
# PLDI 2012

Beijing, China, Jun. 11th - Jun. 16th

**소프트웨어 무결점 연구센터 이원찬**

## PLDI 2012

### 8,000미터 고봉을 오르다

PLDI 2012 학회에 다녀왔다. 그것도 당당히 두 편의 논문을 들고. PLDI 학회는 POPL 학회와 함께 프로그래밍 언어 분야의 최고 학회이다. 특히나, 한국 학교에서 주도한 연구 성과가 PLDI 학회에 발표된 일은 이번이 처음이라고 한다. 두 논문 모두에 보탬이 되었다는 사실에 뿌듯했다.

## PLDI 학회

올해 PLDI 학회는 ECOOP 학회 및 다양한 위성 학회들(ISMM, LCTES)과 여러 워크샵, 튜토리얼과 함께 열렸다. 큰 잔치인 만큼 볼거리도 풍부할 것으로 기대했지만 입맛에 맞는 발표는 많지 않았다. 소문난 잔치에 먹을것 없는 격이랄까. 분석 밖의 것들을 편식했던 탓도 있었을 것이다. 하지만, 발표 수준이 기대했던 것에 못 미쳤던 까닭도 크다. 최고 학회라는 이름이 무색하게도 "정말 발표 잘한다"라는 인상이 드는 발표가 드물었다. PLDI 학회에 채택될 정도면 대단한 연구성과일 것인데도 말이다. 매 발표에 감탄하고 하나씩 배웠던 지난 POPL 학회와는 사뭇 분위기가 달랐다. 소통의 중요성을 다시금 느꼈다. 매끄럽지만은 않았던 학회 준비도 아쉬움이 남는다. 처음으로 아시아에서 열린 PLDI 학회인 만큼 기억에 남는 학회가 되길 바랐다. 안타깝게도 많은 부분에서 부족함이 드러났다. 학회 프로그램 책자가 예상보다 부족해 부랴부랴 다시 인쇄하기도 하고, 휴식시간에 나오는 음식도 부족하기 짝이 없어 이내 동나는 모습이 학회 내내 연출됐다. 중국의 길거리 음식을 선보인 뱅킷도 시도는 좋았지만 좀 더 정돈됐으면 했다. 학회 인원수에 비해 턱없이 좁은 장소에 길거리 음식을 손에 들고 서있는 모양이 참 서글펐다. 돌아갈 차편도 준비가 안된채 개별적으로 택시를 타야만 했다. 도대체 100불이나하는 뱅킷 비용이 다 어디에 쓰였는지 궁금했다. 기념품으로 나누어준 독서등과 티셔츠는 역시 메이드 인 차이나라는 말이 나오는 수준이었다. 만약 다음에 PLDI 학회가 한국에서 열린다면 좀 더 잘 준비하면 어떨까 하는 생각이 든다.

## 발표 이야기

우리 연구실이 이번에 발표한 논문은 두 개다. 하나는 제네릭 프로그래밍의 핵심을 담은 계산언어에 대한 논문이다. 제네릭 프로그래밍의 목적은 한 알고리즘 구현을 여러 다른 타입에 대해 사용하는 것이다. 이를 가능하게 하기 위해서는 타입에 따라 필요한 인자를 함수에 자동으로 넘겨주는 언어의 지원이 필수다. 타입을 보고 필요한 인자를 자동으로 찾아내어 넘겨주는 과정을 resolution이라고 한다. 우리의 계산 언어는 바로 이 resolution의 이론적 토대를 제공한다. 다른 하나는 정적 분석기에서 불필요한 계산을 없애 성능을 높이는 스파스 분석 기법에 대한 논문이다. 스파스 분석은 프로그램 실행 흐름이 아닌 분석에 사용되는 요약 메모리의 의존 관계를 따라 분석하는 방법이다. 이 아이디어는 컴파일러에서 사용하는 분석을 빠르게 하는데 이미 사용되어 왔다. 우리가 새롭게 발견한 결과는 스파스 분석을 하면서도 분석의 정확도를 유지할 수 있다는 사실이다. 요약 해석 기반 정적 분석에 적용할 수 있는 일반적인 스파스 분석 이론 또한 연구로 얻어진 새로운 결과다.

