교육 및 연구계획서

전민석

April 9, 2024

교육과 연구는 순환관계를 형성한다. 교육은 연구의 결과물들을 학생들에게 전달하고, 학생들은 이를 소화하여 자신의 문제를 해결하는 능력을 키우고 새로운 문제에 도전하며 이 과정에서 연구를 통해 새로운 지식을 창출한다. 새로이 창출된 지식은 다시 교육에 반영되어 새로운 학생들에게 전달된다. 나의 목표 또한 새롭고 가치 있는 지식의 순환을 일으키는 것이다.

새롭고 가치있는 순환을 만들어내기 위해선 문제의 본질을 꿰뚫어 보고자 해야 한다. 연구는 문제를 풀어내는 과정이지만, 문제의 본질을 이해하는 과정이기도 하다. 주어진 문제를 제대로 해결하기 위해서는 현재 문제가 발생한 본질적인 원인이 무엇인지, 왜 이것이 본질적인 원인인지 이해하며, 이를 근본적으로 해결하기 위한 방법이 무었인지를 고민해야 한다. 사람마다 문제의 본질에 대한 고민과 이해가 제각기 다르기 때문에, 제각기 다른 해결책이 제시되고, 이를 통해 기존의 지식이 발전되기도 하고 뒤집어지기도 하며 교육과 연구의 건강한 순환을 일으킨다. 반면, 문제의 본질을 스스로 고민하지 않은 채 다른이들이 내놓은 연구들을 따라 문제에 접근하는 것은 제대로된 해결책을 내놓을 수 없을 뿐더러 교육과 연구의 건강한 순환을 만들어낼 수 없다.

문제의 본질을 꿰뚫어 보기 위한 본 연구자의 방법은 문제에 대한 도메인 특화 프로그래밍 언어 디자인 (domain-specific programming language design) 및 특화 언어 프로그램 합성 알고리즘 개발이다. 문제에 대한 해결책을 찾기 위해서는 해결책을 표현할 수 있는 언어가 필수적이다. 특화 프로그래밍 언어 디자인의 목표는 문제의 정답을 하나의 프로그램으로써 표현할 수 있는 언어를 디자인하는 것이고, 특화 언어 프로그램 합성 알고리즘의 목표는 특화 프로그래밍 언어로 표현된 공간 안에서 문제의 해답이 되는 정답 프로그램을 자동으로 찾아내는 것이다. 문제의 정답을 표현할 수 있는 언어를 디자인하는 과정속에서 문제의 본질에 대한 고민을 하게 되고, 정답 프로그램을 합성하는 알고리즘을 개발하는 과정에서 문제의 성질을 이해하게 된다. 위 철학을 기반으로 지금까지 정적 프로그램 분석, 동적 프로그램 분석, 설명가능한 기계학습, 프로그래밍 교육 등 다양한 컴퓨터학 분야의 문제에 대한 해결책들을 제시해왔다. 위 경험과 철학으로 교육과 연구를 진행하고자 한다.

1 강의 가능 과목

학생들에게도 특화 프로그래밍 언어 디자인 및 프로그램 합성 알고리즘 개발을 통한 문제 접근을 전달하고자 한다. 이를 위해 기반이 되는 과목인 계산이론, 프로그래밍 언어, 프로그램 분석, 컴파일러, 소프트웨어 공학, 알고리즘을 강의하고자 한다. 이후 특화 프로그래밍 언어 디자인 및 프로그램 합성 과목을 개발하여 학생들에게 본 연구자의 방법론을 전달할 것이다.

계산이론. 이 과목의 목표는 언어의 문법을 정의하는 법 및 튜링머신과 계산 가능성에 대한 이해를 제공하는 것이다. 과목의 첫번 째 목표는 학생들은 오토마타, 정규표현식, 문맥 자유 문법을 전달하고, 이들의 표현력과 한계에 대해 이해하게 하는 것이다. 이 과정에서 (특화) 프로그래밍 언어에서의 문법을 정의하는 방법에 대한 기초적인 이해를 제공한다. 이후 컴퓨터과학의 기초가 되는 튜링머신을 배우게 되고, 이를 통해 계산 가능성에 대해 이해하게 된다. 마지막으로 계산 불가능성에 대해 배우게 되고, 이를 다양한 분야의 문제들(특히 정적 프로그램 분석)과 연관지어 설명할 것이다.

