
# APLAS+CPP'2011

Kenting, Taiwan, Dec. 4th - Dec. 9th

소프트웨어 무결점 연구센터 이원찬

## 요약

공동 개최된 APLAS 학회와 CPP 학회에 다녀온 경험을 나누고자 한다. 인상깊었던 발표와 학회를 지켜보며 든 생각을 정리하였다. 또한, 우리 센터와 인연이 깊은 왕교수님과 나눈 이야기도 실었다.

## APLAS+CPP'2011

APLAS(Asian Symposium on Programming Languages and Systems) 학회는 이름과는 달리 세계 각국의 연구자들이 모이는 국제학회이다. 올해 참석자중 프랑스인이 가장 많았을 정도로 유럽 및 북미 연구자들도 논문을 많이 내는 학회이다. 특히 올해 처음 시작하는 CPP(Certified Programs and Proofs) 학회와 공동개최를 한 덕분에 프로그램 검증과 자동 증명에 강한 유럽인들의 참석이 두드러졌다.

CPP 학회는 올해 처음으로 열린 자동 증명 및 검증된 소프트웨어에 대한 학회이다. 검증된 소프트웨어 및 검증기 개발, 수학 이론의 자동 증명, 증명 보조기(proof assistant)를 사용한 교육 등을 다룬다. 증명기 자체를 다루는 ITP 학회나 증명기의 근간의 이론을 다루는 POPL 학회가 놓치고 있는 영역을 다룬다는 점에서 검증 기술 발전의 중요한 첫 걸음이라고 생각한다.

## 왕교수님과의 재회

이번 학회의 가장 큰 소득은 왕교수님을 다시 뵙게 된 것이다. 왕교수님은 CPP 학회의 홍보 의장(publicity chair)으로서 학회 운영 전반의 일을 맡으셨다. 특유의 환한 미소로 당신의 나라를 방문한 우리를 기쁘게 맞아주셨다. 안타깝게도 왕교수님과 그리 많은 시간을 함께 하지는 못했다. 몸담고 계신 Academia Sinica에는 대학원생이 없어 모든 학회 일을 직접 처리하셔야 했기 때문이다. 주로 식사시간에 교수님과 만나 이야기를 나눌 수 있었다.

주된 대화주제는 물론 왕교수님과 함께 연구하고 있는 학습 기반의 프로그램 분석(learning-based program analysis)이었다. 시급한 것은 지금 하고있는 학습 알고리즘을 사용한 종료 분석(termination analysis)을 마무리 하는 일이었다. 왕교수님은 지난 TACAS 학회의 제출 기한을 안타깝게 놓친 것을 다시 상기시켜주셨고, 나는 1월 첫 째주까지 초안을 작성하기로 약속드렸다. 이번 논문이 내가 처음으로 1저자가 되는 논문이라는 생각에 어깨가 무거워졌다.

왕교수님은 우리의 연구를 통해 도달하고 싶은 최종 목표에 대해서도 들려주셨다. 왕교수님이 그리시는 큰 그림은 학습 기반 프로그램 분석의 핵심에 대한 일반적인 이론이다. 왕교수님 생각은 그동안 학습 알고리즘을 적용한 성공적인 사례들이 단지 우연일리가 없다는 것이다. 나도 크게 공감했다. 프로그램 분석은 크게보면 프로그램이 가지는 상태에 대한 불변식(invariant)를 찾는 문제이다. 불변식은 프로그램의 상태변화를 기술하는 전이 함수(transfer function)의 고정점(fixpoint)으로 기술되며, 고정점 계산은 대부분 고정점 반복(fixpoint iteration)에 의존해왔다. 왕교수님과 나는 학습 알고리즘이 고정점을 계산하는 새로운 방식이 될 수 있다는 희망에 부풀었다. 지난 연구들을 돌이켜봐도 우리 연구는 불변식 찾는 새로운 기법에 대한 것이었다. 지난 TACAS 논문에서는 기능 검증(functional verification)을 위한 상태 불변식을, 이번 논문에서는 종료 분석을 위한 이행 불변식(transition invariant)을 구했다. 왕교수님과 나눈 이번 대화로 학습 기반 프로그램 분석의 끝에 도달해보고 싶은 마음이 간절해졌다.

