1st International SAT/SMT Solver Summer School

Sunday, June 12- Friday, June 17, 2011 MIT, CAMBRIDGE, MA, USA

서울대학교 프로그래밍 연구실

이우석

이번에 MIT 에서 열린 국제 SAT/SMT solver 여름학교는 SAT/SMT solver 에 관련된 여러 주제를 다루는, 전례가 없는 행사였다. 홈페이지에는 이번 여름학교의 목표는 다음과 같다고 명시되어 있다.

1. SAT/SMT solver 개발자와 사용자의 참신한 아이디어 시장 마련
2. SAT/SMT solver 의 개발자와 사용자 연결
3. SAT/SMT solver 개발자와 복잡도 이론 연구자 연결
4. CDCL 알고리즘 연구자와 비 CDCL 알고리즘 연구자 연결
5. 멀티 코어를 이용한 SAT 해결기에 대한 토론 활성 등

위에 목표에서 알 수 있듯이 SAT/SMT solver 사용자 층 확대라던가 초심자에게 심도있는 이해 제공 등의 목표는 찾아볼 수 없다. 그래서 초심자에게 결코 친절하지 않은 방향으로 전개되었다. 여름학교라기 보다는 학회 느낌이 물씬 났는데, 발표 사이사이 쉬는시간이 길고, 그 동안 이 분야 대가들이 서로 이야기하는 모습이 학회장과 닮았었다. 발표 내용들도 매우 전문적인 것들이 많았기에 나를 비롯하여 우리 연구실에서 함께 간 4 명은 강의들을 이해하는데 매우 힘든 시간을 보냈다.

그렇지만 어려운 내용이기는 해도 직관적인 이해가 가능한 좋은 발표들도 여럿 있었다. 그래서 나는 강의를 통해 바로 명확한 이해를 하지는 못해도, 최대한 여러 다양한 주제들에 대해 (모호하지만)직관적인 수준의 이해와 호기심을 쌓은 다음, 한국에 돌아가서 더 스스로 공부하기로 마음먹었다.

강의는 크게 세 가지 범주로 분류되었다.

1. SAT/SMT solver 의 기초 (Foundational Aspects of SAT/SMT Solvers)
2. 특화된 SAT/SMT solver (Description of SAT/SMT Solvers with tight focus on an application)
3. SAT/SMT solver 를 사용하는 툴 (Description of tools using SAT/SMT Solvers)

1 에 속한 발표는 SAT/SMT solving 의 복잡도와 관련된 얘기나 내부적인 알고리즘과 같은 이론적인 것들이었다. 2 에 속한 발표는 SAT/SMT solver 의 개발자들이 직접 자신들의 solver 에 대한 이야기를 했다. 3 에 속한 발표에서는 SAT/SMT solver 의 응용 사례를 다루었다. 그런데 실제로는 범주 2 의 발표들이 주로 Solver 의 응용에 초점을 맞추었기 때문에, 결국 강의들은 크게 기초와 응용으로 나뉘었다고 볼 수 있다. 나는 응용보다도 이번 여름학교에서 SAT/SMT 해결 자체에 대한 이론적이고 기초적인 내용의 강의를 많이 듣고 싶었다. 기초를 안다면 응용은 나중에 공부하기 수월할 거라고 생각했기 때문이다. 그런데 첫 이틀 정도는 그런 강의들 보다

응용에 대한 것이 주를 이루었다. 응용 중에서도 특히 기호 연산(Symbolic Execution)에 관한 것이 아주 많았다. 계속 기호 연산 도구에 관한 발표만 이어지니까 나는 점점 흥미가 떨어지는데 사람들 반응은 아주 좋았다. 반면 이론적인 발표는 반응이 시원찮았다. 거기에서 만났던 한양대 박용수 교수님께 나중에 들어서 알게 된 것인데, 이번 여름학교에 버그 사냥꾼들이 많이 와서 그랬다고 한다. 마이크로소프트와 구글 같은 몇몇 회사들은 자사 프로그램에 문제를 찾아주는 사람에게 거액의 상금을 준다고 한다. 그리고 그에 고무되어 프로그램 허점을 찾으려는 사람들이 기호연산 도구를 그 수단으로 많이들 쓴다고 한다. 그들 중 기호 연산 및 기호 연산에서 많이 사용하는 SAT/SMT 해결기의 내부에 대해서 더 깊이 알고 싶은 사람들이 이번 여름학교를 찾았고, 그러한 관심이 반영된 결과 그런 현상이 나타났다는 것이다. 흥미로운 뒷얘기였다. 역시 세상 만사 돈이 얽히지 않은 일이 없다.

