A Language for Writing Prompts

Dongjae Lee

2025.07.16

Abstract

Since prompts are written in natural language, it is very difficult to identify the cause of a problem when one arises. Just as inconsistencies can arise between specifications written in natural language and their actual implementation, prompts are also highly prone to inconsistencies when the subject they describe is constantly changing. This article discusses the causes of these problems and potential solutions.

It is very difficult to read and remember a whole prompt written in natural language. For example, Claude's system prompt is 16,000 words long. If the subject described in the prompt is constantly changing (e.g., API specification), the prompt must be continuously updated, which is a very difficult task.

This problem is very similar to the issue of inconsistency between traditional software specifications and their implementation. Because specifications are written in natural language, if the person in charge forgets, changes in the implementation may not be reflected in the specification, and discovering such inconsistencies is also very difficult.

Developing a programming language for writing prompts could be one solution. This is because if prompts are written in a formal language, it is possible to develop tools to analyze and verify their correctness. However, writing everything in a formal language, while powerful, is very difficult and inefficient for the general public.

Another approach is to find an intermediate form between natural language and formal language, and to use language models to verify the correctness of the prompt. One example is to wrap the changing parts with a `<Mutable>` tag, and have the language model repeatedly check the content of this section in a CI process. A kind of Markup Language for writing prompts could be created by introducing various tags.