PLDI 2025 Conference Report

Programming Language Design and Implementation

Seoul, Republic of the Korea

By Jungyeom Kim | PLRG

2025. 07. 11

1. Introduction

이번에 PLDI'25가 한국에서 개최되어서, 많은 국내 학생들이 참여할 수 있게 되었다. 나는 이번 학회에서는 Volunteer에 지원했다. Volunteer에 지원하면 큰 혜택 2가지가 있는데, 첫번째는 학회 등록비가 무료라는 것이고 두 번째는 PLDI 티셔츠를 준다는 것이다(잠옷으로 사용하기 매우 적절하다). 내돈내산 학회가 되었으므로 보고서를 쓰지 않아도 될 줄 알았는데 이미 여비가 소매넣기 되어서 보고서를 써야 한다는 점이 조금 아쉽기는 하다.

다만 만약에 학회를 제대로 즐기고 싶다면, 봉사자로 지원하는 것은 피하라. 봉사자는 크게 다음과 같은 일을 한다.

- 학회의 특정 세션들을 맡아, 마이크를 교체하고 질문을 받으러 뛰어다니는 등의 잡일을 한다.
- 사람들의 질문이나 요청이 있으면 운영진에게 전달하고 소통을 돕는다.

문제는 잡일을 하다 보면 발표를 제대로 듣지 못하는 경우가 있고, 무엇보다 체력을 심하게 뺏긴다. 작년에는 괜찮았는데 올해는 상당히 많이 졸기도 했고 발표를 온전히 집중해서 듣지 못한 경우도 있었다. 물론 등록비가 한두푼이 아니기는 하지만, 연구실이 그것을 커버할 수 있을 만큼 충분히 부유하다면(다행히도, 우리 연구실은 교수님이 차력쇼를 잘 하시는 편이다) 그냥 지원을 받아서 학회에 참가하는 것을 권하고 싶다.

2. SOAP and RPLS

이번 PLDI에서는 SOAP과 RPLS에 참가했다. SOAP같은 경우에는 키노트를 Xavier Rival 교수님이 맡아서 진행해 주셨는데, 상당히 재밌었다. 전반적인 발표의 개요는 seperational logic을 활용해서 자료구조의 abstract domain을 구성하는 방법론에 관한 것이었다. 아무래도 우리 연구실이 Xavier Rival 교수님의 영향을 꽤 받았다 보니 설명이나 멘탈 모델 자체가 익숙해서 재밌게 들을 수 있는 발표였다. 다만 abstract interpretation 자체가 아무래도 최근에 하는 사람이 많이 없다 보니 같은 연구실에 있는 동료들이 잘 이해하지는 못한 듯 싶다. 다른 SOAP 발표의 경우에는 상당히 실망스러웠다(개인적인 의견으로는, SIGPL이 훨씬 더 수준 높은 연구들을 보여주었다).

하지만 RPLS는 정말로 재밌었다. RPLS는 내가 하루 종일 volunteer shift에 배정되어 있었는데, 덕분에 맨 앞자리에 발표자들과 앉아서 발표를 눈 앞에서 볼 수 있었다. 개인적으로 프로그래밍 언어를 사용하는 것 보다는 작성하는 쪽에 관심이 많기 때문에 프로그래밍 언어를 설계하고 그 명세를 정의하는 것에는 관심이 어느 정도 있는 편이다. 이번에는 K-framework, P4, Redex, SpecTec, ESMeta, MiniRust 등 전 세계의 스펙 작성하는 사람들이 모두 모여서 하루 종일 떠들었는데 각 연구팀마다 확연한 개성이 느껴지는 것이 흥미로웠다. 비록 방향에는 공감하지 못하지만 Ralf Jung을 만나 이야기를 들어볼 수 있어서 좋았고(이 분 발표하실 때는 학회장이 꽉 차고도 넘쳤다) ECMA-262 PR을 날릴 때마다 리뷰를 제공해주는 Micahel Ficarra와 대화를 해볼 수 있어서 좋았다. 내년에 열릴지 여부가 불확실한 것으로 알고 있는데, 꼭 열렸으면 좋겠다(그리고 참석할 수 있었으면 좋겠다).

