자기소개서

전민석

March 26, 2024

모든 문제는 알맞은 언어를 디자인 및 사용하여 풀어야 한다. 문제와 알맞는 언어가 만나면, 문제의 본질을 꿰뚫어볼 수 있게 되고 그에 따라 문제를 제대로 해결하는 방법을 찾을 수 있게 되기 때문이다. 이는 컴퓨터학 분야에서도 마찬가지이다. 컴퓨터학 각 분야의 문제들의 본질을 꿰뚫어보고, 그에 따른 해결책을 찾기 위해 적합한 언어를 디자인하고 사용하는 것이 중요하다. 이 철학을 기반으로 컴퓨터학 분야의 다양한 문제들의 해결책을 표현하기 위한 언어들을 디자인해왔고, 언어 위에서 정답을 찾는 알고리즘을 개발해왔다. 디자인한 언어들은 문제의 본질적인 해결책을 만들 수 있도록 해주었고, 이를 통해 혁신적인 연구 및 수업 결과물들을 만들어 내었다. 컴퓨터학 분야에서 문제를 해결하기 위한 언어를 디자인한다는 것은 각 분야의 특화된 프로그래밍 언어(domain-specific programming language)를 디자인하는 것을 의미한다. 디자인 한 도메인 특화 프로그래밍 언어들은 문제의 정답을 프로그램으로써 표현할 수 있도록 디자인 되어야 하며, 이를 위해 문제의 본질적인 특성을 반영하여야 한다. 이러한 언어들은 각 문제의 정답(해결책)을 표현하기 위한 문법과 의미론을 제공하며, 디자인 한 언어 위에서 문제의 해결책을 찾아내는 알고리즘을 개발할 수 있게 해준다.

설명 가능한 그래프 기계학습, 소프트웨어 테스팅, 소프트웨어 정적 분석, 프로그래밍 교육등 다양한 컴퓨터학 분야의 문제들의 본질적인 해결책을 제시해기 위해 도메인 특화 프로그래밍 언어들을 디자인하고 알고리즘을 개발해왔다. 설명 가능한 그래프 기계학습을 위해 다지인 한 특화 프로그래밍 언어는 본질적으로 설명 가능한 그래프 기계학습 모델을 표현할 수 있도록 디자인 되었으며, 이를 통해 학습한 그래프 기계학습 모델은 내놓은 모든 결과에 대해 올바른 설명을 제공할 수 있게 된다. 정적 분석을 위한 특화 프로그래밍 언어는, 분석을 빠르고 정확하게 해주는 정적 분석 전략을 표현할 수 있도록 디자인 되었으며, 소프트웨어 테스팅 문제를 해결하기 위한 특화 프로그래밍 언어는 효과적인 테스트 케이스를 쉽게 생성할 수 있도록 디자인 되었다. 학계에서는 디자인한 특화 프로그래밍 언어의 우수성을 인정 받아, 소위 최우수 국제 학술대회들(PLDI 2024, POPL 2022, OOPSLA 2020, OOPSLA 2018, OOPSLA 1017)에서 논문들을 발표해왔다. 제안한 언어 위에서 후속 연구들이 활발히 진행되고 있으며, 산업계에서도 제안한 언어를 사용하여 문제를 해결하고 있다. 프로그래밍 수업에서도 또한 수업에서 발생하는 문제들을 해결하기 위해 디자인한 특화 언어로 학생들이 프로그래밍을 배우면서 발생하는 문제들을 해결하고 있다.

