에 대한 설명을 나의 인턴 경험과 함께 얘기했더니 사람들이 흥미로워했다. 350만줄 프로그램을 분석하면 알람이 1000개 나오는데, 나는 100개 보는데 하루 종일 걸렸다고 얘기했더니 사람들이 웃었다. 덕분에 긴장이 풀어져서 초반부터 마음편하게 발표를 이어나갈 수 있었다.

물론 이야기를 할 때 너무 개인적이거나 논문의 요지와 깊은 관계가 없는 부분은 적당히 빼야 할 것이다.

2. 너무 쉽게 발표를 구성하는 것은 좋지 않다.

작년 학회에서 지나치게 기술적인 세부내용들을 열거한 발표들을 많이 봐서, 나는 그렇게 하지 않으려고 최대한 쉽고 직관적으로 발표를 구성하려고 애썼다. 그런데 이것이 지나쳐서 내용이 너무 쉬웠던지 일부 사람들이 지루해하였다. 그래서 청중 반응을 보아가며 쉬운 부분을 빨리빨리 넘겼더니 발표가 예상보다 일찍 끝났다. 좀 더 내용을 전달할 수 있었을텐데 하는 아쉬움이 남았다. 지나치게 기술적인 발표도 문제지만 너무 쉬운 발표도 문제임을 알았다. 그 중간의 좋은 지점을 여러 사람들의 피드백과 자신의 거듭된 시행착오로부터 빨리 찾는 것이 중요할 것 같다.

3. 발표할 때 발표 스크립트를 띄워놓으면 요긴하게 쓸 수 있다.

나는 발표할 때 무슨 말을 이어나가야 할 지 모를 상황을 대비해서 슬라이드 노트에 스크립트를 써놓았다. 그리고 발표할 때 노트북에는 스크립트가 뜨게 하고, 발표장 화면엔 슬라이드가 뜨게 하였다. 발표하면서 기본적으로는 스크립트를 보지 않고 대부분의 시간을 청중을 보면서 얘기를 하였지만, 간혹 다음 할 말이 생각이 안 나서 말이 끊길 것 같으면(주로 다음 슬라이드 얘기하기 시작할 때) 바로 스크립트를 보고 다음 문장을 얘기했다. 그러면 그 다음 이어질 내용은 또 기억이 나서 스크립트를 볼 필요 없이 청중을 보며 얘기를 이어나갈 수 있었다. 이렇게 거의 끊김 없이 얘기를 계속할 수 있었다.

발표 스크립트를 준비하고 발표하는 것에 대해서는 부정적인 사람도 있고, 긍정적인 사람도 있는 것 같다. 무엇이 좋은지는 상황에 따라, 그리고 개인차에 따라 다른 것 같다. 내 경우에는 적당히 의지할 수 있는 장치가 되었다.

4. 발표에 필요한 준비물은 빈틈없이 챙기자.

깜빡 잊고 레이저 포인터와 슬라이드를 넘기는 리모컨을 챙기지 못했다. 학회장 주변에 딱히 구입할 장소가 없자 불안감이 엄습했다. 다행히 마지막 날 같은 연구실 승중이 형이 시내로 나가서 구해줘서 발표를 잘 끝마칠 수 있었지만 못 구한 동안 은근히 신경이 쓰였다. 그런데 나만 그런 것은 아니었고, 발표자들 중 종종 포인터가 없어서 곤란해하는 사람들이 있었다. 준비물은 사소한 듯 중요한 문제이다.

5. 실제 질문들은 예상 질문의 범주를 벗어날 수 있다.

나는 영어 듣기를 잘 못해서 질문 답변 시간이 가장 걱정이 되었다. 그래서 미리 내가 생각할 수 있는 한 가능한 질문들과 그에 대한 답변을 영어로 정리해놓았다. 그리고 이와 관련한 백업 슬라이드들도 만들어 놓았다. 그러나 내가 준비한 것들을 하나도 쓸 수 없었다. 질문이 4개 정도 나왔는데 예상한 질문들 중 맞아 떨어진 게 단 하나뿐이었기 때문이다. 그 하나는 논문 리뷰 때도 제기되었던 것이어서 답변을 미리 준비해놓은 원찬이 형이 대신 쉽게 대답하였지만 나머지는 전부 생각지도 못한 것들이었다. 당황해서 잘 알아듣지 못하고 결국 제대로 대답을 못했다. 예상 시나리오에 의존하지 말고 최대한 의연하게 대처했으면 좋았겠다 싶기도 했지만, 한편으로는 예상 밖 질문들이 내 수준에는 약간 난해해서 잘 알아들었어도 대답하기 어려웠을 것 같다. 뻔한 결론이긴 하지만 영어와 연구 면에서 기본기와 내공을 키우는 수 밖에는 답이 없을 것 같다.

6. 지인들 앞에서 리허설을 많이 해보자.

이것은 같이 간 지인들이 있거나 학회에서 도움을 줄 수 있는 사람을 만난 경우에만 해당될 수 있을 것 같다. 나는 학회에 같이 참석한 같은 연구실 사람들 앞에서 리허설을 해보면서 부족한 부분을 보완할 수 있었다. 또한 마찬가지로 VMCAI에 논문을 발표한 한국인 유학생 어덕기씨를 만나서 그 분 앞에서도 리허설을 해보고 많은 조언을 들을 수 있었다. 여러 피드백

도 도움이 되었지만, 무엇보다 리허설을 반복하면서 자신감이 붙었다. 이 점이 큰 이득이었던 것 같다.

학회에서 만난 어덕기 씨는 현재 IOWA 대학교에서 박사학위 취득을 앞두고 계신 분인데, 공교롭게도 ROSAEC 센터 박사 후 연구원 지원에 관심이 있다고 하셨다. 인연은 참 오묘한 것 같다.

POPL 학생 발표 소개

오학주

나를 비롯해서 우리랩에서 간 대부분의 학생들이 학생발표에 참여했다. ROSAEC 워크샵에서의 lightning talk과 비슷했다. 다만, 시간이 4분으로 칼같이 정해져 있고, 슬라이드 장수도 딱 12장으로 정해져 있었다. 게다가 한 슬라이드당 20초의 시간이 주어지고, 20초가 지나면 자동으로 슬라이드가 넘어간다. 슬라이드가 강제로 넘어가기 때문에, 발표시간은 지킬 수 밖에 없는 상황이고, 20초동안 각 슬라이드의 핵심만 간결하게 설명하고 넘어가야 한다.

20초 제한은 결과적으로 괜찮은 시도인것 같았다. 처음에는 4분만 맞추면되지, 슬라이드 장수 제한이나 각 장별 시간제한은 심하다고 생각했다. 하지만, lightning talk의 특성상 연구의 동기 및 중요성, 키 아이디어 정도로 내용을 구성한다고 할 때 전체 슬라이드를 12개로 균일하게 쪼깨면서 구성하는 방식이 핵심적인 사항을 간결하게 표현하는데 도움이 되었다.

학생발표 세션은 첫째날 저녁에 있었다. 신청자가 많아서 (32명) 모두 다 발표할 수는 없고, 예선을 거쳐서 16명만 POPL 청중앞에서 발표하는 식으로 진행되었다. 그래서 32명이 4조로 나뉘어서 8명씩 예선이 진행되었다. 예선은 8명이 돌아가면서 발표를 하고, 서로 투표하여 최종참가자 4명을 선발하는 식이었다. 우리 조에는 원찬이와 내가 함께 있었는데, 나는 예선에서 떨어졌고, 원찬이가 본선에 진출했다. 모두들 lightning talk답게 슬라이드를 재치있게 준비해온 것을 볼 수 있었다. 각자의 발표스타일도 제각각 개성있었다. 얼마나 오랫동안 연습했는지, 각 슬라이드당 20초씩 정확하게 말하면서, 남거나 모자르지 않고 물흐르듯이 발표하는 사람들도 많았다. 재미있는 경험이었다.

POPL 학생 발표 참가

이원찬

POPL 학회에서 막간의 행사로 열렸던 번개 발표에 참가했다. 학생들에게 발표 기회를 주고자 만든 순서로 작년에 이어 열렸다고 한다. 모든 발표자는 총 12 장의 슬라이드를 4분 동안 발표해야 했다. 재밌는 것은 슬라이드 한 장에 꼭 20초씩 써야 했다는 점이다. 20초가 지나면 다음 슬라이드로 자동으로 넘어간다. 이런 발표 방식은 pecha-kucha라고 하는, 디자이너들의 행사로 유명해진 방식이라고 한다.

발표는 총 두 단계에 걸쳐서 진행이 되었다. 첫 단계에서는 32명의 참가자중 16명을 뽑는다. 이를 위해 8명씩 4개 조로 나뉘어 발표를 진행한 다음, 서로의 발표를 평가하여 각 조에서 4명씩 뽑았다. 이렇게 뽑힌 16명은 본 학회장에서 발표할 기회를 얻었다.

나는 학습 알고리즘을 사용한 종료 분석에 대해 발표했다. 당시 막 CAV 학회에 제출했던 논문의 주된 내용이였다. 논문도 썼고 세미나, 워크샵 등에서 발표했던 주제라 준비가 수월할 줄 알았다. 하지만 발표에 가해진 20초 제약 덕분에 머리를 짜내야만 했다. 무엇보다 20초 발표를 해본 적이 없었기 때문 에 슬라이드 한 장에 내용이 얼마나 돼야 하는지 감이 없었다. 미리 준비해서 감이라도 있었으면 하는 생각이 절실했다. 우선 20초라는 시간이 짧을 것으로 예상하고 최대한 내용을 뺐다. 한 슬라이드에는 한 가지 내용만. 글 보다는 그림 위주로. 이정도면 발표때 영어를 떠듬떠듬 말해도 시간이 충분할 것이라고 생각했다.

첫 번째 단계를 위해 8명이 모인 자리로 갔다. 같은 조에 학주 형이 있어서 조금은 안심이 되었다. 곧 8명이 모이고 서로를 평가하기 위한 평가지를 받아 들었다. 종이를 보니 내가 마지막에서 두 번째였다. 긴장감에 심장이 입 밖으로 튀어나올 것 같았다. 첫 번째 국제 학회에서의 발표이기 때문에 더 가슴이 두근거렸다.

