강지훈

ROSAEC 학회 참가 보고서

APLAS 2014

11/17-11/19/2014

APLAS 2014를 다녀와서

Gardens by the Bay, Singapore

1. 소개

   Asian Symposium on Programming Languages and Systems (APLAS) 는 아시아의 POPL 이며, 훌륭한 PL 연구자들이 모이는 인정받는 학회이다. 규모는 다소 작지만 아시아 지역 그리고 나아가 세계의 연구자들이 모이는 연구의 장이다. 올해 12회 APLAS는 싱가포르에서 열렸으며, 11/17-11/19에 개최되었다. (참고로 내년 13회는 포항에서 열린다.) 24편의 흥미로운 논문과 18편의 포스터가 발표되었다.

2. 포스터 발표

포스터 세션에 두 편의 포스터를 제출하였다:

A. Extensible Verified Validation for LLVM Optimizations
Sungkeun Cho, Joonwon Choi, Jeehoon Kang, Chung-Kil Hur, Kwangkeun Yi

내가 발표를 맡았다. 우리의 주된 아이디어를 다음과 같이 설명했다:

- 실용적인 컴파일러인 LLVM를 검증하고 싶다. 가장 큰 문제는 검증 비용이다.
- 많은 LLVM 최적화는 비슷한 꼴임을 발견했고, 이를 이용해 **많은** 최적화를 검산할 수 있는 **하나의** 검산기를 만들었다. 이를 통해 검증 비용을 줄일 수 있었다.
- 검산기가 복잡해지지 않도록, 그래서 검증 비용이 낮도록, 컴파일러가 검산에 필요한 정보를 많이 제공하도록 디자인했다.
- 그 결과 instruction combining pass의 1/4에 해당하는 100여개의 꼬마 최적화를 성공적으로 검증할 수 있었다. 검산은 Python과 같이 큰 프로그램에서도 빠짐 없이 성공했다.

많은 사람들이 관심을 가졌다:

- 많은 사람들이 LLVM의 semantics가 좀 더 정리되어야 한다는 우리의 의견에 동의했고, LLVM을 검증한다는 목적이 유의미함에 동의했다. 또한 검증 비용이 중요한 문제라고 파악한 우리의 의견이 잘 받아들여진 것 같았지만, 우리가 정말로 적은 비용으로 LLVM을 검증할 수 있을지에는 의문을 표하는 사람들이 종종 있었다.
- CompCert 팀: 우리는 CompCert와는 달리 진짜 쓰이는 컴파일러를 검증하고 싶다는 점이 차이점이었고, CompCert 팀도 이를 유의미하게 받아들인 것 같았다.
  특히 APLAS에 pointer-integer casting과 관련한 논문을 제출한 분이 있어서, 우리가 똑같은 문제를 다른 방식으로 풀었다고 귀띔해 주었다. 자세한 내용은 인상깊은 발표 섹션에서 좀 더 다루고자 한다.
- 싱가포르 국립대학에서 LLVM static analyzer를 만드려던 사람: 정리된 semantics가 있어야 한다는 의견에 더 크게 동의했다.
- 한 싱가포르 국립대학 학생이 포스터 구성에 의견을 주었는데, 새겨들을 여지가 있었다. 좀 더 에제를 중심으로 포스터를 구성했으면 좋았으리라 의견을 말해주었고, 나도 동의했다. 성실하게 포스터를 읽어줘서 고마웠다.

포스터 세션을 통해 학회가 의견을 제시하고 나누고 토론하는 장이라는 것을 더 크게 느낄 수 있었다. 많은 사람들이 우리 의견에 동의하는 것 같아 기분이 좋았다. 다른 사람들에게 내 의견을 설명하고 또 피드백을 받기에는 포스터 세션만큼 좋은 기회가 없는 것 같다. 발표하기 잘했다고 생각한다.

B. Towards Scalable Verified Validation of Static Analyzers
Jeehoon Kang, Sungkeun Cho, Joonwon Choi, and Chung-Kil Hur

성근이형이 발표를 맡았다. 바로 옆에 있었던 포스터임에도 불구하고, 내가 발표하는 포스터를 설명하기에 바빠 분위기가 어떤지 잘 살펴보지 못했다. 성근이형 학회 참가 보고서에 더 잘 정리된 내용이 있다.