1 설명 가능한 그래프 기계학습을 위한 특화 프로그래밍 언어

프로그램 분석, 이상 거래 탐지, 프로그램 수정, 약물 발견 등 다양한 현실 세계의 문제들은 그래프 문제로 표현될 수 있다. 이러한 문제들은 그래프 기계학습 문제로 모델링될 수 있으며, 현재 인공신경망 기반의 그래프 기계학습 모델(Graph Neural Networks)이 많은 그래프 문제들을 해결하고 있다 [8]. 그러나, 이러한 인공신경망 기반 모델들은 내놓은 결과에 대한 설명을 제공하지 않는 단점이 있다. 이상 거래 탐지 및 신약 개발과 같은 사람의 안전과 관련된 중요한 문제들에서는 모델의 결과에 대한 설명이 필수적이기 때문에, 인공신경망 기반 그래프 기계학습 방법은 적합하지 않다. 그래프 기계학습 모델의 결과에 대한 설명을 제공하는 방법들이 다양하게 제안되어 왔지만, 인공신경망 설명의 본질적 어려움 때문에 제안된 방법들 또한 모델의 결과에 대한 실제 이유를 반영하는 제대로 된 설명을 제공하지 못한다 [9].

프로그래밍 언어 분야 최우수 국제 학술대회인 PLDI2024에 발표하게 될 논문에서는 설명 가능한 그래프 기계학습 문제의 본질적 해결책을 제시하기 위해, (1) 새로운 특화 프로그래밍 언어인 그래프 패턴 언어(Graph Description Language)를 디자인하고, (2) 이를 이용하여 설명 가능한 그래프 기계학습 모델을 학습하는 알고리즘을 개발하였다. 정확한 모델을 표현할 수 있도록 다양한 그래프 패턴을 표현할 수 있는 언어를 디자인 하여 각 그래프 데이터의 핵심적인 특성을 표현할 수 있도록 하였다. 모델 합성 알고리즘은 주어진 학습 데이터에서 핵심 그래프 패턴을 추출하고, 추출한 패턴을 이용하여 설명 가능한 그래프 기계학습 모델을 학습한다. 학습된 모델은 임의의 새로운 그래프 데이터에 대하여 이전에 학습한 그래프 패턴을 이용하여 분류를 수행하며, 분류 결과에 대한 설명으로 사용한 그래프 패턴을 제시한다. 실제로 분류에 사용한 그래프 패턴을 제시하기 때문에, 제공된 설명은 옳은 설명임을 보장한다. 실험 결과는 제안한 설명 가능한 그래프 기계학습 모델이 기존의 인공신경망 기반 모델들과 비교해 비슷한 정확도를 보이지만, 월등히 높은 설명력을 가지고 있음을 보여준다. 현재 개발한 특화 언어 위에서 학생들과 다양한 후속 연구들을 진행하고 있다. 현재 (1) 연구에서 개발한 언어의 표현력 확장에 대한 연구와 (2) 인공신경망 기반 모델과의 결합 방법, (3) 프로그램 분석에서의 활용에 대한 연구가 진행되고 있다.

2 고성능 정적 분석 전략 합성을 위한 특화 프로그래밍 언어 및 합성 알고리즘

정적분석은 주어진 프로그램이 소스코드로부터 프로그램이 실행 중 가질 수 있는 성질을 분석하는 방법이고, 현재 소프트웨어 개발에서 매우 중요한 역할을 하고 있다. 성공적으로 정적 분석을 수행하기 위해서는 정적 분석을 빠르고 정확하게 해주는 분석 전략이 필수적이다. 전통적인 분석 전략들은 전문가들에 의해 수동으로 만들어져 왔는데, 수동으로 분석 전략을 디자인 하는 것은 매우 어렵고 시간이 많이 걸리는 작업이다. 또한, 수동으로 디자인한 분석 전략들은 낮은 성능을 보이는 경우가 많다.

이전 연구들에서는 자동으로 고성능 분석 전략을 합성하기 위해 (1) 분석 전략을 표현하는 특화 프로그래밍 언어(Feature Language)를 디자인하고, (2) 이를 이용하여 고성능 분석 전략을 합성하는 알고리즘을 개발하였다 [6, 3, 4, 5, 1, 2]. 해당 결과들은 POPL 2022, OOPSLA 2020, OOPSLA 2018, OOPSLA 2017 등 최우수 국제 학술대회에서 발표되었으며, 현재 타 연구 그룹에서도 제안한 언어 위에서 후속 연구들을 활발하게 진행하고 있다.