이윽고 발표가 시작되었다. 다른 친구들 발표를 들으니 더 긴장이 되었다. 다들 번개 발표를 꽤나 충실히 준비해온 모습이었다. 더군다나 영어가 익숙하다보니 20초라는 시간을 충분히 활용하는 모습이었다. 어느새 학주형의 순서가 됐다. 학주형은 100만라인 C코드 분석을 주제로 발표했다. 같은 주제로 여러 번 발표했던 만큼 학주형의 발표는 순조롭게 끝났다.

학주형 발표에 이어 두 번의 발표가 이어졌고 내 차례가 돌아왔다. 발표가 시작되고 나니 내가 엄청 말을 빨리 하고 있다는 사실을 깨달았다. 첫 장에서 발표 제목과 공동 연구자의 이름을 언급하고 나니 시간이 꽤나 많이 남게 되었 다. 남아있는 십여초가 영겁의 시간처럼 느껴졌다. 어색함을 어찌하지 못하고 "엄"하는 소리만 내면서 다음장을 하릴없이 기다려야만 했다. 어색한 상황은 뒤의 슬라이드에서도 반복됐다. 20초라는 시간을 내가 너무 안일하게 생각했구나 하는 마음에 속이 탔다. 어색한 미소를 지을 뿐이었다. 속도 모르고 내가 "엄" 소리를 낼 때마다 청중들은 소리내어 웃었다.

발표가 다 끝나고 서로의 점수를 매겼다. 나는 차마 내 발표에 점수를 줄 염치가 없어서 다른 네 발표에 점수를 주었다. 점수를 합산한 결과 뜻밖에도 내가 4명 중 한 명으로 뽑혔다. 이름이 호명되는 순간 기쁨보다는 걱정이 앞섰다. 이런 긴장을 또 한 번, 그것도 본 무대에서 겪어야 한다니. 너무 긴장돼서 멀미가 날 지경이었다. 발표 장소를 박차고 나와 리셉션하는 곳에서 맥주를 한 병 마셨다. 기분탓이었겠지만 긴장이 조금 가시는 느낌이었다.

두 번째 단계가 곧이어 시작되었다. 본 학회때 만큼 사람들이 많이 모여있지는 않았지만 그래도 백 여명은 족히 앉아있는 것처럼 보였다. 학회장에서 눈에 띄는 한 사람은 Andreas Podelski였다. 이행 불변식 (transition invariant) 을 사용한 종료 분석을 시작한 분이다. 다른 사람은 몰라도 이 분은 내 발표를 들어주었으면 했다. 그래서 이전 단계에서와 같은 어색한

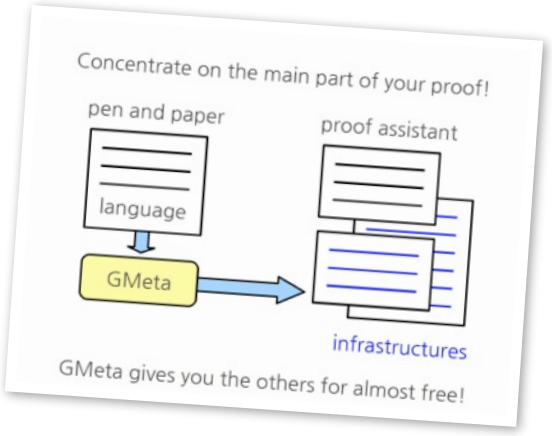
상황은 최대한 피해야겠다고 생각했다. 다른 사 람들이 발표하는 동안 발표 대본을 정비했다. 아까 시간이 남았던 슬라이드에는 말을 좀 더 집어넣었다.

곧 내 차례가 돌아왔고 발표를 시작했다. 처음부터 곤경에 처했다. 내가 사용한 학습 알고리즘이 비통계적 학습 알고리즘이라는 말을 한 것이 화근이었다. 학습이라는 단어에 많은 사람들이 헛갈려하는 것 같아서 괜히 집어넣어본 말 이었다. 학습 알고리즘에 대해 짧게 말하고 나니 공동연구자 이름을 말하기도 전에 다음 슬라이드로 넘어가버리는 것이 아닌가. 아찔했다. 그 뒤로도 몇 초간 멍하게 있었고 사람들은 웃기 시작했다. 정신을 차리고 발표를 재개했다. 중간에 내가 "우리는 이행 불변식을 사용한 종료분석에 관심이 있습니다"라는 말을 하는 순간 Podelski가 안경 너머로 나를 바라보는 것 같은 모습이 보였다. 아무래도 슬라이드에 적힌 Podelski et al.을 확인한 것 같다. 용기를 얻어 무사히 마무리했다.

발표를 마치고 자리에 돌아오니 Podelski가 내게 다가왔다. 발표에 대해 좀 더 듣고 싶으니 논문을 보내달라고. 그리고 시간내서 미팅을 갖자고. 그 말을 건네고 학회장을 표표히 빠져 나갔다. 기뻤다. 내 발표에 관심을 가져주어 고마운 마음이 들었다. 나중에 Podelski를 따로 만나 아이디어를 좀 더 잘 설명해주었다. 내 이야기를 듣고는 학습 알고리즘이라는 것이 있는지 몰랐다며 재밌어 했다. 제출한 논문 잘 되길 바란다는 말도 잊지 않고 해주었다. 아쉽게도 길게 이야기 나누지는 못했지만 내 연구를 남에게 알릴 수 있다는 사실이 기뻤다. 다음에는 꼭 본 학회에서 연구 성과를 발표할 수 있으면 좋겠다고 생각했다.

조성근

지난 POPL에서 시작한 학생들을 위한 번개발표 세션이 올해에도 있었다.



작년에 반응이 좋아 올해에도 하게 되었다고 한다. 국제 발표 경험이 없는 학생들을 위해서는 발표 연습을 위한 좋은 기회였다. 발표자 수와 청중의 수를 생각하면 작년에 비해 올해에 더욱 성공적이었기 때문에 내년에 또 하지 않을까? 생각한다.

나는 증명을 할 때에 변수묶임(variable binding)과 관련하여 귀찮게 등장하는 정의와 증명들(infrastructure)을 만들어주는 GMeta에 대해 발표하였다. GMeta의 핵심은 자료타입에 구애받지 않는 프로그래밍(Datatype Generic Programming, DGP)이다. GMeta는 프로그래밍 언어의 문법을 정의할 수 있는 좀 더 일반적인(generic) 문법을 포함하고 있고, 이 문법으로 표현할 수 있는 프로그램 언어에 적용가능한 일반적인 함수나 정리들이 정의/증명되어 있다. 따라서 일반적인 문법을 이용하여 특정 언어를 정의할 때, 미리 정의/증명되어 있던 일반적인 함수와 정리들도 함께 구체화된다. 이러한 구체화는 모듈 프로그래밍(Modular Program-

ming)을 통하여 이루어지게 되며, 이때에 locally nameless나 de Bruijn과 같은 여러 변수묶임 방법 또한 모듈인자로 받아 선택할 수 있게 된다.

슬라이드가 12장으로 제한되었고, 각 장당 20초의 시간이 강제 되었으며, 슬라이드에 수식이나 문장은 허용되지 않았다. 처음에는 이러한 제한이 이상하게만 느껴졌지만 특이한 발표 덕분에 얻은 것이 있다.

20초의 시간동안 한 페이지의 내용을 설명하려 하니 대본의 문장이 점점 간결해지는 것을 느낄 수 있었다. 최종적으로 이해를 위해 꼭 필요하다고 생각되는 문장들만이 남았다. 그동안 나의 발표에서 얼마나 쓸데없는 말을 많이 하고 있었는지 느낄 수 있었다. 슬라이드 한 장당 20초라는 황당한 규칙을 넣은 것이 번개발표의 시간초과를 걱정한 것이 아닌 발표연습을 위한 것이라는 생각이 들었다.

전하고자 하는 내용을 짧은 어구나 그림으로 표현하는 것이 효과적이라는 것도 다시 한 번 느낄 수 있었다. 슬라이드에 수식이나 문장이 허용되지 않았기에 대부분의 내용을 그림으로 그려야 했다. 그림을 그리면서 내가 한 일이 무엇인가에 대한 직관이 더욱 또렷해지는 것을 느낄 수 있었다.

김진영

지난 1년간 연구했던 ScanDal을 주제로 학생 번개톡에 참여하였다. 비록 정식 발표는 아니지만 그래도 저명한 국제학회에서 내 이름을 걸고 발표한다는 사실이 굉장히 떨렸다. 슬라이드 12장을 가지고 한 슬라이드당 20초씩 4분간 발표하는 것이다.

1년간 한 연구를 4분에 발표하려니 준비가 여간 어려운 것이 아니었다. 도대체 어떤 내용을 넣어야 할지, 그리고 그 내용을 짧은 시간에 강렬한 인상으로 전달할 수 있는 슬라이드를 구성하는 것이 너무 어러웠다. 결과적으로는 대부분의 슬라이드를 단어 하나, 또는 그림 하나를 넣는 식으로 구성하였다. 나는 나름대로 열심히 고민하여 만든 것인데 다 만들어 놓고 보니 어쩐지 성의가 부족해 보이는 것 같아 좀 찝찝한 느낌도 들었지만..

떨리는 발표 시간이 다가왔고 8명이 한 조가 되어 예선을 펼쳤다. 이 중 참가자들의 투표를 통해 4명을 선발하여 전체 청중 앞에서 발표하는 것이라고 하였다. 학생들의 발표는 참 다양했다. 한 중국 학생은 20초마다 다음 슬라이드로 넘어간다는 규칙을 제대로 이해하지 못해 발표를 완전히 망쳤다. 한 프랑스인은 슬라이드 12장 전부 깨알같이 글씨를 써 온 뒤 이를 20초간 읽기도 했다. 하지만 이 둘을 제외하고는 다들 아주 열심히 준비해 온 모습이었다. 특히 미국의 한 학생은 20초를 완벽히 구성하여 정확히 문장이 끝나기 무섭게 다음 슬라이드로 넘어가는 철저한 모습을 보여주기도 했다. 나는 다소 긴장한 나머지 말을 굉장히 빨리 해 버리는 실수를 했는데, 덕분에 할 말을 너무 일찍 해 버려서 다음 슬라이드로 넘어갈 때까지 굉장히 어색한 시간을 보내야 했다. 지금 생각해 보면 고작 8명의 생판 모르는, 아마도 평생 다

시는 못 볼 가능성이 높은 학생들 앞에서 4분 발표하고 마는 것인데 뭐가 그리 긴장되었던 것인지 모르겠다.