올해 PLDI 학회에서는 처음으로 소개 발표(teaser), 본 발표, 포스터 발표로 나누어 진행하는 방식을 도입했다. 논문 편수가 많은 PLDI 학회에서는 두 세션을 같은 시간에 동시에 진행한다. 그래서 종종 듣고 싶은 발표를 놓치게 되는 경우가 있다. 새로운 진행 방식은 이러한 문제를 어느정도 해결하고자 도입한 장치인 것 같았다. 소개 발표는 본 발표의 내용을 요약하여 1분간 발표하는 시간이었다. 학회 일정 제일 첫 머리에 둠으로써 이후에 들을 발표를 미리 정할수 있게 했다. 포스터 발표는 논문 내용을 담은 포스터를 준비해서 포스터를 찾은 사람들에게 설명해주는 시간이었다. 나는 첫 번째 논문의 포스터 발표를 맡았는데, 본 발표와 소개 발표는 브루노가 맡았다. 두 번째 논문은 우석이가 소개를, 학주형이 본 발표를, 기홍이가 포스터를 맡았다.

브루노의 발표는 학회 첫 발표였다. 복잡한 타입 시스템을 하나하나 설명하기 보다는 예제 위주로 핵심을 전달하려 애썼다. 하지만, 제네릭 프로그래밍이나 스칼라 임플리싯을 모르는 사람에게는 다소 어려웠을 것 같다는 느낌이 들었다. 학주형의 발표는 좋은 평가를 받았다. 전반부에는 연구의 동기와 스파스 분석의 핵심을 그림으로 쉽게 전달했고 후반부에는 정확한 정의를 제시함으로써 오해를 없앴다. 발표 당일날까지 자료를 가다듬고 연습하는 학주형의

모습은 정말 존경스러웠다. 발표 후 Anders Møller 교수님과 식사를 함께 했다. Anders Møller 교수님은 학주형 논문의 팬이라고 하셨다. 그동안 학주형의 분석기를 빠르게 만드는 연구들을 관심있게 지켜보고 계신다고 한다. 역시 훌륭한 연구는 사람들의 관심을 받는구나 하는 생각을 했다.

내 포스터는 생각보다 흥행했다. 많은 사람이 찾아주었고 나는 최선을 다해 설명했다. 대부분의 사람들이 발표를 듣지 않고 찾아온 경우가 많았다. 브루노의 발표가 학회 첫 발표라 그랬을 것이다. 포스터를 만들면서 발표를 듣지 않았어도 이해할 수 있을 포스터를 만들기 위해 애썼다. ROSAEC 워크샵 때 포스터 발표를 했던 경험이 도움이 됐다. 포스터 발표하면서 가장 즐거웠던 순간은 사람들이 지나쳐가다가 되돌아와 내게 말을 걸었을 때다. 내가 만든 포스터가 그래도 이해할 수 있을 것 같이는 보였나 보다. “꼭 당신 논문 읽어볼게요”라는 말을 들었을때는 참 고마운 생각이 들었다. 물론, 예의상 한 빈말일 수도 있었겠지만 말이다.

## 기억에 남는 발표들

### Deterministic Parallelism via Liquid Effects

Ming Kawaguchi, Patrick Rondon, Alexander Bakst, Ranjit Jhala

리퀴드 타입 시스템을 확장하여 병렬 프로그램이 안전함을 보이는 방법에 대한 발표였다. 안전한 병렬 프로그램이란 동시에 실행되는 코드가 접근하는 힙 영역이 겹치지 않는 경우를 말한다. 겹치는 경우란 같은 힙 주소에 동시에 값을 쓰는 경우를 말한다. 핵심 아이디어는 1) 코드가 접근하는 힙 범위에 대한 제약식을 코드의 타입으로서 모으고, 2) 동시에 실행되는 코드의 제약식들의 논리곱이 만족될 수 없음을 보이는 것이다. 예를 들어 함수 foo(n)가 배열의 n번째부터 n+4번째까지 값을 쓰는 경우 두 함수 호출 f(0)과 f(5)는 동시에 실행할 수 있다. 왜냐하면 foo(n)의 제약식은 $n \leq i \leq n+4$가 되어 $0 \leq i \leq 0+4$와 $5 \leq i \leq 5+4$를 동시에 만족시키는 i가 없기 때문이다.