내 포스터를 발표하느라 다른 포스터 발표는 잘 살펴보지 못했다. 슬쩍 살펴보기로는 JVM garbage collector 관련 일이 재밌어 보였는데, 나중에 보니 1등상을 받았다. 우석이형이 포스터 발표 2등상을 수상했다. 나는 상을 타지 못했지만 내 포스터를 크게 실수 없이 발표했음에 만족하고, 다소 미진했던 부분은 다음부터 더 잘해야 겠다.

3. 인상깊은 발표

A. A precise and abstract memory model for C using symbolic values
(Frédéric Besson, Sandrine Blazy and Pierre Wilke)

C의 중요한 기능중에 하나인 정수-포인터 변환을 잘 정의한 (formalize) 일이다. CompCert 등 이전의 일들은 이 기능을 지원하지 않았다. 왜냐하면 포인터를 어셈블리 언어에서와 같이 숫자가 아니라 (block번호, offset)와 같이 특별하게 표현했는데, 이런 경우 포인터를 숫자로 바꾸는게 자명하지 않았기 때문이다. 포인터를 숫자로 표현하지 않는 이유는 최적화를 잘 지원하기 위해서이다. 포인터가 숫자라면 상수 전파 등 기본적인 최적화도 증명할 수 없게 된다. 정리하면, (1) 최적화를 잘 지원하면서 (2) 정수-포인터 변환을 잘 지원하는 C 모델을 고안하는 것은 중요한 문제였고, 이 논문은 이 문제를 풀기 위한 하나의 답이다.

우리 연구그룹도 이번에 이 문제에 대한 하나의 답을 만들었기 때문에, 특별히 이 발표를 신경써서 들었다. 우리는 우리의 논문을 준비하면서 우리의 답이 이 논문의 답보다 더 나은 답임을 확신하고 있어서 발표 내용이 크게 기대되지는 않았지만, 나는 개인적으로 이 그룹에서 **지금** 어떤 연구를 하고 있는지 궁금했다. 발표 끝나고 질문하는 시간을 기다렸다.

이 논문의 아이디어를 정리하면 다음과 같다. (1) 어떤 식을 계산할 때, 포인터들에 임의의 정수를 대입해도 같은 결과가 나오면 식의 결과는 잘 정의된다. 아니면 undefined behavior. 예를 들어 p & 0xFFFF는 undefined behavior이고 (p의 alignment를 잘 모를 경우), p & 0xFFFF0000 + p & 0xFFFF는 p로 잘 정의된다. (p에 무엇을 넣든 p가 나오므로) (2) 이를 알기 위해 SMT Solver를 쓴다.

그러나 이 방법은 (1) non-determinism을 본질적으로 지원하지 못하고, 보다 중요하게 (2) hash table등 중요한 응용을 undefined behavior로 만들어버리는 약점이 있으며, 또한 (3) 이 모델 위에서 최적화를 증명하는 것이 불가능하다.

우리는 포인터를 정수로 변환하려 하는 경우 실제로 block번호에 실제로 정수를 주는 방식으로 해결했다. 이를 통해 (1) non-determinism도 잘 지원하고, (2) hash table도 설명할 수 있으며, (3) 최적화를 지원함을 증명할 수도 있었다.

질문 시간에 우리가 발견한 단점들을 질문했고, 기대와는 달리 이 단점을 해결할 의지를 가지고 있어보이지는 않았다. 장기적으로는 우리가 제안한 답이 표준이 되리라 더욱 더 크게 기대하게 되었다.

B. Automatic Memory Management Based on Program Transformation using Ownership (Tatsuya Sonobe, Kohei Suenaga and Atsushi Igarashi)

Ownership을 분석해서 메모리가 샐 만한 곳에 자동으로 `free` 구문을 넣어주는 일이었다. 이 문제는 halting problem과 동등하게 어려운 문제이기 때문에 항상 잘할 수는 없고, 이 일은 soundness(찾은 곳은 모두 메모리가 새는 곳이다)는 만족하되 completeness(새는 곳은 모두 찾는다)는 만족하지 못하는 모델을 만들었다.

발표는 매우 인상적이었고 잘 했다. 영어가 매끄럽다고 하기는 힘들지만, (허충길 교수님께서도 말씀하셨듯) 강조할 부분에 강조하고 전체적인 호흡을 잘 조절하는 것이 우수한 발표였다고 생각한다.

기본적으로 타입 시스템에 ownership 정보를 얹었고, SMT Solver를 이용해서 type checking을 하는 방식이다. 다만 이 ownership 타입 시스템이 얼마나 실용적으로 쓰일 수 있을지는 다소 의문이다. 메모리 모델도 다소 현실과는 거리가 있어보인다.

