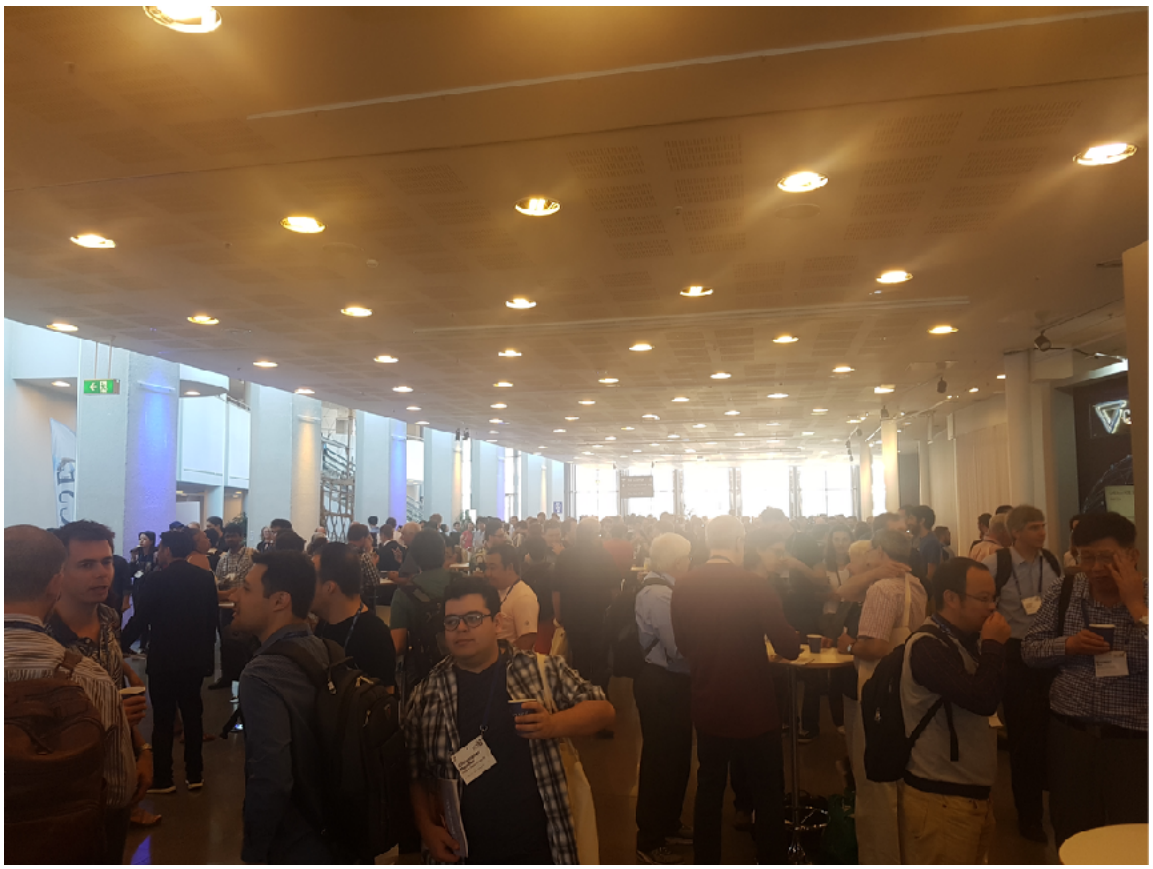
ICSE'18 Trip Report

예테보리, 스웨덴

홍성준

개요

FSE와 더불어 소프트웨어공학 분야에서 가장 우수한 논문들이 채택되는 ICSE에 논문 2저자로서 참여하게 되었다. 올해는 소프트웨어공학 50주년과 40번 째 ICSE라는 두 이벤트 때문인지, ICSE 사상 최대의 학회 참가 인원을 기록했다. ICSE는 굉장히 활동적인 학회이다. Facebook을 비롯한 여러 기업에서 다양한 부스를 열어놓았고, 자율주행 자동차를 학회장에 전시해놓기도 했다. 메인 컨퍼런스 오프닝에서 학회 대장이 프로그램 위원부터 시작해 우수 논문 수상 저자, 결국 참가자 전원을 자리에서 일으켜 세우면서 이번 학회의 신기록에 기여했음을 축하하기도 했다.