항상 느끼지만 리퀴드 타입 시스템 연구는 참 꾸준하다는 생각이 든다. 2008년을 시작으로 매년 PLDI, POPL, CAV와 같은 좋은 학회에 연구를 발표하고 있다. 한 가지 주제를 가지고 꾸준히 탐구하는 자세는 본받을만 하다. 또한 느끼는 것은, Ranjit Jhala교수님 학생들은 항상 발표를 잘한다는 것이다. 한눈에 들어오는 예제를 가지고 문제와 논문의 핵심 아이디어를 알기 쉽게 풀어낸다. 영어가 모국어라도 저런 발표는 쉽지 않을것 같다는 인상을 매번 받는다.

한 가지 드는 의문은 안전한 병렬 프로그램을 위해 이렇게나 복잡한 타입 시스템을 사용할 필요가 있는가 하는 점이다. 결국 논문에서 하고 싶었던 것은 프로그램에서 경쟁 상태가 발생하지 않음을 보이는 것이다. 이는 일반적으로 어려운 문제이기 때문에 논문에서는 동적으로 스레드가 생성되지 않는 프로그램을 가정하고 있다. 게다가 힙에 대한 제약식을 만들기 위한 템플릿도 사용자가 입력하도록 되어 있다. 이정도 가정이 있다면 논문에서 같은 복잡한 타입 시스템을 새로 만들어서 안전성을 증명하기 보다는 요약해석 기반의 분석기를 만드는 것이 더 쉬웠을 것이다. 제약식이 붙은 타입 시스템의 타입 추론을 이미 발표한 터라 쉬운 확장을 찾아 논문을 썼다는 느낌을 지울수가 없다.

## Parallelizing Top-Down Interprocedural Analyses

Aws Albarghouthi, Rahul Kumar, Aditya Nori, Sriram Rajamani

병렬로 함수 간 분석을 하는 방법에 대한 발표였다. 함수 간 분석을 하는 방법에는 두 가지가 있다. 하나는 메인 함수에서부터 실행 흐름과 함수 호출관계를 쫓아가며 분석하는 방법이다. 다른 함수를 호출하지 않는 함수들의 요약된 분석 결과로부터 호출관계를 역순으로 쫓아가며 요약을 갱신해 나가는 방법이 다른 하나다. 후자의 경우 서로 호출하지 않는 함수들은 동시에 분석해서 요약을 만들 수가 있어 병렬로 처리기가 용이하다. 하지만, 전자의 경우에는 순차적으로 프로그램 실행 흐름을 쫓는 방식이라 병렬화가 어렵다. 알려진 병렬화 방법으로는 분석 작업을 관리하는 큐에서 서로 의존성이 없는 작업을 동시에 꺼내어 수행하는 방법이 있다. 연구실을 졸업한 희종 선배의 경험에 따르면 이 같은 방법으로 최대 4배까지 성능 향상이 있었다고 한다. 발표 또한 같은 종류의 함수간 분석을 대상으로 하고 있어 관심을 가지고 듣게 됐다.

기본 아이디어는 기존과 크게 다르지 않은데, 동시에 할 수 있는 작업은 동시에 하자는 것이다. 다만, 분석 작업이 필요에 따라 만들어지는 분석을 대상으로 하는 점이 다르다. 분석은 주어진 프로그램의 속성에 대한 질의에 답을 주는 계산으로 가정한다. 질의가 곧 분석 작업인 셈이다. 질의에 대한 답은 분석을 담당하는 각 스레드가 만들어 낸다. 만일, 질의에 답하기 위해 정보가 더 필요한 경우에는 분석 결과 대신 질의들을 돌려준다. 예를 들면, 분석을 하다가 함수 호출을 만나는 경우, 해당 함수의 분석 결과에 대한 질의를 만들어낸다. 이렇게 새로 생긴 질의들을 답을 기다리는 질의들과 취합한 다음 의존성이 없는 것들만 추려 각 스레드에 나누어주는 과정을 반복한다. 논문에서는 이 아이디어를 잘 알려진 맵-리듀스 모델로 설명하고 있다. 분석을 각 스레드에 질의로 나누어 실행하는 과정이 맵, 분석으로 얻어지는 질의에 대한 답과 새 질의를 취합하는 과정이 리듀스가 된다.

