The Next Compiler

Dongjae Lee

2025.03.20

Abstract

The advent of compilers has made machine and assembly languages largely obsolete for most programmers. While some developers still use assembly language for specific purposes, it generally requires more development time and resources compared to high-level languages. Now, with large language models, some developers use natural language for programming, bypassing conventional programming languages entirely. In this era of next-generation compilers, I explore these changes from the perspective of programming language research.

The evolution of technology follows a pattern of high expectations, setbacks, and improvements. The development of compilers and high-level languages exemplifies this pattern. When high-level languages emerged, many believed they would simplify programming and reduce the need for technical expertise. However, these technologies have created new opportunities for programming language researchers.

The era of natural language programming has arrived, blurring the boundaries between programming languages and human communication. Does this transformation render programming language research obsolete?

Not at all. New technologies introduce new challenges, particularly the gap between natural language specifications and code implementations. In conventional compilers, translation validation ensures the compiled program matches the source code. We need similar tools to bridge the gap between natural language and code.

Another challenge lies in high-level language optimization. Unlike conventional optimization in compilers using rule-based optimization, high-level optimization often requires algorithmic changes that are difficult to represent in formal rules. Language models could help by generating optimization guidelines beyond strict rules. These models could then identify and apply optimizations to existing programs, with verification mechanisms to ensure success.