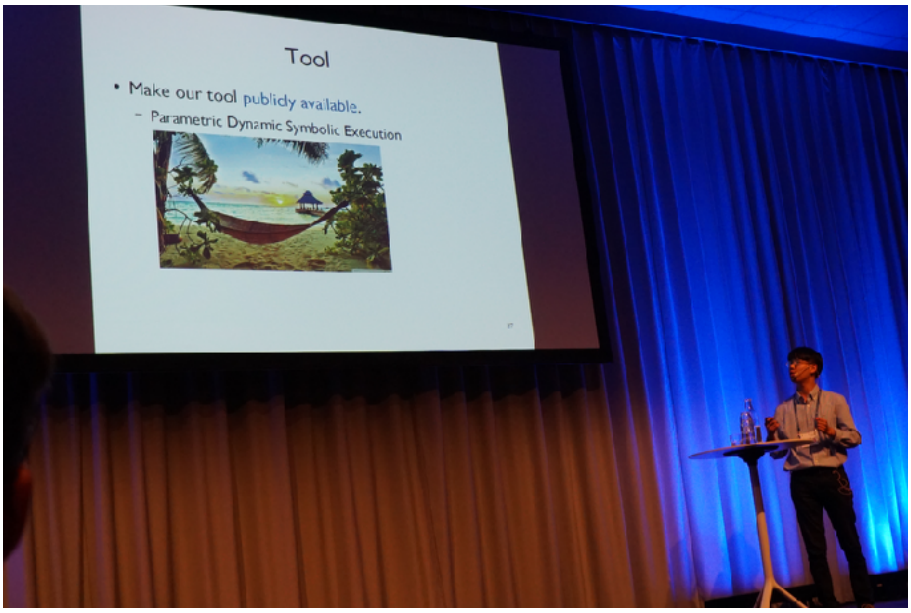


학회에서

• 논문발표

Automatically Generating Search Heuristics for Concolic Testing

차수영, 홍성준, 이준희, 오학주



이번 학회는 처음으로 논문 저자로서 참가하게 되었다. 우리 논문은 콘콜릭 테스팅의 고질적인 문제인 path-explosion을 해결하기 위해 주로 사용하는 탐색 전략을 자동으로 생성하는 방법을 제안했다. 기존에 제안된 많은 탐색 전략들은 실제로는 대부분 sub-optimal한 성능을 갖는다는 관찰로부터 시작한 연구이다. 이를 해결하기 위해 우리가 제안하는 방법은 상식적이다. 브랜치의 특성을 나타내는 feature를 정의하고 이를 통해 탐색 전략을 매개화 한 뒤, 좋은 탐색 전략을 만드는 문제를 좋은 매개변수를 찾는 문제로 reduce했다. 전략을 학습하는 과정 자체는 조악하지만, 문제가 굉장히 중요하고 테스팅 프로그램별로 탐색 전략을 생성하는 최초의 시도라는 점이 논문 accept에 크게 작용한 것 같다.

발표는 적어도 해당 세션에서는 가장 잘한 것 같았다. 무엇보다 동기가 매우 뚜렷했고, 덕분에 무슨 문제를 푼 것인지 확실하게 전달이 됐던 것 같다. 초반에 발표하면서 긴장하는 수영이 형의 모습을 볼 수 있었다. 네 개나 열린 테스팅 세션을 놔두고 Search-based SE세션에 배정된 것은 아쉬웠다. 그래도 질문도 많고 실제로 콘콜릭 테스팅을 이용하는 연구자가 우리 기술에 큰 관심을 보여서 기뻤다.