5 일 동안 있었던 발표들 중 인상 깊은 것들을 정리해 보았다.

SAT/SMT solver 의 기초

SMT Theory and DPLL(T)

- *Albert Oliveras*

이 발표는 SMT 문제에 대해 소개하고, 그것을 푸는 내부 알고리즘에 대해서 다루었다. SAT 가 명제논리식에 대해서 변수에 참 혹은 거짓을 할당해서 식 전체를 참으로 만드는 만족할당을 찾는 문제라면, SMT 는 선형 산술식(linear arithmetic), 배열(array) 등 특정 도메인에 특화된 추론이 필요한, 만족할당을 찾는 문제이다. (예, $a = b + 2 \wedge a \geq 0 \wedge b < 0$ 를 참으로 만드는 a 와 b 의 값은? 답: 없음) SMT 는 기본적으로 SAT 해결기를 이용하여 푼다. 이 때 표현하는 식에 관련된 정보가 관여되는데, 이를 Background Theory 라고 한다. Theory 는 구성요소와 공리로 구성되어 있다. 가령 선형 산술에 대한 Theory 는 구성요소로 숫자와 사칙연산 기호, 등호, 부등호들을 갖는다. 공리는 우리에게 이미 익숙한 결합법칙, 교환법칙, +항등원, +역원 등이다. 구성요소들로 만들어 질 수 있는 모든 1 차논리식(First order logic formula)식들 중에서 공리를 만족하는 식들의 집합으로 Theory 를 이해할 수 있다. SMT 를 SAT 해결기를 이용하여 푸는 방법은 크게 두 가지, 성급한 방법(Eager approach)와 느긋한 방법(Lazy approach)이 있다. 성급한 방법은 주어진 SMT 식을 동일한 SAT 식으로 바꾼다음에 SAT solver 에게 해를 물어보는데, 식을 바꿀 때 Theory 의 공리를 첨가한다. 느긋한 방법은 SMT 식의 literal 들을 부울 변수들로 바꾼 다음에 SAT solver 에게 해를 물어본 후, SAT solver 가 알려준 해가 공리에 어긋나면(이를 판단하는 모듈을 Theory solver 라 한다) 그 해는 안된다는 정보를 식에 추가해서 SAT solver 에게 다시 물어보는 식이다.

또한 어떤 SMT 식이 여러개의 Background Theory 와 관여되어 있을 수 있는데, (예, $a = b + 2 \wedge B[a+1] \leftarrow 4 \wedge (A[b+3] = 2 \vee f(a-1) \neq f(b+1))$와 같은 식. 선형 산술, 배열, 모르는 함수가 낀 방정식(Equality with uninterpreted functions)이 관여되어있는 식이다) 이러한 경우는 식을 하나의 Theory 만 관여된 여러 부분들로 나누고, 각 부분을 해당 Theory solver 들에게 맡겨서 푼다. 이 때 각 Theory solver 들은 서로 알게 된 정보를 공유하면서 푼다.[1]

MaxSAT for Optimization Problems

- *Joao Marques-Silva*

MaxSAT 은 Maximum Satisfiability 의 약자로 어떤 CNF 식이 UNSAT 일 경우 그냥 UNSAT 이라고 말하고 끝내는 것이 아니라, 그나마 최대한 많은 절들이 SAT 이 되게 하는 해를 찾는 문제이다. 이 문제에서 꼭 만족시켜야 할 절에 대한 제약을 추가하면 Partial MaxSAT, 각 절마다 가중치를 두고 UNSAT 이 된 절들의 가중치 합이 최소가 되어야 한다는 제약이 추가 되면 Weighted MaxSAT 이 된다. 이 둘이 합쳐진 Weighted Partial MaxSAT 도 있다.