실험 결과도 알려진 것과 크게 다르지는 않았다. 8개 코어를 사용해서 평균 4배 정도, 최대 7.4배정도 좋아졌다고 한다. 여기서 들었던 근본적인 의문은 이런 방식의 분석이 과연 병렬화하는게 가능한가이다. 직관적으로 생각해보면 이런 종류의 분석은 분기문을 만날때만 작업을 병렬로 실행할 수 있다. 그렇다면 병렬화가 잘 되려면 1) 많은 갈래로 분기가 생기고 2) 각 분기에서 분석해야 할 양이 많아야 한다. 하지만, 사람이 짤 수 있는 코드 중에 이런 코드가 과연 얼마나 될 것인가. 평균 4배 좋아졌다는 말은 분기문 깊이가 두 단계정도라는 것을 방증할 따름이다. 그 외에도 논문에서 제기한 문제가 분석의 scalability였음에도 실험 결과는 문제를 해결했다고 보기에 무리가 있다. 문제인 scalability를 분석할 수 있는 프로그램 크기로 잡았다면 고작 2만 라인으로는 부족하다(이 부분은 실제로 이광근 교수님께서 발표가 끝나자 마자 바로 지적하셨다). 만일 scalability가 늘어나는 코어 수를 잘 활용하는 것을 의미했어도 실험 결과는 여전히 실망스럽다. 코어 8개로 한 실험을 가지고 시스템이 scalable한지를 논의하는 건 무리가 있다. 더군다나 논문에서는 맵-리듀스 모델을 고려하고 있음에도 말이다. 발표를 보면서 우리 연구실이라면 같은 주제로 더 나은 논문을 쓸 수 있지 않았을까 하는 생각이 들었다. 안그래도 우석이가 스파스 분석을 병렬화하는 연구를 하고 있다고 하니 기대를 해 볼만 하다.

## Test-case Reduction for C Compiler Bugs

John Regehr, Yang Chen, Pascal Cuoq, Eric Eide, Chucky Ellison, Xuejun Yang

C 컴파일러 테스트 케이스의 크기를 자동으로 줄여주는 도구에 대한 발표였다. 이 논문을 발표한 팀은 작년에 C 컴파일러의 테스트 케이스를 자동으로 생성하는 Csmith라는 도구를 발표한 바 있다. Csmith는 무작위로 컴파일러 버그를 일으키는 C 코드를 생성하는데, 코드가 불필요하게 복잡했다고 한다. 비단 Csmith뿐 아니라 실제 GCC 버그 리포트에도 수십 킬로바이트의 코드가 첨부돼있는 경우가 허다하다고 한다. 하지만, 버그를 재현하는데 가장 좋은 테

스트 케이스는 버그를 내면서 “군더더기 없는” 코드일 것이다. 버그와 관련 없는 코드가 많으면 그 원인을 사람이 파악하는데 방해가 되기 때문이다. 그래서 군더더기 없는 테스트 케이스를 만들어주는 C-Reduce라는 도구를 만들었다고 한다.

C-Reduce는 C 테스트 케이스를 받아서 50여가지의 변환 규칙을 반복 적용하여 크기를 줄여나간다. 그 과정에서 버그가 더 이상 발생하지 않는 케이스는 제거한다. 또한, 의미가 정의되지 않는 코드도 제거한다. 예를 들어, 변수를 초기화 없이 사용하는 경우 코드의 의미가 정의되지 않는다. 변환 규칙은 코드의 크기가 줄어드는 방향으로 코드를 변화시키며 의미를 꼭 보존하진 않는다. 예를 들어 사용된 이름을 간단하게 한다든지, 함수 인자를 없앤다든지, 타입이 같은 변수들을 한 변수로 치환한다든지 같은 방법을 쓴다. 기술적으로 특별한 것은 없지만 50가지가 넘는 변환 규칙을 생각날때마다 하나씩 구현하고 추가한 그 노고가 인상깊었다.

도구의 성능도 꽤나 훌륭했다. 30~300킬로바이트 크기의 테스트 케이스를 평균 500바이트 이하로 줄일 수 있었다. 이는 기존도구들에 비해 25배나 나은 결과라고 한다. 기존 컴파일러들의 버그 관리 시스템에서 채택해도 괜찮은 수준일 것 같다는 생각이 들었다.

