SAS 2009, PCC 2009

Univ. of California LA, CA, U.S.
8 - 17 August, 2009
공순호, 서울대학교 프로그래밍 연구실
soon at ropas.snu.ac.kr

0. Abstract

2009년 8월 8일부터 17일까지, 미국 LA의 UCLA에서 열린 SAS'09, PCC'09에 참석하였다. 여러 사람들을 만나고, 다양한 논문들을 통하여 프로그램 분석 분야의 요즘의 흐름과 연구 분야들을 접할 수 있었다. PCC'09에서 Abstract Parsing을 이용한 PCC Framework을 발표하였다. 이 글을 통하여서 10일간의 경험을 공유하고자 한다.

1. 배운 것들

2007년에 San Jose에서 열린 ESC(Embedded System Conference)에 참석했지만, 전시회의 성격이 강한 학회였고, 이번 SAS/PCC가 사실상 처음의 학회 경험이었다. 생각하고 배운 것들을 다음과 같이 정리해보았다.

1.1 발표

SAS와 PCC에서의 많은 발표들을 듣고, 다른 사람들의 반응을 살피면서 발표를 어떻게 하는 것이 좋을가에 대해서 다음과 같은 생각을 하게 되었다.

- 동기 부여: 모든 사람들이 발표에 집중하고 관심있게 들을 것을 기대하였는데, 그렇지는 않았다. 각자가 관심있는 것들에만 집중하고 다른 것들에 대해서는 자기 할 일을 진행하는 분위기였다. 내가 한 일에 대해서 자세히 소개하기 전에, 사람들에게 이것이 어떠한 도움이 되는지, 어떻게 유용하게 쓰일 수 있을지를 분명하게 이야기해주는 것이 필요하다는 것을 느꼈다.
- 침착함: 인상 깊었던 발표자들은 모두가 침착하게, 천천히, 웃으며, 당당하게 이야기했다. 실제로 PCC 발표를 준비해보니 그렇게 하는 것이 참 어렵다는

것을 느꼈다.

1.2 학회 준비

학회 참석을 제대로 활용하기 위해서는 많은 준비가 필요함을 배웠다.
적어도,
 - 참석자들의 얼굴, 이름, 연구 분야, publication list 파악하기
 - 발표 논문들 미리 읽기
이 두 가지가 필수적이라는 것을 느꼈다.

1.3 학회

사람들과 밥을 같이 먹어야한다. 함께 이야기하고 싶은 사람이 있다면 그 사람과, 그런 사람이 없을지라도 누군가와 식사를 함께 해야한다. 일대일로 이야기할 거리가 없다면, 여러 사람들이 모여 있는 테이블에 합석해서 그 사람들의 이야기를 듣고 대화에 참여하는 것도 좋은 방법이다(처음에 원태와 내가 그랬다.)

Session 사이의 쉬는 시간에도 사람들과 이야기하는 것이 중요하다. 처음에는 혼자 서있는 것이 어색하고 이상했지만, 사람들과 이야기하다보면 금새 이야기할 것들이 생기게 되었다. 주로 1) 발표자에게 발표한 내용에 대해서 궁금한 것들을 물어보거나, 2) 처음 만나는 사람들과 인사하고, 서로의 연구 분야와 연구 내용을 물어보았다.

1.4 영어

외국에 나가면 늘 겪는 것이지만 영어가 참 중요하다.

말하기가 서투르면 내용이 전달되지 않는다. SAS에서 Hirotoshi Yasuoka라는 사람이 "Polymorphic Fractional Capabilities"에 대해서 발표하였는데, 영어가 서투르다보니 사람들의 집중도가 급히 떨어지는 것이 느껴졌다. 다른 나라 말이어서 서툰 것뿐인데 억울한 면이 없지 않다. 그래도 어쩔 수 없다. 아쉬운만큼 열심히 영어 발표 연습을 꾸준히 해야겠다고 느꼈다.