DNF 기반 분석 전략 합성을 위한 특화 프로그래밍 언어 및 합성 알고리즘 [6, 2, 1]. 정적 분석에서 성능에 가장 큰 영향을 미치는 요소 중 하나는 함수 문맥 민감도(context sensitivity)이다. 함수 문맥 민감도는 함수 호출 시 호출된 함수의 문맥을 고려하여 정확하게 분석하는 것을 의미한다. 프로그램의 모든 함수 호출에 문맥 민감도를 적용하면 분석의 정확도는 높아지지만, 분석 시간은 급격히 증가한다. 반면, 모든 함수 호출에 문맥 민감도를 적용하지 않으면 분석 시간은 줄어들지만, 분석의 정확도가 매우 낮아진다. 이 문제를 해결하기 위해 선택적 문맥 민감도 전략이 제안되었는데, 문맥 민감도를 정확하게 분석해야 할 함수호출들에게만 선택적으로 적용하여 높은 분석 정확도를 유지하면서 분석 시간을 최소화 방법이다.

자동으로 선택적 문맥 민감도 전략을 합성하기 위해 (1) DNF 식으로 분석 전략을 표현하는 특화 프로그래밍 언어를 디자인하고, (2) 이를 이용하여 선택적 문맥 민감도 전략을 합성하는 알고리즘을 개발하였다. 정확하게 분석해야 할 함수들은 DNF 식으로 표현되며, 합성 알고리즘은 학습 데이터로부터 DNF식을 자동으로 합성하게 된다. 실험에서는, 제안한 합성 알고리즘들이 자동으로 합성해낸 분석 전략들이 기존의 전문가들이 수동으로 디자인한 분석 전략들과 비교해 월등히 더 높은 성능을 보이는 것을 확인하였다.

컨텍스트 터널링 [3]. 정적 분석에서는 함수 호출을 요약하는 방법이 필요하다. 함수 호출요약 없이 정적 분석을 수행하면 일반적으로 무한히 많은 함수 호출을 만나게 되어 분석이 끝나지 않는다. 현재 함수 호출을 요약하기 위해 널리 사용되고 있는 방법은 k개의 함수호출 요소로 함수 호출을 요약하는 k-제한 함수 호출 요약 방법이고, 표준은 가장 최근 k개의 함수 호출 요소로 함수 호출을 요약하는 마지막 k-제한 함수 호출 요약 방법이다. 하지만, 표준으로 사용되고 있는 마지막 k-제한 함수 호출 요약의 치명적인 단점을 발견하였는데, 이

방법은 중요한 함수 호출 요소가 마지막 k개에 포함되지 않으면 중요한 함수 호출 요소를 잃어버리게 된다는 것이다. 이로 인해 분석의 정확도가 현저히 떨어진다는 것을 발견하였다.

이 문제를 해결하기 위해, 컨텍스트 터널링을 제안하였다. 컨텍스트 터널링은 중요한 k-제한 함수호출 요소로 함수 호출을 요약할 수 있게 해주는 방법이다. 하지만, 컨텍스트 터널링을 적용하기 위해선 중요한 함수 호출 요소를 알아내야 하는데, 분석 이전에 중요한 함수 호출 요소를 알아내는 것은 매우 어려운 문제이다. 컨텍스트 터널링 전략을 자동으로 합성하기 위해 (1) 중요한 함수 호출 요소를 알아내는 특화 프로그래밍 언어를 디자인하고, (2) 이를 이용하여 중요한 함수 호출 요소를 자동으로 찾아내는 알고리즘을 개발하였다. 실험에서는, 합성해낸 컨텍스트 터널링 전략으로 인해 분석의 정확도가 획기적으로 향상됨을 보였다.