처음으로 출판까지 마무리된 연구라 기억에 더 남는다. 허기홍 박사님의 도움을 받아 Sparrow 위에 SE 엔진을 구현했다가 버렸던 것, 특히 연구 초기에 한가지 탐색 전략으로 모든 프로그램을 이기려고 고생했던 기억이 뚜렷하다. 프로그램의 특성별로 탐색 전략을 만들어야 한다는 우리 주장에 정면으로 반대되는 일인데도 말이다. 어쨌든 FSE에서 쓴맛을 한 번 봤지만 생산적인 리뷰 덕에 좀더 탄탄해진 상태로 ICSE에서 발표하게 되어서 다행이란 생각이 든다.

• 재밌었던 논문

이번 ICSE는 4~6개의 세션이 병렬로 열렸고, Empirical study, Human and Social aspects of Computing 등의 세션은 SE학회에 온 것을 실감하게 했다. 특히 Program repair 기술의 관심이 뜨거웠다. 최근들어 수많은 자동 패치 관련 논문이 쏟아져나오고 있는데, 올해 ICSE에서는 자동 패치 기술의 over-fitting 문제가 화두였다. 테스트 케이스 기반의 패치 기술의 근본적인 한계인 완전하지 않은 패치 명세를 어떻게 보완할 것인지, 테스트 케이스를 어떻게 구성해야 좋은 품질의 패치를 얻을 수 있을지에 대한 연구 등이 있었다.

나는 Program repair/testing/analysis 세션을 주로 들었는데, 그 중 재밌었던 논문들을 소개한다.

Static Automated Program Repair for Heap Properties
Rijnard van Tonder, Claire Le Goues

자원 누수, 널 참조 오류와 같이 힙에 관한 성질에 대해 정적 분석기가 내는 알람을 자동으로 고치는 기술에 대한 연구이다. 기존의 테스트케이스 기반 패치 기술은 메모리 누수와 같이 입/출력으로 명세하기 모호한 오류에는 사용하기 힘들고, 특히 테스트 케이스를 만들기 전에는 분석기가 새롭게 발견한 버그를 고치기에 곧바로 사용할 수가 없다. 이 연구에서는 정적 분석을 활용하여 테스트 케이스가 필요 없는 패치 시스템을 만들었다. Facebook Infer 분석기 위에서 패치 시스템을 구현했는데, 심볼릭 힙 위에서 심볼릭 실행을 통해 미리 정의된 힙 성질을 위반하는 경로를 찾아낸다. 패치는 프로그램에서 이러한 fault를 일으키지 않도록 하는 specification을 만족하는 프로그램 조각을 local reasoning을 통해 찾아낸다.

분석을 Smallfoot의 중간 언어 위에서 하기 때문에 다양한 언어 위에서 동작한다. 실험에서도 C와 Java 프로젝트 위에서 유용성을 보였다. 100K가 넘는 C 프로젝트 위에서 오류를 고칠 정도로 scalable하고, 11개의 알려지지 않은 버그를 패치하기도 했다. 그리고 Infer라는 우수한 정적 분석기 뒤에 패치 시스템을 붙였다는 것 자체가 이 연구의 큰 메리트 인 것 같다.

정적 분석을 이용하여 패치를 만들어 낸다는 점에서 예전부터 관심이 갔던 논문이다. MemFix와 다른점은 명확한 것 같다. 첫째로는 분석기 자체가 unsound하기때문에 패치의 correctness가 분석 결과가 사실이라는 가정 위에서만 보장된다. 이 부분은 실험 결과로 잘 방어한 것 같다. 둘째로는 패치 방식이 비교적 제한적이라는 것이다. 논문에서는 기존의 프로그램에 새로운 부분을 추가하는 방식(additive transformation)으로만 패치를 하고, 누수가 발생한 알람 지점에만 패치를 삽입한다. 아마 자원 누수를 고치기에는 충분할지 모르겠지만, 다른 종류의 오류를 고칠 때 trivial하게 확장할 수 있지는 않을 것 같다.