## 타이완, 그리고 타이완 음식

한국에 돌아와 돌이켜봐도 타이완하면 제일 먼저 떠오른 것은 음식이다. 타이완 음식에서 마음에 들었던 점은 무엇보다 싱거웠다는 점이다. 짠 음식을 그다지 좋아하지 않기 때문에 내 기억속에 외국학회때 먹은 음식이라 하면 너무 짜서 혀가 마비되었던 기억밖에 없다. 찍은 사진을 봐도 음식 사진은 몇 장 없었다. 하지만 이번 학회 사진을 들추어보니 음식을 찍은 사진이 제법 많아서 적잖이 놀랐다. 음식 가격도 저렴해서 모두가 배불리 먹고 이 가격이 맞냐며 되물은 적이 한 두번이 아니었다. 만일 타이완을 방문하고자 하는 사람이 있다면 길에서 파는 만두와 우육면을 꼭 먹어보라고 권하고 싶다. 공항 면세점에서 파는 갖가지 과자들(특히, 파인애플 케익)도 맛이 좋았다. 입맛 까다로우신 할머니께서도 한 개를 다 드셨다고 한다.

# 견문록

## Tutorial

### Dependently Typed Programming in Agda

Shin-Cheng Mu

의존 타입 시스템(dependent type system)을 갖춘 Agda(“아그다”라고 읽는다) 언어를 배워보는 시간이었다. Coq 증명 보조기를 알고 있던터라 재미있게 들을 수 있었다. Agda는 Coq과 마찬가지로 마틴 뢰프(Martin Löf)의 타입 이론(type theory)에 기반을 두고 있다. 하지만, 증명에 좀더 초점을 맞춘 Coq과는 달리 Agda는 더 쉽게 의존 타입을 사용한 프로그램 작성하는데 초점을 맞추었다. 비록 세 시간의 짧은 튜토리얼이었지만 간단한 언어의 검증된 컴파일러의 구현같이 간단하지 않은 주제까지 다루었다.

Agda는 의존 타입으로 선언된 값의 패턴 매칭에 있어서 강력한 면모를 보였다. 예를 들어 길이가 1 이상인 리스트를 인자로 받는 함수의 경우, 그 인자에 대한 패턴 매칭 시에 길이 0에 해당하는 경우를 처리하지 않아도 되었다. 이는 Coq을 사용해오면서 꼭 있었으면 했지만 최근에 와서야 지원되기 시작한 기능이다. 오랬동안 Coq에서는 타입에 의해 인자가 가지는 경우가 제한되더라도 패턴 매치시에는 항상 모든 경우를 다 다루어야만 했다. Agda에는 택틱(tactic)이 없다는 점도 이맥스와 연동된 자동완성 기능을 통해 어느정도 보완이 되었다.

또한, Agda는 Coq에 비해 간결하고 배우기가 더 쉽다는 인상을 받았다. Coq과 비교해보기 위해 한국에 돌아와 튜토리얼 내용을 구현해보았다. Coq에서는 같은 내용을 더 어렵게 구현해야만 했다. 구현이 복잡해진 것은 내 Coq 실력이 미진한 탓이 클 것이다. 하지만, 고작 세 시간 배운 Agda로 만든 코드와 동일한 것을 꽤 오래 공부했던 Coq으로 작성할 수 없었던 사실에 놀랐다.

기회가 된다면 다음 연구에 Agda를 사용해보고 싶다는 생각이 들었다.

# Invited Talks

## Program Analysis and Machine Learning: A Win-Win Deal

Sriram Rajamani

기계 학습(machine learning)을 프로그램 분석에 적용한 APLAS의 첫 초청 강연이었다. 연사는 SLAM 모델 검증기를 만든 Sriram Rajamani였다.

발표의 첫 머리에서 대부분의 사람들이 놓치고 있는 주석(annotation) 찾기 문제의 중요성이 언급됐다. 주석이란 함수의 선, 후 조건이나 프로그램 중간 지점에서 만족하는 불변식과 같은 것을 말한다. 주석이 첨가된 프로그램에 대해 검증을 빠르고 정확하게 하는 기술은 이미 많이 다져져 있다. 하지만 주석을 붙이는 일은 프로그램의 의미를 이해해야 하기 때문에 사람의 지능이 요구되어 왔다. 그래서 그동안의 연구들은 주석이 주어져 있다고 가정하고 진행된 것들이 많다.