듣기도 쉽지가 않다. 10년 동안 들은 영어가 "젊은, 미국인, 백인, 남성/여성이 이야기하는 영어"라는 생각이 들었다. 특히나 SAS 발표자 중에서 인도 사람들이 많았는데, 도무지 알아 듣기 힘든 발표들도 있었다. 그래도 그 사람들을 원망할 수 없다, 왜냐하면 다른 사람들(유럽, 미국사람들)은 다 잘 알아들었으니까. 못 알아들은 것은 내 탓이다. 영어 듣기에 갈 길이 멀다는 것을 느꼈다.

2. PCC 발표

8/15 PCC workshop에서 제출했던 논문을 발표했다. 논문과 발표 slide는 각각 다음에서 찾아볼 수 있다.

논문 : http://ropas.snu.ac.kr/~soon/paper/pcc09.pdf
발표 slide: http://ropas.snu.ac.kr/~soon/talk/20090815.pdf

발표는 순조롭게 진행되었고, 발표 후에 다음과 같은 질문/답변 시간을 가졌다.

Q1. LR parsing으로 설명하였는데, 다른 grammar 예를 들어서 Tree grammar여도 작동하는건가?
A1. Parsing algorithm은 component로 사용할 수 있다. efficient한 parsing을 위해서 LR parsing을 사용하는 것을 보여준 것이다. parsing algorithm을 가지고 있는 grammar라면 해당 grammar를 이용할 수 있다.

Q2. Syntactic property에 대해서만 작동하는 것인가? Semantic property에 대해서 방법이 없는가?
A2. Grammar로 만들 수 있는 property에 대해서 작동한다.
Semantic property를 기술하는 grammar를 만들 수 있다면 가능하다.

Q3-1. Code consumer side에서 fixed point checking을 1 iteration으로 함으로써 아낄 수 있는 계산량이 얼마나 되는가? 즉, Code producer side에서 fixed point computation을 할 때에 몇 iteration 정도 진행되는가?

A3-1. 주어진 프로그램과 grammar에 따라 다르다.
보여준 예제에서는 2 iterations이 필요했고, 따라서 1 iteration만 아꼈지만 복잡한 프로그램과 문법에 대해서는 더 커질 수 있다.

Q3-2. 그렇다면 fixed point computation이 가질 수 있는 maximum iteration이 존재하는가? lattice height?

A3-2. 좋은 질문, 예제에서 사용한 도메인은 2^P였고, infinite height이기 때문에 fixed point computation이 끝나지 않을 수 있다.
프리젠테이션에서 설명하지 않았는데 GPCE'09 논문에 있는 것과 같이 abstract parsing domain을 사용할 수 있고, 따라서 finite height의 abstract parsing domain을 사용하거나 widening을 이용하여서 분석을 유한시간안에 종료시킬 수 있다.

발표 후의 점심 식사를 여러 사람들과 함께 했는데, syntactic property만 분석 가능한 것인지에 대한 질문을 다시 받았다. 원태와 함께 집에 오는 길에 abstract parsing을 semantic analysis로 확장하는 것이 불가능한 것인지에 대해서 고민해보았다. 좋은 아이디어가 떠올랐고 현재 이것을 정리해보고 있다.

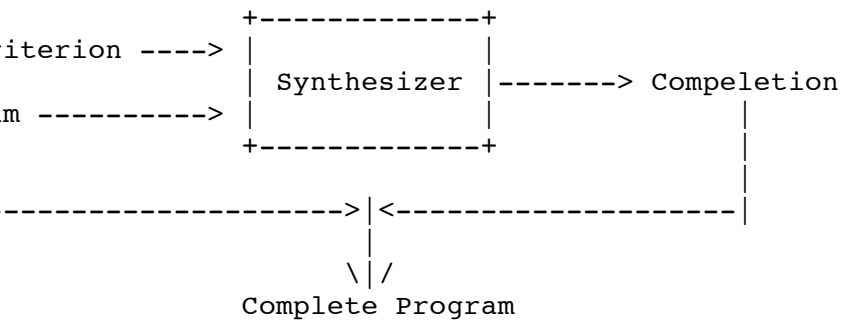
3. SAS 2009 발표 정리

8월 9일부터 11일간 SAS가 진행되었다. 총 21편의 논문과 2개의 invited talk이 있었다. 학회에 관한 자세한 정보는 학회 홈페이지인 http://sas09.cs.ucdavis.edu/ 에서 찾아볼 수 있다. 학회 proceeding은 이광근 교수님께 1권, 그리고 서울대학교 138동 215호 연구실에 1권 비치되어있다. 전체 발표된 논문과 talk중에서 인상적이 었던 것들은 정리하면 다음과 같다.

