Trip Report

OPLSS, MLPA, CADE

July 22 – August 10, Eugene, Oregon, US & Montreal, Quebec, Canada

들어가기에 앞서 이 이야기는 포항공대 통합과정 임현승군의 경험을 토대로 작성하였음을 알려드립니다.

1. Summer School on Theory and Practice of Language Implementation

July 23-31, 2009

University of Oregon, Eugene, Oregon, US

Oregon Programming Language Summer School (OPLSS) 은 올해로 8번째 개최된[1] 깊이 있고 체계화된 교육 프로그램입니다. 올해는 Theory and Practice of Language Implementation이라는 주제로 흥미진진한 강좌가 많이 개설되었는데요, 본 글에서는 그 중에 인상 깊었던 강좌 몇 개를 소개하고자 합니다. OPLSS-09 홈페이지에서 강좌 소개뿐만 아니라 강의 자료 및 동영상도 제공하고 있으니 자세한 내용은 홈페이지를 참고하시길 바랍니다.

사진 1 박종현군(왼쪽)과 필자(오른쪽),
우리에게 여름학교 기간 내내 맛있고 영양가 있는 다양한 음식을 단돈 $5에 제공해준 학생식당 앞에서

포항공대를 비롯하여 우리나라 대학 대부분에서는 프로그래밍 언어를 공부하시는 교수님들이 없거나 한 명뿐이어서 관련 수업을 다양하게 수강하기가 참 어렵습니다. 따라서 OPLSS 같은 여름학교를 통해서 각 분야의 전문가들에게 직접 가르침을 받는 것은 정말 좋은 기회라고 생각합니다. 이번 여름학교에는 한국에서는 저와 함께 역시 포항공대에 재학 중인 박종현군과 카이스트의 김철주씨, 이철우씨, 그리고 창병모

[1] 어디까지나 개인적인 추정으로 확실하지는 않습니다. 2002년부터 올해까지 OPLSS 홈페이지는 다음 사이트에서 찾아볼 수 있습니다: http://ix.cs.uoregon.edu/~jallen/

숙명여대 교수님도 참여하셨습니다.

이번 여름학교에서는 총 8개의 주제로 32개의 80분 강좌가 진행되었는데요, 아침 9시부터 오후 5시까지 매일 4개의 강의를 들어야 하는 강행군이 계속되었습니다. 이러한 강행군 덕에 오른쪽 사진에 보이는 바와 같이 수업 사이사이 쉬는 시간에 단잠에 취하시는 분들이 꽤 보였습니다. ㅎㅎ 물론 강의내용을 숙지하고자 쉬는 시간에도 열공하시는 학구파 분들도 꽤 있었습니다. ㄷㄷㄷ

자, 그럼 이제부터 각 강의 주제에 대해 간략하게 소개의 한 말씀 올리도록 하겠습니다. ^^

A. Algorithmic Program Synthesis

Ras Bodik University of California – Berkeley

서울대 공순호씨의 trip report를 보니 Bodik 교수님이 SAS-09에서 초청강연을 했더군요. 강의내용은 초청강연의 240분 확장판이라고 생각하시면 될 것 같습니다. 공순호씨가 잘 정리해주셔서 이에 대한 내용은 공순호씨의 trip report를 참조하시길 바랍니다. 쿨럭...

B. Continuations to Go

Olivier Danvy Aarhus University

Danvy 교수님은 Continuations에 관한 한 최고의 지성이죠. 또한 굉장히 열정적입니다. 번외편으로 거진 4~5시간 동안 Practical PhD Requirements라는 주제로 총합 242페이지에 달하는 방대한 양의 조언을 해주셨습니다. (주옥 같은 조언을 많이 해주셔서 좋기도 했지만 참 힘들었습니다. (>.<))

이 강의는 이론과 실재를 겸비한 참 재미있는 강좌였습니다. Continuations에 대해서 강의해주시고 예제 코드를 각자 continuation-passing style (CPS) 바꿔보는 연습을 할 수 있었습니다. 총 3개의 연습문제가 있었는데요, 심심하신 분들을 위해 문제를 소개하자면 (Objective Caml 문법으로 문제를 소개하겠습니다.) 첫 번째로 다음 두 함수를 어떤 list comprehensions (map, reverse, foldr 등)도 사용하지 않고 순순하게 CPS 함수로 작성하는 것입니다.