프로그래밍 언어. 이 과목의 목표는 프로그래밍 언어의 주요 개념들을 이해하고, 이를 통해 프로그래밍 언어를 디자인하는 방법을 배우는 것이다. 학생들은 우선 기본적인 프로그램 언어의 컨셉을 배우게 되고, 실제로 프로그래밍 언어를 구현해보는 과정을 통해 프로그래밍 언어의 문법과 의미를 정의하는 방법을 익히게 된다. 이후 타입 시스템과같이 진보된 프로그래밍 언어 개념들을 배우고 실제로 구현해봄으로써 프로그래밍 언어에 대한 이해를 높인다. 마지막으로, 학생들은 자신들만의 문제를 해결하기 위한 특화 프로그래밍 언어를 디자인해보는 프로젝트를 수행하게 하여 특화 언어 디자인을 통한 문제 접근을 경험해볼 수 있도록 할 것이다.

컴파일러. 컴파일러 과목의 목표는 컴파일러의 중요성 및 주요 개념들을 이해하고, 컴파일러를 설계하는 방법을 배우는 것이다. 학생들은 어떻게 하나의 프로그램밍 언어가 다른 프로그래밍 언어로 의미 변화 없이 안전하게 번역되는지에 대해 배우게 된다. 학생들은 컴파일러의 주요 구성 요소인 렉서, 파서, 번역기, 최적화기를 구현해보는 프로젝트를 수행하게 되어 컴파일러의 동작 방식을 이해하게 된다. 더 나아가 정적 분석 기술이 컴파일러 최적화에 어떻게 활용되는지에 대해 배우게 될 것이다.

소프트웨어 공학. 이 과목의 목표는 소프트웨어 개발의 주요 단계들을 이해하고, 소프트웨어 개발 방법론을 배우는 것이다. 학생들은 소프트웨어 요구사항, 소프트웨어 디자인, 소프트웨어 개발, 소프트웨어 분석, 소프트웨어 유지보수등 소프트웨어 공학의 주요 개념들을 배우게 된다. 이 과정에서 최근의 소프트웨어 공학 연구들을 소개하고, 학생들은 최신 소프트웨어 공학 연구 위에서 이를 개선해보는 프로젝트를 수행하게 하여 최신 소프트웨어 공학 기술 연구를 경험해볼 수 있도록 할 것이다.

알고리즘. 알고리즘 과목의 목표는 다양한 알고리즘의 작동 원리를 이해하고, 각 알고리즘이 어떤 문제를 해결하는 데 적합한지 파악하는 것이다. 학생들은 우선 알고리즘의 성능을 분석하고 비교하는 방법을 배우고, 알고리즘이 얼마나 효율적으로 동작하는지 평가하는 것을 익힌다. 이 후 다양한 문제들을 해결하기 위해 개발되어온 알고리즘들을 배우게 되고, 이를 실제로 구현해 보는 과정을 통해 알고리즘에 대한 이해를 높인다. 더 나아가 최근 학계에서 연구되고 있는 프로그램 합성 알고리즘 연구들을 소개하고, 이들을 개선해볼 수 있는 기회를 학생들에게 프로젝트로써 제공할 것이다.

프로그램 분석. 이 과목의 목표는 프로그램 분석의 기본적인 개념들을 이해하고, 이를 통해 프로그램 분석 도구를 디자인하는 방법을 배우는 것이다. 학생들은 다양한 프로그램 분석 기술들을 배우게 되고, 이를 실제로 구현해봄을 통해 프로그램 분석 기술의 동작 방식을 이해하고 개선하는 방향을 고민해보게 된다. 더 나아가 프로그램 분석 기술이 다양한 분야에 어떻게 적용되는지 배우게 될 것이다. 특히 프로그램 자동 합성에 프로그램 분석 기술이 어떻게 활용되는지에 대해 이해하게 될 것이다.