Sriram Rajamani는 이 주석 찾기 문제를 기계 학습으로 해결하고자 했다. 두 가지 예로, 소스-싱크 분석에서 소스와 싱크, 입력 세정기(sanitizer)를 기계학습으로 알아내거나, 메소드의 선조건(pre-condition), 후조건(post-condition)을 유추하는 문제에 적용했다. 그동안 기계 학습은 사물들을 분류하거나 연관시키는 것과 같이 사람의 지능이 요구되는 영역에서 능력을 발휘해왔다. 예로 언급된 주석 찾기도 사람이 더 잘 할 수 있는 부분이기 때문에 기계 학습을 적용한 것은 어찌보면 당연한 귀결인지도 모른다. 기계 학습으로 찾아진 주석은 그동안의 검증, 분석 기술로 안전하게 검사될 수 있기 때문에 옳은 분석 결과를 얻을 수 있음에는 변함이 없다.

우리의 분석 문제에도 기계 학습을 적용해 볼 수 있을 것 같다. 프로그램 의미를 따라 요약 상태를 계산하는 것 자체는 기계가 더 잘 할 수 있는 부분이다. 하지만, 프로그램 각 지점에서 어느정도 문맥에 민감하게 (context sensitive) 분석할지, 혹은 경로

에 민감하게 (path sensitive) 분석할 지 결정하는 것은 사람의 판단이 필요하다. 왜냐하면, 우리가 필요로하는 정확도를 얻을 수 있는 최소한의 문맥과 최소한의 경로는 사람만이 정확히 알 수 있기 때문이다. 이 문제에 기계 학습을 적용해서 정확하면서도 비용이 크지 않은 분석을 달성해보고 싶은 마음이 들었다.

## Software Verification with Liquid Types

Ranjit Jhala

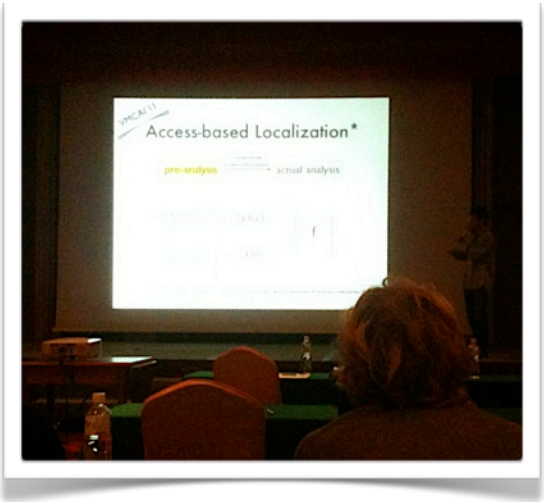
Ranjit Jhala의 최근 연구인 리퀴드 타입에 대한 강연이었다. 리퀴드 타입은 타입에 값에 대한 제약식을 포함하게 만든 타입 시스템이다. 값에 대한 제약식은 사용자가 넣어준 부품식의 견본을 사용해서 만든다. 그리고 이들 제약식 사이의 유추 관계는 타입 검사를 할 때 발생하는 하위 타입 관계에서 찾는다. 만일 찾아진 제약식 사이의 관계가 항상 참이라면 프로그램은 타입 검사를 통과하게 된다. 예를 들어 함수 호출 f x에서 f의 타입이 $\{v{:}int \mid 0 \le v \le 10\} \rightarrow int$ 이고 (여기서 타입 $\{v{:}int \mid 0 \le v \le 10\}$ 은 0과 10사이의 정수 값만을 가리키는 타입이다) x의 타입은 $\{v{:}int \mid 0 \le v \le 5\}$ 였다고 하자. x의 타입은 함수 인자 타입의 하위 타입이어야 하므로 $\{v{:}int \mid 0 \le v \le 5\} <: \{v{:}int \mid 0 \le v \le 10\}$을 만족해야 한다. 이 하위 타입 관계를 제약식 사이의 유추 관계로 바꾸면 $\{v{:}int \mid 0 \le v \le 5\} \Rightarrow \{v{:}int \mid 0 \le v \le 10\}$이 되고 이 식이 항상 참이 된다는 것은 SMT 풀이기를 사용해서 증명한다. 리퀴드 타입에 대한 연구는 처음 발표된 이래로 많이 진행되어 리스트와 같은 복잡한 자료구조에 대한 제약식도 유추하고 검사할 수 있다고 한다.