3. Research

솔직히 이번 PLDI'25는 작년에 덴마크에서 개최된 PLDI'24에 비해 재밌는 연구가 크게 없었다. 작년에는 타입이나 정적 분석에 관한 논문들도 많았던 것 같은데 이번에는 대체적으로 인공지능이나 검증, 또는 확률이나 KAT 등의 특정한 도메인에 관한 논문들이 많았던 것 같다. 기껏 나온 분석 논문들도 거의 특정한 언어에 국한해서 가벼운 정적 분석을 진행하는 논문들이었고 반면 전통적인 abstract interpretation에 대해서는 거의 다루지 않았다. 심지어는 컴파일러 세션조차 재미없었다.

하지만 그럼에도 불구하고 머리를 탁 치게 만드는 논문들이 드물게나마 있었기 때문에, 이런 논문들에 대해 간단하게 소개하고 넘어가도록 하겠다.

3.1 Tree Borrows

Rust는 borrow checker를 통해서 메모리 안전성을 보장하는데, 기존에는 stacked borrows를 통해서 이를 구현했다. 하지만 stacked borrows는 기본적으로 내가 지금 수정하고 싶은 변수가 나올 때까지 위에서 invalidate를 greedy하게 해야 하다 보니 조금만 프로그램 흐름이 엉켜있어도 바로 UB를 띄우는 문제점이 있었다. 그래서 이 논문에서는 stacked borrows보다 크게 느려지지 않으면서도 UB 비율을 줄이기 위해 스택 대신 트라이와 비슷한 트리 구조를 사용하는 Tree Borrows를 제안했다.

기본적으로 UB를 막는 방법의 핵심은 두 변수가 aliasing 관계에 존재할 수 있는지 여부를 판단하는 것이다. 따라서 각 변수가 어떤 상태에 있는지를 간단하게 오토마타 형태로 요약한 다음 정보를 트리 위에서 전파한다면(PS에서 사용하는 것과 은근히 비슷하다) 효과적으로 sound한 analysis를 진행할 수 있다. 말은 쉽지만 당연히 효과적으로 state machine을 구성하는 것은 꽤 까다로운 고려들이 필요한데, 변수들의 scope와 lifetime을 생각해야 하의 포인터와 shared reference, mutable reference 등도 존재하기 때문이다. 뭔가 대략 Euler tree와 lazy propagation 비스무리한 접근법을 사용해서 효율을 높였다고 생각해도 반쯤 맞는 것 같은데, 이런 느낌으로 최적화를 하면 복잡한 코드에 대해서도 현실성 있는 수준의 분석을 제공할 수 있을 것 같아서 인상적이었다.

3.2 Taking out the Toxic Trash: Recovering Precision in Mixed Flow-Sensitive Static Analyses

정말정말 간단하게 요약하면, Mixed Flow-sensitive analysis에서 global variable을 잘 핸들링 하는 방법에 대한 연구이다. Global variable은 당연히 프로그램 전반에서 폭넓게 쓰이기 때문에 만약 mutate된다면 쉽게 정확도가 낮아지기 쉽다. 특히 interval analysis와 같이 height가 무한한 도메인인 경우, widening을 사용하는 것이 불가피하므로 몇 번만 루프를 돌아도 top으로 가기 쉽다. 보통 widening의 정확도가 심하게 떨어지는 문제를 해결하기 위해서는 narrowing을 사용하는데, global variable의 경우 readsite가 너무 많아 narrowing을 적용하더라도 결과가 잘 바뀌지 않는다.

이 논문은 global variable에 대한 분석을 각 readsite마다 쪼개서 sensitive하게 분석하면, 훨씬 더 정확하게 narrowing할 수 있음을 보인다. 기본적으로 문제는 서로 다른 program flow가 실제로 섞일 수 있기 때문에 생기는 것인데, narrowing을 적용할 때 너무 부정확한 곳에서 시작하지 않도록 적당히 보정해주면 각각의 비교적 정확한 분석 결과에 대해서는 narrowing이 작동할 수 있기 때문에 최종 결과가 더 정확해진다. 사실 어떻게 생각하면 일반적인 context-sensitive analysis와 와 같은 접근법이라고 볼 수도 있겠지만 이게 실제로 작동할 거라는 직관을 얻기에는 꽤 어려워 보인다.