Towards Practical Program Repair with On-Demand Candidate Generation
Jinru Hua, Mengshi Zhang, Kaiyuan Wang, Sarfraz Khurshid

Generate-and-validate 기반의 패치 기술은 보통 프로그램을 고치는 수정 단계와 패치의 올바름을 검증하는 단계가 분리되어 있는데, 이러한 시스템에서 검증해야할 패치 후보가 많아질 경우 각각의 후보 프로그램을 매 번 컴파일하고 실행하는 과정이 굉장히 비효율적일 수 있다. 이 연구에서

는 localized 된 프로그램 부분을 hole로 뚫어놓고, 테스트 케이스를 실행하는 과정에서 hole에 들어갈 expression을 합성해낸다. 이렇게 테스팅 실행중에 on-demand로 후보를 생성하면 infeasible한 패치들을 (e.g., `if (false) hole`) 무시할 수 있으므로 search space를 많이 pruning할 수 있다고 한다.

합성으로 패치를 찾는다는 점과 후보 생성에 실행 정보를 사용한다는 점에서 우리 연구실의 FixML과 상당히 비슷한 부분이 많은 것 같았는데, 컴파일에 소요되는 시간은 생각해보지 않았던 것 같다. 발표에서는 기존의 기술로는 JFreeChart 프로그램의 특정 함수를 패치하는데 8만 7천개의 후보 프로그램을 validate하는 과정에만 꼬박 하루가 걸린다고 한다.

Chopped Symbolic Execution
David Trabish, Andrea Mattavelli, Noam Rinetzky, Cristian Cadar

Symbolic execution을 할 때 유저가 굳이 탐색하고 싶지 않은 코드 부분을 안전하게 무시할 수 있도록 한 연구이다. 명호 형이 시도했었던 target testing 연구와 거의 같은 문제이다. 내가 탐색하고자 하는 프로그램 파트와 전혀 상관없지만 굉장히 복잡한 함수 f가 있다고 할때, f를 심볼릭 실행에서 배제하고 싶을 수 있다. 문제는 f가 parameter나 global 변수를 통해 symbolic state에 영향을 줄 수 있는 경우이다. 이 경우 함부로 f를 무시하면 심볼릭 실행 결과가 unsound 해질 수 있다. 이 연구의 아이디어는 f를 만날때 현재의 symbolic state의 스냅샷을 찍어놓고, 심볼릭 실행중에 f의 side-effect에 영향을 받을 수 있는 상황이 오면 스냅샷의 상태로 돌아가 비어있는 실행을 채우는 식이다. 이 때 f의 side-effect는 포인터 분석을 통해 미리 계산해놓는다.

나는 타겟 노드의 control/data-dependency를 미리 구해놓고 static하게 프로그램을 축소 시키는 방향으로만 생각했었는데, 스냅샷을 찍어서 필요할 때마다 실행을 채워나가는 방식으로 좀 더 과감하게 탐색공간을 줄일 수 있겠다는 생각이 들었다. 기본적인 아이디어는 간단명료하지만, recovery state가 여러 개가 되거나, 다수의 함수를 skip하려고 할 때 어떤 이슈가 있는지 등을 잘 정의하고 해결한 점이 돋보였다.

Towards Optimal Concolic Testing
Xinyu Wang, Jun Sun, Zhenbang Chen, Peixin Zhang, Jingyi Wang, Yun Lin

우리 연구와 목표가 같아서 주의깊게 살펴봤던 연구이다. 콘콜릭 테스팅은 랜덤 실행(concrete)과 심볼릭 실행을 병행한다. 주어진 시간이 무한하다면 두 실행 모두 결과적으로 같은 커버리지를 내뱉겠지만, 현실에서는 당연히 적은 시간동안 많은 커버리지를 달성하는 것을 목표로 하게된다. 랜덤 테스팅은 심볼릭 실행보다 비용이 싸므로, 랜덤 테스팅으로 쉽게 도달할 수 있는 부분은 랜덤 테스팅으로 커버하고, if (x == 1)과 같이 랜덤 테스팅으로 도달하기 어려운 부분은 심볼릭 실행으로 커버하는 것이 효율적일 것이다. 이 연구에서는 콘콜릭 테스팅 중에 랜덤 실행과 심볼릭 실행 중 어떤 실행을 선택할지를 결정해주는 최적의 전략을 정의하고, 최적에 근사한 전략을 구해주는 greedy 알고리즘을 제안한다.