Invited Talk :
Algorithmic Program Synthesis with Partial Programs and Decision Procedure (8/9)
Ras Bodik

프로그램 합성기(program synthesizer)의 목표는 프로그램이 만족해야하는 성질(high-level spec)으로부터 실제 프로그램(low-level implementation)을 만들어 내는 것이다. 이러한 연구는 1960년대에서부터 이미 진행되어 왔지만, 이것을 완전 자동화하는 것은 1) 최종 구현물의 성능이 좋지 못하고, 2) 프로그래머가 알고 있는 지식들을 효과적으로 이용하지 못한다는 문제점을 가지고 있었다.

UC Berkeley의 Ras Bodik교수님의 연구 그룹에서는 프로그램의 일부분이 빈칸으로 남아있는 부분 프로그램(partial program)과 이 프로그램이 만족해야할 성질들(correctness criterion)을 입력으로 받아서 부분 프로그램의 빈칸을 채워주는 프로그램 합성기를 제안하고 개발하였다. 큰 그림은 다음과 같다.



이러한 방법은 마치 model checking을 연상케 한다. Model checking이 프로그램이 어떠한 성질을 만족하는지를 살피기 위해서 문제 공간을 탐색하는 것과 같이, 부분 프로그램(partial program)을 이용한 프로그램 합성기(program synthesizer)는 만족해야할 성질(correctness creterion)을 만족하는 프로그램이 존재할 수 있는지를 탐색해나간다.

이러한 발상을 구체화시키는데는 여러가지 문제가 있었을테다. 예를 들면 다음과 같은 것들이 있겠다. 1) search space가 무한하다면 끝나지 않을 수도 있을텐데? 2) 만들어낸 프로그램이 올바른지는 어떻게 검증할 수 있는가?

첫번째 문제를 위해서 SKETCH라는 문제를 기술하는 언어를 고안하였다. 이 언어를 통하여서 구멍이 있는 partial program을 기술할 수 있고, 동시에 각각의 구멍이 가질 수 있는 경우를 유한하게 제한할 수 있다. SAT와 같이 최적화가 잘 되어 있는 decision procedure를 사용하는 경우 큰 크기의 문제 공간에 대해서도 잘 작동함을 볼 수 있다. 두번째 문제의 경우에는 여러 가지 decision procedure를 사용하여 해결한다. 따라서 사용하는 decision procedure가 풀 수 없는 것들은 해결하지 못할 것이다.

이 방법은 여러 가지로 model checking과 닮아 있다. Model checking의 CEGAR(counterexample guided abstract refinement)와 상응하는 개념인 CEGIS(counterexample guided inductive synthesis)라는 개념도 존재하며 그 역할도 비슷하다.

Automatic Parallelization and Optimization of Programs by Proof Rewriting (8/9)
Clément Hurlin

프로그램과 그것의 separation logic proof를 이용하면, 1) 프로그램을 병렬화하거나 최적화할 수 있고, 2) 병렬화/최적화된 프로그램에 대한 separation logic proof를 만들어낼 수 있음을 보여준다.

유사한 idea를 이용했던 ESOP'09 논문 "Automatic parallelization with separation logic"과의 차이점이 궁금했는데, 핵심적인 차이점은 ESOP'09이 heap에 label을 붙이고 이를 이용해서 parallelism을 발견하는 반면, SAS'09에서는 separation logic의 frame rule을 이용해서 parallelism을 발견한다는 것이다. SAS'09의 방법이 더 깔끔하다고 생각할 수도 있지만, SAS'09의 방법은 separation logic proof를 요구하지만, ESOP'09는 proof가 아닌 shape analysis 결과에서 parallelism을 탐지하기 때문에 어느 쪽이 더 좋다고 이야기하기는 어려운 것 같다는 생각이 들었다.