리퀴드 타입이 유용한 이유는 두 가지이다. 하나는 타입 검사 만으로도 프로그램이 안전하다는 것을 일부분 증명할 수 있다는 점이다. 발표를 듣고 논문을 찾아 읽어보았는데, 리퀴드 타입으로 OCaml 프로그램의 배열 접근이 안전하다는 것을 증명할 수 있었다고 한다. 이는 기존의 타입 검사만으로는 보장할 수 없었던 성질이다. 다른 하나는 리퀴드 타입으로 얻어진 값에 대한 제약식은 더 자세한 정적 분석을 위한 정보가 될 수 있다는 점이다. 리퀴드 타입을 사용한 타입 검사를 비용이 크고 더 자세한 정적 분석을 위한 전처리 분석으로 사용하면 어떨까 하는 생각이 들었다.

# 논문 이야기

## Access-Based Localization with Bypassing

Hakjoo Oh, Kwangkeun Yi




함수간 분석(inter-procedural analysis)의 성능을 높이기 위한 방안을 다룬 학주형의 논문이다. VMCAI'11에 발표된 지역화(localization)기법을 한 단계 더 향상시켰다. 지역화는 분석할 함수에서 접근하는 메모리 내용만 분석에 사용하는 테크닉이다. VMCAI'11에서는 한 함수 f에서 접근하는 메모리 내용이 그 함수가 호출하는 다른 함수에서 접근하는 메모리까지도 포함하도록 했다. 이번 APLAS 논문에서는 f에서 호출하는 함수들이 접근하는 부분의 경우 f를 거치지 않고 바로 해당 함수들의 분석에 넘겨지도록 바꾸었다. 이렇게 하면 많은 함수들이 재귀호출로 묶여있는 경우에도 각 함수들이 접근하는 메모리 부분만 가지고 분석할 수 있게 된다. 실험 결과 VMCAI'11에 비해 42%나 성능이 향상되었다고 한다. 특히, 재귀 함수 호출이 많이 사용된 프로그램에서는 77%나 좋아졌다고 한다.

APLAS 학회의 첫 날 발표 중 가장 인상적인 발표였다고 생각한다. 무엇보다 실제로 사용되는 GNU 소프트웨어를 분석대상으로 하는 점이 의미가 크다. 학회장에 있었던 다른 사람들도 많은 관심을 보였다. 특히, 오랫동안 분리 논리를 사용해서 형태 분석을 연구해온 Peter O'Hearn이 지대한 관심을 보였다. 발표 후에도 학주형은 Peter O'hearn과 함께 식사하며 논문에 발표된 아이디어를 발전시킬 방법에 대해 이야기 나누었다고 한다.

## Extending Hindley-Milner Type Inference with Coercive Structural Subtyping

Dmitriy Traytel, Stefan Berghofer, Tobias Nipkow

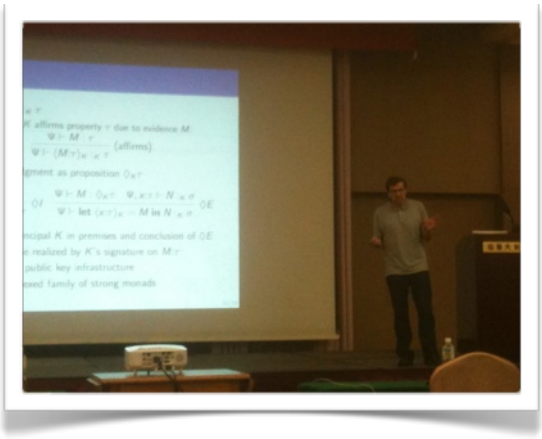
하위 타입(subtype)이 있는 타입 시스템에서 타입 강제(type coercion) 구문을 유추하는 완전한(complete) 알고리즘을 최초로 제안한 논문이다. 핵심은 하위 타입 관계에 대한 제약식(constraint)을 모아서 한꺼번에 푸는 것이다. 기존의 방법에서는 하위 타입에 대한 제약식이 발생할 때마다 지역적으로 해를 구해나갔다. 하지만, 이렇게 하면 해를 구하는 순서에 따라 타입 강제 구문의 유추가 가능한데도 실패하는 경우가 생긴다. 예를 들어, 정수 타입 Z와 자연수 타입 N이 주어져 있고 N이 Z의 하위 타입이라 하자. 또, N 타입의 값을 Z 타입의 값으로 바꾸는 함수를 int라고 하자. 함수 호출 leq i n에 대해 i가 정수 타입 변수이고 n이 자연수 타입 변수라 하면, 첫 번째 인자의