MaxSAT solving 은 최적화 문제에 이용되기도 한다. 문제 성질 자체가 최적화와 깊은 관련이 있기 때문이다. 주어진 제약 조건 중에서 최선의 해를 찾아낸다는 점에서 선형계획법(Linear programming)과 닮아있다. 그래서 이 문제를 0-1 정수 선형 계획법(0-1 Integer Linear Prorgramming)과 동일하게 보기도 한다. 이것의 응용의 한 예는 최적의 소프트웨어 패키지 설치 솔루션을 찾는 것이다. 가령 패키지들 마다 이것을 설치하기 위해 필요한 다른 패키지들이 있고, 동시에 설치될 수 없는 패키지들에 대한 제약이 있을 때, 설치 용량을 최소화 하는 설치 패키지 리스트를 찾는 것이다. 또한 최소 정점 덮개(minimum vertex cover)같은 문제의 해를 구하는 데에도 사용될 수 있다.

이를 푸는 알고리즘으로 분기 한정법(Branch and bound method)이 있다. 핵심은 모든 가능한 후보해들 중에서 최적해를 찾는데, 더 이상 진행해볼 필요 없는 후보는 일찌감치 버리는 것이다. 예컨데 지금까지 찾은 것 중 제일 좋은 해가 불만족 시키는 절의 개수를 UB 라고 하자(적을수록 좋다). 그리고 현재 조사하고 있는 후보 해의 일부분으로부터, 이것이 완성될 경우 불만족시킬 거라고 예상되는 절의 개수가 LB 라고 하자. 만약 LB >= UB 면 그 해는 버리고 다른 새로운 해를 찾는다. 이런 식으로 모든 가능한 해를 다 뒤져보기 때문에 최악의 경우 지수 시간이 걸리지만 구해진 해는 최적의 해라는 것이 보장된다. 실제로는 UB 와 LB 를 더 잘 구하기 위한 여러가지

---

[1] 저는 이 내용으로 6 번째 ROSAEC 워크샵에서 튜토리얼을 했는데, 발표 자료를 보시면 더 이해가 쉬울 것입니다. http://rosaec.snu.ac.kr/meet/file/20110626o2.pdf

기술이 사용되기 때문에 지수시간 보다는 훨씬 빠르게 최적해를 구한다. 성능을 위해서, 가령 UB 와 LB 를 Binary Search 를 이용하여 구하기도 하고, 조금 다른 방법으로 UNSAT core(CNF 가 UNSAT 이 되게 하는 부분)에 대한 정보를 이용하여 더 효율적인 알고리즘을 구현하기도 한다.

특화된 SAT/SMT solver

SAT solving in AI

- *Henry Kautz*

SAT solving 을 인공지능에 적용한 예도 있다. 초기 조건과 목표 조건, 취할 수 있는 행동의 전후 조건(precondition, postcondition)들이 주어졌을 때, 가능한 행동의 시퀀스를 구하는 것을 Automated planning 이라 하는데, 인공지능의 한 지류이다. 이러한 문제를 명세하는 언어 중에 STRIPS 라는 것이 있다. 가령 초기 조건으로 현재 위치와 바나나의 위치가 주어지고, 이동, 박스 위에 올라서기, 내려가기, 박스 옮기기, 바나나 집기 등이 가능한 행동들로 주어졌을 때, 상황을 다음과 같이 간단히 명세할 수 있다.

```
Initial state: At(A), Level(low), BoxAt(C), BananasAt(B)
Goal state:    Have(Bananas)
Actions:
              // move from X to Y
             _Move(X, Y)_
             Preconditions:  At(X), Level(low)
             Postconditions: not At(X), At(Y)
             // climb up on the box
             _ClimbUp(Location)_
             Preconditions:  At(Location), BoxAt(Location), Level(low)
             Postconditions: Level(high), not Level(low)
             // climb down from the box
             _ClimbDown(Location)_
             Preconditions:  At(Location), BoxAt(Location), Level(high)
             Postconditions: Level(low), not Level(high)
             // move monkey and box from X to Y
```

```
_MoveBox(X, Y)_
Preconditions:  At(X), BoxAt(X), Level(low)
Postconditions: BoxAt(Y), not BoxAt(X), At(Y), not At(X)
// take the bananas
_TakeBananas(Location)_
Preconditions:  At(Location), BananasAt(Location), Level(high)
Postconditions: Have(bananas)
```