### RockSalt: Better, Faster, Stronger SFI for x86

Greg Morrisett, Gang Tan, Joseph Tassarotti, Jean-Baptiste Tristan, Edward Gan

x86 실행 코드가 보안 규칙을 준수하는지 검사하는 분석기를 Coq으로 구현한 논문이다. SFI(Software-Fault Isolation)이란 소프트웨어에 오류가 발생해도 전체 시스템에는 영향을 미치지 않도록 격리하는 것을 말한다. 이를 위해 소프트웨어를 특정 메모리 접근이나 특정 동작만 허용하는 공간, 일명 샌드박스에 가두는 방법을 사용한다. 실행하기 전에 실행 코드가 샌드박스의 보안 규칙을 준수하는 지 검사하는 분석기로 분석하는 것이다. 예를 들어, 분석기는 실행 코드가 샌드박스가 허용한 메모리 영역에만 접근하는지를 검사해야 한다. 여기서, 분석기를 올바르게 만드는 것이 매우 중요해진다. 왜냐하면 분석기의 결함은 곧 보안 취약점으로 연결될 수 있기 때문이다. 논문은 이 분석기를 Coq을 사용하여 구현함으로써 그 안전성을 검증했다. 논문에서 대상으로 하는 샌드박스 시스템은 구글의 Native Client이다. 더 나은 사용자 경험을 위해 크롬 브라우저 안에서 실행코드를 수행할 수 있도록 하기 위해 만든 시스템이다. 기존 Native Client에도 실행 코드를 분석하는 분석기가 포함돼있지만 그 안전성은 증명되지 않았다. 논문에서는 기존 분석기의 안전성을 증명하는 대신 Coq으로 새로 만드는 방법을 택했다.

앞의 논문과 마찬가지로 결과에 이르는 과정에 수반된 노력이 인상깊은 연구였다. x86 실행 코드의 보안 규칙 분석기를 만들기 위한 기술적 어려움은 기존 연구들에서 대부분 해결됐다고 한다. 거의 모든 노력은 x86 아키텍쳐의 실행 모델을 Coq 안에서 구현하는데 들어갔다고 한다. x86 아키텍쳐 자체가 매우 복잡하고 명령어 갯수도 수백가지나 된다. 게다가 개발 언어가 Coq이니, 그 어려움이 오죽하랴. 연구 과정에서 구축된 x86 실행 모델의 Coq 라이브러리는 다른 검증된 프로그램 개발에도 활용될 수 있을 것 같다.

### Application-only Call Graph Construction (ECOOP)

Karim Ali, Ondřej Lhoták

사용자 프로그램에 대해서만 메소드 호출 그래프를 만드는 방법에 대한 발표였다. 자바 분석에서는 바이트 코드를 사용하기 때문에 사용자 코드와 라이브러리 코드에 구분이 없다. 덕분에 소스가 없는 라이브러리도 함께 분석할 수 있다는 것은 장점이다. 하지만, 그 탓에 Hello, World! 예제만으로도 5천개가 넘는 메소드에 대한 호출 그래프가 만들어지는 것은 문제다. 논문에서는 라이브러리 안에서 일어나는 일을 안전하게 요약함으로써 사용자 코드에 대한 메소드 호출 그래프를 만드는 방법을 제안했다. 아이디어는 라이브러리가 사용자 코드와 따로 컴파일되었다는 사실을 활용하는 것이다. 따로 컴파일되었다는 말은 라이브러리가

사용자가 정의한 클래스나 메소드를 직접 접근할 수 없다는 말이 된다. 이 사실을 활용하면 라이브러리에서는 어떤일도 일어날 수 있다고 가정하는 경우보다 더 정확한 요약을 얻을 수 있다. 새로운 방법은 라이브러리 코드를 다 분석하는 기존의 방법보다 3배 정도 빠르며 7배 작은 호출 그래프를 만든다고 한다. 물론, 코드를 전체 다 분석하는 경우보다는 라이브러리에서 일어나는 일을 부정확하게 묘사하는 그래프다.