제약으로 부터 leq의 타입이 정수 두 개를 받는 함수임을 할 수 있게 된다. 따라서 타입 강제 구문이 추가된 함수 호출은 leq i (int n)와 같이 된다. 하지만, 함수 호출 leq n i의 경우 첫 번째 인자의 제약으로부터 leq의 타입이 자연수 두 개를 받는 함수가 되어 버리기 때문에 유추에 실패하게 된다. 제약식을 모았다가 나중에 한꺼번에 푸는 방법으로 이 같은 불완전성(incompleteness)를 해결할 수 있다.

이 논문에서 제일 마음에 드는 점은 실제 Isabelle을 쓰면서 사람들이 겪은 문제를 해결하려는 시도라는 점이다. 수학 정리의 증명을 자동화하면서 정수를 실수로 바꾸는 일이 빈번하게 발생했고, 자동화로 이 귀찮음을 줄이는 것이 목표였다. 실제로 실수에 대한 부등식의 성질을 증명하면서 총 1061번의 타입 변환이 필요했는데, 이 중 80%나 유추할 수 있었다고 한다. 아직 불완전한 유추 알고리즘을 쓰는 Coq에도 이 아이디어를 적용해보면 많은 도움이 될 것 같다.

## Proof-Carrying Code in a Session-Typed Process Calculus

Frank Pfenning, Luis Caires, Bernardo Toninho



증명을 담은 코드 (이하 PCC)란 코드의 일부분으로서 그 코드가 올바로 동작한다는 증명을 가지고 있는 코드를 말한다. 용도는 남이 보내준 코드를 클라이언트가 다시 살펴보지 않고도 첨부된 증명이 올바른 것을 통해 안전한 코드라는 사실을 믿게 하기 위함이다. PCC를 위해 코드와 별도로 증명을 첨부할 수도 있지만 의존 타입을 갖는 코드를 사용하면 간편하다. 왜냐하면 커리-하워드 동일성(Curry-Howard Isomorphism)으로 통해 그 의존 타입을 갖는 코드가 바로 증명이 되기 때문이다. 예를 들어, 어떤 함수가 의존 타입 ∀x:nat.∃y:nat.y > x을 갖는다고 하자. 의존 타입이 의미하는 바에서 이 함수는 입력 값보다 큰 값을 돌려주는 함수라는 것을 알 수 있다. 그리고 이 의존 타입을 만족하는 함수의 구현이 바로 그 함수가 올바르다는 증명이 된다.

발표된 논문에서는 PCC를 더 잘 지원하기 위해 기존의 의존 세션 타입에 두 가지 기능을 추가했다. 하나는 클라이언트가 관심 없는 증명을 타입에 표시할 수 있게 하는 기능이다. 관심 없는 증명을 표시할 수 있으면 그 타입을 갖는 구현에서 증명을 다른 프로세스에 넘기지 않아도 된다. 예를 들어 타입이 ∀f:nat→nat.∃y:nat.∃q:y = f(y)라고 하자. 타입을 통해 알 수 있는 사실은 이 타입의 프로세스가 어떤 함수 f의 고정점을 구한다는 것이다. 하지만 이 프로세스가 넘겨주는 y를 f에 넣어보는 것으로 고정점임을 확인할 수 있으므로 증명 q는 보내지 않아도 된다. 따라서, 논문에서는 q가 필요 없는 증명이라는 사실을 ∀f:nat→nat.∃y:nat.∃q:[y = f(y)]와 같이 표현했다 ([A]는 A에 대한 증명이 어떻게 돼도 관계 없음(proof irrelevance)을 뜻한다). 다른 한 가지