출처 : http://en.wikipedia.org/wiki/STRIPS

이 때 목표는 결국엔 바나나를 집게되는 행동의 시퀀스를 구하는 것이다. 발표자는 처음으로 1992 년에 이러한 STRIPS 문제를 SAT 문제로 변환하여 푸는 방법을 제시하였는데, 이 무렵 SAT 해결기의 성능이 비약적으로 발전하고 있었기 때문에 그런 시도를 했다고 한다. 이들은 SAT 해결기의 뛰어난 성능에 힘입어 STRIPS 대회에서 수 차례 우승했다. 구체적인 내용은 어렵지 않고 재미있는 편이었는데 안타깝게도 발표자료가 홈페이지에 올라와 있지 않다.[2] 요점은 SAT 해결기가 인공지능 분야에서도 사용된다는 것이다. SAT 해결기의 참신한 응용 방법들 중 한가지를 엿볼 수 있어 좋았다.

SAT/SMT solver 를 사용하는 도구

Constraint Solving Challenges in Dynamic Symbolic Execution

- *Cristian Cadar*

기호 실행(Symbolic Execution)이란 말 그대로 실제 실행과 달리 모르는 값을 기호로 두고 프로그램을 실행해 나가는 것이다. 이 방식은 정적으로 테스트 입력을 생성하는 방법으로 많이 애용되어 왔다. 그러나 이 방식은 기호 실행하기 어려운 부분(예, 시스템 콜, 비선형 표현식)이 존재하면 정확도가 크게 떨어진다는 단점이 있다. 그래서 새롭게 각광받는 방법이 동적 기호 실행(dynamic symbolic execution)인데, 이는 실제 실행을 기호 실행과 같이 하는 것이다. 기호

---

[2] 보다 자세한 내용을 알고 싶다면 Armin Biere et al., *Handbook of Satisfiability* 의 Chapter 15. Planning and SAT 을 참조하세요.

실행하기 어렵거나 기호와 연관되지 않은 부분은 아예 실제로 실행을 하고, 커버리지를 높이기 위해 다른 경로를 탐색하는 테스트 입력을 만들고자 할때는 기호 실행으로 생성된 경로 제약 조건에서 일부를 바꿔서 그것을 만족하는 새로운 입력으로 똑같은 일을 반복한다.

이 발표는 동적 기호 실행을 보다 효율적이고 정확하게 만들기 위해서 사용했던 여러가지 기술과, 동적 기호 실행의 응용사례들을 자신들의 기호 실행 도구 KLEE 를 위주로 다루었다. 정확성을 위해 이들은 대상 언어를 C 같은 언어가 아니라 LLVM 바이트코드를 대상으로 하며 비트 연산도 지원하기 위해 비트벡터를 다루는 SMT 해결기인 STP 를 사용한다. 보다 빠르게 만들기 위해서 이들은 경로 제약식과 상관없는 제약식들 없애기, 제약식과 SMT solver 의 결과를 캐싱하여 나중에 같거나 유사한 제약식이 들어올 때 이용하기 등 상식적인 방법들을 사용하고 있다. 이들은 KLEE 를 이용하여 GNU core utils 에 대해 10 개의 치명적인 버그를 발견했고, 어떤 프로그램에 대해서는 15 년 동안의 수동 테스팅 보다 높은 line coverage 를 달성했으며 (68% 대비 91%), Coreutils 과 그것의 임베디드 시스템 구현 버전인 Busybox 간의 여러 개의 불일치를 찾았다고 한다.