논문에서는 최적의 전략을 정의하기 위해 프로그램을 각각의 control location(브랜치) 사이의 전이 확률을 표현하는 DTMC(discrete time Markov Chain)으로 요약한다. 예를 들어, 이전 브랜치에서 if (x == 1)의 then-branch로 전이될 확률은 1/2^32인 식이다. 이제 랜덤 테스팅과 심볼릭

실행의 비용을 정의하면 최적의 전략은 모든 control-location을 최소의 기대 비용으로 도달하게 하는 policy로 정의된다. 이때 비용을 최소화하는 policy를 찾는 알고리즘은 여러개가 알려져있다고 한다. 그러나 실제로는 계산 복잡도때문에 정확히 구하는 것은 어렵기 때문에 본 연구에서는 이를 근사해서 구하는 greedy 알고리즘을 제시했다.

확률을 이용해서 최초로 최적의 탐색 전략을 제대로 정의했다는 것이 이 연구의 주요 contribution인 것 같다. 기존 연구에서는 모두 실험적으로 전략의 유용함을 보이기에 그쳤기 때문에, optimal strategy를 정의했다는 것 자체로 가치있는 일인 것 같다. 이론적인 부분은 기여가 큰 반면 실용적인 부분에서는 한계가 많은 연구이다. 실험으로는 두 종류의 세팅에서 유용함을 보였다. 첫째로는 5~20개의 상태를 가진 Markov Chain 모델을 임의로 생성하여, 이 연구에서 제안하는 greedy 알고리즘으로 구한 전략과 기존에 제시된 전략과의 비용을 비교했다. 둘째로는 GNU Scientific Libary(GSL)의 여러 함수에 대해 비교를 했다. 모두 심볼릭 실행만을 실행해도 무리가 없을 정도의 벤치마크여서, 실험 결과만을 보면 과연 최적의 전략이 필요한 것인지에 대한 의문이 들었다.

• 리셉션

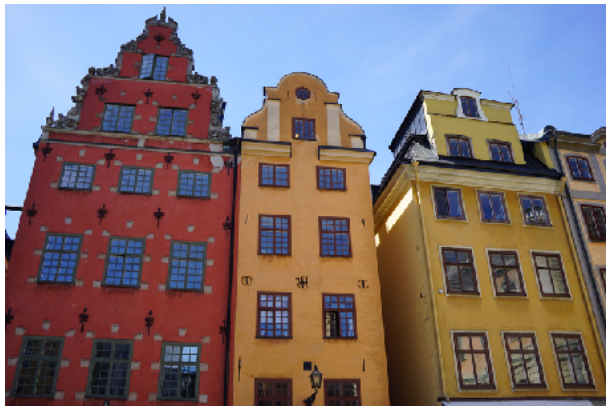
첫 날 리셉션은 학회장소 근처의 박물관에서 열렸는데, 스칸디나비아에서 가장 큰 자연 박물관이다. 사진에서 보이는 것처럼 수족관, 열대우림등 여러 층으로 이루어져있었다. 음식으로는 또띠아가 나왔는데 먹기가 불편했고 크게 맛있지는 않았다. 나무늘보, 상어 등 평소에 잘 볼수없는 동물들이 많았고 왜 여기서 리셉션을 했는지 이해가 잘 안갔다. 장소가 너무 컸고 리셉션은 작년 ICSE처럼 홀에서 하는게 나은 것 같다.