C. Inferring Grammatical Summaries of String Values (Se-Won Kim, Wooyoung Chin, Jimin Park, Jeongmin Kim and Sukyoung Ryu)

결론부터 말하면, 문자열을 문법으로 요약하는건 어렵다는 것이 톡의 요지였다. 이 톡의 훌륭한 점은 어려운 지점을 분별해내고 어려운 지점에서 어떤 시도를 해봤는지 그 경험을 잘 설명해냈다는 데에 있다고 본다. 구체적으로 어떤 어려움과 해결책이 있었는지 잘은 기억나지는 않지만, 연사가 문제에 접근하는 방식이 훌륭해서 깨닫는 바가 있어 적는다.

D. Hereditary history-preserving bisimilarity: logics and automata (Paolo Baldan and Silvia Crafa)

사실 정확히 이해하기는 어려웠던 톡이었지만, 그래도 재미있는 포인트가 있었다.

두 프로그램이 동등(equivalence)하다는 데에는 여러가지 관념이 있다. (1) 행동의 동등 behavioral equivalence, 즉 입출력이 같다는 것이 두 프로그램이 동등하다는 정의가 될 수도 있고, (2) 바이시뮬레이션bisimulation, 즉 step별로 동등함이 유지된다는 것이 두 프로그램이 동등하다는 정의가 될 수도 있다. (이 때 2=>1은 성립하지만 1=>2는 성립하지 않는다.) 둘중 무엇이 더 중요한 동등의 관념이냐는 상황마다 다를 수 있다. 지금까지 CompCert 등 컴파일러 검증 쪽에서는 (1)이 가장 중요한 정의라고 생각해왔다.

하지만 동시성을 고려하고 동시성 프로그램에 대해 논증하거나 성질을 알아낼 때, 행동의 동등은 너무 다루기 어려운 관념이다. 바이시뮬레이션은 한 step만 보면 된다는 지역성을 가지고 있지

만, 행동의 동등은 전체 행동에 대해 정의되기 때문에 이런 지역성이 없다. 지역성의 부재는 대체로 문제를 어렵게 만든다. 예로, 이용할만한 지역성이 없으면 model checking에서 state explosion을 일으키기 더 쉽다.

이 논문은 그냥 바이시뮬레이션도 논증하기에 적합한 수준이 아니고, 더 구조를 준 동등성을 도입해야 한다고 주장한다. 그게 바로 “hereditary history-preserving bisimilarity”인데, 톡만으로는 자세한 내용을 이해하기는 너무 어려웠다. 왜 더 많은 구조가 필요한지도 잘 이해하지 못했다.

다만 (1) 행동의 동등이 무조건 좋은 정의라고 생각했던 내게 새로운 관점을 주었고, (2) 앞으로 동시성 메모리 모델, 동시성 C11을 다루게 될 때 이 관점에서 영감을 많이 얻을 수 있을 것 같아 나에게는 매우 유익한 톡이라고 생각했다. 좀 더 읽어보고, 면밀히 파악해서 앞으로 동시성을 다루는 또 하나의 툴을 얻어야 겠다고 생각했다.

E. NetKAT: A formal system for the verification of networks (Dexter Kozen)

NetKAT은 네트워크 시스템을 기술하는 언어이다. 최근에는 다소 범용적으로 네트워크 장비를 만들고, 소프트웨어적으로 장비의 행동을 기술하는 방식이 유행한다고 한다. NetKAT은 행동을 기술하는 언어이다. 명시적인 언어를 사용하는 이점은 (1) 네트워크 상황이 바뀔 때, 더 좋은 알고리즘이 개발될 때 하드웨어 변경 없이 소프트웨어 변경만으로 대응할 수 있다는 점, (2) 하드웨어는 언어를 잘 구현한다는 것을 검증하고, 로직은 언어 레벨에서 한 번만 검증하면 된다는 점이 있다.
NetKAT은 Kleene 대수를 기반으로 만들어진 언어라는 점이 독특하다. Kleene 대수는 정규식의 성질을 기술하는 방식이라 한다. 네트워크 장비는 오토마타이므로, Kleene 대수로 표현하는 것이 자연스러운 방향이라고 한다. 하지만 참/거짓을 판단하는 장치가 없는 것이 장비를 Kleene 대수로 표현하는데에 문제점이었다고 한다. 그래서 test도 지원하도록 확장한 것이 Kleene Algebra with Test (KAT) 이라고 한다.
KAT 대수 자체는 오랜 이론이라 이 일의 “이론적인” 기여는 크진 않지만, 이 일의 훌륭한 점은 이론이 잘 적용될만한 지점을 찾아 훌륭하게 이론과 실제를 접합한데에 있다고 할 수 있다. 이론은 이론만으로 남아선 곤란하고 항상 활용되는 지점을 찾아야 한다. NetKAT은 이 교훈을 내게 다시 깨우쳐 주었다.

