Super Abstracted Programming Language

Dongjae Lee

2025.05.31

Abstract

Programming languages have become increasingly abstracted over time. This is because advances in hardware and compilers have eliminated the need to consider low-level elements when programming. As a result, productivity and efficiency have dramatically increased. Language models will further abstract programming languages to a higher dimension, making programming tasks easier and more efficient.

In the past, due to limited hardware performance and compilers, programmers had to program with hardware considerations in mind. As hardware has advanced and compilers have added numerous optimizations, most programmers no longer need to consider hardware. Thanks to this, software development productivity has been able to increase dramatically.

Abstraction, in other words, means that you don't have to specify concretely how to do it. Now programmers don't have to specify concretely what sorting algorithm to use, how to manage memory, how to handle network communication, etc.

Language models will enable programmers to program declaratively at a very high level of abstraction. Compilers of existing programming languages translated declarative elements into procedural machine code according to predetermined rules. However, language models can understand user intent and convert it into procedural algorithms even in situations where there are no predetermined rules.

However, to ensure correctness, there must be a guarantee that the converted algorithm is correct. In the case of existing compilers, this is guaranteed through numerous tests or verification techniques. However, since static inspection is virtually impossible for language models, it is necessary to dynamically check whether the converted algorithm meets the user's instructions. I would like to guarantee this through EXPECTO and create a new programming language and compiler based on this.