Microsoft Research Cambridge 방문기

LLVM 컴파일러의 검증된 검산기를 위한 방문 연구

뉴턴의 사과나무. 아마도...

172013년 4월 15일 (일)
소프트웨어 무결점 연구센터
서울대학교 컴퓨터공학부 대학원 1학기
강지훈(jhkang@ropas.snu.ac.kr)

허충길 박사님을 만나 공동 연구를 수행하기 위해 조성근, 최준원 연구원과 함께 Microsoft Research Cambridge에 2013년 3월 18일부터 2013년 4월 2일까지 방문했다. LLVM 컴파일러의 검증된(verified in Coq) 검산기를 만들기 위한 연구를 시작하기 위해, 연구 동기와 방법론을 정리하고 할일을 일별해보고 일정을 짜는 작업을 수행하였다. 허충길 박사님이 연구에 필요한 지식을 전해주시기도 했다.

이국에 간다는, 그리고 새로운 연구를 한다는 설레임으로 가득찬 방문이었다. 영국 케임브리지는 날씨는 그다지 좋지 않지만, 날씨가 안좋아도 괜찮을 정도로 도시가 예쁘고 아기자기했다. 시내는 천 년이 된 교회 건물이 아직까지도 남아있을 정도로 역사적인 장소이기도 하다. 천 년의 시간을 넘나드는 느낌이 어우러진 케임브리지에서 2주 넘게 머무른 것만으로도 기쁜 일이었다. 사람들도 여유가 있어보이고, 음식도 맛있었다.

연구 이야기

1.방문 연구

학회 참석차 방문한 적은 예전에도 있었지만, 공동 연구를 위한 방문은 이번이 처음이었다. 학회 참석의 목적이 최신 연구 동향을 파악하고 서로 인사하는 것인 반면, 공동 연구는 보다 구체적으로 할 일과 함께 일할 사람이 정해져 있다는 특징이 있다. 예전에 다녀온 학회에서는 활발한 교류를 못했던 반면, 이번 방문 연구에서는 허충길 박사님과 짧지만 알찬 교류를 할 수 있었다고 생각한다. 학회는 학회의, 방문 연구는 방문 연구의 목적과 특징이 있고 다른 준비가 필요하다는 것을 깨닫게 되었다.

방문 연구를 하기 위해서는 사전에 준비가 많이 필요한 것 같다. 만나야만 할 수 있는 것이 무엇인지 세심하게 검토하고, 만나기 전에 할 수 있는 일은 최대한 많이 해놓고 가는 것이 좋다고 생각했다. 이번 방문 전에 LLVM과 Vellvm 관련 문서를 읽어보긴 했지만, 더욱 많이 읽어보고 더욱 많이 이해하고 갔더라면 더욱 좋았으리라고 생각한다.

2.LLVM 컴파일러의 검증된 검산기

a. 연구 동기

방문 기간 전에도 가장 힘들었던 작업은 강력한 연구 동기를 생각하는 것이었으며, 방문 기간중 첫 2-3일은 연구 동기를 생각해 내는 것에 집중할 정도로 정리하지 못했던 문제였다. 방문 후에 있었던 연구실 내부 세미나에서도 연구 동기를 잘 설명하기 힘들었었다. 연구 동기는 앞으로 연구를 잘 수행하기 위해 우리가 지금 가장 중점적으로 다뤄야 할 것이라고 생각한다.

영국에서 토의한 우리의 연구 동기는 다음과 같다. 우리가 MSR Cambridge에 방문해서 풀고자 했던 문제는 다음과 같다: 어떻게 신뢰할 수 있는 실용적인 컴파일러를 만들까? 이때 “실용적인” 컴파일러는 GCC나 LLVM같이 현장에서 제품으로 만들어진 컴파일러를 뜻한다. “신뢰할 수 있다”는 말은 컴파일러가 옳다는 것을 알기 위해 믿어야 하는 trust base가 작다는 것을 뜻한다. 신뢰할 수 있는 컴파일러를 만들기 위한 두 가지 방법이 있다. 하나는 컴파일러 자체를 기계적으로 검증(verification in Coq)하는 것이고, 다른 하나는 컴파일러의 검산기를 기계적으로 검증(verification in Coq)하는 것이다. 우리는 후자의 방법을 택하여 LLVM에 적용하기로 했다. 그 이유는 기존에 작성된 LLVM 컴파일러를 재사용하는 것이 가능하기 때문이다. 따라서 CompCert와 Vellvm과는 달리 빠른 컴파일이 가능하며, 검증된 컴파일러를 작성하는 것보다 더 적은 노력을 들이고도 검증된 검산기를 작성할 수 있다.