F. Incremental Adoption of Static Typing (Julien Verlaguet)

연사는 Facebook에서 PHP에 타입을 넣는 작업을 하는 엔지니어였다. 개인적으로 Facebook에서 애초에 PHP를 선택한건 불운한 선택이었다고 보는데, (물론 초창기 폭발적인 성장은 PHP가 아니었으면 불가능했을 것 같기도 하지만) 이제는 그 불운을 완전히 딛고 일어난 것 같다. 더이상 Facebook이 쓰는 PHP는 원래 PHP가 아니고 많이 패치가 된 Facebook의 PHP가 된 것 같다. 회사를 위해서는 좋은 일이라 생각한다.

타입을 점진적으로 받아들이는 것은 이러한 방향에서 중요한 단계라고 보는데, 왜냐하면 타입은 명세를 작성하는 가장 기본적인/싼 방법이라고 보기 때문이다. 그러나 거대한 리거시가 있는 시스템에 타입을 도입하는건 매우 도전적인 일이다. 연사는 Facebook에서 이 과업을 이루는 과정에

서 만난 문제를 소개하고 해결하는 과정을 발표했다.

연사는 이론적인 내용이라기보다는, (1) 기존의 프로그래머들을 어떻게 설득할 것인가? (2) 어떻게 기존 시스템에 최소한의 변경을 가하면서도 타입을 도입할 수 있을까? 와 같이 실용적으로 중요한 질문에 집중했다. 매우 흥미로웠고, 거대한 시스템에 적용되는 경험을 듣는 것은 매우 좋은 일이라고 생각했다. 특히 구체적으로 기존 개발팀과 있었던 일들을 경험담으로 설명해준 일이 인상깊었다.

얼마 전 POPL에서 발표되었던, ActionScript에 gradual typing을 넣은 일과 비슷하다고 생각했다. 다만 Facebook은 이 일을 산업에서 찾을 수 있는 가장 큰 규모의 코드베이스에 했다는 데에 중요한 의의와 참고점이 있다고 생각했다. 좀 더 많은 정보를 얻고싶지만, 기업이 하는 일이라 많은 것들이 비공개인 것 같아 다소 아쉽다. 큰 기업에서 PL 관련 지식이 실용적으로 쓰일 수 있는 중요한 응용을 본 것 같아 매우 좋았다.

4. 싱가포르

   싱가포르는 적도 부근의 더운(혹은 따뜻한) 도시국가로, 말레이반도 끝자락에 붙어있다. 아주 깨끗하고 쾌적하다는 말이 어울릴만한 도시였다. 학회에 참석하는 것 외에 싱가포르 시내는 거의 돌아다니지 않아 약간 아쉽다. 누군가 서울이 “하루만에 만들어진 도시”같다는 지적을 했는데, 싱가포르도 약간 그런 것 같다. (혹은 내가 싱가포르의 도심만 돌아다닌 것일 수도 있다.) 차이나 타운, 리틀 인디안 등을 더 잘 돌아다니지 못한 것이 다소 아쉽다.

   학회가 열린 싱가포르 국립대학가 다소 인상적이었는데, 데이타베이스 수업에 거의 200여명이 수강하는 것을 보았다. 컴퓨터공학과가 아주 큰 학교라는 생각이 들었다. 우리 학부는 다소 작은 것 같아 차이가 좀 느껴졌다. 싱가포르 국립대학이 아시아의 손꼽히는 명문이라고 하는데, 앞으로 어떻게 발전해나갈지 궁금하다.

5. 감사 인사

   학회는 짧았지만 배운 것은 많았습니다. 특히 학회장에 가만히 앉아있는게 참가의 목적이 아니라, 제가 지금 하고 있는 연구를 발표하고, 피드백을 얻고, 다른 사람에게 피드백을 주는 과정이 학회의 본질임을 다시금 깨달을 수 있었습니다. 학회에 참가할 수 있도록 지원해주신 이광근 교수님, 허충길 교수님, 그리고 무결점 소프트웨어 연구 센터 여러분들께 정말 감사드립니다. 앞으로 꾸준히 좋은 연구할 수 있도록 더욱 노력하겠습니다.