ㄱ. val list_of_suffixes: 'a list -> 'a list
e.g., list_of_suffixes [1;2;3;4;5] -> [[1;2;3;4;5]; [2;3;4;5]; ... ; [5]; []]
ㄴ. val list_of_prefixes: 'a list -> 'a list
e.g., list_of_prefixes [1;2;3;4;5] -> [[]; [1]; ... ; [1;2;3;4]; [1;2;3;4;5]]

참고로 CPS transformation의 기본 원칙은 다음과 같습니다.

1. Names intermediate results.
2. Sequentializes their computations.
3. Introduces continuations (first-class functions).

두 번째 문제는 주어진 regular expression과 string list가 일치하는지 검사하는 함수를 작성하는 것입

니다. 아래의 예제를 만족하는 함수를 CPS 함수로 작성하시면 됩니다. 힌트는 tail-recursive 함수입니다.

type regexp = Elem of string | Seq of regexp * regexp
e.g., Seq (Seq (Elem "임", Elem "현"), Elem "승")
= Seq (Elem "임", Seq (Elem "현", Elem "승"))
= ["임";"현";"승"] != ["임";"현승"]

세 번째 문제는 두 개의 lists [x1; ... ; xn], [y1; ... ; yn]이 주어졌을 때, [(x1,yn); ... ; (xn,y1)]을 계산하는 CPS 함수를 작성하는 건데요, 여기에서 주목할 점은 n 값을 모른다는 것입니다.

앉아서 수동적으로만 이론 강의를 듣는 것이 아니라 적극적으로 continuation에 대해 생각해보고 참여하는 멋진 수업이었습니다. 여담으로 대학원에 처음 들어와서 Selective CPS transformation에 대해 공부했던 기억이 새록새록 떠오르더군요. ㅎ

C. Control-flow Analysis of Higher-Order Languages

Matt Might University of Utah

이번 강좌에서는 함수형 언어에서 많이(?) 사용하는 정적분석기법 중 하나인 Control-flow analysis (CFA) 에 대해 자세하고 깊이 있게 다뤘습니다. 간단히 말해 CFA는 함수호출 시에 어떤 함수가 호출될지를 보수적으로 추정하는 기법인데요, 예를 들어 코드에 f (x) 라는 함수호출이 있을 때 f가 어떤 함수인지 정적으로 어림잡는 것입니다. 0CFA, 1CFA, *k*-CFA, ... , Environment analysis 등 종류도 다양한데요, Matt 교수님이 어찌나 빨리 발표를 하든지 처음에는 좀 따라가다가 엎친 데 덮친 격으로 몇 가지 배경지식까지 가정하고 진행하다 보니 나중에는 정신을 놓게 되더군요... (>.<) 다행히 (?) 다른 많은 친구들도 정신 줄을 놓고 있었다는 것을 나중에 알았습니다. ㅎㅎ

사진 2 매일 저녁 1~3시간 가량 학교 주변을 산책하고 중간에 근처 아이스크림 가게, 커피숍, 술집 등을 들리더군요. 처음 두세 번은 함께 했는데 나중에는 지쳐서 그냥 숙소에서 쉬게 되었습니다. ^^;

D. Managed Runtime Environments: Implementations and Opportunities

Chandra Krintz University of California – Santa Barbara

Krintz 교수님은 Java, C#, Python과 같은 언어에서 사용되는 bytecode 및 runtime environments에 대해서 소개해주셨습니다. 또한 Web service와 business applications 에서처럼 여러 언어로 작성된 응용 프로그램을 managed runtime environments를 이용해서 어떻게 support 할 수 있는지에 대해서도 다루시고 끝으로 현재 진행중인 cloud computing 연구에 대해서도 소개해주시더군요. 굉장히 자신감이 넘치시는 분이었는데, 강의 내용이 워낙 방대해서 기억에 남는 것은 별로 ... ^^;