2 개발하고자 하는 과목

위 과목들을 기반으로 본 연구자의 연구 방향인 특화 프로그래밍 언어 디자인 및 프로그램 합성 과목을 개발하고자 한다.

특화 프로그래밍 언어 디자인 및 프로그램 합성. 본 과목의 목표는 주어진 문제에 대한 특화 프로그래밍 언어를 디자인 하고, 디자인 한 언어에서 문제의 정답 프로그램을 합성하는 방법을 배우는 것이다. 수업은 아래의 네가지 파트로 구성될 것이다.

- ***Part 1.*** 학생들은 우선 기초적인 특화 프로그래밍 언어 디자인에 대한 개념을 배우게 되고, 프로그램 합성의 목표를 이해하게 된다.

- ***Part 2.*** 기본적인 프로그램 합성 알고리즘인 하향식 (Top-down) 및 상향식 (Bottom-up) 알고리즘을 배우게 되고, 이를 구현해봄으로써 프로그램 합성의 기본적인 개념을 이해하게 된다.

- ***Part 3.*** 학생들에게 프로그래밍 교육/머신러닝/소프트웨어 공학/프로그래밍 언어 등 다양한 분야에서 문제에 접근하기 위해 디자인 된 실용적인 특화 프로그래밍 언어들 [5, 6, 2, 1, 8, 7, 3, 4, 9] 및 합성 알고리즘들을 소개한다.

- ***Part 4.*** 학생들 개인이 직면한 문제들을 정의하고, 이를 해결하기 위한 특화 프로그래밍 언어 및 합성 알고리즘을 개발하는 프로젝트를 수행하게 할 것이다.

위 과정을 통해 학생들에게 특화 프로그래밍 언어 디자인 및 프로그램 합성을 통한 문제 접근 방법론을 전달하고자 한다.

3 중단기 연구 계획

이 연구 방향의 중단기적 목표는 (1) 지금까지 개발해온 특화 프로그래밍 언어 및 합성 알고리즘을 개선하거나 (2) 적용 범위를 확장하는 것이다. 현재까지 개발한 특화 언어들은 초기 단계의 프로토타입 언어들이며, 특화 언어와 합성 알고리즘 모두 아래와 같이 개선의 여지가 매우 많이 남아있다. 또한, 더 많은 문제들에 대해서 위 방법론을 적용하여 문제 해결의 범위를 확장해 나가고자 한다. 현재 세명의 후배들을 지도해가며 아래와 같은 연구들을 진행중이다.

적용 범위 확장. 현재 특화 언어 기반 문제 접근법을 소프트웨어 엔지니어링 분야의 자동 결함 위치 추정 문제에 적용한 연구를 진행중이다. 자동 결함 위치 추정은 소프트웨어에서 오류가 발생하였을 때 오류의 원인이 되는 결함의 위치를 자동으로 찾아내는 기술로써 현업에서 적극적으로 사용되고 있는 실용적인 기술이다. 현재 진행중인 연구에서는 효과적인 자동 결함 위치 추정을 위해, 프로그램의 결함 패턴을 표현할 수 있는 특화 프로그래밍 언어를 디자인하고 학습 데이터로부터 결함 패턴을 합성하는 알고리즘을 개발하였다. 현재 마무리 단계에 있으며 소프트웨어 공학 분야 최우수 학술대회인 ICSE 2025에 제출할 예정이다. 이 후에도 다양한 분야의 문제들에 대해 이 방법론을 적용하여 문제 해결의 범위를 확장해 나갈 것이다.