나는 아쉽게도 4명 안에 들지 못했고, 모여서 다른 학생들의 발표를 구경하였다. 우리 중에서는 원찬이형이 홀로 결선에 진출하였다. 아주 다채로운 16개의 발표를 듣는 재미가 상당했다. 거의 개그 쇼에 가까운 슬라이드를 구성해 온 재간둥이도 있었고, 시종일관 긴장한 표정이 역력했지만 준비해 온 할 말은 또 다 조곤조곤 확실히 전달하는 소심하지만 실속있는 발표도 있었다. 결과적으로는 우리 조의 그 철저히 계산된 발표를 보여 준 학생이 우승하고 끝났다.

최준원

POPL 에서는 작년부터 연구 중인 학생을 대상으로 자신이 연구하고 있는 내용을 짧게 설명할 수 있는 번개 세션을 마련하였다. 작년의 성원에 힘입어 올해도 번개 세션을 진행하였고 연구실의 많은 사람들이 세션 발표를 위한 준비를 하였다. 올해 번개 세션에는 pecha-kucka 라는 발표 방식을 도입하였다. 12 슬라이드에 한 슬라이드당 20초의 고정 시간을 부여하여 4분에 발표를 끝내는 방식으로, 간단하고 명확한 전달이 중요한 요소가 된다. 작년에도 있었는지는 모르겠지만 올해 세션에는 예선이 있었고 나는 안타깝게도 결선에 진출하지 못하였다. 하지만 처음으로, 비록 짧막하지만 현재 연구에 대한 영어 발표를 준비하면서 많은 것을 느끼고 배웠다. 가장 크게 든 생각은 영어의 벽은 역시 높다는 것이었다. 단순한 영어 공부가 아닌 실제로 발표하고 질문에 답하고 많은 사람들과 대화해 봐야겠다는 생각이 들었다. 또 다른 생각은 연구에 대한 발표 준비를 하면 자신의 연구를 신중하게 돌이켜 볼 수 있다는 것이다. 이번 발표는 그 자체로 내가 하고 있는 연구 내용을 정리할 수 있었다는 점에서 의미가 컸다.

POPL 2012

오학주

POPL은 프로그래밍 언어 분야 최고 학회중 하나로써 가장 최신, 최고의 연구결과들이 발표되는 곳이다.

역시 POPL은 여러모로 최고의 학회 다웠다. 발표된 논문들도 하나같이 훌륭했고 유익했다. 정적분석 세션이 없었음에도 불구하고 제작년, 작년보다 흥미로운 논문이 많았고 사람들이 발표도 더 잘 하는것 같았다. 또한 올해는 참가인원이 아주 많았다. 작년, 제작년 인원의 배에 가까운 사람이 참석하여 학회장 앞의 홀은 쉬는시간마다 사람들로 발디딜틈없이 가득찼다. 들리는 말에 의하면 미동부의 대학원생들이 대거 참여하였기 때문이라 한다. 이번이 세번째 POPL참석인데, 이번 POPL의 프로그램들이 가장 유익했던 것 같다. 논문들도 그렇지만, 학생발표(student lightning talks)등의 다른 세션들도 더 체계화되고 눈에 띄게 진행되었다. 특히, 나는 학생발표에 참여했었는데, 발표도 재미있었고 다른 학교 학생들도 알게되는등 여러모로 유익했다.

여러 다른 워크샵들도 많이 개최되었다. 타입의 대가들이 주도하는 TLDI를 비롯해서, 요즘 점점 눈에 띄는 VSTTE, 전통적으로 좋은 논문들이 발표되는 워크샵인 PEPM 등등 많았다. 이들 워크샵들도 관심이 많이 갔지만, VMCAI와 시간이 겹치고, POPL이 끝난후에 열리는 것들이 많아 별로 참석하지 못해서 아쉬웠다. 또한 Tutorial Fest라고 해서 각 분야의 연구내용을 자세하게 튜토리얼식으로 진행하는 시간도 있었다. 우리는 비행기시간때문에 참석하지 못했는데, Byron Cook의 종료분석, Peter O'Hearn의 분리논리 등 좋은 튜토리얼들이 많았다.

흥미로운 발표가 많았다. 처음엔 프로그램을 보고 정적 분석 세션이 아예 없는 것을 보고 실망했다. 대신 semantics, type theory 등 고전적인 프로그래밍 언어 주제들이 많은 것을 보고 지레 못알아듣는 발표가 많을것이라 생각했다. 그런데 막상 프로시딩을 보니 정적 분석 논문이 각 세션별로 많았을 뿐 아니라, 분야가 달라도 재미있는 것들이 많았다.

POPL 포스터는 시청 건물 옆의 LOVE 마크를 닮았다. (사진 제공: 이승중)

SIGPLAN Distinguished Achievement Award Presentation and Interview

Tony Hoare

강지훈

Tony Hoare의 마이크 상태가 좋지 않아 말을 잘 들을 수가 없었지만, 학계 초창기부터 학계와 함께 성장해온 한 노회한 대학자의 면모를 살펴볼 수 있었던 강연이었다. 중간 중간 손으로 쓴 것 같은 원고를 책상에서 들춰보고 하던 모습은 약간 감동적이기까지 했다.

평생 소프트웨어 검증에 더할 나위 없이 기여해온 Hoare는, 자신이 왜 이 분야에서 연구하고 있는지 담담하게 설명했다. 회사에서 학교로 올 때 소프트웨어 검증을 하게 된 이유는 회사는 이 분야와 관계가 없으리라 생각했기 때문이라고 반 농담으로 밝히면서, 지금 다시 MSR에서 실제로 소프트웨어 검증이 현장에서 쓰이는 것을 보고 있노라고 담담히 말하는 모습도 감동적이었다.

세션을 진행했던 Peter O'Hearn가 Tony Hoare의 작업 방식을 극찬하면서, 다른 사람들이 이제 일을 마무리하고 학회에 제출할만한 즈음에 Hoare는 다시 그 일을 붙들고, 설명을 간단히 하고, 간단히 하고, 또 간단히 해서, 누가 봐도 자명하게 보일 때까지 다듬는다는 이야기를 들려주었다. 평소 교수님께서 하시던 말씀이기도 하거니와, 1970년대의 Hoare의 논문을 보면서 느꼈던 명쾌함이 다시 생각나 다시금 감동을 느낄 수 있었다.

학자로서의 태도, 일을 하는 자세, 학계를 이끌어온 선배들의 모습을 담담하게 배울 수 있었던 그야말로 감동 그 자체인 강연이었다.

Underspecified harnesses and interleaved bugs

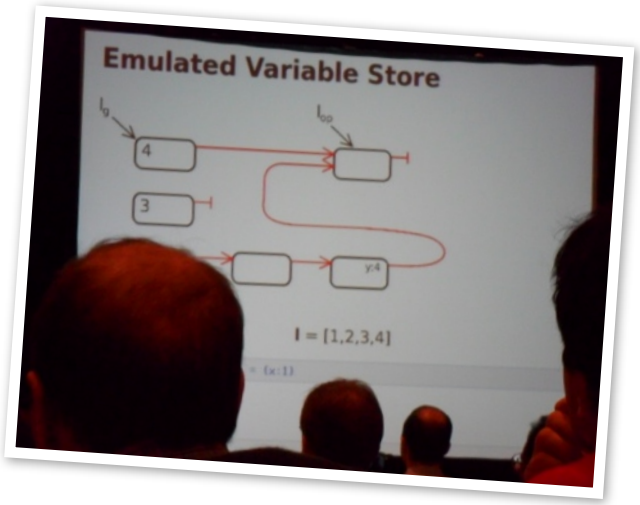
Saurabh Joshi, Shuvendu K. Lahiri, and Akash Lal

오학주

문제가 흥미로웠다. 라이브러리 등의 오픈 프로그램 (main함수가 없고 API 셋만 제공하는)들을 분석할 때는 일종의 함수를 부르는 방법(harness)이 주어져야 한다. 그런데 이를 사람이 일일히 손으로 제공하는 것은 어렵고 오류를 만들기 쉽다. 예를들어, read 함수를 부르기 전에는 항상 파일핸들이 초기화되어 있어야 한다는 등의 규칙을 지키면서 함수를 호출해야 한다. harness가 제대로 주어져 있지 않은 상황(underspecified harness)에서 정적분석을 시도하면 이로인한 허위경보가 발생한다. 즉, 파일핸들을 초기화하는 루틴을 부르지 않고 바로 read함수를 부르는 harness를 바탕으로 분석하면 read함수에서 경보가 발생하지만 이는 상황을 고려하지 않았기 때문에 생기는 허위경보이다. 이를 줄이는 방법이 이 논문의 내용이다. 스레드를 사용하는 프로그램에 대해서 적용가능한 방법을 제시했는데, 각 스레드를 독립적으로 분석했을때는 알람이 나지 않지만, 스레드를 고려했을때 발생하는 오류들(interleaved bugs라고 한다) 을 이용해서 underspecified harness로 인해 발생하는 버그를 줄이는 아이디어를 제시했다. 문제와 해결방법이 흥미롭다.

Towards a Program Logic for JavaScript

Mikolaj Bojańczyk, Laurent Braud, Bartek Klin, and Slawomir Lasota
조성근



이 발표는 상당히 인상적이었는데 그 이유는 발표 슬라이드가 자바스크립트로 되어 있었기 때문이다. 지금까지 본 학회에서의 모든 발표 슬라이드는 비머, 키노트, 파워포인트로 만들어진 것이었다. 이들을 사용하면 아름다운 수식을 표현하거나 직관적인 그림을 넣는 일이 수월하기 때문이다. 자바스크립트로 슬라이드를 만들어 웹브라우저로 발표하는 것이 어떤 이득이 있을까? 자바스크립트로 만들어진 발표는 대화형(interactive)일 수 있다는 점을 들 수 있다. 발표자는 발표 도중 자바스크립트의 직관적이지 않은 동작들을 설명해야 했는데 이 때 직접 자바스크립트를 작성하여 실행하는 과정을 보여주었다. 물론 예를 보여줄 때에만 브라우저를 띄워 보여줄 수도 있었겠지만, 하나의 자바스크립트 페이지 안에서 모든 것을 이야기하는 것이 발표에 집중하는데 도움을 주었다.