• 포스터세션

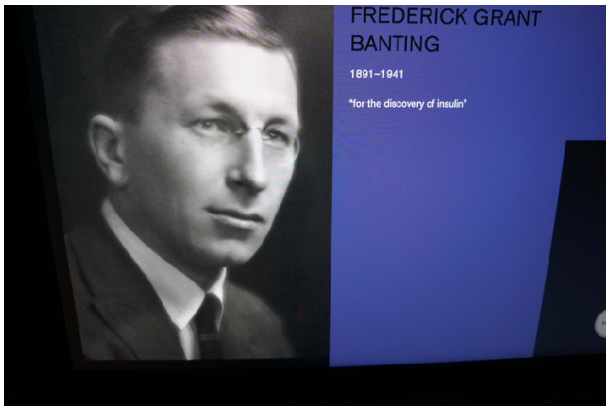
포스터는 3일 내내 꽤 크게 열렸다. 포스터 세션도 초록을 제출하고 accept를 받아야 참가할 수 있는지는 이번에 처음 알았다. 결국 데드라인을 놓쳐서 정식으로 참가하지는 못했는데, 나와 같은 사람들이 많았는지 학회장 남는 벽에 그냥 붙여논 사람들이 있었다. 우리도 발표가 있는 날에 홍보용으로 붙여놓았다. 내가 느끼기에 포스터보다는 발표가 더 재밌고 유익했던것 같다.

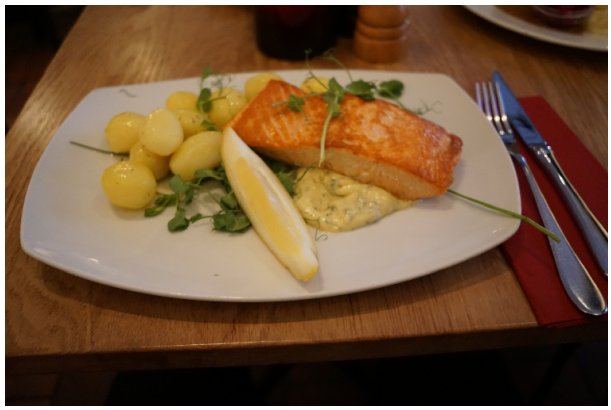
여행기

이번 여정은 스톡홀름에서 예테보리, 그리고 예테보리에서 웁살라로 이어지는 루트를 따라갔다. 북유럽이라 밤 10시가 넘어도 해가 지지않아서 알차게 돌아다닐 수 있었다. 낮에는 주로 스톡홀름의 감라스탄(구 시가지)를 위주로 돌아다녔는데, 볼거리가 꽤 많았다.



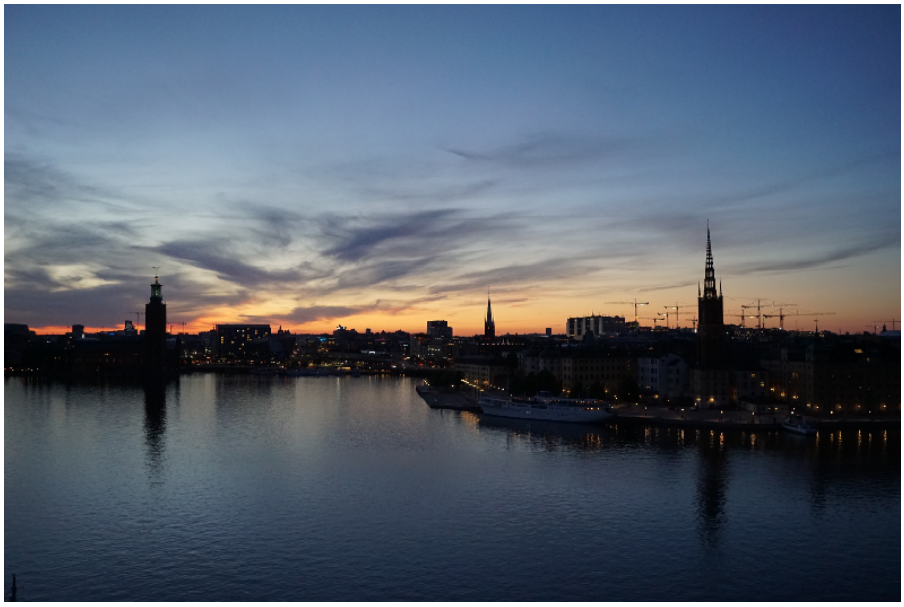
감라스탄 광장에 들린김에 노벨 박물관을 갔다, 박물관에서는 역대 수상에 관한 정보를 전부 볼 수 있다. 아래는 인상깊었던 수상자 중 하나. 연구 업적은 “인슐린의 발견”이다. 단 한 문장으로 누구나 인정하는 업적을 설명할 수 있다는게 정말 멋있었다.