Sketching: Program Synthesis using SAT Solvers

- *Armando Solar-Lezama*

프로그램 합성(Program Synthesis)는 자동으로 제약 조건들을 만족하는 프로그램 코드를 만들어내는 기술이다. 이는 안전한 프로그래밍을 위한 한 대안으로 연구되어 왔지만 오랫동안 장난감 수준에 머물러왔다. 그런데 Armando 라는 사람으로 인해 이 분야 판도가 바뀌었다고 한다. 이 사람은 프로그램 합성이 나아가야 할 길을, 곡선 그리기에 비유하면서, 컴퓨터에게 아예 빈 화면에 곡선을 그리게 하기는 어렵지만, 만약 사람이 그려져야 할 곡선의 점 몇 개를 찍어준다면 문제가 훨씬 쉬워진다는 좋은 비유를 들어 설명했다.

Sketch 는 Armando 가 만든 C 언어 프로그램 합성 도구인데, 합성하고 싶은 프로그램 코드의 제약조건(pre, postcondition 등)을 주고, 만들어질 프로그램 코드의 생김새를 정규 표현식 형태로 주면 코드를 합성해준다. Sketch 는 DES 암호체계의 암호화/복호화 모듈을 구현하는데 사용되었는데 순전히 사람이 구현한 것 보다 성능이 50% 좋다.

이 도구에서 Solver 가 이용되는 부분은 CEGIS(Counter Example Guided Inductive Synthesis)라는 알고리즘에 있다. 이 알고리즘의 목표는 모든 가능한 입력 예제들에 대해서, 제약조건을 만족하는 코드를 만드는 것이다. 그런데 합성된 코드가 모든 가능한 입력 예제에 대해서 제약조건을 만족하는지 검사하는 것은 불가능하므로, 일부 코너케이스 입력 예제(corner case input)들만을 이용한다. 입력들 중 제약조건을 만족하지 않는 반례를 통해서 합성할

프로그램을 더 정교하게 다듬게되는데, 이 부분에서 Solver 를 쓴다. 조건을 만족하는 함수를 찾는 것이 코너 케이스에 집중한 몇 개의 입력만으로 충분하다는 것이 발표자의 주장이다.

구경거리

MIT

그림 1 STATA center 건물 외부

그림 2 STATA Center 건물 내부

이번 여행 통틀어 가장 강한 인상이 남은 곳은 여름학교가 열렸던 MIT 이다. MIT 의 컴퓨터공학부 CSAIL 건물은 굉장히 특이하게 생겼다. Ray and Maria Stata Center 라고 불리는 이 건물은 매우 정돈이 안 된 느낌이다. 보스턴 글로브 지의 건축 칼럼니스트 Robert Campbell 은 이 건물에 대해서 다음과 같이 찬사를 보냈다고 한다. "Stata 센터는 늘 미완성작품같이 보인다. 그것은 무너질 것 같이 생겼고, 기둥은 무서운 각도로 기울어져 있으며 재질은 볼 때마다 벽돌, 알루미늄, 철 등 달라진다. 이런 모습은 자유, 모험심, 건물안에서 일어나는 연구들의 창조성을 상징한다."[3] 이 건물은 학교가 기존 관념을 깨는 창조적인 아이디어를 얼마나 중요시 하는지 보여주는 일면 같았다.