자바스크립트로 작성된 프로그램 의미를 설명하는 것은 어려운데 그 이유 중 하나가 변수의 범위(scope)를 설명하기가 어렵다는 점이다. 자바스크립트에서 변수의 범위는 범위객체(scope object)의 사슬(scope chain)들로 표현될 수 있다. 위의 그림은 저자가 예를 들고 있는 자바스크립트에서의 값 저장공간이다. 각 상자 안에 변수와 그게 묶여있는(binding) 값이 저장되어 있고 이를 범위객체라 부른다. 범위객체들이 사슬처럼 연결되어 있고 이러한 사슬들이 스택에 쌓여있는 구조이다. 따라서 특정 변수의 값을 구하기 위해서는 스택의 위에 쌓여있는 사슬부터 차례로 찾아보면 된다. 문제는 이러한 사슬을 프로그래머 마음대로 수정할 수 있다는 점에 있다. 그림에서처럼 스택에서 서로 떨어진 사슬들이 연결될 수 있어 변수의 범위는 복잡해진다.

이 연구팀은 자바스크립트 프로그램에서의 변수의 범위를 설명하기 위하여 분리논리를 확장하였다. 위에서 이야기한 것처럼 범위객체의 사슬들이 공유하는 사슬이 있기 때문에 단순한 분리논리만으로 이를 설명하는 것은 어렵다. 이를 위하여 분리논리에 부분적인 분리를 의미하는 sepish connective(P ⋈ Q)를 추가하였다. 의미는 P와 Q를 각각 만족하는 메모리 영역이 있고 이 두 메모리 영역이 일정 메모리 영역을 공유한다는 것이다. 식으로 표현하면 다음과 같다.

$$h, L, \epsilon \models P \bowtie Q \iff \exists h_1, h_2, h_3.$$
$$h \equiv h_1 * h_2 * h_3 \wedge (h_1 * h_3, L, \epsilon \models P) \wedge (h_2 * h_3, L, \epsilon \models Q)$$

이것만으로 훌륭한 연구라 생각되었지만 이들의 연구는 이것이 끝이 아니었다. 이들은 while과 eval을 제외한 자바스크립트 프로그램을 입력으로 받아 pre-condition과 post-condition을 추론하는 규칙을 제안하였고 이것이 안전하다는 것을 증명하였다. while과 eval이 제외되었기 때문에 모든 자바스크립트 프로그램에 대해 적용가능하지는 않지만 프로그램의 의미를 기술하는 데에 부분적인 자동화가 가능하다는 점에 의미가 있다고 생각한다. 하나의 문제를 해결하기 위하여 노력하고 끝장을 본다는 것이 어떤 것인지 생각해 볼 수 있었다.

An Abstract Interpretation Framework for Termination

Patrick Cousot, and Radia Cousot

오학주

Cousot 교수님이 기존의 모든 종료분석(termination analysis)를 아우르는 요약해석 기법의 이론적인 틀을 발표하셨다. Cousot교수님의 전형적인 이론논문이다. 즉, 기존에 알려진 분석이 알고보니 일종의 요약해석이다라는 메시지다. 발표는 늘 하시던대로 슬라이드를 화려한 폰트의 수식으로 가득 채우시고는 스피디하게 발표를 진행하셔서 대략 슬라이드 3쪽부터 내용보다는 발표하시는 모습을 구경하는 양상으로 갔다. 한참, 빽빽한 설명을 마치신 후 결론부분에서 하신 말씀이 인상적이다. 지금까지는 아주 간단하고 직관적인 큰그림 위주로 설명하였으니 자세한 내용은 논문을 읽으라는 것이었다. 읽어보려고 했지만 짧은시간에 읽을 엄두가 나지 않는 논문이어서 포기했다.

A Language for Automatically Eonforcing Privacy Policies

Jean Yang, Kuat Yessenov, Armando Solar-Lezama

오학주

프로그래밍을 할 때부터 privacy policy를 지정할 수 있도록 하는 언어 설계에 대한 논문이었다. 핵심은 프로그래머가 프로그램의 기능에 대한 부분과 보안기능에 대한 부분을 따로 고려하면서 프로그램을 작성할수 있도록 최대한 언어설계에 고려했다는것 같았다. 발표자인 Jean Yang은 중국계 여학생이었는데, 예전부터 연구를 아주 잘한다고 들었었다. 발표도 아주 잘했고, 훌륭한 일을 체계적으로 했다는 느낌을 받았다. 하지만, 새로운 언어를 정의했는데 과연 이 언어가, 혹은 그 중 일부기능을 실제로 쓸수 있게 되기까지 얼마나 걸릴지, 과연 쓰게는 될지에 대한 의문이 들었다.

Higher-Order Functional Reactive Programming in Bounded Space

Neelakantan R. Krishnaswami, Nick Benton and Jan Hoffmann

최준원

Reactive Programming 에 대한 정의를 정확히 몰랐던 상태에서 발표를 들어 조금 생소하였지만 lazy evaluation 에 의해 값이 생긴다는 것을 명시적으로 지정해 주겠다는 아이디어를 인상 깊게 들었다. FRP(Functional Reactive Programming) 에서는 무한대로 길이가 늘어날 수 있는 스트림을 생성할 수 있고 따라서 메모리 관리가 중요한 이슈 중 하나이다. 기본적으로 이 스트림에 의해 메모리 누수가 생길 가능성이 가장 큰데 이를 한정할 수 있는 syntax, type, semantics 를 구현했다는 것이 발표의 핵심 내용이었다.

논문에서는 기존 lambda calculus 의 타입 시스템에 크게 세 가지를 추가하였다. ●(modality) 는 다음 clock tick 에서 값이 생성될 것이라는 것을 의미하는데 스트림의 길이가 하나씩 늘어나는 것으로 생각할 수 있다. ◇는 새 스트림을 생성함을 알려주고 R→A 는 리소스 R 에서 A 가 생성되었음을 나타낸다. FRP 에서 메모리 누수가 생길 수 있다는 기초적인 사실을 알았고, 깔끔한 해결책도 알 수 있었던 좋은 발표였다.

The Marriage of Bisimulations and Kripke Logical Relations

Chung-Kil Hur, Derek Dreyer, Georg Neis, and Viktor Vafeiadis

이원찬

프로그램 동등성 증명에 대한 허충길 박사님과 MPI-SWS팀의 최근 연구 성과다. 이번 논문에 발표된 증명 기법은 그동안 한 기법에서 동시에 지원되지 않던 세 가지를 모두 지원한다. 1) 서로 다른 언어로 작성된 두 프로그램의 동등성 증명. 2) 조립 가능한 증명. 3) 징검다리 증명 (transitive proof). 증명이 조립 가능하면 A과 A'가 동등함을 증명하고 B와 B'이 서로 동등함을 증명하기만 해도 A+B와 A'+B'도 저절로 동등해진다. 증명이 징검다리라는 말은 프로그램 A와 B가 동등함을 증명하고 B와 C가 동등함을 증명하면 A와 C는 저절로 동등해지는 것을 뜻한다. 기존의 증명 기법들에서는 같은 언어로 작성된 프로그램만 지원하거나 닫힌 프로그램만 지원하는 등 제약을 가했다. 하지만 새 증명 기법은 우리가 가지고 있는 프로그램에 대해서만 동등함을 보이는 것에 집중함으로써 제약들에서 벗어날 수 있었다고 한다. 이는 우리가 가지지 않은 프로그램에서 일어나는 일들이 무엇이 되든 상관없이 두 프로그램이 똑같이 동작해야 한다는 것을 요구한다.

새 증명 기법을 통해 유용한 일들이 가능해지는 데, 대표적인 예가 실제적인 컴파일러의 검증이라고 한다. 컴파일러가 올바르다는 것은 소스 프로그램과 컴파일러가 만들어 내는 실행파일이 같은 일을 한다는 것이다. 실제적인 컴파일러에 대해서는 이같은 검증이 간단하지 않다. 우선, 실제 컴파일러들은 소스 프로그램을 실행파일로 변환할 때 중간 언어로의 변환을 여러 번 거친다. 게다가 컴파일러는 전체를 한 번에 컴파일하지 않고 부분으로 나누어 컴파일 한 다음 서로를 연결(linking)한다. 이 말은 서로 다른 두 컴파일러가 있어서 그 둘이 만들어내는 라이브러리를 나중에 연결할 수도 있다는 뜻이 된다. 이런 상황에 모두 대처하려면 다음 셋이 필요하다:

1) 중간 언어로 된 프로그램과 소스 프로그램 사이의 동등성을 보일 수 있어야 한다.

2) 각 컴파일 단계의 변환이 프로그램 동등성을 보존한다는 것만 증명해도 소스 프로그램과 생성된 실행 파일이 동등해져야 한다.
3) 올바르게 컴파일된 각 실행 파일을 연결해도 올바로 컴파일된 실행 파일이 돼야 한다. 실제로, 잘 알려진 CompCert 컴파일러도 3번을 제대로 지원하지 못하고 있다고 한다. 허나, 새 증명 기법은 이 셋을 모두 지원하기 때문에 이를 통해 실제적인 컴파일러의 검증이 가능해 질 것이다.

Multiple Facets for Dynamic Information Flow

Thomas H. Austin and Cormac Flanagan

이영석

이 연구에서는 faceted value 라는 것을 도입하여 information flow controls 를 웹브라우저에서 구현하였다. 이 발표는 도입부부터 청중들의 시선을 확 잡는 예시로 시작하였다. 페이스북, 트위터의 예를 들면서 많은 보안 문제와 자바스크립트의 문제점을 이야기 하면서 청중들의 흥미를 유도하였다. 저절로 관심이 가고 귀 기울여 듣게 되는 도입부였다. 실행을 누가 하냐에 따라서 볼 수 있는 정보가 달라야 하고 그것을 런타임 시에 해결 할 수 있는 calculus를 만든 것이 이 발표의 주된 내용이다. 결론적으로 Cross-Site Scripting 이라는 기술을 이용하여 비밀번호를 유출하는 공격을 이 faceted value를 도입하였을 때에는 잘 방어를 할 수 가 있었다. 대략적인 흐름을 쉬운 은행의 예를 들면서 설명을 하여서 듣기 편한 발표였다. 아이디어가 굉장히 직관적이어서 좋았다. 잘 모르는 상태에서 발표를 듣고 했던 생각은 아이디어 하나로 POPL에서 발표를 한다는 느낌이 들었다. 하지만 발표를 들은 이후에 논문을 차근차근 읽어보니 생각보다 간단한 내용이 아니었다. 복잡한 내용을 간단하게 설명을 한 발표자의 능력에 다시 한 번 감탄하였다. 또 이 발표를 듣고 나도 언젠가 저런 무대에서 멋있게 발표를 하고 싶다는 생각이 들었다.