Invited Talk:
Algorithmic Verification of Systems Software Using SMT Solvers (8/10)
Shaz Qadeer, Microsoft Research

Microsoft Research에서 진행하고 있는 HAVOC project에 관한 내용이었다. 정확한 C 분석을 위해서 annotation을 사용자에게 요구하고, 이를 이용해서 C 분석을 진행하는 도구인 HAVOC에 관한 내용이었다. 8/9 저녁 식사 때에 Shaz Qadeer가 동일한 이야기를 해주었는데, "annotation이 항상 나쁜 것이 아니다, 좋은 면이 있다."라는 내용이 인상적이었다. 어떤 좋은 면인가하면, 1) 우선 문서화에 도움이 되고, 분석하고자하는 성질이 어떤 것인지를 annotation해두면 추후에 코드 관리할 때에 도움이 된다. 2) 두번째는 annotation 정보를 이용하면 false alarm을 줄일 수 있다.

이러한 annotation에는 사용자가 직접 코드와 별도로 정보를 주어야한다는 단점이 분명 존재한다. HAVOC에서는 SMT solver를 이용하여서 필요한 조건들을 유추해내는 방법을 적용하였고, 이를 통해서 분석을 위해서 필요로하는 annotation의 수를 줄이는데 성공하였다.

Abstract Interpretation of FIFO Caches (8/10)
Daniel Grund, Jan Reineke

FIFO(First-in First-out) policy를 사용하는 cache에 대해서 abstract interpretation을 이용한 분석을 제안하는 논문. 목표는 프로그램

point마다 가질 수 있는 cache contents를 파악하는 것인데, 이를 위해서 may-/must-analysis가 필요하다. 이 논문에서는 새로운 may analysis를 고안하였고, 이를 통해서 기존의 FIFO cache 분석보다 좋은 결과를 보여주었다.

핵심 아이디어는 간단한 것이었다. "FIFO cache에서 새롭게 추가된 항목은 k개의 miss가 나야 비로소 cache에서 제거된다."라는 관찰에 근거해서 분석을 고안하였고, 그 결과가 기존의 분석들보다 좋게 나타났다.

A Verifiable, Control Flow Aware Constraint Analyzer for Bounds Check Elimination (8/10)
David Niedzielski, Jeffery von Ronne, Andreas Gampe, Kleanthis Psarris

목표 : JAVA에서는 실행시간에 배열의 bounc check이 일어나는데, 인하여 성능이 저하될 수 있다. 정적 분석을 통하여 불필요한 check을 제거하고자 한다.
방법 : e-SSA + Fourier-Motzkin Var. Elim + Control-related rules
결과 : imporved precision + less runtime overhead
전략 : constraint-based bound check elimination에서는 프로그램에서 부등식들을 추출하고 그것들을 풀이함으로써 bounc check elimination을 진행한다. 이 논문에서는 부등식 추출 과정에서 FMVE(Fourier-Motzkin Variable Elimination) 알고리즘을 이용하면 불필요한 부등식을 줄일 수 있음을 보였고 이를 적용하였다.

Abstract parsing: static analysis of dynamically generated string output using LR-parsing technology (8/10)
Kyung-Goo Doh, Hyunha Kim, David Schmidt

David Schmidt교수님께서 먼저 전체적인 내용 설명을 15분 정도에 걸쳐서 발표해주셨다. Abstract parsing에 대해서 세세한 내용은 설명하지 않으면서도 전체의 그림과 직관에 대해서 잘 설명하셨다. 발표를 어떻게 준비해야하는가에 관해서 배운 점들이 많았다.