LLVM을 선택한 이유는 현장에서 제품으로 만들어진 컴파일러중에서 가장 이해하기 쉬운 컴파일러이며, 이미 University of Pennsylvania의 Vellvm 그룹이 Coq 위에서 LLVM semantics를 이론화하고 또 LLVM semantics에 관한 여러 성질을 증명했기 때문이다. 기존 연구를 공부하기 위해 LLVM Reference Manual/Pogrammer’s Manual 등 다양한 LLVM 문서와, Vellvm 그룹에서 작성한 논문과 코드를 보았다.

b. 검산 방법

검산기는 기본적으로 입력 프로그램과 출력 프로그램, 그리고 컴파일러가 제공하는 힌트를 입력받아 두 프로그램이 동등한지 검사한다. “검증된 컴파일러를 작성하는 것보다 더 적은 노력을 들이고도 검증된 검산기를 작성할 수 있”게 하기 위해, 검산기를 최대한 간단하게 설계한다. 검산기가 간단하려면 컴파일러가 최대한 구체적이고도 자세한 힌트를 제공해야 한다.

검산기가 최대한 구체적이고도 자세한 힌트를 제공받기 위해, 우리는 각 program point에 대한 invariant를 힌트로 제공받도록 디자인했다. 보다 구체적인 설명은 조성근 연구원의 슬라이드 (http://ropas.snu.ac.kr/talk/2013/0412_1.pdf) 7-13쪽에 서술되어 있다.

c. Instruction combine: 검산할 최적화

LLVM의 다양한 최적화중에서, 우리는 instruction combine을 검산할 최적화로 골랐다. 그 이유는 다음과 같다: 1) Yang et al.에 따르면 (Finding and Understanding Bugs in C Compilers. PLDI 2011) instruction combine은 LLVM 최적화 중에서 가장 오류가 많이 발견된 최적화이다. 2) Instruction combine은 간단하다. 비록 수행하는 최적화의 종류는 많으나, 종류 하나 하나가 전부 간단하고(예를 들면 (a+b)+c를 a+(b+c)로 바꾼다), control flow graph를 변경하지 않는 것으로 보인다. 위에서 언급한 조성근 연구원의 슬라이드 6-14쪽에 구체적으로 설명되어 있다.

3.Coq

하루는 허충길 박사님이 Coq에 대해 말씀해 주셨는데, 그 요점은 증명 스크립트를 최대한 구조화하고 자동화하라는 것이었다. 지금까지 Coq에 대해 배워오고 공부한 것이 “Coq으로는 무엇무엇이 가능하다”를 파악하는 것이었다면, 허충길 박사님께서 말씀해주신 내용은 “어떻게 하면 Coq으로 잘 개발할 수 있을지”에 대한 것이었다.

구조화의 예를 들면 다음과 같다:

```
Inductive sem_expr (state:State.t): forall (e:expr)
(val:Value.t), Prop :=
| sem_const: forall n, sem_expr state (expr_const n) n
| sem_undef: forall n, sem_expr state expr_undef n
| sem_var: forall x v (Hmapsto: Mem.MapsTo (State.varToLoc
state x) v (State.mem state)),
  sem_expr state (expr_var x) v
| sem_numop: forall o e1 e2 n1 n2 n
  (Hlhs: sem_expr state e1 n1)
  (Hrhs: sem_expr state e2 n2)
  (Hop: sem_op o n1 n2 n),
  sem_expr state (expr_numop o e1 e2) n.
```

Lemma를 쓸 때, 이름을 붙일 수 있는 곳에는 항상 이름을 붙여야 한다. 예를 들어 “forall x, P x -> Q x”와 같이 쓰는 것보다는 “forall x (Hpremise:P x), Q x”와 같이 쓰는 것이 좋다(위 그림의 Hmapsto, Hlhs, Hrhs, Hop). 이러면 Coq이 자동으로 변수를 생성하는 것을 최대한 피할 수 있고, 증명하는 도중 나타나는 이름들이 우리가 명시적으로 준 이름들이 되기 때문에 가독성을 높일 수 있

다.

```
Lemma sem_exprs_proper: Proper (State.eq ==> eq ==> eq ==>
iff) sem_exprs.
Proof.
  unfold Proper, "==>"; intros; subst.
  revert y1.
  induction y0; intros; constructor; intros; inv H0;
constructor.
  { exploit sem_expr_proper; eauto; intro.
    apply H0; auto.
  }
  { apply IHy0; auto. }
  { exploit sem_expr_proper; eauto; intro.
    apply H0; auto.
  }
  { apply IHy0; auto. }
Qed.
```

그외에도 C언어에서 중괄호 {}로 block을 지정하듯이, Coq 스크립트에서도 중괄호 {}로 block을 지정하면 더 가독성이 높아진다는 것을 배웠다.