특화 프로그래밍 언어 개선. 개발한 특화 프로그래밍 언어들의 표현력을 개선하는 연구를 진행중이다. 특화 프로그래밍 언어의 표현력이 너무 낮은 경우 언어가 문제의 정답을 표현할 수 없게 되고, 특화 프로그래밍 언어의 표현력이 너무 높은 경우 탐색 공간이 매우 넓어져 합성 알고리즘이 문제의 정답을 찾지 못하게 된다. 따라서 특화 프로그래밍 언어는 문제의 정답을 표현할 수 있는 최소한의 표현력만을 가지고 있어야 하는데, 이를 만족하는 특화언어를 디자인하는 것은 매우 어려운 일이다. 현재까지 개발한 특화 언어들은 위를 만족하는 이상적인 언어와는 거리가 멀다. 완벽한 정답을 표현할 수 없는 경우도 많이 있고 탐색 공간 또한 매우 넓어 합성 알고리즘이 많은 시간을 소모하고 있다. 현재 그래프 기계학습을 대상으로 디자인 한 특화 언어에 대해서 이 문제를 해결하기 위한 연구를 후배 학생들과 진행중이다.

프로그램 합성 알고리즘 개선. 합성 알고리즘을 효율적으로 개선하는 연구 또한 진행중이다. 특화 언어가 문제의 정답을 표현할 수 있다고 하더라도, 합성 알고리즘이 정답을 찾아낼 수 없다면 문제를 해결할 수 없다. 따라서 주어진 탐색 공간(특화 언어)에서 문제의 정답을 효율적으로 찾아낼 수 있는 합성 알고리즘이 필요하다. 이전 연구에서 개발한 합성 알고리즘들 모두 잘 알려진 기본적인 탐색 알고리즘들만을 적용한 것이며, 이들을 개선하기 위한 연구 또한 진행중이다.

4 장기적 연구 목표

궁극적으로는 주어진 문제에 대해서 자동으로 특화 프로그래밍 언어 및 특화 언어 프로그램 합성 알고리즘을 생성하는 시스템을 개발하고자 한다. 현재까지 개발한 특화 언어 및 합성 알고리즘들은 모두 수동으로 직접

문제를 분석해 디자인한 것들이고, 이 과정은 시간과 노력이 매우 많이 소모되는 작업이었다. 자동으로 언어 디자인 및 알고리즘 개발 작업을 수행하는 시스템을 개발하여 위 수고를 덜어주고, 특화언어 기반 문제 접근법을 하나의 대중적인 방법론으로 만들어내는 것이 장기적 목표이다.

References

[1] Ria Das, Joshua B. Tenenbaum, Armando Solar-Lezama, and Zenna Tavares. Combining functional and automata synthesis to discover causal reactive programs. *Proc. ACM Program. Lang.*, 7(POPL), jan 2023.

[2] Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. Inversecsg: Automatic conversion of 3d models to csg trees. *ACM Trans. Graph.*, 37(6), dec 2018.

[3] Kevin Ellis, Adam Albright, Armando Solar-Lezama, Joshua B. Tenenbaum, and Timothy J. O'Donnell. Synthesizing theories of human language with bayesian program induction. *Nature Communications*, 13(1):5024, 2022.

[4] Sumit Gulwani. Automating string processing in spreadsheets using input-output examples. In *Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '11, pages 317–330, New York, NY, USA, 2011. Association for Computing Machinery.

[5] 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.

[6] Jinkook Kim, Minseok Jeon, Sejeong Jang, and Hakjoo Oh. Automating endurance test for flash-based storage devices in samsung electronics. In *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*, 2023.

[7] Vu Le and Sumit Gulwani. Flashextract: A framework for data extraction by examples. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '14, pages 542–553, New York, NY, USA, 2014. Association for Computing Machinery.

[8] Oleksandr Polozov and Sumit Gulwani. Flashmeta: A framework for inductive program synthesis. In *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA 2015, page 107–126, New York, NY, USA, 2015. Association for Computing Machinery.

[9] Megan Tjandrasuwita, Jennifer J. Sun, Ann Kennedy, Swarat Chaudhuri, and Yisong Yue. Interpreting expert annotation differences in animal behavior. *CoRR*, abs/2106.06114, 2021.