값 기반 함수 호출 요약을 더 정확한 함수 호출 요약으로 변환하기 [5]. 객체 지향 프로그램을 대상으로 하는 함수 호출 요약에서는, 함수 호출시 전달된 값들을 이용하여 함수 호출을 요약하는 값 기반 함수 호출 요약 방법이 높은 정확도를 이유로 가장 좋은 함수 호출 요약방법으로 여겨지고 있다. 전세계 정적 분석 강의에서도 값 기반 함수 호출 요약이 가장 좋은 방법이라고 가르치고 있으며, 대부분의 연구들은 값 기반 함수 호출 요약의 성능을 향상시키는 방법을 연구하고 있다 [7]. 반면, 함수의 호출 위치를 이용하여 함수 호출을 요약하는 위치 기반 함수 호출 요약 방법은 낮은 정확도를 이유로 가장 나쁜 함수 호출 요약 방법으로 여겨지고 있다.

이러한 분위기 속에서 이 연구는 위치 기반 함수 호출 요약 방법이 오히려 값 기반 함수 호출 요약보다 더 좋은 방법이라는 것을 보였다. 이 연구에서는, 컨텍스트 터널링이 적용된 경우 주어진 값 기반 함수 호출 요약을 더 정확한 위치 기반 함수 호출 요약으로 변환할 수 있음을 보였다 [5]. 이러한 결과는 현재 널리 알려져 있는 값 기반 함수 호출 요약이 가장 좋은 방법이라고 가르치는 것이 옳지 않음을 보여주며, 위치 기반 함수 호출 요약이 가장 나쁜 방법이라고 가르치는 것 또한 옳지 않음을 보여주는 큰 임팩트를 가지는 결과이다.

특질 자동 생성을 위한 특질 언어 및 합성 알고리즘 [4]. 위 연구들에서 분석 전략을 DNF 식으로 표현하기 위해선 DNF 식의 재료로 사용되는 특질(feature)가 필요하다. 또한, 고성능 분석 전략을 학습해내기 위해선 고품질 특질들을 사용하여 분석 전략을 학습해야 한다. 이전에는 이러한 특질들은 사람이 수동으로 디자인 해왔으며 이 또한 매우 어려운 작업이었다. 이 연구에서는 이 문제점을 해결하기 위해 (1) 특질을 표현하는 특화 프로그래밍 언어를 디자인하고, (2) 이를 이용하여 고품질 특질을 자동으로 생성하는 알고리즘을 개발하였다. 실험 결과는 제안한 특질 언어와 합성 알고리즘으로 인해 고품질 특질을 자동으로 생성해냈음을 보였다. 이 연구는 현재 타 그룹들에서도 후속 연구들을 진행하고 있다.

3 고성능 소프트웨어 테스팅을 위한 특화 프로그래밍 언어

소프트웨어 테스팅은 오류를 일으키는 입력을 찾아내는 것을 목표로 하며, 소프트웨어 품질을 향상시키는데 중요한 역할을 한다. 소프트웨어 테스팅은 산업에서도 매우 적극적으로 사용되고 있는데, 삼성전자에서는 플래쉬 기반 저장장치의 수명을 평가하기 위해 소프트웨어 테스팅을 사용하고 있다. 플래쉬 기반 저장장치는 데이터 쓰기 요청을 처리하는 데 한계가 있기 때문에, 제조사는 제품이 안정적으로 견딜 수 있는 최대 데이터 쓰기 용량을 정확하게 테스트하고 사용자에게 제공해야 한다. 저장장치의 수명을 정확하게 측정하기 위해선 효과적인 테스트 케이스를 사용해야 하는데 수명평가용 테스트 케이스 또한 현장에서 사람에 의해 수동으로 만들어져왔으며 이는 매우 어려운 작업이고 만들어진 테스트 케이스 또한 효과적이지 못하였다.