자동화의 예를 들면 다음과 같다. Tactic들을 최대한 잘 이해하고 활용해야 한다. 허충길 박사님과 토의하며 “match goal with”와 같이 똑똑한 tactic을 어떻게 하면 잘 사용할 수 있을지 많이 배웠고, 특히 허충길 박사님이 직접 개발하신 tactic들을 하나 하나 설명듣는 시간을 가졌다. Coq 스크립트를 자동화하면 할수록 개발 기간이 단축될 수 있으며, 스크립트 변경에 더 잘 대응할 수 있게 된다 (definition을 조금 고쳐도 증명이 깨지는 경우가 많은데, 자동화하면 definition을 조금 고치면 증명도 조금만 고쳐도 될 가능성이 커진다).

Coq으로 개발할 일이 많은 이상, 지금 눈앞에 닥친 것을 개발하는 것과 동시에, 어떻게 하면 더 잘 개발할 수 있을지에 대해 고민해야 한다는 생각이 들었다. 더 나은 디자인, 구조화, 자동화를 찾아 나서는 지적인 여정은 아마 앞으로도 계속될 것이다.

다른 이야기

1.영국

영국은 날씨가 좋지 않다. 첫 날과 마지막 날에만 날씨가 좋았고, 다른 날은 흐리고 비오고 눈오고 춥고 바람불고 구름끼고 해도 없고 난리도 아니었다. 그런데 나는 별로 좋지 않은 영국 날씨가 좋았다. 특히 아기자기하고 예쁜 Cambridge와는 더 잘 어울린다고 생각했다. 하지만 방문하는 것이 아니라 생활하는 것이라면 어떨지는 궁금하긴 하다.

밥은 들어왔던 악명이 무색할 정도로 맛있었다. 지금까지 먹은 이탈리아 음식중 가장 맛있는 것은 Cambridge Pizza Express에서 먹은 것이고, 가장 맛있는 버거는 Cambridge Gourmet Burger이고, 가장 맛있는 치즈는 허충길 박사님 댁에서 먹은 치즈이고, 가장 맛있는 인도 요리는 Cambridge 것이고, 가장 맛있는 프랑스 음식은 Cambridge의 Galleria에서 먹은 것이며, 가장 맛있는 편의점 샌드위치는 London에서 먹은 것이다.

Cambridge는 매우 역사적인 도시인 한편, 생활감도 강하게 느껴진다. 시내에 천 년이 된 교회 건물이 아직까지도 남아있을 정도로 오랜 역사를 자랑한다. 그러나 역사속에 박제되었다는 느낌은 전혀 받을 수 없다. 오래된 교회 건물 역시 교회로 사용중이고, 800년된 University of Cambridge 건물에서 세계를 선도하는 연구를 하고 있다. 천 년의 시간을 넘나드는 느낌이 어우러져 있었다.

London은 영국의 중심지일 뿐만 아니라 한때 세계의 중심지였던 흔적을 찾아볼 수 있는 웅장한 도시이다. The British Museum에 전시된 물품과 The National Gallery에 걸린 그림에 압도당하는 즐거움을 느꼈다. Buckingham 궁전은 현존하는 왕의 거처라는 위엄과 5관광객들의 생기를 동시에 찾아볼 수 있는 곳이었다. Big Ben은 세계에서 제일 오래된 의회인 영국 의회가 지닌 자부심을 형상화한 것만 같았다. Covent Garden에서는 재래시장의 활기가 느껴졌다.

영어가 그나마 할 줄 아는 외국어인 만큼, 영국이 다른 외국보다 가깝게 느껴졌다. 다른 나라 공항과는 달리 영국 Heethrow 공항의 안내문은 영어로만 작성되어 있다. 세계 공용어라는 영어가 모국어라는 자부심이 느껴지는 한편, 그나마 내가 외국 생활을 할

수 있는 나라가 영어권 국가뿐일 것같다는 생각이 들었다.