나도 이런 논문을 쓰고 싶다는 생각이 든다. 요즘 안드로이드 앱 분석을 위해 자바 분석기를 만들고 있어 비슷한 경험을 하고 있다. 하지만 바이트 코드를 분석하기 때문에 라이브러리까지 같이 분석되는 것을 나는 장점으로만 여겼던 것 같다. 같은 현상을 바라보고도 나는 왜 이런 생각을 못했을까. 발표를 들으며 이렇게 생각한 사람이 비단 나뿐 만은 아닐 것이다. 그렇기 때문에 이런 종류의 재발견, “리”서치가 소중한 것 같다. 나도 언젠가는 한참 머리를 긁적이며 ‘이게 정말 이렇게 되야만 할까’하고 궁리하다 무릎을 탁 치며 새로운 발견을 하는 순간을 맞이하고 싶다.

### Yogi: Property Checking via Static Analysis and Testing (Tutorial)

Aditya Nori,Sriram Rajamani

모델 검증은 프로그램을 요약한 모델을 사용하여 오류가 없음을 보이는 검증 기법이다. 모델이 도달할 수 있는 상태는 프로그램이 도달 가능한 상태의 안전한 요약이다. 따라서, 도달가능한 상태 집합의 안전한 요약에 오류 상태가 포함되있지 않으면 프로그램 또한 안전하다. 만일 모델이 오류 상태에 도달할 수 있는 경우 모델 검증 알고리즘은 오류 경로를 돌려준다. 오류 경로는 실제 프로그램의 오류 경로이거나 안전한 요약으로 인한 허위 경로일 수 있다. 실제 오류를 발견한 경우라면 프로그램의 오류를 발견한 것이 된다. 그렇지 않은 경우 허위 경로가 발생할 수 없는 이유를 사용하여 모델의 요약을 더 정확하게 만든다. 이 과정을 안전하다고 증명하거나 실제 오류를 발견할 때까지 반복한다. 모델 검증은 따라서 프로그램이 무한히 많은 상태를 가질 수 있는 경우 끝나지 않을 수 있다.

모델 검증의 성능을 좌우하는 것은 요약을 더 정확하게 만드는 단계이다. 요약이 정확할 수록 도달 가능한 상태를 탐색하는 과정에 시간이 많이 걸린다. 따라서, 오류 발견하거나 증명할 수 있을 만큼만 요약을 정확하게 만드는 것이 가장 이상적이다. 즉, 요약을 정확하게 만드는 횟수를 최소한으로 하는 것이다. 아직 오류 검증에 “필요한 만큼”이 얼마인지 모르기 때문에 기존 연구들은 요약을 정확하게 만드는 횟수를 최소화하는 데 집중됐다. 예를 들어, 주어진 허위 경로가 일어날 수 없는 이유를 일반화하여 요약을 정확하게 만드는 방법이 있다. 이는 요약 개선 한 번으로 정확히 “같은” 이유가 아니라 “비슷한” 이유로 허위 경로가 되는 것들까지도 없애기 위함이다.

Yogi는 테스팅을 사용해서 오류를 검증하는데 필요없는 허위 경로 판별을 없애는 방법으로 성능을 높인다. 우선, 모델 검증 단계 이전에 테스팅을 수차례 수행한다. 이때, 방문한 요약 상태를 기록해둔다. 이후 모델 검증에서는 방문한 요약 상태들에서 방문하지 않은 요약 상태로 진행하는 요약 경로 중 오류 상태에 도달하는 것을 찾는다. 만일, 그 경로 실제로 일어날 수 있는 경로라면, 경로 조건을 만족하는 변수의 값으로부터 새로운 테스트 케이스를 만들수 있다. 새로운 테스트 케이스는 이전에 방문하지 않은 요약 상태를 적어도 하나 방문하게 된다. 만일, 실제로 일어날 수 없는 경로라면, 아까 방문한 요약 상태들 중 오류에 가장 가까운 것을 오류에 도달하게 만드는 것들의 요약 상태와 그렇지 않은 것들의 요약 상태로 나눈다. 오류에 도달하게 만드는 조건은 오류 조건으로부터 가장 약한 선조건(weakest precondition)을 역산하여 구할 수 있다. 이 과정을 테스팅으로 오류를 찾거나 모델 검증이 오류에 도달하는 요약 경로를 찾을 수 없을 때까지 반복한다. 만일 테스팅 없이 모델 검증만 한다면 실제 방문 가능한 요약 상태와 아닌 것을 구별하기 위해 수 차례 요약을 정확하게 만들었을 것이다.