Defining Code-Injection Attacks

Donald Ray and Jay Ligatti

윤용호

Code-injection 중에서도 CIAO(code-injection attacks on outputs)를 정의하는 새로운 방법을 제시하였다. CIAO는 사용자의 입력으로부터 만들어진 출력을 이용해 취약점을 공격하는 방식이다. 이 공격을 판정하는 기존의 방법들은 공격을 탐지하지 못 하거나, 공격이 아닌 것을 공격으로 잘못 판단하는 경우가 있었다. 이 연구로는 기존의 허위 경보를 내던 케이스나 경보를 못 내던 경우를 제대로 판단한다고 한다.
기존의 SqlCheck, Candid 등을 비교 대상으로 하였다. SqlCheck은 입력이 파싱하였을 때 온전한 파스 트리를 구성하는가를 판단 기준으로 하였다. 그러나 이 경우 exit()이나 1000>GLOBAL 등을 입력으로 하는 공격을 공격으로 판단하지 못하는 문제가 있었다. 또한 WHERE filename='f.e'에서의 f.e와 같이 적법한 입력에 대해서 공격이라고 판단하기도 한다. Candid는 사용자의 입력과, 그로부터 만들어진 출력이 파싱하였을 때 동일한 구조를 갖는지

를 기준으로 하여, 둘의 구조가 다르다면 공격으로 판단한다. 그러나 이 방법 역시 몇몇 입력에 대해 공격이 아닌 입력을 공격으로 판단하는 문제가 있다.
이 연구에서는 CIAO를 두 단계로 나누어 정의한다. 먼저 출력 중 일부가 코드인가 아닌가를 정의하고, 그 코드가 삽입(injection)된 것인지를 판단한다. 기존의 연구들은 이 둘을 따로 구분하지 않고 뭉뚱그려 판단하였다. 코드는 먼저 코드가 아닌 것이 무엇인지 정의한 후, 그 반대를 코드로 정의한다. 코드가 아닌 것은 더 이상 값매김 될 수 없는 부분, 즉 값으로 정의한다. 람다식의 normal form에 해당한다. 주어진 임의의 문자열에 대해 그 문자열이 값인지 아닌지를 판단하는 NCV 진리 함수를 엄밀히 정의하고 Code는 ¬NCV로 정의하였다. 삽입된 문자열인지는 taint 분석을 활용하였다. 사용자의 입력에 taint 표시를 한 후, 모든 문자열 연산에서 taint를 전파시켜 최종 결과에서 taint가 남아있는 곳이 삽입된 문자열로 정의한다.

Verification of Parameterized Concurrent Programs By Modular Reasoning about Data and Control

Azadeh Farzan and Zachary Kincaid

오학주

내 연구주제인 sparse analysis를 활용한 논문이어서 관심있게 들었다. 문제는 sparse analysis자체보다도 이를 이용하여 스레드의 개수를 예측할 수 없는 프로그램의 분석을 어떻게 잘 할 것인가였다. 이 논문에서 직접적으로 언급하지는 않곤있지만, 이 문제를 핵심적으로 이용한것이 우리가 말하는 sparse analysis였다. 이 부분에서 우리가 이번에 PLDI 2012에 채택된 논문의 내용과 상당히 유사한 점이 많았다.

이 논문에서는 sparse analysis라는 용어를 직접 사용하고 있지는 않지만 data flow graph라는, 우리 표현으로는 data dependency를 정의하고 그 위에서 요약해석을 수행한다. data dependency를 정의하고, 그것의 올바름을 보인 점, 일반적인 요약해석의 틀 (여전히 dataflow analysis 스타일로 정의하기는 했지만) 위에서 설명한 점, sparse analysis를 관계를 고려하지 않는 값분석(non-relational analysis)에만 적용한게 아니라 변수그룹(variable packing)을 이용하여 관계분석(relational analysis)에도 적용한 점이 우리 논문과 매우 유사했다. 그래서 이 발표를 들으면서 우리끼리 이번 PLDI에서 떨어지면 이 논문을 어떻게 대응해야 하나 걱정하기도 했다.

발표가 끝나고 발표자인 Zachary Kincaid에게 이것저것 물어보았다. 다행히 전날에 학생발표에서 같은 조에 속해있어서 안면이 있는 사이였다. Zachary Kincaid는 토론토 대학의 박사과정 학생이었는데, 학생발표때 내 발표를 듣고는 여러가지 관심을 보였었다. 학생발표는 워낙 간결해서 이 정도로 비슷한 내용일지는 몰랐기 때문에 왜 그런가 싶었는데, 이 논문을 보니 그 이유를 알것 같았다. 쉬는시간에 sparse analysis를 어떤 세팅에서 고려하는것인지, 관계분석은 어떤식으로 하는지 등을 물어보았다.

얘기를 해보니 우리논문과 전반적으로 비슷하지만 결정적으로 다른점이 있었다. 우리논문의 핵심은 포인터가 있을때, 그래서 data dependency를 미리 알 수 없고 어림잡은 정보를 사용할때도 정확도를 잃지 않고 sparse analysis를 할 수 있다는 것인데, 이 논문은 포인터가 없다고 가정하고 있었다. 이렇게 간단한 세팅이면 사실 정확도가 갖다는 것을 증명할 것도 없는 세팅이다. 우리는 일반적인 세팅에서 정확도를 보존한다고 했더니, 매우 흥미로워했다. 어쨋든, 생각지도 못한 논문에서 비슷한 내용을 발견하고, 또 학생발표도 우연히 같은 조였다니 신기하고도 재밌는 경험이었다.

Canonicity for 2-Dimensional Type Theory

Daniel R. Licata and Robert Harper
강지훈

POPL에서 발표되는 타입 이론은 수업에서 배운 타입 이론의 수준보다 훨씬 높아서 들어도 잘 이해가 잘 안된다고 선배들이 충고해주었는데, 역시 타입에 관한 6개의 논문 중에서 단 하나만 조금이라도 이해할 수 있었다. 이 논문이 바로 유일하게 조금이라도 이해한 논문이다.

이 논문은 두 타입의 동등함에 대해 다룬다. 예를 들어, list X 라는 X의 list를 나타내는 타입과, forall n, {0, 1, ..., n-1} -> X 은 동등하다. (후자는 자연수 n에 대해, 0, 1, ..., (n-1)을 받으면 X를 내놓는 의존 타입dependent type을 의미한다.) 왜냐하면 X의 list의 길이와 n을 대응시키면 자연스럽게 isomorphism이 있기 때문이다.

이와같이 두 타입이 동등하다는 이야기는 보통 언어 내에서 표현하기는 힘들고 메타-언어 수준에서 다룰 수 있는 이야기이다. 하지만, 메타-언어 수준에서 두 타입이 동등함을 보인다고 해서 대상 언어에서 두 타입 사이를 자유롭게 오갈 수 있는 것은 아니다. 예를 들어 f: X -> Y가 있을 때, l: list X의 각 원소에 f를 적용하여 list Y를 얻는 함수 map: (X -> Y) -> list X -> list Y를 생각해보자. list X와 forall n, {0, 1, ..., n-1} -> X는 동등하기 때문에, 사실 map만 가지고 map': (X -> Y) -> (forall n, {0, 1, ..., n-1} -> X) -> (forall n, {0, 1, ..., n-1} -> Y)를 상상할 수 있다. 하지만 기존의 타입 이론만으로는 map만 가지고 map'을 손쉽게 이끌어낼 수 없다.

동등한 타입 사이에 변환이 부자연스러운 문제를 해결하기 위해, 타입이 동등하다는 이야기를 대상 언어에서 표현하고 이를 타입 이론 안에서 설명하기 위해 동등 타입 Id_(A, B)를 고려한다. 즉 두 타입 A, B가 동등하다는 것은 표현식 e: Id_(A, B)가 있다는 것이다.

이제 list X와 forall n, {0, 1, ..., n-1} -> X가 어떻게 동등한지 그 방법에 대해 생각해보자. 즉 표현식 e: Id_(A, B)에 대해 생각해보자. list X를 원소가 나타난 순서대로 배열해서 두번째 타입의 원소를 얻어서 두 타입의 동등함을 보일 수도 있을 것이고, 역순으로 배열해서 두번째 타입의 원소를 얻어서 두 타입의 동등함을 보일 수도 있을 것이다. 즉 list X와 forall n, {0, 1, ..., n-1} -> X는 여러가지 방법으로 동등할 수가 있다. 즉 동등함의 방법끼리 동등하지 않을 수도 있다!

이 논문에서는 어떤 방법으로 동등한지를 나타내는 Id_(A, B)의 여러 표현식을 분류하여 구별할 수 있는 타입 이론을 만들어냈다. 즉 어떤 Id_(A, B)의 원소들이 A와 B가 동등함을 보이는 방법이 동등한지를 구별해내는 타입 이론을 만들어냈다. 논문 제목의 '2-dimensional'의 의미는 바로 타입의 동등함의 동등함을 대상 언어가 나타낼 수 있음을 의미한다. 타입의 동등

함을 동등함을 동등함을... 동등함을 대상 언어가 나타낼 수 있다면 multi-dimensional type theory가 될 것이다. 논문 제목의 'canonicity'란 타입 이론이 동등함의 동등함을 구별해낼 수 있는 능력을 가지고 있음을 의미한다.

앞으로 연구를 확장하여 canonicity of multi-dimensional type theory를 만들고 싶다고 포부를 밝히던 젊은 발표자의 패기가 발표회장 전체에 퍼지는 것 같았다. 타입이 동등하다는 것을 대상 언어에서 표현한다는 아이디어를 이 논문 발표를 들으며 처음 접해며 나도 약간 흥분했었다.

혹시 이 아이디어를 POPL 2010, Hutchins의 'Pure Subtype Systems'에 적용할 수 있을지 생각해보고 싶다. 이는 서브타입 이론을 집대성하려는 목적을 가진 논문이었는데, 안타깝게도 결정 여부decidibility를 증명하지 못했다. 서브타입 관계 역시 대상 언어로 표현할 수 있도록 설계하면 충분히 일반적이면서도 결정 여부를 확인할 수 있도록 pure subtype systems를 바꿀 수 있지 않을까 생각한다. 지금 하고 있는 일이 마무리되면 이쪽으로 좀더 공부해보고 싶다.

