Thesis Project Portfolio

Towards General Compilation for Heterogeneous Backends: The Unified Compiler

(Technical Report)

Generative AI on Computer Science Education: From the Computer Science Faculty Perspective

(STS Research Paper)

An Undergraduate Thesis

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Neil Phan

Spring, 2024

Department of Computer Science

Contents of Portfolio

Executive Summary

This summary covers the technical paper component and STS research question component that has been worked on over the past year. Each component will be addressed separately due to their lack of connection with one another. The technical paper covers a proposal in the field of programming languages and compilers. Given the numerous hardware devices that are being developed with their own in-house compiler and programming language, the need to pick up new languages and overcome programming barriers has been larger than ever. Under the unified compiler proposal, instead of having to learn many programming languages, one can use a singular programming language that is able to translate into many hardware backends. The STS research question paper covers a completely different topic: how has computer science education shifted from the introduction of generative AI, and what is the perspective of computer science educators on the matter? The paper covers multiple computer science professors that have been interviewed on questions on how their course structure has changed, what they anticipate in the future with generative AI, and much more. Afterwards, a conclusion on general trends in how education is shifting will be discussed and analyzed.

The technical report paper covers a proposal that plans on tackling a common problem in programming: how can one simplify the barrier to programming on new hardware and new technologies on the edge? The existence of compilers allows programmers to write higher level programs that translate down into lower level binaries that will be run on hardware. However, there exists a plethora of programming languages that exist that all work for different use cases and different hardware devices. This requires programmers to have to learn completely new languages and understand their use cases in order to be proficient in programming for that hardware. This proposal introduces a unified compiler, which attempts to encapsulate a

magnitude of different hardware backends under one programming language. By introducing "middle-end" languages via MLIR, a tool that is used to describe different languages at different levels, the proposal defines how the unified compiler can use MLIR to define many different middle-ends that reduce down to different hardware devices. The overall design involves a front-end for the high-level programming language, a middle-end that maintainers will introduce translations for, and the back-ends which have already been developed by the hardware companies. For example, there exist two different types of GPU brands, NVIDIA and AMD, each with their own compiler (nvvm and rocdl respectively). The unified compiler would build middle-ends that would translate the high-level programming language down to the respective GPU compiler, as long as the programmer specifies the back-end they would like to compile down to. To verify that the design works, a suite of benchmarks will be developed that will verify that the file will run on the hardware properly and efficiently. Some future works of the proposal include creating the baseline for the unified compiler for GPUs and introducing more niche hardware backends to the proposal.

The STS research question paper covers the following question: how are computer science educators reflecting and adapting to changes in computer science education from generative AI? The methods for investigating this major question revolved around conducting in-person interviews with computer science professors at the University of Virginia. Two professors were selected and interviewed and asked the same questions related to computer science education and Generative AI. The four question asked are as follows:

- How has generative AI been handled in courses you have taught recently?
- How do you feel that generative AI has changed the student's learning experience from your own experience?

- How does the future of generative AI play into how your courses are constructed in the future?
- From your perspective, what are the benefits and downsides of introducing generative AI into computer science education?

After interviewing each of the professors and collecting their answers, some commonalities and differences between the two were summarized in the conclusion. It can be concluded that with generative AI assisting in assignments, there will be a shift in grading schemas for computer science courses to focus more on exams and in-person assignments. The difference between undergraduate and graduate courses is also substantial, where lower-level courses that need students to learn fundamentals are hesitant on the adoption of generative AI, while higher-level courses welcome generative AI to support tasks such as debugging code to streamline the process further. Ultimately, the adoption of generative AI in computer science courses is uncertain and requires professors to be keen to future updates. For now, leaning towards caution by restricting generative AI usage where possible is ideal.

Overall, the technical project was an interesting way to dive into the world of compilation and programming languages and see where the future holds with it. Although no actual results were obtained from the proposal, it can serve as a placeholder for future engineers to take upon action. For the STS research paper, I was satisfied with the results of the interviews and learned a lot about the perspectives that professors take for educating their students. While some of the trends were expected, there were other trends that were interesting with computer science education that will depend on the future of generative AI.