E. Multi-Threaded Programming and Transactional Memory

Yannis Smaragdakis University of Massachusetts, Amherst

병렬 프로그래밍이 앞으로 대세가 될 것이라는 점에는 누구나 공감하실 거라 생각합니다. Smaragdakis 교수님은 병렬 프로그래밍의 한 유형인 multi-threaded programming에 대해서 소개하였습니다. Multi-threaded programming 방법으로 먼저 현재 가장 많이 사용되고 있는 monitor style programming을 설명하고 현재 활발하게 연구되고 있는 transactional memory에 대해 소개했습니다. 병렬 프로그래밍이라고는 박성우 교수님의 Parallel Programming이라는 대학원 과목을 수강하며 OpenMP와 MPI를 이용하여 C programming을 해본 게 전부인 저에게는 상당히 유익하고 재미있는 강의였습니다.

Monitor style programming에 대해서 잘 모르시는 분들을 위해 간략히 소개하면 mutexes (lock) 와 condition variables (wait, signal, broadcast) 을 이용해서 프로그래밍 하는 방법으로 Java에서는 모든 Object가 mutex 이고 condition variable 입니다. lock은 synchronized로, wait는 Object.wait로, signal은 Object.notify로, broadcast는 Object.notifyAll로 구현되어 있고, C와 C++에서는 PThreads를 이용해서 monitor style programming이 가능합니다. 자세한 내용 및 예제는 OPLSS-09 홈페이지에서 제공하고 있는 강의자료를 참고하시기 바랍니다.

Transactional memory는 monitor style programming의 단점을 (no modularity, lock is a global property, lack of composability) 보완하기 위한 대안으로서 database에 있던 개념을 차용한 것입니다. Monitor style programming의 단점을 잘 시사하는 가장 유명한 예는 여러분들도 다들 아실 bank account 문제입니다.

```
class Account {
        int balance = 0;
        public synchronized int withdraw(int amt) { ... }
        public synchronized void deposit(int i) { ... }
}
class Client1 {
        public synchronized void move (Account a1, Account a2) {
                a2.deposit(a1.withdraw(10));
        }
}
```

위 Java code에서 withdraw 함수와 deposit 함수를 아무리 올바르게 구현했다고 하더라도 두 함수를

이용하여 구현된 move 함수는 올바르지 않을 수 있습니다. Transactional memory는 atomic code sections을 (all-or-nothing) 도입함으로써 위와 같은 문제를 우아하게 해결하는데요, 현재는 성능이 별로 안 좋다는 큰 단점이 있습니다. Software Transactional Memory: Why Is It Only a Research Toy? 라는 제목으로 ACMQUEUE에 개제된 article은 이와 같은 문제점을 강력히 시사하고 있는데요, Smaragdakis 교수님의 말을 빌리면 현재의 TM은 80~90년대의 garbage collector가 처한 상황과 유사하기에 앞으로 더 많은 연구가 진행되고 컴퓨팅 파워가 증가하면 transactional memory가 병렬 프로그래밍을 쉽고 안전하게 할 수 있는 가장 좋은 방법 중에 하나가 될 것이라 더군요. 앞으로 병렬 프로그래밍이 엄청 중요해질 것이라는 점은 자명하니 TM의 성능이 더 좋아지기를 기대해봐도 좋을 것 같네요. 관심 있는 분들을 위해 참고로 현재 Haskell에서 STM을 사용해볼 수 있답니다.

여담으로 Smaragdakis 교수님도 PhD rants and raves 라는 주제로 번외강의를 해주셨는데요, 상당히 흥미롭더군요. 이 분 자료는 꽤 유명한 것 같던데, 관심 있으신 분들은 링크를 따라가서 한 번 읽어보시길 추천합니다.