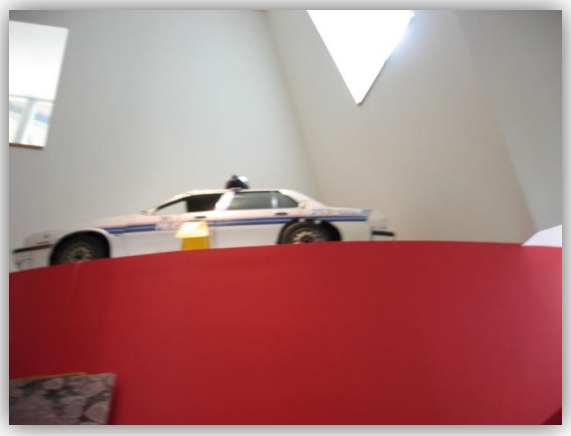
---

[3] http://en.wikipedia.org/wiki/Ray_and_Maria_Stata_Center#Critical_response

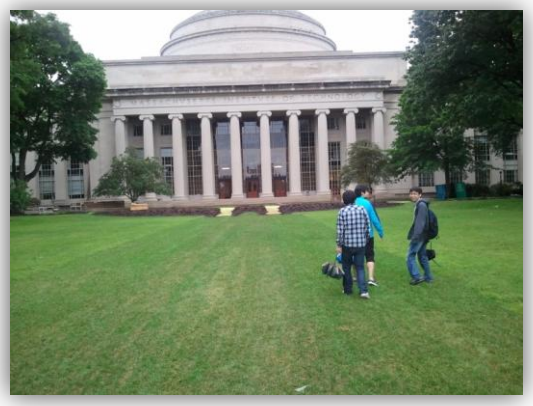
그림 1 건물 외부 사진에 노란 드럼통 같은 부분은 세미나실인데 내부가 위와 같다.

이 곳에서 공교롭게도 우리가 머물고 있는 동안에 지도 교수님이신 이광근 교수님께서 발표를 하셨다. 미국에서 교수님 영어 발표를 들으니 매우 감회가 새로웠다.

또 Stata Center 건물 내부에 경찰차와 소 모형이 높은 곳에 설치되어 있는 것을 보았는데, 학생들의 퍼포먼스를 기념하기 위한 것이라고 한다. 그 퍼포먼스는 MIT 학생들이 하룻밤 사이에 경찰차와 소 (진짜 소 인지 모형인지는 확실치 않다)를 학교 건물 꼭대기에 올려놓은 일이었다. 연구실 선배이고 현재는 MIT 에서 박사과정을 밟고 계신 덕환이 형께서는 이런 이상한 애들(?)이 학교에 꽤 있다고 했다.



Stata Center 뿐만 아니라 MIT 에는 돔 건물도 매우 유명한데 돔 아래에는 매우 예쁜 도서관이 있다. 학생 뿐 아니라 외부인도 자유롭게 출입할 수 있어서 점심시간에 시간이 빌 때 이곳에 가서 공부하곤 했다.



학교에는 이런 예쁜 도서관이 몇 군데 더 있다고 하는데(유리창 벽면으로 맞은편 강을 바라보면서 책을 읽을 수 있는 도서관도 있었다) 덕환이 형 말로는 학생들이 잘 안 들른다고 한다. 실제로 우리들이 갔을 때도 학생들은 별로 없었다. 방학이라서 라기보다 학생들이 도서관보다는 아무곳에서나 책, 노트북을 펴고 공부하는 것을 더 좋아하기 때문에 그렇다고 한다. 실제로 이상하게도 밖에는 벤치가 별로 없지만 건물 내부 곳곳에는 앉아서 공부할 수 있는 곳들이 매우 많이 있다. 학교 곳곳을 둘러보면서 학교가 학생들 편의 시설을 위해 돈을 아낌없이 펑펑 쓴다는 느낌을 받았다. 그리고 그러한 시설들을 외부인들이 전혀 제한없이 쓸 수 있다는 점도 인상깊었다. 이러한 MIT 의 개방적인 태도가 학교가 지역 공동체의 사랑을 받는 것과 무관하지 않다는 생각이 들었다. 옆에 사진은 올해 치뤄진 MIT 설립 150 주년 기념 행사때 찍힌 사진이다. 보스턴에서 꽤 높은 건물 들 중 하나인 프루덴셜 타워에서 32~49 층 사이에 조명을 이용해서 MIT 150 주년을 축하하고 있다. 학교가 이렇게 지역 공동체의 존경과 사랑을 받는다는 것이 매우 부러웠다.

