

# 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 reduced the need to consider low-level elements when programming. As a result, productivity and efficiency have dramatically increased. Language models will push programming languages to an even higher level of abstraction, making programming tasks easier and more efficient.

In the past, due to limited hardware performance and compiler capabilities, programmers had to develop software with hardware limitations in mind. As hardware has advanced and compilers have added numerous functionalities, most programmers no longer need to consider hardware. Thanks to this, software development productivity has increased dramatically.

Abstraction, in other words, means not needing to specify the concrete implementation details. Now, programmers are often not required to explicitly define what sorting algorithm to use, how memory should be managed, or how network communication should be handled, among other details.

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