역시 타입 이론은 참 어려운데, 다단계 언어처럼 사람을 끄는 마력이 있는 것 같다. 좀만 생각해보면 될 것 같은데... 좀만 더 하면 될 것 같은데... 하며 유혹하면서도 오래된 분야라서 그런지 좀처럼 새로운 발견을 허락하지 않는 도도한 모습에 이끌린다. 설레이는 나였다.

Abstractions From Tests

Mayur Naik, Hongseok Yang, Ghila Castelnuovo, and Mooly Sagiv

오학주

양홍석 교수님이 직접 발표하셨다. 전공분야이고 또 한국인이어서 친숙한점도 있었겠지만, 개인적으로는 가장 잘 이해한 발표가운데 하나였다. 문제가 무엇인지, 해결방안에 대한 키 아이디어는 무엇인지가 첫 부분에서 명료하게 언급되어서 앞으로 어떤 내용이 전개될지 예측하면서 들을 수 있었기 때문이다.

다루는 문제는 주어진 질의(query)를 증명하기 위해 어떻게 정적분석기를 튜닝할 것인가이다. 정적분석은 프로그램의 실행흐름에 대한 어떤 질의(query)가 올바름을 증명하는 것이 목적이다. 그런데 정적분석에는 항상 요약이 수반되는데, 얼마만큼의 요약을 하느냐에 따라서 그 질의를 증명할수도 있고, 못할 수 도 있다. 증명할수 없다면, 요약을 덜해서 더 자세한 분석으로 튜닝해야한다. 또한 증명할 수 있어도, 그 요약이 그 질의를 증명하는데 최적이 아닐 수 있다. 즉, 더 요약해도 증명가능한데 지나치게 자세한 (따라서 비싼) 분석을 하고 있을수 있는 것이다. 이렇게 주어진 질의별로 최적의 (질의를 증명할 수 있으면서, 너무 오버하지 않는) 요약을 찾는 것은 일반적으로 어려운 문제이다. 이를 제대로 할 수 없기때문에 기존에는 요약을 고정된 어느하나로 정하게 되고 따라서 분석기가 최적으로 동작하기를 기대하기가 어렵다.

이 문제는 정적분석기를 써본 사람이면 누구나 겪는 실질적인 문제기 때문에 흥미로웠다. 스패로우의 경우에도 주어진 프로그램을 어느정도의 자세함으로 분석해야 할지 판단하는 기

능이 없기 때문에 (정확도 조절을 위한 여러 버튼들이 있음에도 불구하고) 거의 항상 고정된 정확도로 프로그램을 분석한다. 그러다보니, 어떤 프로그램들에 대해서는 필요한만큼의 정확도를 얻지 못하거나 (false alarm으로 나타난다) 어떤 경우에는 지나친 분석을 하게되는 경우가 흔하다. 예를들어서, 어떤 프로그램을 분석하는데는 배열의 원소를 일일히 구분할 필요가 없지만, 또 다른경우에는 구분할 필요가 있고.. 등이다.

이 논문에서 제시하는 해결방법은 동적분석을 먼저 수행해서 프로그램의 실제 실행 흐름을 찾고, 이를 활용하는 것이다. 즉, 동적분석결과로부터 최적의 요약이 무엇인지에 대한 힌트를 얻는다. 이 때 재미있는 것은 동적분석으로부터 얻은 결과가 주어진 질의를 증명하기 위한 필요조건(necessary condition)을 보장하도록 프레임웍을 제시했다는 것이다. 다시말해서, 동저분석이 가이드해주는 요약은 최소한 그 질의를 증명하는데 필요한 것이다. 그 이하로 요약하면 절대 그 질의를 증명할 수 없다. 이 방식이 좋은 이유는 두가지 생각해 볼 수 있다. 첫번째는 요약을 결정하는 인자들이 실제 실행(테스팅, 동적분석)으로부터 나온다는 것이다. 모든 실행경로를 고려하는게 아니라, 테스팅이므로 간단하고 효율적으로 인자들을 추론할 수 있다. 두번째는, 필요조건을 추론하기 때문에 애초에 조건을 만족하지 않는 요약들은 배제할 수 있다.

논문의 주 흐름은 두 가지 분석예를 통해서 전개된다. 발표에는 그 중 하나인 thread-escape analysis 하나만 사용되었다. thread-escape analysis란 리소스들을 각 스레드 안에서만 접근가능한것들(thread-local)과 두개 이상의 스레드에서 접근할 수 있는 (global) 것들로 구분하는 분석이다. 이 경우에 여러가지 요약을 이용하여 어떤 경우에 주어진 질의를 증명할 수 있는지 보여졌고, 이를 동적분석으로부터 어떻게 추론하는가가 설명되었다. thread-escape analysis를 수분내로 수행하면서 50%이상의 질의를 증명하는 실험결과가 있었다. 한가지 논문의 결과를 바로 쓰기 어려운 점 하나는 필요조건을 일반적으로 추론하는 방법이 있는것이 아니고, 각 분석별로 디자인 해야 한다는 것이다. 이를 일반적으로 분석이 주어졌을때 어떻게 적용할 수 있는건지는 이해하기 어려웠다.

지난번 아플라스에서 양홍석 교수님을 만났을때, 정적분석과 동적분석을 결합하는 연구를 하신다고 하셨는데, 이 논문을 말씀하신것이었다. 발표도 재미있었고, 유창하고 멋지게 발표하시는 모습이 인상적이었다.

이우석

이 논문의 주제는 대상 문제에 특화된 효율적인 정적 분석을 어떻게 만드느냐에 대한 것이다. 특화된 효율적인 정적 분석이란, 원하는 프로그램 성질의 증명을 위해 필요한 만큼의 요약을 사용하는 정적 분석이다. 방법은 테스팅 결과를 이용하는 것이다. 즉 실제 실행 결과로부터 좋은 요약에 대한 힌트를 얻는 것이다.
개략적인 방법은 다음과 같다. 예컨대 어떤 프로그램 지점에서 한 객체 x가 다른 전역변수를 통해서도 접근할 수 있는지 분석을 통해 알고싶다고 하자. 요약 도메인은 각 객체들을 {L,

E}로 연결하는 사상들의 집합이다. L은 확실히 전역변수로 접근 못한다는 뜻이고, E는 접근 가능할지도 모른다는 뜻이다. 이 때 테스트를 수행하여 실제 실행 결과들을 얻는데, 테스트 결과 모두에서, 해당 프로그램지점에서 x와, x를 가리키는 다른 객체 y가 늘 전역변수로 접근할 수 없다는 사실을 관찰했다고 하자. 그러면 우리는 x와 y를 L로 요약해야 한다는 조건을 얻는다. 이는 분석을 위한 필요조건이다. 즉, x와 y 둘 다 L로 요약하는 것은 증명을 위해 필요하다. 하나라도 E로 하면 증명이 되지 않는다(만약 y가 E라면, y가 가리키는 x 또한 E가 되어 증명이 안된다). 예로 든 분석은 Thread escape analysis 의 간단한 예로, race detection등에 사용될 수 있다.
논문은 이론적, 실험적 기여를 모두 하고 있다. 분석을 통해 어떤 성질을 증명하기 위한 최소한의 요약을 찾는 문제를 필요조건 문제(necessary condition problem)로 형식적으로 정의하고, 두 가지 예 thread escape analysis와 points-to analysis 두 가지 분석에 대해 이를 찾는 알고리즘을 제시하였다. 실험결과는 자바 벤치마크 프로그램들에 대해 수 분내로 분석하고 약 50%의 질의를 증명하였다.
현재 논문의 방식의 한계는 3가지 정도 인 듯 하다. 우 선 이 방법은 현재 points-to analysis나 thread escape analysis 같이 도메인이 유한하고 비교적 간단한 분석에 대해서 적용 가능하다. 또한 아직 테스팅을 통해 최적의 요약을 구하는 일반적인 프레임워크는 존재하지 않는다(논문은 문제를 수학적으로 정의하고, 두 가지 분석에 대해서만 구체적 알고리즘을 제시한다). 그리고 테스트 coverage가 낮은 경우 잘 안될 수 있다. 한계들 내에서 이 방법을 연구실 분석기 Sparrow에 적절히 적용할 수 있다면 정확도 향상에 도움이 될 것 같다. 하지만 Sparrow는 더 복잡한 값 분석을 하고, 도메인이 무한하여 간단치는 않을 듯 하다.

Meta-level Features in an Industrial-Strength Theorem Prover

J Strother Moore
조성근

"Hoare logic의 Hoare에 이어 이번에는 Boyer-Moore 문자열 탐색 알고리즘의 Moore라니, POPL은 정말 큰 학회이구나!"

이 발표는 ACL2 증명보조기(theorem prover)에 대한 초청강연이었다. 새로운 언어나 증명보조기를 접하는 것은 재미난 일이다. 지난번에 참석한 학회(VMCAI)에서 Agda에 대한 튜토리얼을 듣고 재미있었기 때문에 이번 발표도 꽤 기대되었다.
ACL2는 first-order logic에 기반을 둔 증명보조기이다. 이미 알고 있던 Coq이나 Agda와는 달리 ACL2는 Common Lisp의 문법을 따르고 있었기 때문에 따로 문법을 배울 필요가 없었다. 제목에서 알 수 있듯이 ACL2는 실제 산업에서 사용되고 있다. AMD 마이크로프로세서의 부동소수점 연산자(floating point operations), Rockwell Collins(항공기의 통신장비 제작 업체)의 cryptoprocessor, Centaur (VIA) CPU의 미디어 유닛이 ACL2로 검증되었다고 한다.

ACL2의 다음의 두 가지 특징을 갖는다.

1) 증명 자동화: ACL2 언어는 상당히 원시적이다. 타입도 없고 모든 함수는 전역함수(total function)이어야 하며, 정량자(quantifier)도 사용할 수 없다. 또한 ACL2는 first-order 언어이기 때문에 복잡한 predicate를 표현할 수 없다. 하지만 이러한 단점들 덕분에 자동화가 매우 강력하다. Moore는 증명보조기가 위와 같이 실제 산업에서 사용될 수 있는 이유가 자동화에 있다고 이야기하였다. 슬라이드에 "40년의 자동화 기술"이라는 문구가 기억에 남는다.