하버드

하버드는 비록 전부 둘러보지는 못했지만 MIT 와는 분위기가 확연히 달랐다. MIT 가 새로운 신식 건물들로 꽉 차있다면 하버드는 좀 더 고풍스럽고 유럽 같은 분위기를 자아낸다. 저 사진은 설립자 존 하버드의 동상인데 발을 만지면 자기 자식이 후에 하버드에 입학한다는 미신이 있다. 얼른 만졌다. 여러 사람들이 발을 자꾸 만져서 동상의 발만 흰색이다.

학교 앞 분위기도 매우 달라서, 비로소 대학가 같은 느낌이 났다. MIT 는 비교적 조용한 반면, 하버드 앞은 거리 공연 예술가들, 개성있는 패션의 학생들, 관광객들이 어우러져 활기찬 분위기를 느낄 수 있었다.

그림 4 도서관 전경

그림 3 (확실치 않지만) 빌게이츠가 머물렀던(머물렀을지도 모르는) 기숙사 건물

보스턴 시내 구경

보스턴에 대한 첫인상은 자유롭고 개방적인 이미지인데, 때마침 열린 게이 축제인 Boston Pride 때문이다.

동성애자들이 여러가지 치장을 하고 거리 행진을 하는데, 동성애자 이성애자 할 것 없이 모두 어우러져 함께 즐기는 모습이 좋아보였다. 그렇지만 나는 오래 즐기지는 못하고 웃통 벗은 게이들이 뛰어다니는 모습에 식겁하여 얼른 숙소로 들어왔던 것이 안타깝다.

여름학교가 끝나고 우리는 제대로 도시 탐방을 해보려고 나섰다. 나름 계획을 잡고 움직이려는데 때마침 보스턴의 아이스하키팀 우승을 축하하는 퍼레이드가 열려서 사람들이

많을거라고 호스텔 매니저가 일부 지역 관광을 만류했다. 보스턴 레드삭스 홈 구장 등 몇몇 볼거리를 포기하고 도시를 걸어다니면서 둘러봤는데 높은 건물들이 아주 많았고 항구를 중심으로 뻗은 시가지가 멋졌다.

여름학교를 통해 얻은 것들

여름학교 전에 나는 SAT/SMT solver 를 고작 몇번 써본 정도에 불과했지만, 이제는 부족하게나마 내부 알고리즘이 어떻게 돌아가는지, 그리고 어떠한 일들에 쓰이는지 알게 되었다. Solver 내부에 대해 알게 된 것도 큰 소득이지만 무궁 무진한 응용 사례들에 대해 알게 된 것 또한 큰 소득이다. 계속 많아지고 다양해지는 쓰임새로 보건데, 앞으로 SAT/SMT Solver 를 잘 아는 것은 확실히 큰 자산이 될 것 같다. 그래서 여름학교 동안 배웠던 내용들에 대해 꾸준히 관심을 가지고 이해 못했던 부분들을 채워나갈 생각이다.

또한 말로만 듣던 MIT 에서 여러가지를 보고 듣고, 여름학교에 참석한 전 세계에서 온 똑똑한 학생들의 예리한 질문들을 보면서 받았던 자극들도 가능한 한 오랫동안 잊지 않아야겠다.

마치며

무엇보다 많은 걸 배울 수 있는 큰 기회를 주신 이광근 교수님께 정말 감사드립니다.

그리고 여름학교 기간동안 연구실 선배이자 현재 MIT 에서 박사과정을 밟고 계신 덕환이 형한테 정말 많은 도움을 받았습니다. 바쁘신데도 학교 이곳 저곳을 안내해주시는 건 물론 주변

맛집에도 매일 데려가 주셨고, 강의도 같이 들으시면서 우리가 몰랐던 부분들에 대해 설명도 많이 해주셨습니다. 덕환이 형께도 정말 감사드립니다.

아울러 거기에서 처음 뵙게 된, 저녁과 맥주를 사주시면서 좋은 말씀 많이 해주신 한양대 박용수 교수님께도 감사드립니다.