확장은 다른 프로세스로부터 증명 대신 인증서를 넘겨받을 수 있게 하는 기능이다. 위의 예에서 프로세스가 이번에는 함수 f가 단조 증가 함수라는 증명을 받는다고 하자. 타입은 증명 p를 추가한 ∀f:nat→nat.∀p:Πx:nat.x≤f(x).∃y:nat.∃q:[y = f(y)]가 된다. 하지만 증명 p가 복잡해서 직접 증명이 올바른 것을 검사하기 힘들 수 있다. 따라서 증명 검사를 믿을만한 다른 검사기에 맡기고 인증서만 받는 방법을 생각해 볼 수 있다. 논문에서는 이를 타입에 ∀f:nat→nat.∀p:◇V[Πx:nat.x≤f(x)].∃y:nat.∃q:[y = f(y)]와 같이 표현했다. 여기서 p는 증명이 아니라 어떤 검사기 V가 발급한 인증서를 말한다. 이 두 가지가 장착된 의존 세션 타입과 프로세스 계산 언어를 가지고 실용적인, 증명을 담은 파일 시스템을 구현했다고 한다.

자료로만 만나던 Frank Pfenning 교수님을 발표로 만날 수 있어서 좋았다. 의존 세션 타입의 두 가지 확장에 대해 차근차근 설명하셔서 따라가기 쉬웠다. 특히나 발표 때 등장했던 예제들이 이해에 많은 도움을 주었다. 타입만 가지고도 서로 다른 구현들이 어떻게 동작할 지, 차이가 무엇인지 한 눈에 들어왔다. 타입의 강력함이 빛나는 순간이었다.

# 글을 맺으며...

학회 내내 머리를 맴도는 생각은 어떻게하면 도움이 되는 연구를 할 수 있을 까였다. 해마다 수많은 연구들이 여러 학회를 통해 발표된다. 허나, 모두가 실제로 도움이 되는 연구인지는 의문이 든다. 이번 APLAS도 CPP도 예외는 아니었다. 이유를 생각해보았다. 양홍석 교수님과 왕교수님과도 이야기를 나누어보았다. 그 까닭은 명료했다. 도움이 되는 연구에 이르는 길은 험난하지만 보상은 적기 때문이다. 예를 들어, 분석 기술이 사람들에게 쓰이려면 GCC, 아파치 같이 사람들이 지금 쓰고 있는 프로그램을 분석할 수 있어야 한다. 뿐만 아니라 분석기는 쓰기 편해야하고 분석 결과는 알기 쉬워야 한다. 이에 이르는 과정에는 새로운 결과보다는 지루하지만 꼭 필요한 공학적인 사안들이 많이 도사리고 있을 것이다. 내가 ROPAS에 합류하기 전 선배들이 Sparrow 제품을 만들어가는 과정이 그러했을 것이다. 허나, 도움이 되는 지경에 이르는 데 필요한 노력을 흔쾌히 인정해주는 학회는 많지 않아보인다. 학회는 대체로 새로운 연구 성과를 선호하기 때문이다. 기존의 기술을 잘 조합해서 실제 프로그램에 적용해본 내용의 논문을 낸다면 심할 경우 “이 논문은 단지 경험 보고서(experience report)에 지나지 않습니다”는 리뷰를 받을수도 있다.

무관의 위험을 뚫고 진정 도움이 되는 연구에 이르는 길은 진부하지만 부단한 노력뿐이라는 생각이 들었다. 얼마전 학교를 방문한 세계적인 로봇 공학자이신 데니스 홍 교수님의 강연이 생각난다. 학생들이 시키지 않아도 새벽을 넘어서까지 학교에 남아 연구한다고 한다. 이유는 간단하다. 쓸모있는 연구 결과에 이르려면 할 일이 너무나 많이 때문일 것이다. 그런 각고의 노력들이 쌓여 앞 못보는 사람들을 위한 자동차를 낳은 것이 아닐까. 마찬가지로 ROPAS의 많은 선배들의 노력의 결실로 Airac이라는 분석

기가 만들어지고 그 위에서 우리가 쓸모있는 여러 연구를 꿈꿀 수 있는 것 아닐까. 다만 한 가지 바람은, 도움이 되는 연구를 향한 많은 노력이 공허하지 않도록 공학적인 사안들, 방법론적인 측면을 다루는 학회가 많아졌으면 한다.

마지막으로 좋은 학회, 반가운 만남, 유익한 생각의 시간을 누릴 수 있게 해 주신 이광근 교수님께 진심으로 감사드린다.