F. Program Analysis for Computing Symbolic Complexity Bounds

Sumit Gulwani Microsoft Research

이 강좌에서는 특정 procedure에 특정 input이 주어졌을 때, procedure 안에 선언된 특정 control-location이 (loop의 conditional expression 등) 얼마나 많이 방문되는지를 계산하는 방법에 대해 다루었습니다. 물론 upper bound를 계산하는데요, 이것을 symbolic bound라고 한다고 하더군요. Symbolic bound를 계산하기 위해 먼저 procedure의 invariants를 찾아야 되는데요, colorful logic, fixpoint brush, program transformations 등과 같은 방법을 이용해서 invariants를 찾아낼 수 있다고 합니다. 이러한 방법을 통해 invariants를 찾고 나면 symbolic bound를 계산할 수 있다더군요. 주제 자체는 굉장히 흥미로웠습니다.

G. Pointer analysis

Ondrej Lhotak University of Waterloo

Lhotak 교수님은 Java와 같은 object-oriented languages를 위한 포인터분석기법에 대해서 소개해주셨습니다. 포인터분석은 주로 C 언어에서 많이 하는 줄로만 알았는데 Java에서도 많이 하고 있나 보더군요. 상속이나 캐스팅, 오버로딩 등 때문에 포인터분석이 어렵고 또한 필요하다네요. 한가지 흥미로웠던 점은 서로 별 연관이 없어 보이던 control flow analysis와 pointer analysis 사이에 유사점이 꽤 많다는 것이었습니다.

H. Garbage Collection and the Metronome GC

David Bacon IBM Research

이번 강좌에서는 IBM에서 개발했던 garbage collector 중에 하나인 Metronome Garbage Collector에 대해 소개했습니다. Metronome은 현재 IBM에서 개발한 WebSphere Real-time Java Virtual Machine에서 사용되고 있습니다. 강의는 기술적인 세부사항을 다루기 전까지는 정말 흥미로웠고 재미있었습니다. Garbage collection의 방법으로는 크게 네 가지가 있습니다: "Stop the world", parallel, concurrent, and incremental garbage collection. Metronome은 그림 5에서 볼 수 있듯이, parallel, incremental, and concurrent garbage collection을 실시간으로 지원합니다. 흥미로운 데모로 실시간 피아노연주를

일반 Java garbage collector를 이용하여 처리한 경우와 Metronome을 이용해서 처리한 경우를 비교해보시기 바랍니다. 자세한 Metronome 설계 및 구현 방법에 대해서는 OPLSS-09 홈페이지에서 제공하고 있는 강의자료를 참고하시기 바랍니다.

그림 1 "Stop the world" garbage collection

그림 2 Parallel garbage collection

그림 3 Concurrent garbage collection

그림 4 Incremental Garbage collection

그림 5 Parallel, incremental, and concurrent garbage collection

I. Abstract Interpretation

Patrick Cousot Ecole Normale Superieure / New York University

Cousot 교수님이 abstract interpretation에 대해서 세 번에 걸쳐서 강의를 해주셨는데요, 강의가 끝날 때마다 매번 ASTRÉE Static Analyzer 광고를 잊지 않으시더군요. ㅎㅎ ASTRÉE는 Airbus 비행제어소프트웨어에 런타임에러가 없다는 걸 증명하는데 사용되었다고 하더군요.

총평:
각 분야의 최고 전문가들에게 직접 강의를 들을 수 있다는 것은 참 즐거운 일인 것 같습니다. 강의 스케줄이 빽빽하여 다소 힘들기는 했지만 강사들의 친절하고 자세한 설명 덕분에 많은 것을 배울 수 있었던 귀중한 경험이었습니다. 기회가 되신다면 적극 추천하고 싶네요. 위의 왼쪽 사진은 저녁 피크닉 때 (피크닉도 두세 번은 했던 것 같네요.) 찍었던 사진이고 오른쪽 사진은 저녁식사 후 교정을 거닐다가 한 컷 찍어봤습니다. ^^

2. Modules and Libraries for Proof Assistants (MLPA) & 22nd International Conference on Automated Deduction (CADE)

August 2 – 7, 2009
McGill University, Montreal, Quebec, Canada

8월 3일에 개최된 MLPA 워크숍에서 "A module system independent of base languages" 라는 주제로 발표를 해야 하기에 Summer School이 끝난 다음날 부랴부랴 Eugene에서 그레이하운드 버스를 타고 Portland로 가서 비행기를 타고 Vancouver를 잠시 들렸다가 비행기를 갈아타고 Montreal에 가게 되었습니다.