ICST 2023 (industry track)에서 발표한 이 산학 협력 연구에서는 삼성전자에서 생성하는 플래쉬 기반 저장장치의 수명을 정확하게 평가하기 위한 고품질 테스트 케이스들을 자동으로 생성하기 위해 (1) 테스트 케이스를 표현하는 특화 프로그래밍 언어를 디자인하고, (2) 이를 이용하여 효과적인 테스트 케이스를 자동으로 생성하는 알고리즘을 개발하였다. 테스트 케이스를 표현하는 언어는 테스트 케이스의 탐색 공간을 효과적으로 줄여주는 역할을 하며, 이를 이용하여 합성 알고리즘은 효과적으로 고품질 테스트 케이스를 생성하였다. 실험에서는 현재 현업에서 사용하고 있는 테스트 케이스보다 26% 더 효과적인 테스트 케이스를 자동으로 생성해냈음을 보였다. 26% 더 효과적이다는 것의 의미는 현재 제품에서 보장해주고 있는 수명보다 더 빠르게

수명을 소진 시키는 테스트 케이스를 자동으로 생성해냈음을 의미한다.

4 프로그래밍 수업에서 발생하는 문제 해결을 위한 특화 프로그래밍 언어

프로그래밍 수업에서는 프로그래밍 과제가 주를 이루게 되는데, 이와 관련하여 다양한 문제들이 발생한다. 그 중 한 문제는 학생의 사소한 실수에 대해 처리하는 것이다. 프로그래밍 과제는 특성상 학생의 사소한 실수가 0점으로 이어질 수 있고, 이와 관련된 학생과 조교 사이의 갈등을 많이 경험하였다. 사소한 실수로 인해 많은 점수가 깎인 학생의 입장도 이해가 가지만, 조교 또한 사소한 실수라는 것이 다양하고 객관적으로 정의하기 어렵기 때문에 학생의 요청을 받아들이기 어렵다. 이 문제를 해결해보고자 조교로 참여한 수업에서, 사소한 실수를 명확한 언어로써 정의하고 이를 이용하여 과제 제출물에서 발생한 학생들의 사소한 실수에 대한 수정을 받아주는 방법을 개발하였다. 개발한 언어는 수업에 참여한 학생들과 함께 확장을 해나 갔었는데, 첫번째 버전에서는 코드를 지우는 것만으로 수정 가능한 실수들을 사소한 실수로 정의 하였지만, 학생들이 제안한 아이디어들로 인해 오타, 띄어쓰기등의 사소한 실수들도 표현할 수 있도록 확장되었다. 학생들 또한 실수에 대한 명확한 정의가 있음에 만족하였고, 자신의 점수에 대한 부정적인 항의들은 실수 표현 언어에 대한 건설적인 확장 제안들로 바뀌었다.

5 최종 목표

References

[1] Donghoon Jeon, Minseok Jeon, and Hakjoo Oh. A practical algorithm for learning disjunctive abstraction heuristics in static program analysis. *Information and Software Technology*, 135:106564, 2021.

[2] Minseok Jeon, Sehun Jeong, Sungdeok Cha, and Hakjoo Oh. A machine-learning algorithm with disjunctive model for data-driven program analysis. *ACM Trans. Program. Lang. Syst.*, 41(2):13:1–13:41, June 2019.

[3] Minseok Jeon, Sehun Jeong, and Hakjoo Oh. Precise and scalable points-to analysis via data-driven context tunneling. *Proc. ACM Program. Lang.*, 2(OOPSLA):140:1–140:29, October 2018.

[4] Minseok Jeon, Myungho Lee, and Hakjoo Oh. Learning graph-based heuristics for pointer analysis without handcrafting application-specific features. *Proc. ACM Program. Lang.*, 4(OOPSLA), November 2020.

[5] Minseok Jeon and Hakjoo Oh. Return of cfa: Call-site sensitivity can be superior to object sensitivity even for object-oriented programs. *Proc. ACM Program. Lang.*, 6(POPL), jan 2022.

[6] Sehun Jeong, Minseok Jeon, Sungdeok Cha, and Hakjoo Oh. Data-driven context-sensitivity for points-to analysis. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA), 2017.

[7] Yannis Smaragdakis and George Balatsouras. Pointer analysis. *Foundations and Trends in Programming Languages*, 2(1):1–69, 2015.

[8] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[9] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–19, 2022.