The Next Compiler

Dongjae Lee

2025.03.20

Abstract

The advent of compilers has made machine and assembly languages largely obsolete for most programmers. Few can still use assembly language, but it is inefficient in terms of time. High-level languages, being closer to human speech, are convenient and allow for concise coding. Now, with language models, some people bypass programming languages altogether. In this era of next-generation compilers, I explore perspectives on these changes as someone involved with programming languages.

The rise of new technologies often brings high hopes, big disappointments, and eventually, better living standards. Compilers and high-level languages are no different. Many thought high-level languages would make programming too simple and reduce expertise. However, new tech has brought fresh challenges, benefiting programminglanguage researchers.

Language models seem to disrupt this status quo, as programming languages face replacement by natural language. Techniques like type checking, and static analysis are too flexible for natural language.

However, new tech inevitably creates new issues, such as the inconsistency between natural language and code. In compilers, Translation Validation checks for mismatches between original and complied program. Creating tools to find these gaps between natural language and code could be promising.

Another challenge is optimizing in high-level languages, which requires changing algorithms, making rule-based methods tough like optimization pass in conventional compilers. However, using language models to draft a manual of optimization guidelines, not strictly rule-based, could help. Language models can use this guide to find and apply optimizations to current programs. Adding a feature to verify optimization success based on guidelines would be ideal.