우리나라 버스와 비교하면 그레이하운드는 생각보다 많이 좁고 더럽더군요. California 밑에서부터 출발하여 Portland까지 가는 버스다 보니 어쩔 수 없는 것 같다는 생각이 들었습니다. 특히 주목해야 할 점은 그레이하운드가 좌석제가 아니라 선착순제로 운영되고 있다는 점이었습니다. 정류장에 주차되어 있던 텅텅 빈 버스를 (오른쪽 사진) 타고 갈 줄 알고 안심하고 있었는데, 그게 아니더군요. 정류장에 제일 일찍 갔음에도 불구하고 줄을 늦게 서서 하마터면 버스도 못 타고 비행기도 놓칠 뻔 했습니다. (——;) 다행이 턱거리로 승차하게 되어서 무사히 몬트리올로 갈 수 있었지요. ㅎㅎ

A. MLPA

MLPA는 올해 처음 열리는 워크숍이지만 Derek Dreyer를 비롯하여 module system 전문가들이 꽤 참여하기에 다소 두려우면서도 기대되는 워크숍이었습니다. 저는 현재 진행 중인 연구에 대해서 발표하였는데 박성우 교수님의 귀중한 코멘트를 바탕으로 발표 슬라이드를 정리하고 Summer School 기간 동안 계속 발표 연습을 해서 그런지 개인적으로 평하자면 발표 자체는 그럭저럭 봐줄 만 했던 것 같습니다. 다만 발표 후 질문 답변은 다소 매끄럽지 못 했던 것 같은데, 영어로 실시간으로 자기 생각을 적절하게 표현하는 것은 역시 쉽지 않은 것 같습니다. 학회에 참석할 때마다 항상 느끼는 거지만 좀 더 분발해야겠습니다.

지난 7월에 있었던 두 번째 ROSAEC workshop에서도 간단하게 소개했지만, 제 연구 주제는 여러 언어에서 사용할 수 있는 모듈 시스템을 설계하는 것입니다. 고려하고 있는 모듈 시스템은 ML 계열의 언어에서 제공하는 모듈 시스템의 변형이고요. 기반 언어에 상관없이 간단한 interface 함수만 제공하면 nested

modules과 higher-order functors를 지원하는 것이 목표입니다. 현재 전체 시스템을 일차적으로 곧 완성할 수 있을 것 같으니 조만간 ROSAEC technical memo로 만나 뵙길 기대합니다. ^^

Invited talk은 four color theorem 증명을 Coq proof assistant를 이용하여 mechanize한 Georges Gonthier 박사님이 "Combinatorics for Theorem Proving" 이라는 주제로 말씀해주셨습니다. Gonthier 박사님은 Ariane 5 로켓 폭파 원인을 규명하는 팀에도 속해 있었더군요. 정말 대단하신 분인 것 같습니다. 그 외에 한 가지 흥미로웠던 발표는 Michael Franssen과 Mark van den Brand의 "Design of a Proof Repository Architecture" 라는 주제의 발표였는데요, 취지는 기존 증명을 데이터베이스화하여 새로운 증명에 이용하자는 것이었습니다. 기존 증명과 유사한 증명을 자동으로 해주자는 것이었는데요, 아이디어는 참 좋은 것 같은데 정말로 잘 될지는 두고 봐야 알 것 같습니다.