2) 사용자의 증명 특화: ACL2에 의한 증명은 사용자에 의해 쉽게 특화될 수 있다. 이것의 핵심은 ACL2가 사용자가 명세나 모델링에 사용할 언어와 같은 언어로 짜여 있다는 점이다. 따라서 사용자는 ACL2이 증명하는 과정에서 사용하는 데이터를 쉽게 가공할 수 있고, 이를 통해 사용자가 원하는 방향으로 증명을 유도할 수 있다고 한다.

Moore는 ACL2의 튜토리얼을 직접 보여주었다. 리스트의 기본 성질에 대한 증명과 정수(integer)의 overflow에 대한 증명들이었다. 모든 증명이 자동으로 이루어졌다. 증명은 간단한 기호실행, 귀납법을 통해 이루어진다. 따라서 ACL2가 증명할 수 있는 간단한 증명은 성공하게 되고 그렇지 않은 복잡한 증명은 실패하게 된다. 복잡한 증명을 성공하기 위해서는 그러한 증명을 위한 sub-lemma를 먼저 증명해야 한다. 그러면 ACL2는 이후의 증명에서 sub-lemma를 활용할 수 있게 된다. 처음에는 이렇게 sub-lemma를 도입하여 자동으로 증명하는 것이 좋지 않다고 생각했다. 어떤 sub-lemma를 도입해야 하는지 직관적이지 않을 수도 있으며 Coq이나 Agda에서는 불필요한 것이라 생각했기 때문이다. 하지만 간단하게 증명 가능한 sub-lemma를 증명하고 이를 이용하여 복잡한 증명을 하는 것은 Coq이나 Agda에서도 흔히 있는 일이다.

발표를 듣고 증명보조기의 선택은 증명할 대상에 맞게 해야 한다는 생각을 했다. ACL2가 자동으로 증명을 하니 Coq이나 Agda보다 낫다고는 이야기할 수 없다. 복잡한 증명은 자동으로는 어려울 뿐 아니라 복잡한 성질에 대해서는 ACL2에서는 기술조차 할 수 없기 때문이다. 반면 ACL2를 이용하여 자동으로 증명 가능한 문제를 Coq이나 Agda로 하는 것은 쓸데없이 비싼 비용을 치르는 일일 수도 있다.

A Unified Approach to Fully Lazy Sharing

Thibaut Balabonski
최준원

함수형 언어, 기본적인 lambda calculus 부터 출발하면 계산 과정에서 beta reduction 을 얼마나 적게 하느냐가 퍼포먼스에 큰 영향을 주게 된다. sharing 은 같은 계산을 한꺼번에 하는 것을 의미하는데, 예를 들어 (λx. x ... x ...) e 의 경우 e 가 x 에 바인딩될 때 두 번 치환이 일어나고 이 때 call-by-name 일 경우 치환 이후 e 의 같은 계산을 두 번 하게 된다. sharing 은 이를 한꺼번에 같이 계산하여 퍼포먼스를 올린다. sharing 을 하는 방법으로 많은 방법이 제시되었는데 이 논문에서는 그 방법을 묶어서 다른 하나의 방법을 제시하였다. 많은 방법들 중 치환시

라벨링을 하여 같이 계산하는 방법을 사용하였고, 추가로 람다 리프팅이라는 이론을 설명하였는데 완벽하게 이해하지 못하였다.

발표 슬라이드가 특히 인상적이었는데, 간단한 그래프와 라벨로 슬라이드를 만들고 설명을 하면서 그 설명으로 슬라이드를 채워나가는 방식이 매우 효율적으로 보였다. 다만 좋은 슬라이드를 설명하는 발표자의 어투와 속도가 매우 부담스러워 거의 슬라이드만 보면서 내용을 이해했던 점이 아쉬웠다.

The Ins and Outs of Gradual Type Inference

Aseem Rastogi, Avik Chaudhuri, and Basil Hosmer

김진영

Gradual type inference는 프로그램의 일부는 동적으로 타입을 추론하고 다른 일부는 정적으로 타입을 추론하는 타입 시스템에 대한 것이다. 말은 재미있어 보이긴 하지만 실제로 어떻게 응용되는지, 정말 쓸만하긴 한 건지 궁금해서 열심히 들어 보았다.

실제로 어도비의 연구원들이, 플래시 어플리케이션을 개발할 때 많이 사용되는 액션스크립트의 타입추론 알고리즘을 개선한 연구였다. 동기와 목적이 명확하면서 이론과 실제를 잘 결합한 흥미로운 연구라는 생각이 들었다. 실제로 잘 되면 플래시 어플리케이션의 개발자도 편해지고, 앱 자체의 성능도 개선될 것이기 때문이다.

현재 액션스크립트에서의 상황은 이렇다고 했다. 먼저 모든 타입을 일일이 기입해 주는 방법이 있다. 개발자 입장에서는 매우 번거로운 일이다. 하지만 편하게 안 써도 실행에는 문제가 없다. 다만 타입 정보가 불완전하면 불완전할수록 매우 느린 성능을 감수해야 한다. 타입이 기입되지 않은 변수를 현재는 모두 동적 타입으로 보기 때문이다.

이 연구는 타입이 기입되지 않은 변수들의 타입을 동적(dynamic) 타입이 아닌 모르는(unknown) 타입으로 놓는 방식으로 알고리즘을 개선하였다. 이렇게 하면 성능 최적화 같은 정적 타입의 장점을 가져올 수 있다고 했다. 그리고 모르는 타입들에 타입 변수를 주고 방정식을 만든 뒤 그 방정식을 최대한 풀어낸다. 방정식은 이 변수로 흘러들어오는 값들의 타입과, 이 변수가 흘러나가는 값들의 타입들을 보고 세운다.

그리고 이 모든 과정이 하향 호환 가능(backward compatible)함도 보였다. 즉, 타입 정보를 개선해나가는 과정에서 새로운 런타임 에러가 출현하지 않는다는 것이다. 발표를 할 때에는 사실 이 부분이 어려웠다고 하며 그 이유와 해결과정을 설명했는데, 잘 이해하지는 못하였다.

이 알고리즘을 액션스크립트에 실제로 구현하여 벤치마크 테스트로 성능 개선을 보였다. 앞으로도 계속 액션스크립트의 타입 시스템에 대한 연구를 계속 진행할 것이라고 했다.

Clarifying and Compiling C/C++ Concurrency: from C++11 to POWER

Mark Batty, Kayvan Memarian, Scott Owens, Susmit Sarkar, and Peter Sewell

이승중

C++에 관심이 많아서 이번 세션을 듣게 되었다. 원래 C++ 분석을 하고 싶었던 나로서는 흥미로운 주제이기도 했다. 비록 이번에는 C++의 모든 부분을 formalize 하지는 못했지만 차츰 모든 부분을 지원해 나가려고 하는 것 같다.

이 발표는 C++11에 확정된 메모리 모델을 formalize 하는 내용이었다. 이번에 발표된 C++11에서는 멀티 스레드 지원이 추가되면서, 원래 컴파일 되는 머신 마다 행동이 달라졌던 메모리 모델을 하나로 확정 지었다. 이 연구는 그 메모리 모델에 따라 C++11로 작성된 코드가 Power 아키텍처로 컴파일 될 때, 제대로 동작하는지를 검증하였다.

이 발표를 보면서 C++이 갈수록 복잡해 지고 있구나 하는 생각이 들었다. 비록 분석이 용이하고, 시류에 따른 멀티 스레드 지원이 필요하다고 하더라도 사용방법이 너무 복잡해서 제대로 쓸 수 있는 사람이 별로 없을 것 같다. 기존의 C++도 단순하게 만들어서 분석을 용이하게 하고 사람들이 실수 할 수 있는 여지도 줄여야 할 것 같은데 반대로 가고 있는 것 같아서 아쉬웠다.

POPL을 정리하며

이승중

이번 학회에는 재미있는 주제의 발표가 많았다. 스펙을 가지고 프로그램을 만들어내는 합성부터 시작해서 바이너리 분석, LLVM과 C/C++의 의미구조에 대한 주제까지 다양했다. 제목들만 봐도 프로그래밍 언어 분야에 관심이 있는 사람들이라면 한번쯤 호기심을 갖고 들어봄직한 주제들이었다.

주제뿐 아니라 발표방식도 흥미롭게 꾸민 것들이 많았다. 듣기 어려울 것이라고 생각했던 발표들을 의외로 쉽게 접근할 수 있어서 좋았다. 발표를 듣기전에 주제만 보았을 때에는 무슨 내용인지를 이해하기가 힘들었었다. 하지만 여러 발표들이 전문 분야가 아닌 사람들을 대상으로 준비를 해서, 동기와 하는 일들을 대략적으로 알 수 있었다. 보다 구체적인 내용은 논문을 찾아봄으로써 궁금한 부분을 찾을 수 있었다. 특히 K. Rustan M. Leino가 발표한 'Automating Induction with an SMT Solver' 는 구체적인 예와 함께 유머도 추가하여 다른 발표에 비해 받아들이기가 편했다.

그 동안 책이나 논문에서만 보아왔던 사람들의 발표나 이야기를 들을 수 있어서 더욱 좋았다. POPL의 첫 순서였던 Peter W.O'Hearn의 진행으로 이루어진 Tony Hoare 와의 인터뷰는 연구자로서의 삶을 들여다볼 수 있는 좋은 기회였다. Patrick Cousot의 'An Abstract Interpre-

tation for Termination'은 비록 많은 생소한 내용과 빠른 진행으로 많이는 알아들을 수 없었지만 Cousot의 발표를 들을 수 있다는 것만으로도 감회가 새로웠다.

PHILADELPHIA

미국의 고도

Philly Cheese Steak

오학주

POPL 홈페이지를 보면 필라델피아에서 치즈스테이크를 먹지 않으면 필라델피아 여행이 완성될수 없다고 쓰여있다. 그 동안 미국에서 먹었던 음식들 중에서 인상적이었던게 스테이크밖에 없던터라 이번에도 기대를 했었다. 하지만, 며칠후 치즈스테이크를 검색해보니 일반스테이크가 아니라 바게트같은 빵에 소고기와 치즈를 잔뜩 넣은 일종의 샌드위치였다. 그래도 맛있다고 하니까 조금 기대를 했었는데, 필라델피아에 가서 직접 먹어보고 완전 실망했다. 고기가 특별히 맛있는것도 아니고, 느끼하기만한 전형적인 미국음식이었다. 게다가 양도 아주 많아서 웬만한 사람은 일인분을 다 먹지 못할 것 같았다. 치즈스테이크에 실망하고 마지막날밤에 정식 스테이크를 먹으러 여러 레스토랑을 돌아다녔는데, 사람들이 너무 많아서 결국 먹지 못한게 아쉬웠다.