감라스탄을 구경한 뒤 배를 타고 스웨덴 왕궁인 드로트닝홀름 궁전에 갔다. 스톡홀름에 가면 배를 타라고 추천해주고 싶다. 한 시간 정도의 거리를 1~2만원 내외로 탈 수 있기 때문에 왠만한 관광버스나 택시보다 싸다. 실제로 스톡홀름에서는 배를 관광용이 아닌 교통수단으로 많이 사용하고 있었다. 돌아오는 길에는 배가 버스 정류장처럼 섬 곳곳을 들러 현지인들을 태우고 갔다.

감라스탄에서 평이 좋은 연어 스테이크집을 찾아가서 저녁을 먹었는데, 식사를 하는 도중 누가 가방을 훔쳐가서 많은 것을 잃었다. 마땅히 갈 식당이 없다면 이 곳을 추천한다. 가격은 비싼편이지만 양도 많고 맛도 괜찮다.



가장 기억에 남는 풍경은 스톡홀름에서의 마지막 날에 본 야경이다. 밤 10시가 넘어가면서 해가 뉘엿뉘엿 질 무렵에 높은 곳에 올라가 바라본 도시 전경은 정말 멋있었다. 작년 아르헨티나에서 이과수 폭포와는 다른 느낌의 광경이었다.

스톡홀름에서 관광을 많이 하기도 했고, 예테보리는 학회를 참가하느라 따로 돌아다니지는 못했지만, 교수님 덕분에 맛있는 저녁을 많이 먹었다.

학회가 끝난 뒤에는 스톡홀름 근처의 웁살라로 향했다. 스톡홀름에서 웁살라까지는 30분정도 통근 열차를 탔는데, 퇴근시간인지 사람이 꽉 차있어서 캐리어를 들고 고생하며 겨우 겨우 낑겨 탔다. 사람사는데는 다 비슷하다는 생각이 들었다. 웁살라에선 스칸디나비아에서 가장 오래된 웁살라 대학교를 방문했다. 주요 관광지가 대학교를 중심으로 모여있어서 금방 돌아볼 수 있었다. 마지막엔 주로 강가나 시내에 앉아서 웁살라 주민들처럼 여유를 만끽하며 수영/준희 형과 이런 저런 얘기를 했다. 마지막 일정을 웁살라로 정하길 잘했던 것 같다.

마치며

이번 ICSE에서의 가장 큰 수확은 자신감이다. 진부한 말이지만 좋은 문제를 풀었다면 꼭 멋있게 풀지 않아도 다들 좋은 연구로 인정해준다는 것을 직접 확인할 수 있었다. ICSE에 갈 때마다 예상치 못한 일들을 하나씩 마주치는데, 이번 여행에서 있었던 일들도 살아가면서 한 번쯤은 도움이 될 것 같다는 생각이 들었다. 마지막으로 항상 좋은 연구를 할 수 있도록 자극과 격려를 주시는 오학주 교수님께 감사드린다.