# 베이징 단상

아직도 베이징의 탁한 공기와 덕분에 걸린 여름감기가 기억에 남는다. 공항을 나서면서부터 베이징의 공기가 폐를 꽉 메운 탓에 답답한 느낌이 들었다. 무더운 중국의 여름날씨도 한몫을 했다. 택시를 타고 가는 길이 너무 갑갑해 창문을 열었지만 도통 개운한 느낌이 들지 않았다. 창 밖 도로는 어마어마한 수의 차들 가득 메워져 있었다. 공기가 탁한 이유를 알 것도 같다. 공기 탓인지 학회 셋째날부터는 목과 코에 감기가 찾아들었다. 마지막으로 감기에 걸렸던 때가 언제인지 기억도 안나는 나였는데 말이다. 그리 심하지는 않았지만 만리장성을 다녀오던 날에는 몹시 피곤했다. 이 탁한 공기가 바람을 타고 서울까지 불어올 것을 생각하니 갑자기 짜증이 났다. 더운 날씨와 감기 덕에 베이징 이곳 저곳을 돌아볼 기력이 없었던 것이 아쉬울 따름이다.

수도라는 명함이 무색하게 베이징 풍경은 90년대의 한국을 연상케 했다. 물론 몇몇 곳곳은 개발이 잘 된 모습이었으나 택시로 조금만 들어가도 어릴적 본 것만 같은 모습이 나타났다. 넓게 뚫린 8차선 도로, 그 위를 달리는 온갖 수입차들과 함께 그림에 놓기에는 어쩐지 위화감이 든다. 풍경 뿐 아니라 사람들의 행동양식도 90년대인것 같았다. 파란불에 건널목을 건너다 차에 치일뻔한 적이 한 두번이 아니다. 그때마다 당황스러웠던 것은 주위 사람들이 너무나도 태연하다는 것이다. 아무도 신호를 지키지 않는 것에 불만이 없는 눈치였다. 택시 운전도 위험천만이었다. 서울 사람들이 위험하게들 운전한다고 얘기하지만 베이징에 비하면 정말 안전운전이다. 만리장성 다녀오는 길에 앞이 보이지 않는 오르막 급커브에서 태연히 앞차를 추월하던 택시 운전기사 아저씨를 생각하면 지금도 주먹이 꽉 쥐어진다.

베이징에서 하나 정말 마음에 들었던 것은 음식이다. 지난 APLAS 학회때 타이완에서 기가 막힌 만두를 먹은 이후로 음식에서 그만큼의 만족을 느낄 수 있을지 의문이었다. 하지만, 역시 본토의 음식은 그 깊이가 달랐다. 추천하고 싶은 음식을 꼽으라면 북경 오리와 양고기 요리다.

북경 오리는 학회 첫 날 우리 연구실 사람들과 중국 국방대학의 Liqian Chen, 구글의 하정우 박사님과 함께했다. 전취덕이라는 베이징에서 가장 유명한 식당을 찾았다. 총 세 마리의 북경 오리를 시켰는데, 각 마리마다 한 명의 요리사가 배정되어 오리를 손질했다. 인건비가 싼 중국이라 가능한 일이라는 생각이 들었다. 맛은 생각보다 담백했다. 간을 연하게 하고 손질 과정에서 기름이 많이 제거된 덕분인것 같다. 실제로 요리사가 살을 한 칼씩 베어 나갈 때마다 기름이 새어나와 아래 쟁반을 가득 채웠다. 식감은 껍질이 바삭하면서도 살은 부드러웠다. 파와 춘장으로 추정되는 소스를 곁들여 전병에 싸먹으니 진정 일품이었다. 구운 오리를 충분히 맛보고 난 다음에는 손질하고 남은 부분을 튀겨서 내어왔다. 튀긴 오리는 간을 몹시 짜게 해서 내어 왔는데 혹시라도 혀에 남았을 오리 요리의 느끼함을 지워주었다. 음식을 내어오는 그 순서에 감탄했다. 마지막으로 남은 뼈로 고아서 만든 오리 탕이 나왔다. 정말 오리 한 마리를 다 먹은 인상을 받았다. 꽉 찬 위장 만큼이나 마음도 맛에 대한 감동으로 가득 찼다.