3.3 Relational Abstractions Based on Labeled Union-Find

이 논문이 모든 PLDI'25 논문 중에 가장 재밌었고, 다른 논문들보다 압도적으로 발상이 신선했다. 기본적으로 우리가 relational abstract domain 을 잘 사용하지 않는 이유는 무겁기 때문이다. 단순 ax+by=c 꼴의 relation을 저장하는 것만으로 closure을 계산하는데 $O(n^3)$의 시간이 걸린다. 이 논문에서는 labeled union-find를 사용해서 이 문제를 해결하는 방법을 제안한다. 기본적으로 union-find는 disjoint set을 관리하는 자료구조인데, 구현이 tree로 되어 있어서 결합법칙이 성립하는 구조에 대해서는 임의의 부모-자식 관계에 대한 경로 쿼리를 $O(\log n)$에 처리할 수 있다. 만약에 결합법칙 뿐만 아니라 역원까지 존재한다면, 임의의 부모-자식을 넘어서 임의의 경로에 대한 쿼리를 $O(\log n)$에 처리할 수 있다. 따라서 domain을 하나의 스패닝 트리로 정의할 수 있다면 매우 빠른 시간 내에 항상 closed 상태를 유지할 수 있다는 것이다. 사실 이런 구조가 매우 표현력이 떨어질 것이라고 생각했는데, 이 논문에서는 이런 구조를 만족하는 abstract domain이 생각보다 많고 분석에 실제로 효과적임을 보였다. 방법보다는 실제로 효과가 있을 것 같다고 생각하고, 그것을 현실에서 보였다는 데에 큰 의의를 두고 싶다.

딱 한 가지 아쉬운 점이 있다면 저자가 자료구조에 대해 완벽하게 알지는 못해서, persistent union-find tree의 존재를 몰랐다는 것이다. 갑자기 union-find의 path compression을 언급하면서 mutablity를 핸들링하기 위해 SSA form을 고려했다고 하길래 무슨 소리인가 했는데, rank-by-optimization을 이용해도 시간복잡도가 동일하게 보장되면서 immutable하게 자료구조를 구성할 수 있다는 사실을 몰랐던 것 같다(질문했더니 persistent array를 생각했다는 엉뚱한 답변이 되돌아왔다). 결국 끝나고 저자한테 직접 가서 얘기를 조금 더 했는데, 나는 아마 속도에 유의미한 향상이 있을 것

같다고 생각해서 나중에 구현체가 업데이트되면 좀 확인해보고 싶다.

4. Food

항상 말하지만 밥은 학회의 전반적인 연구 수준을 대표하는 만큼 정말 중요하다. 첫 날은 이상한 튀김만 잔뜩에 갈비 두 조각 있는 도시락을 줘서 매우 기분이 좋지 않았는데(심지어 10만원이 넘었다는 이야기를 전해 들었다), 둘째 날부터는 나름 풍성하고 맛있는 뷔페를 제공해서 좋았다. 메인 음식은 그렇게까지 맛있지는 않았지만 충분히 좋은 퀄리티였고 특히 디저트가 훌륭했는데, 특히 블루베리가 올려진 한 입 크기의 치즈 케이크와 레몬 타르트가 매우 맛있었다. 거기에 과일도 비싸서 잘 먹지 못하는 용과가 제공되어서(퀄리티가 그렇게 좋지 않기는 했다), 용과를 잔뜩 쌓아두고 하루에 한 개 정도 분량은 먹었던 것 같다. 음료가 없었던 것은 조금 아쉬웠다!

반면에 뱅퀫과 커피 브레이크에 제공되었던 간식은 너무나도 실망스러웠다. 뱅퀫에서 나온 갈비는 다 식어서 질겼고 양도 적었다. 커피 브레이크때는 편의점 과자보다 맛없는 과자와 뜨거운 커피만이 제공되었는데, 호텔의 명성을 순식간에 떨어지게 할 정도의 퀄리티였다. 이번 PLDI에서는 최첨단 garbage collecting 시스템이 적용되어서 직원 분들이 쉴새없이 돌아다니며 구석구석의 빈 접시와 쓰레기들을 수집해가셨는데, 그럴 돈으로 코스트코에서 간식을 조금 더 공수해주셨으면 어떨까 싶은 마음이었다.

Figure 1: 샐러드와 두부 버섯 구이. 두부 버섯 구이는 생각보다 괜찮았다!

Figure 2: 나물쌈밥, 갈비와 패션프루트 아이스크림을 곁들인 망고 판나코타. 판나코타는 훌륭해서 두 개 먹었다.