도경구 교수님께서 발표를 이어 받아서 10분 정도 abstract parsing의 세부적인 내용과 실험 결과를 이야기해주셨다. Abstract parsing이 SAS에서 가장 화제가 되는 발표였다는 느낌을 받았다. Patrick Cousot 교수님은 흥미로운 연구라는 코멘트와 함께 날카로운 질문을 던졌다. "선형적으로 프로그램이 만들어지는 경우에만 잘 되는 것 아닌가?" 사실 abstract parsing을 shape analysis에 적용하는 일을 진행할 때의 어려움이 이 부분이었는데, 한번의 발표만에 찝어내어서 놀라웠다.

Zhendong Su와 Naoki Kobayashi도 흥미로워하는 반응이였다. 아마도 그들의 지난 연구가 string analysis이고 그것과 관계 있는 연구였기 때문인 것으로 보였다. 자신들의 연구와의 차이점에 관해서 질문과 답을 주고 받았다.

Invited talk:
Model Checking: My 27 Year Quest to Conquer the State-Explosion Problem (8/11)
Edmund M. Clarke, Carnegie Mellon University

전반적인 강연의 내용은 2007년 ROPAS에서 했던 강연과 크게 다르지 않았다. verification의 정의를 "intelligent exhaustive search of the state space of the design"로 하는 것을 보고 "Model checking의 아버지"라고 불리우는 분이시구나라는 생각을 했다.

Model checking을 모르는 사람이 들어도 Model checking에 대해서 이해할 수 있을만큼 친절하고 자세하게 강연해주었다. CTL, LTL과 같은 temporal logic에서 시작하여서, state explosion problem을 어떻게 풀어나갔는지를 시간 순서에 따라서 기술해나갔다.
- Symbolic Model Checking
- Partial Order Reduction
- Bounded Model Checker (SAT solver)
- Counterexample Guided Abstraction Refinement(CEGAR)

27년간의 연구가 1시간에 정리되는 강연이었다.

Future Challenge로 다음과 같은 것들을 이야기하였다.

- Constraint solver - SAT, SMT
- Compositional Model Checking
- Software Model Checking + Static Analysis
- Verification of embedded systems - timed and hybrid automata
- probabilistic / statistical model checking

Interval Polyhedra: An Abstract Domain to Infer Interval Linear Relationships (8/11)
Liqian Chen, Antoine Mine, Ji Wang, Patrick Cousot

Polyhedra domain은 a_1 x_1 + a_2 x_2 + ... a_n x_n <= b 와 같은 꼴의 수식을 표현하는 domain인데, interval polyhedra domain은 여기에서의 coefficient를 a_i가 아닌 [a_i, b_i]와 같이 interval로 표현하는 것이다.

논문의 내용은 interval polyhedra domain을 정의하고 그것을 이용한 프로그램 분석 결과가 어떻게 달라지는가를 보여준다. interval linear algebra를 이용하여서 domain에서 필요로하는 연산들을 정의하였고 interval polyhedra domain을 polyhedra domain과 비교하여 실험하였다.

비판적으로 바라본 것은
1) 12개의 testset 중에서 3개에서는 기존의 polyhedra domain을 이용하는 것이 보다 정확한 invariant를 구할 수 있었다.
2) testset에 있는 프로그램들이 모두 toy program 수준의 프로그램이다.
3) talk 자체가 너무 세부적인 내용에 집중했다. 전체적인 일의 의미와, 이것의 필요성에 대해서 더 많이 이야기해주었으면 보다 많은 사람들의 관심을 받지 않았을까 생각했다.

4. PCC 2009 발표 정리

8월 15일에 PCC workshop이 진행되었다. 2개의 invited talk과 7편의 논문이 발표되었다. Workshop 정보와 발표된 논문을 http://ti.arc.nasa.gov/event/pcc09/에서 찾아볼 수 있다. 전체 발표된 논문과 talk중에서 인상적이 었던 것들은 정리하면 다음과 같다.

Invited Talk:
Mending the Gap, An effort to aid the transfer of formal methods technology (8/15)
Kelly Hayhurst, NASA Langley.