2.University of Cambridge

허충길 박사님이 안내해 주셔서 University of Cambridge를 관광할 수 있었다. 가장 오래된 College중 하나인 King's College, Trinity College, St. Jones College 등 여러 칼리지들을 둘러보고, 각 칼리지 안에 있는 교회도 구경했다. Henry 8세의 흔적이 곳곳에 남아있는 King's College를 둘러보며 규모에 압도당하기도 하고, Trinity College에 남아있는 뉴턴의 사과나무(허충길 박사님의 말씀으로는 가짜라고 한다)를 보며 즐거워하기도 했다.

영국 방문의 최대 재미였던 Punting을 하며 Cam강을 돌며 University of Cambridge를 구경하기도 했다. Punting은 아주 긴 막대로 강 바닥을 짚으며 배를 조종하는 놀이이다. 허충길 박사님은 많이 해보셨는지 매우 익숙하신 반면, 내가 하면 헛돌기만 하고 잘 나가지 않았다. 다시 영국에 오면 꼭 다시 해보고 싶은 놀이이다. 오리들에게 빵 던져주며 즐겁게 놀았다.

3.Microsoft Research Cambridge

MSR Cambridge에는 PL 분야의 대가들이 많이 있었다. 우리 앞방에는 Tony Hoare가 있었고, Simon Peyton Jones는 맨발로 복도를 다니곤 했다. Georges Gonthier, Andrew Kennedy, Nick Benton, Josh Berdine 등 수없이 많이 들어본 이름들의 주인들을 직접 만나볼 수 있는 기회였다.

MSR에는 우리 분야인 PL 말고도 다양하고 재미있는 연구가 많았다. Xbox Kinect가 MSR이 만들어낸 작품이라고 한다. 로비층에 있는 Kinect로 함께 테니스 게임을 하기도 했다. Wii에서 컨트롤러를 잡고 하는 것과는 다른 맛이 있었다. 더 많은 동작을 인식할 수 있는 것 같았다. 하드웨어를 연구하는 그룹에서는 여기 저기 조립이 안된 부품들이 널려 있었고, 신기해 보이는 하드웨어들을 발견할 수 있었다.

음료가 무료로 제공되어 하루에 탄산수를 거의 2L는 마신 것 같다. 콜라, 사이다와 같은 탄산 음료, 차, 커피, 물 등 다양한 음료를 무료로 즐길 수 있었다. 계속 맛있는 것만 먹다가는 살이 많이 찔 것같아 물만 많이 마셨다. 덕분에 하루에 화장실에 15번은 간 것같다. 매일 책상에 수북히 쌓인 페트병을

보고 왠지 모르게 건강해진(?) 느낌이 들어 기분이 좋았다.

MSR Cambridge는 시내에서 15분 정도 거리에 위치한 교외에 자리잡고 있었다. 기차역에서 가까웠고, 우리도 MSR Cambridge 근처 5분 거리 정도에 호텔을 잡았다. 원래는 University of Cambridge 안에 있었는데, 얼마전에 이사했다고 했다. 심지어 우리가 방문한 기간중에도 마감 공사가 진행중이고 여러 장식이 추가되고 있었다. 이사했다고 실시한 소방 훈련에도 참여했다.

그외에도 많은 추억이 남아있다. 임시 통행증을 발급받아 사용하다가, 마지막 날에야 비로소 정식 통행증이 발급되어 무용지물이었던 것이 생각난다. 로비층의 식당이 싸고 맛도 상당히 괜찮았다. 특히 수요일마다 인도 음식을 파는데, 가격을 생각한다면(4500원 상당) 매우 훌륭했다는 것이 기억에 남는다. 점심 식사 후 잠시 즐겼던 당구대와 Kinect도 생각난다.

감사 인사

좋은 연구 기회를 주신 이광근 교수님과 소프트웨어 무결점 연구센터, 그리고 한국연구재단 여러분들께 진심으로 감사드립니다. 2주가 넘도록 거의 하루도 빠짐없이 토의하고, 방문 기간을 알차게 보낼 수 있도록 도와주신 허충길 박사님과, 저희를 집으로 초대해 주신 사모님께 진심으로 감사드립니다. 행정일을 물심양면으로 도맡아주신 행정원 여러분들께도 감사드립니다. 더욱 열심히해서 좋은 연구할 수 있도록 노력하겠습니다.