B. CADE

올해로 22번째인 CADE는 logic 분야의 권위 있는 학회로 8월 4일부터 7일까지 총 4일간에 걸쳐서 진행되었습니다. Logic 관련 다양한 주제들이 발표되었는데요, 배경지식이 부족해서 그런지 이해 가능했던 발표는 거의 손가락에 꼽을 것 같네요. (>.<) 총 3개의 Invited talks이 있었는데요, 첫째 날에는 Martin Rinard 교수님께서 "Integrated Reasoning and Proof Choice Point Selection in the Jahob System: Mechanisms for Program Survival" 주제로 말씀해주셨습니다. 제 기억이 맞는다면 거의 300장에 달하는 방대한 양의 슬라이드 자료를 준비해오셔서 엄청 빠르게 진행하시더군요... 셋째 날에는 "Building Theorem Provers"라는 주제로 Mark E. Stickel 박사님께서 invited talk을 해주셨는데요, theorem prover를 기초부터 실제로 만드는 방법에 대해서 이야기 해주실 줄 알았는데, 기대와는 달리 거의 theorem provers의 역사 정도의 강의가 아니었나 싶었습니다.

발표 중 개인적으로 가장 흥미 있고 재미있었던 것은 Sean McLaughlin과 Frank Pfenning 교수님의 "Efficient Intuitionistic Theorem Proving with the Polarized Inverse Method" 란 주제의 발표였습니다. (사실 다른 발표는 정말 거의 이해를 못 하겠더군요.) Intuitionistic logic은 박성우 교수님께서도 강의하시고 계시고 theorem proving은 평소에도 관심 있어 하던 분야라서 더욱 흥미진진한 발표였습니다. First-order logic 같은 경우에는 특별한 decision procedure가 없기 때문에 보다 효율적이고 expressive한 theorem prover를 만드는 게 중요한데요, 기존의 inverse method를 polarization, focusing이라는 기법을 이용하여 확장함으로써 보다 효과적으로 theorem proving을 할 수 있더군요. 이 기법은 Imogen이라는 theorem prover에 구현되어 있는데요, 관심 있으신 분들은 링크된 홈페이지를 참고하시기 바랍니다.

C. Montreal and Quebec City

학회 일정에 Old Montreal 관광이 포함되어 있어서 잠시나마 Montreal을 둘러볼 수 있었고, 학회 마지막 날에는 교수님과 함께 Old Quebec을 찾아가봤습니다. Montreal도 그렇고 Quebec city도 그렇고 정말 아름답고 좋은 곳인 것 같습니다. 그러나 안타깝게도 관광을 할 때마다 비가 오고 날씨가 안 좋았습니다. 특히 Old Quebec은 중세 유럽풍의 건물들로 이루어져있어서 굉장히 고풍스럽고 멋들어진 곳이었습니다. 기회가 된다면 맑은 날씨일 때 다시 한 번 가보고 싶은 곳입니다.

3. 맺음말

비행기만 총 7번을 타고 출발하여 돌아오기까지 장장 20일이 걸린 다소 긴 여정이었습니다. 정말 많은 것을 배우고 느끼고 돌아왔는데 글 솜씨 부족으로 여러분에게 얼마나 잘 전달했을지는 모르겠네요. 특히 프로그래밍 언어 관련 학회로는 이번 MLPA 발표가 저에게 있어서는 첫 발표였는데 영어가 모국어가 아니다 보니 긴장도 많이 하고 두렵기도 했지만 끝나고 나니 한결 홀가분하고 앞으로는 부족한 점을 보충하기 위해 더더욱 분발해야겠다는 다짐이 생기더군요. 한가지 언급하고 싶은 것은 CADE 학회에 CMU 졸업생 및 학생 분들께서 꽤 오셨는데, 참 좋아 보였습니다. 우리도 다들 더욱 열심히 연구해서 좋은 학회에서 많이 많이 만날 수 있으면 좋겠습니다. 끝으로 OPLSS를 비롯하여 MLPA, CADE에 참석할 수 있는 정말 좋은 기회를 주신 박성우 교수님께 감사를 드리며 본 글을 마칩니다.