Kelly Hayhurst는 NASA에서 RTCA/DO-178 이라고 하는 "항공산업체가 지켜야할 소프트웨어 표준"을 손질하는 일을 20년간 하고 있는 사람이다. 컴퓨터를 전공한 사람도, formal method를 세세하게 알고 있는 사람도 아니지만, industry에서 formal method가 어떻게 사용되고 있고, 어떠한 대접을 받고 있는지를 적나라하게 들려주었다.

1) 항공 산업에서 소프트웨어가 차지하는 비중은 막대하다. Cyber Physical Systems - An aerospace industry perspective, Don Winter, VP. Boeing Phantom Works. Nov 2008에 따르면 비행기 1대를 제조하는데 들어가는 비용중 70%가 소프트웨어에 사용된다고 한다. 전체의 20%는 소프트웨어 개발에 사용되고, 50%는 만들어진 소프트웨어의 검증(verification)에 소요된다고 한다. "항공우주 산업은 소프트웨어 산업이 되었다."

2) 이렇게 verification이 중요하지만 정작 formal method가 사용되고 있지 않다. 미국에서 formal method를 이용하는 항공회사는? 없다! 전 세계에서도 Airbus가 유일하다. 왜?
 a) inadequate tools
 b) inadequate exmaples
 c) "build it and they will come" expectation (일단 비행기 system을 만들고 나면, 그 이후에 formal method를 적용할 수 있을 것이라는 생각)
 d) "Formal method is for PhDs only" (어려운 것이라는 막연한 선입견)

3) 그럼에도 불구하고, RTCA에서는 2005년부터 그들의 소프트웨어 표준 문서에 formal method를 추가하는 노력을 기울이고있다. 이것은 항공 시스템이 발전하면서 비행기 뿐만 아니라 전체 항공 기반 시설에서 소프트웨어가 차지하는 비중이 커지고 있기 때문이다. Airbus의 성공적인

사례가 고무적이며, 미국 회사들을 설득하는데도 유용하게 이용된다고
한다.

Efficient Type Representation in TAL (8/15)
Juan Chen

문제 : TAL(Typed Assembly Language)에서 모든 type 정보를 기록하면 type
정보의 크기가 너무 크게 된다. 그렇다고 모든 type 정보를 inference하게
되면 type inference의 부하가 크게 된다. 양 극단 사이의 trade off가
존재할까?

해결책 : "Type checking 시점에서 iteration이 필요하지 않게 한다."는
조건하에서 다음과 같은 방법을 이용해서 type annotation의 양을
줄여나간다. 1) 공통된 subterm의 type을 공유하게하고, 2) pointer type을
function type으로 표현하고, 3) 중복되거나 불필요한 metadata들을
제거하고, 4) data section에 해당하는 type information들을 제거하고, 5)
공간 절약을 위해서 특별한 interger encoding을 사용한다.

Towards a Certified Lightweight Array Bound Checker for Java Bytecode (8/15)
David Pichardie

JAVA 프로그램에 대해서 array bound check이 부분적으로 안전하게
제거되었다고 '주장하는' bytecode를 받았다고 하자. 이 프로그램을
수행하기 전에 이 프로그램의 수행이 안전하다는 것을, 다시 말하면 이
프로그램에서 제거된 array bound check이 정말로 불필요한 것임을, 어떻게
확인할 수 있을까?

Abstraction-carrying code framework에서는 이 확인작업을 위해서 프로그램과
함께 받아 온 고정점(fixed-point)이 실제 고정점인지를 확인한다. 이 확인 작업은
시간 복잡도는 작지만, 고정점 계산을 위한 추상 연산자(abstract operator)들을
구현해야한기 때문에 trusted base가 커질 수 있는 단점을 가지고 있다.

이 논문에서는 고정점 확인 작업에서 고정점 계산 작업에서 사용했던
프로그램 의미(semantics)에 대한 요약 의미(abstract semantics)를
사용하는 방법을 제시하였다. 이 방법을 사용하게 되면 고정점 확인 작업을
위해서 필요한 추상 연산자의 종류나 복잡도를 줄 일 수 있고, 따라서
trusted base의 크기를 줄일 수 있다.

Soonho Kong