시내 구경

조성근

필라델피아에 도착한 다음 날에는 저녁도 먹을 겸 시내 구경을 나갔다. 눈이 많이 와서 길이 미끄러웠다. 숙소에서 15분 정도 걸어 나가자 가로등이 환하게 비추는 시내가 등장했다.

먼저 넓은 옷가게에 들어섰다. 슬리퍼를 사기 위해 들어갔었는데 정작 슬리퍼는 팔고 있지 않았다. 학회 때에 발이 불편해서 중간 중간 갈아 신으려 했는데 결국 학회가 끝날 때까지 구입하지 못했다. 함께 구경을 갔던 지훈이가 청바지가 싸

다며 좋아하며 구입했다.

그리고는 저녁을 먹으러 눈에 보이는 치즈스테이크 집에 들어갔다. 필라델피아가 치즈스테이크로 유명하다니 안 먹어볼 수 없었다. 막상 시키고 나니 치즈스테이크라는 것이 별게 아니었다. 긴 핫도그 빵에 치즈와 얇은 소고기가 들어있을 뿐이었다. 스테이크라고 해서 기대했었는데 약간 실망했다.

어느새 밤이 깊어져 돌아와야 할 시간이었다. 이 동네는 한 골목만 잘못 들어가면 위험한 사람을 만난다고 원찬이에게 전해 들었기 때문에 서둘러 집을 향했다. 돌아오는 길에는 가로등도 없고 사람도 별로 없어 을씨년스러웠다.

사진들

이승중

Benjamin Franklin Bridge. 필라델피아의 넓은 경관을 한눈에 볼 수 있는 곳이었다. 비가 내린 직후라 석양이 멋있었다.

Philadelphia Museum of Art. 필라델피아의 북서쪽 끝에 있는 미술관으로 웅장한 건물에 볼거리가 많다고 한다. 고흐 전을 하고 있었으나, 시간관계상 들어가보지는 못했다.

미술관에서 본 시청의 모습.

시청 야경.

정리하며

김진영

두 번째로 참석한 국제 학회였는데 처음보다는 조금 더 편안했다. 12월의 APLAS에 이어 한 달 반 정도만에 또 이렇게 좋은 기회가 생긴 것이다. 그때는 처음이라 모든 것이 어색했고, 한창 삼성과의 연구과제를 진행 중이라 다소 산만한 느낌도 있었다. 이번엔 그때보다는 좀 나았다고 생각하고 있다.

다만 긴 여정과 시차적응 문제로 몸이 고단하여 약간은 고생하기도 했다. 홍콩에 워크샵을 다녀온 뒤 바로 20여시간을 비행기를 타고 날아온 것이 조금은 몸에 무리였던 듯했다. 그러다 보니 준비도 다소 부실했던 것 같다. 사전에 발표논문을 훑어보고 조금이라도 공부를 해갔으면 조금 더 좋았겠다 하는 아쉬움이 남았다. 특히 POPL은 두 개의 세션이 동시에 진행되는 만큼 사전에 미리 정보를 파악했으면 좀 더 선택이 수월하고 남는 것도 조금이라도 많았지 않았을까 싶었다. 논문 제목과 요약을 읽고 나름대로 요리조리 골라 들었는데 늘 성공한 것은 아니었다. 나중에 이야기해 보니 생각과는 달리 내가 들은 톡은 너무 어려워서 거의 이해하지 못했던 반면 옆 세션의 톡은 재미있고 흥미로운 내용이었던 적도 있었다.

유익한 경험을 많이 했지만 발표 논문 없이 온 것이라 다소 겉도는 느낌이 아쉬웠다. 특히 쉬는 시간이나 식사 시간에 주변 참석자들과 이야기를 나눌 때 그랬다. 안 그래도 말을 해 볼 기회가 거의 없었는데, 그마저도 늘 인사 정도 하는 수준에 그쳤다. 아무래도 내가 할 이야기가 없다 보니 지레 위축되는 느낌 때문인 듯했다.

PL의 다양한 분야들을 조금씩 맛볼 수 있었지만 깊은 이해는 만만치 않았다. 특히 거의 모든 VMCAI 논문 발표의 내용은 새로워 보였다. 예를 들자면 첫날은 프로그램 Synthesis라는 흥미로워 보이지만 생소한 분야의 논문이 대부분이었는데, 또 로직에 대해 잘 알지 못하면 내용의 핵심을 알아듣기 어려운 발표가 많았다.

많은 발표를 들으며, 당연한 것이지만 중요한 것은 결국 소통이라는 것을 새삼스럽게 느꼈다. 저명한 국제학회이다 보니 비록 세세한 기술적인 내용은 알아듣지 못하더라도 연구의 핵심이나 큰 그림은 이해할 수 있게끔 준비한 경우가 대부분이었다. 하지만 모두가 그런 것은 아니었다. 듣고 있으면 이 사람이 뭔가 자기가 많이 한 걸 열심히 설명하기는 하는데 대체 뭘 한 건지 알아들을 수 없는 경우도 가끔 있었다. 그리고 어떤 발표자는 부득이한 사정으로 참석할 수 없었는지 자신의 발표를 녹음한 파일을 보내 왔는데, 솔직히 발표가 전혀 귀에 들어오지 않았다.

나도 어서 논문을 써서 학회장에서 다른 사람들과 나누면 참 재밌겠다 싶었다.

이영석

열 시간 넘게 비행기를 타고 어디론가 떠나는 것이 처음이었고, 난생 처음 시차적응 문제도 겪었다. 낮에 출발하여 10시간을 비행했지만 도착한 샌프란시스코도 낮이어서 환승 시간에

공항에서 잠을 잘 수 밖에 없었다. 몇 시간을 더 걸려서 도착한 필라델피아는 자정이었지만 그때에는 정신이 너무 맑아서 잠을 잘 수가 없었다. 첫 날부터 시차적응의 문제를 겪고 힘든 하루를 보냈다. 거의 매일 학회가 끝나자마자 저녁에 잠을 자서 자정에 일어나고 다시 새벽에 잠드는 생활을 반복하다 학회 막바지가 되어서야 시차에 겨우 적응을 하게 되었다.

나의 영어실력의 한계와 공부의 필요성을 느낀 학회였다.. 워낙 내용이 어려워서 모국어로 들어도 이해하기가 힘들었을 텐데 100% 이해하지 못하는 영어로 들으니 따라가는 데 많이 힘들었다. 학회에 들어가기 전에 논문 요약부분을 읽고 들어가서 무엇에 관한 발표인지는 대충 파악을 했으나, 인트로 부분을 넘어간 후에 자세한 내용을 들어갔을 때에는 백퍼센트 이해한 발표가 거의 없었다. 또한 영어 실력 뿐만 아니라 기본적인 배경 지식도 많이 부족한 부분도 있다. 열심히 공부해야겠다는 열의가 생기게 된 학회였다.

음식문제는 좋은 점과 나쁜 점이 공존했다. 생각보다 음식이 입맛에 맞았고, 어느 식당이나 서비스 수준이 최고였다. 햄버거와 빵 시리얼만 먹다가 마지막 날에 갔던 베트남 식당은 정말 인상 깊었다. 3년 연속 필라델피아 최고의 베트남 레스토랑으로 선정 되었다고 하는데 정말 맛있었다. 하지만 간단한 햄버거도 너무 비쌌을 뿐 아니라 따로 팁까지 내야 해서 금전적인 부담이 상당했다. 그 점이 약간 아쉬웠다.

숙소는 학회장 근처의 3분 거리의 호텔로 잡았다. 학회가 열리는 호텔이 예약이 안되는 날짜가 하루 있어서 어쩔 수 없이 다른 곳으로 잡았지만 굉장히 맘에 들었다. 구글 맵으로 검색했을 때에는 5-7분 걸린다고 되어있었지만 더 빠른 경로가 있어서 잠깐만 걸어서 도착할 수 있었고, 방도 넓어서 편한 1주일을 보냈다. 하지만 무선 인터넷이 유료로만 제공되는 부분이 아쉬웠다. 창문을 내려다 보면 바로 강과 커다란 배들이 보이고, 그 주변으로 공원이 있어 산책하기도 매우 좋았다.

최준원

이번 VMCAI, POPL 은 내가 처음으로 경험해본 국제 학회였다. 학회라는 것은 어떤 분위기를 가지고 있을까 이런저런 생각이 많았는데, 예상보다는 동적인 모습이 많이 보였다. 선배들께서 이번 POPL 에 특히 많은 사람들이 참가했다고 알려주셨는데 정말 큰 규모였다. 많은 사람들이 비슷한 분야를 연구한다는 사실이 꽤 놀라웠다. (학회가 끝날 때쯤에는 비슷한 분야 안에서도 정말 수많은 다른 연구가 이루어지고 있구나라는 사실을 깨달았지만)

사실 VMCAI 학회의 경우 약자 그대로 Verification, Model Checking, AI 의 내용을 다루고 있고 나는 앞의 두 분야에 대해서는 거의 아는 것이 없었기 때문에 대부분의 발표를 이해하기 힘들었다. 다음에 학회를 갈 수 있는 기회가 생긴다면 발표의 Abstract 정도는 미리 읽어보고 가야겠다는 생각을 바로 하게 되었다. POPL 의 경우도 학회의 권위와 규모만큼 어려운 주제가 많았지만 몇몇 발표는 기막힌 아이디어를 제시하였고 개인적으로 충격을 많이 받았다. 덕분에 여러가지 주제도 생각나고 연구해 보고 싶다는 의지도 새로이 생겨나게 되었다.

감사 인사

학회에 참석해서 좋은 사람, 좋은 연구, 좋은 기회들을 접할 수 있도록 지원해주신 이광근 교수님, 소프트웨어 무결점 연구센터, 한국 연구재단에 다시 한 번 진심으로 감사 드립니다.

출장 참가자 일동: 오학주, 이원찬, 이우석, 조성근, 이승중,
김진영, 윤용호, 이영석, 최준원, 강지훈 드림