양고기 요리는 너무 맛있는 나머지 두 번이나 찾았다. 동래순이라는 100년 전통의 양고기 요리 전문점이다. 첫 날에는 양고기 샤브샤브를 주로 먹었다. 육수를 담는 솥이 특이하게 생겼는데, 신선로에 쓰이는 솥을 크게 만든 것 같은 모양이었다. 가운데 우뚝 솟은 부분에는 숯을 담아 솥에 있는 육수를 가운데서부터 데우는 방식이다. 솥의 턱이 꽤나 높아 앉아서는 젓가락으로 안에 있는 음식을 집기가 쉽지 않았다. 덕분에 음식을 먹는 동안 대부분 서 있어야만 했다. 하지만 서 있는 보람이 들게 만드는 맛이었다. 재료가 신선해서 인지 양고기하면 으레 떠오르는 냄새가 거의 문제가 되지 않았다. 옆 테이블 손님이 양꼬치를 먹는 걸 보고 우리도 시켜보았다. 한국에서 보던 손톱만한 고기가 꿰어진 꼬치와 달리 엄지손가락만한 덩어리가 주렁주렁 꿰어져있었기 때문이다. 맛도 정말 눈이 번쩍 뜨이는 맛으로, 고기에 간을 한 정도가 신통방통했다. 덕분에 한 동안은 서울에서 양꼬치를 못 먹을 예정이다. 첫 날 양꼬치에 좀 소홀한 느낌이 들어서 두 번째 방문에는 양꼬치 위주로 주문했다. 짧은 중국어로 “양로취 얼쓰 빠거”(양고기 스물 여덟 개)라며 주문해놓고 기뻐 박수치던 장면이 떠오른다. 같이 시킨 다른 음식들도 정말 맛이 좋았다. 한국에는 중국 음식이 없었구나 하는 깨달음을 주는 맛이었다.

학회 중간에 일상 음식을 시도하느라 먹었던 개구리 요리도 기억이 난다. 학회장 인근의 아파트 단지 안에 꽤나 믿음직스러운 음식점이 있었다. 알고보니 매운 음식을 전문으로 하는 곳이었다. 입구에 들어서니 내 손바닥보다 큰 개구리들이 어항 안에서 우리를 맞았다. 중국스러운 것을 주문해보자는 나의 권유로 개구리 요리를 시키기로 했다. 나중에 학주형이 다시 어항을 찾았을 때에는 우리를 맞아주었던 개구리들 중 다섯 마리가 사라지고 없었다. 개구리 얼굴을 봤던 탓일까 다른 사람들은 나만큼 열렬히 즐기는 눈치는 아니었다. 개구리의 맛은 진화단계를 입증이라도 하듯 어류와 조류의 중간 맛이었다. 닭고기 같은 맛이 나면서도 그 살의 식감은 생선을 떠올리게 했다. 살의 담백한 맛과 음식점 특유의 매운 양념이 어우러져 꽤 괜찮은 맛을 냈다. 다음에 혹시라도 중국을 또 방문한다면 꼭 개구리 요리를 먹어야겠다.

# 맺으며...

“얘들아, 우리 PLDI에 한 번 내보자” 학주형을 도와 스파스 분석 논문을 쓰기 시작할 무렵 이광근 교수님께서 하신 말씀이다. 아무도 가능할 것이라고 생각 못했던 일이었다. 이 말씀이 없었더라면 지금의 논문도 없지 않았을까 생각한다. 끝까지 지도해주시고 격려해주신 이광근 교수님께 감사드린다. 또, 논문을 쓰느라 함께 고생한 학주형, 기홍이, 우석이, 원태, 브루노, 톰, 이 모두에게 고맙다는 말을 전하고 싶다. 앞으로 뻗어나갈 각자의 연구에 건승을 기원한다. 마지막으로, 이 글을 읽지는 못하시겠지만 기꺼기 중국행 비행기를 지원해주신 UC Berkeley의 Dawn Song 교수님께도 감사드린다.